Universidad de las Ciencias Informáticas
Facultad 1

Título:

Administración del servicio SSH2 desde la Herramienta para la Migración y Administración de Servicios Telemáticos HMAST

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

Autora:
Eva Lianet Ferrer Ruano

Tutores:
Ing. Nadia Porro Lugo
Ing. Reidiel Castillo Arbelo

La Habana

Junio 2014

¨Si una persona es perseverante, aunque sea dura de entendimiento, se hará inteligente; y aunque sea débil se transformará en fuerte¨

Leonardo Da Vinci

Declaración de Autoría

Declaro por este medio que Eva Lianet Ferrer Ruano, con carné de identidad 90090532374 estudiante de la Universidad de las Ciencias Informáticas (UCI), soy la autora de la tesis de pregrado "Administración del servicio SSH2 desde la Herramienta para la Migración y Administración de Servicios Telemáticos HMAST" y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio. Para que así conste, firmo la presente declaración jurada de autoría en la Habana el día 10 de Junio del año 2014.

_____

Eva Lianet Ferrer Ruano

_____ _____

Ing: Reidiel Castillo Arbelo Ing: Nadia Porro Lugo

Agradecimientos:

***A mis padres**, que me dieron la vida, y se han mantenido, transmitiéndome su inagotable energía y su dedicación para el desarrollo del saber, así como la enseñanza de hacer las cosas cada vez mejores. Y que sin su eterna paciencia para conmigo, no hubiera sido posible llegar a la culminación de esta investigación.*

***A mi tutor,** Reidiel Castillo Arbelo, que no escatimó un segundo de su preciado tiempo, para ayudarme, guiarme y transmitirme su extraordinaria experiencia profesional, dedicando muchas horas de su descanso a este trabajo.*

***A mi tutora**, Nadia Porro Lugo, quien en todo momento se encontró disponible para orientarme y dirigirme, sobre todo en los procederes metodológicos de la investigación, así como en la revisión de cada paso investigativo. Ha sido su apoyo, con su vasta experiencia profesional y sencillez incalculable un elemento de extremo valor en el desarrollo y culminación de este trabajo.*

***A mis hermanos**, que han sido un apoyo y una fortaleza en todo este tiempo recorrido lejos de casa.*

***A Yanet**, que ha sido un gran apoyo todo este tiempo transmitiéndome su infinito cariño y sabiduría.*

***A mi familia**, que me ayudaron de diferentes maneras en estos largos años.*

***A mis amigos**, que saben los llevo en mi corazón y no es posible mencionarlos uno a uno y no han escatimado el más mínimo esfuerzo para apoyarme en todos los sentidos, que se han mantenido a mi lado en los buenos y en los malos momentos, que han seguido paso a paso el desarrollo de este proceso y no han cesado de darme aliento e impulso para llegar a la culminación del mismo, llegue pues, mi eterno e infinito agradecimiento.*

Y, a todos aquellos, que saben que me han ayudado en este largo camino, muchas gracias.

Dedicatoria

A mis padres,

con los que he contraído un compromiso especial,

con su ejemplo han sido mi guía y empuje para todo en la vida,

y han logrado establecer un equilibrio entre el desenfado y la exigencia.

A mis hermanos,

con quienes tengo una armonía perfecta

y a quienes nunca me daría el lujo de decepcionar.

Para aquellos que están cerca y

van dejando su huella.

Y, para los que aún estando lejos,

están cerca, en el corazón.

Resumen

El proceso de migración hacia tecnologías libres, realizado actualmente en el país y encabezado por el departamento de Servicios Integrales en Migración, Asesoría y Soporte, requiere herramientas que permitan administrar los servicios telemáticos, por este motivo se está desarrollando la Herramienta para la Migración y Administración de los Servicios Telemáticos (HMAST), la cual aún no contaba con la capacidad de administrar el servicio SSH. Para darle solución a esto se planteó como objetivo de este trabajo desarrollar un módulo de administración del servicio SSH2 para HMAST que facilite su uso por parte de los administradores de sistema. Para su desarrollo se analizaron las herramientas que implementan y administran este servicio, seleccionándose OpenSSH como servidor a utilizar por la seguridad y funcionalidades que brinda y utilizándose como base para la definición de requisitos del módulo las herramientas Yast2 y Webmin. La arquitectura del módulo se desarrolló de acuerdo a las especificaciones que HMAST requiere para su integración. Se utilizó para desarrollar el módulo la metodología ágil SXP, el *framework* Spring, el entorno de desarrollo Netbeans, el lenguaje de programación Java; como herramienta CASE Visual Paradigm, para el trabajo con ficheros Augeas y la herramienta de diseño gráfico Péncil. Como resultado de esta investigación se implementó un módulo que dota a HMAST de funcionalidades para la administración de SSH como la gestión de autenticación, de las conexiones y el control de acceso.

Palabras claves: Administración de OpenSSH, Servicio SSH, HMAST, OpenSSH.

Índice

Índice de ilustraciones

Índice de tablas

Introducción

Cuba se encuentra a las puertas de un proceso de migración hacia el uso de aplicaciones de software libre y de código abierto en sustitución del software privativo. Como parte de la estrategia de informatización de la sociedad cubana, varias instituciones de los Organismos de la Administración Central del Estado (OACE) han iniciado el proceso de migración, entre estos se pueden citar la empresa Telemar del Ministerio de la Alimentación (MINAL) y el Centro de Cibernética Aplicada a la Medicina (CECAM). En este sentido, una de las instituciones que mayor impacto ha tenido en el apoyo de este proceso, es la Universidad de las Ciencias Informáticas (en lo adelante UCI), la cual desde sus inicios ha contribuido determinantemente en el desarrollo de software para la informatización del país.

La UCI centra su principal fuerza de desarrollo en sus centros productivos, los cuales son un elemento importante en el proceso de crecimiento de la universidad. Estos cuentan con varias líneas de desarrollo, tales como: telecomunicaciones, biomedicina, realidad virtual, software libre, entre otros. Precisamente para el desarrollo del software libre se crea el Centro de Software Libre (CESOL) que tiene como parte de su misión “conducir Procesos de Migración a Aplicaciones de Código Abierto”. El centro está compuesto por los Departamentos de Sistemas Operativos (SO) y Servicios Integrales en Migración, Asesoría y Soporte (SIMAYS). Este último especializado en brindar servicios relacionados con la migración a aplicaciones de código abierto como asesoría, consultoría, capacitación, personalización de sistemas operativos libres y asistencia técnica. Por la importancia estratégica para las organizaciones y como garantía de la seguridad de sus sistemas, lo primero que se migra en una institución son los servicios telemáticos.

Para dar cumplimiento al objetivo del centro, se desarrolla la Herramienta para la Migración y Administración de los Servicios Telemáticos (HMAST); una solución adaptada a las condiciones de las entidades cubanas. Esta herramienta se encuentra en desarrollo por los especialistas en migración de SIMAYS, mediante la implementación de módulos para la administración de los diferentes servicios telemáticos como son: Correo, FTP, DNS, Proxy, Apache y Firewall.

Sin embargo, no cuenta con la capacidad de administrar el servicio SSH en las conexiones seguras a servidores con GNU/Linux. Lo cual puede permitir la ocurrencia de fallas de seguridad en el envío de datos, provocando el acceso a la información por personas ajenas a la institución. Además, es necesario

el trabajo con la segunda versión del protocolo SSH, ya que la primera presenta un agujero de seguridad haciéndola potencialmente vulnerable a la inserción de datos en la corriente de comunicación mediante intrusos. Actualmente, la administración del servicio SSH se desarrolla con herramientas no adaptadas a las condiciones de las empresas cubanas resultando ser complejas para los administradores a cargo de la gestión.

Una vez planteada la situación problemática existente, se define como **problema científico** a resolver: ¿Cómo dotar a la herramienta HMAST de la capacidad de administrar el servicio SSH2 facilitando su uso por parte de los administradores del sistema?

El **objeto de estudio** de la investigación se centra en la administración del servicio SSH, enmarcándose el **campo de acción** en la administración del servidor SSH2 en la herramienta HMAST.

Una vez definido el problema científico se propone como **objetivo general** desarrollar un módulo de administración del servicio SSH2 para HMAST que facilite su uso por parte de los administradores del sistema.

Para un mejor desarrollo del presente trabajo, se desglosa el objetivo general en los siguientes **objetivos específicos**:

- Sistematizar en los software que implementan el protocolo SSH2.
- Analizar y diseñar la propuesta de solución.
- Implementar y probar el módulo definido.

Para contribuir al cumplimiento a los objetivos específicos, se planifican las siguientes **tareas de investigación**:

- Conceptualización de los elementos necesarios para comprender el protocolo de seguridad SSH2.
- Caracterización sobre los servidores que implementan el protocolo SSH2 y las herramientas que lo administran.
- Definición de las funcionalidades.

- Análisis de la arquitectura de HMAST.
- Diseño del sistema acoplado a la arquitectura de HMAST.
- Implementación del módulo para la administración del protocolo SSH2.
- Diseño y ejecución de pruebas para el módulo implementado.

En el presente trabajo de diploma se plantea como **idea a defender** que el desarrollo de un módulo de administración del servicio SSH2 para HMAST facilitará su uso por parte de los administradores del sistema.

La investigación que se realiza en el presente trabajo se respalda con el empleo de los métodos científicos siguientes:

Analítico-Sintético: Se utiliza en el estudio de diferentes fuentes bibliográficas para extraer los elementos más significativos del protocolo de seguridad SSH2, haciendo una valoración de los conceptos más relevantes asociados al protocolo SSH2 y de las herramientas que administran e implementan el mismo.

Análisis Histórico-Lógico: Se utiliza para determinar los antecedentes y la evolución del protocolo SSH2, así como de las herramientas que lo administran y servidores que lo implementan [1].

La estructura del documento de investigación consta de una introducción, tres capítulos, las conclusiones generales, la bibliografía, las referencias bibliográficas, el glosario de términos donde se explican los vocablos de difícil comprensión que se han empleado en la elaboración de este trabajo y los anexos. La estructura de los capítulos se define a continuación:

Capítulo 1. Fundamentación Teórica: Se plantea una base teórica para entender el problema planteado. Se analizan los conceptos fundamentales acerca del protocolo de seguridad SSH2, mostrándose el resultado del estudio a las herramientas que administran e implementan este protocolo. Además, se describen las tecnologías, lenguajes de programación y la metodología a emplear en la solución del problema.

Capítulo 2. Diseño de la Solución: Se modela la propuesta de solución, para determinar cómo se va a desarrollar esta y se especifican las funcionalidades a implementar. Las cuales son descritas a través de Historias de Usuario, prototipos de interfaz de usuario, entre otros artefactos. También se especifica la arquitectura del sistema y los patrones de diseño a emplear.

Capítulo 3. Implementación y Prueba de la solución propuesta: Se realiza la implementación del módulo propuesto tomando como ayuda las tareas de ingeniería generadas por la metodología SXP. Además, se especifican las clases principales del sistema, se diseña el plan de prueba y se realizan las pruebas necesarias para validar el cumplimiento de los requisitos establecidos por el cliente.

Capítulo 1: Fundamentación teórica

El presente capítulo se centra en realizar un estudio del protocolo SSH2, con el propósito de entender el funcionamiento del mismo. Además, se realiza un análisis de los software que brindan el servicio SSH en sistemas operativos GNU/Linux, quedando detalladas sus características, resaltando las ventajas y desventajas de cada uno de ellos, con el objetivo de seleccionar el idóneo para realizar la administración de dicho servicio. Se analizan también las herramientas para administrar el servicio SSH, con el fin de enriquecer la solución. Igualmente se describe el funcionamiento y las características del sistema al cual va a ser integrado el módulo concebido, detallándose aspectos como: lenguaje de programación, metodología, herramientas y las tecnologías que serán utilizadas para su desarrollo. Estas tecnologías se corresponden con las definidas en estudios iniciales referentes al desarrollo de HMAST, aspectos que se encuentran especificados en el documento Plantilla de Concepción del Sistema HMAST, recogido en el expediente de proyecto de dicho sistema.

1.1 Conceptos fundamentales

Internet es una red mundial de ordenadores la cual proporciona conexión con multitudes de redes informáticas [2]. El funcionamiento del Internet viene dado a través de los protocolos de transmisión. Para esto es necesario conocer los siguientes conceptos fundamentales:

Protocolo de red: es un conjunto de reglas y procedimientos que deben respetarse para el envío y la recepción de datos a través de una red. Existen diversos protocolos de acuerdo al modo en que se espera sea la comunicación. Algunos protocolos, por ejemplo, se especializarán en el intercambio de archivos (FTP); otros pueden utilizarse simplemente para administrar el estado de la transmisión y los errores (como es el caso de ICMP), también está el protocolo internet (IP), Protocolo de Control de Transmisión (TCP), y el protocolo de seguridad SSH, entre otros [3].

Protocolos de Comunicación: Un protocolo de comunicación es un conjunto de pautas que brindan la posibilidad de que distintos elementos que forman parte de un sistema establezcan comunicaciones entre sí, al intercambiar información [4].

Inicialmente se utilizaban en la conexión remota para el envío de información, protocolos como *Telecommunication Network* (*Telnet*), *Remote Login* (*rlogin*) y *Remote Shell* (*rsh*). Estos protocolos eran

utilizados para conectar a un equipo remoto a través de la red, enviándose información por un canal inseguro, el cual no obstante ofrecía cierto nivel de seguridad en las comunicaciones, aunque no el suficiente. Esto se evidenció a través de Telnet, que utilizaba un nombre de usuario y una contraseña, sin embargo el envío de estos datos lo hacía en texto plano, por lo que era posible suplantar la identidad del cliente [5].

Para dar solución a esto surge el protocolo de seguridad SSH, ya que provee una conexión encriptada y comprimida más segura que sus homólogos anteriores.

Protocolo SSH

SSH fue creado en 1995 bajo una licencia *open-source* por el investigador finlandés Tatu Ylonen, quien en ese mismo año fundó la empresa *SSH Communications Security* [6]. Este protocolo está conformado por una arquitectura cliente/servidor, la cual provee la habilidad de que los clientes deben tener una sola fuente de autenticación y autorización: el servidor, que permite el acceso al servicio SSH. Además, de poder ser utilizado en los diferentes sistemas operativos, proporciona los siguientes tipos de protección:

- Después de la conexión inicial, el cliente puede verificar que se está conectando al mismo servidor al que se conectó anteriormente.
- El cliente transmite su información de autenticación al servidor usando una encriptación robusta.
- Todos los datos enviados y recibidos durante la sesión se transfieren por medio de encriptación, lo cual los hace extremamente difíciles de descifrar y leer.
- El cliente tiene la posibilidad usar aplicaciones gráficas sobre una red [7].

En la actualidad existen dos versiones del protocolo, la primera versión de SSH (en lo adelante SSH1) y la segunda versión (en lo adelante SSH2). SSH1 evita ciertos fallos de seguridad, al proporcionar diferentes métodos de autenticación como son *.rhost* junto con RSA, o utilizando solamente RSA. Además, cifra las comunicaciones de manera automática y transparente, mediante los siguientes mecanismos de encriptación DES, 3DES, IDEA, Blowfish; con lo cual protege la integridad de la información. Pero una de las desventajas que presenta es la misma autenticación por *.rhost*, ya que se especifican las máquinas y usuarios que pueden acceder al servidor sin validación de contraseña.

También es vulnerable a un agujero de seguridad que potencialmente permite a un intruso insertar datos en la corriente de comunicación.

SSH2 tiene un algoritmo de intercambio de llaves mejorado que no es vulnerable al hueco de seguridad de la versión 1; sin embargo, soporta las conexiones de la primera versión. Consta de tres componentes:

- Capa de protocolo de transporte: proporciona autenticación del servidor, confidencialidad e integridad. Generalmente se ejecutará sobre una conexión TCP/IP.
- Capa de protocolo de autenticación: autentifica al cliente contra el servidor. Se ejecuta sobre la capa del protocolo de transporte.
- Capa de protocolo de conexión: multiplexa[1] el canal cifrado en múltiples canales lógicos. Se ejecuta sobre la capa de aplicación.

El protocolo de conexión proporciona canales que pueden utilizarse para diferentes propósitos como abrir sesiones interactivas (X11) y redireccionar puertos TCP/IP a través de túneles. Para resolver problemas de seguridad, SSH2 protege contra ataques como: engaños de direcciones de IP, modificación de datos, secuestro de sesión, datos en texto claro; además de muchos otros ataques, los cuales pueden ser directos e indirectos. Para hacer esto el mismo puede cifrar toda la información utilizando alguno de los siguientes algoritmos: 3DES, Blowfish, Twofish y AES (128, 192 y 256), Arc Four, CAST, DES, RC4; la autenticación puede requerir usuario/contraseña o llave pública, además de que estas dos opciones pueden ser utilizadas juntas. Para garantizar la integridad, el protocolo SSH2 crea una firma digital de los datos transferidos utilizando los algoritmos MD5 y SHA1, asegurando que no ocurran modificaciones de ningún tipo [8].

Las diferencias entre estas dos versiones son significativas, ya que SSH2 fue reescrito desde cero, brindando mayor seguridad, funcionalidad y flexibilidad que SSH1. Por lo tanto, para el desarrollo de este trabajo se analizará el uso y la optimización de SSH2. Existen diferentes clientes para conectarse a un servidor por SSH, pero en Linux es suficiente con abrir una terminal y escribir el usuario, el IP, y el puerto por el que se va a conectar. Además, el usuario puede modificar todas las opciones que desee en el fichero de configuración.

[1] Multiplexar: es la combinación de dos o más canales de información en un solo medio de transmisión.

1.2 Servidores SSH

A continuación se describen los servidores que implementan el protocolo *Secure Shell*, detallándose las principales funcionalidades, ventajas y desventajas de cada uno. Para la investigación se seleccionaron los servidores que se ejecutan en sistemas operativos GNU/Linux por ser esta la plataforma donde se va a instalar el sistema HMAST. Los servidores que cumplen con estas características son: Dropbear, OpenSSH, Tectia SSH Server.

Dropbear

Dropbear es una aplicación ligera de un cliente y el servidor SSH. Se ejecuta en una variedad de plataformas basadas en POSIX[2]. Es un software de código abierto, comercializado bajo una licencia estilo *Massachusetts Institute of Technology* (MIT) o licencia X11[3] [9].

Dropbear se usa particularmente en sistemas operativos de Linux y Unix. Utiliza la segunda versión del protocolo SSH, permite la separación de privilegios, utiliza *Secure Copy* (SCP) para el envío de datos, soporta IPv6, y trabaja con la autorización de llaves de OpenSSH. Utiliza SSH y TCP para hacer un túnel a través de múltiples *hosts*.

Se caracteriza por:

- Utilizar el reenvío de X11.
- Poder ejecutarse desde *inetd*[4] o independiente.
- Ser compatible con OpenSSH al utilizar autenticación por llave pública.
- Emplear poca memoria siendo adecuado para entornos con limitaciones de memoria.
- Poder desactivar las características de ser necesario para ahorrar espacio.
- Utilizar algoritmos de encriptación como rsa y dsa, para la generación de llaves.

[2]Posix: Las plataformas posix generalizan las interfaces de los sistemas operativos para que una misma aplicación pueda ejecutarse en distintas plataformas.

[3] MIT: Muestra el software de manera gráfica, lo cual permite reutilizarlo.
[4]Inetd: Es un demonio que gestiona conexiones de varios demonios.

Muestra grandes ventajas ya que consume menos memoria que OpenSSH, al iniciar la sesión como usuario *root* genera un solo proceso, mientras que OpenSSH genera dos. Pero también tiene desventajas como un menor número de opciones de configuración y características. Además, está implementado mediante librerías de terceros, incluyendo su propia distribución [10].

Tectia SSH Server

Tectia SSH Server es una solución de clase empresarial para asegurar la administración del sistema, la transferencia de archivos y la conectividad de las aplicaciones en las redes empresariales.

Se caracteriza por:

- Sustituir FTP, *Telnet* y *rlogin* por tecnología de clase empresarial segura.
- Permitir transferencia de archivos.
- Poseer una arquitectura mínimamente invasiva, al ofrecer el tránsito de datos seguro sin modificaciones a las aplicaciones o la infraestructura.
- Trabajar con *Tunneling* TCP.
- Presentar una interfaz intuitiva para operaciones interactivas de transferencia y de fácil configuración.
- Ofrecer adaptabilidad.
- Generar las llaves durante la instalación.

Pero una de sus desventajas es que es un software de gestión centralizada, el cual posee una licencia EULA [11].

OpenSSH

Open Secure Shell (OpenSSH) es un conjunto de aplicaciones que permite realizar comunicaciones a través de una red utilizando el protocolo SSH. Fue creado por el equipo de *OpenBSD* como una alternativa libre al software propietario *Secure Shell* de Tatu Ylönen. Una de las ventajas que plasma este

software es una mayor seguridad que el original, lo cual se debe a la reputación de los desarrolladores de crear código limpio y perfectamente auditado.

Su seguridad también se le atribuye al hecho de que su código fuente se distribuya libremente utilizando la licencia BSD[5]. *OpenSSH* apareció por primera vez en el Sistema Operativo *OpenBSD* 2.6 y la primera versión portable se hizo en octubre de 1999. Este servidor incluye protocolos como ambas versiones de SSH, SCP, SFTP; y algunas herramientas como:

- *SSH-Keygen:* herramienta usada para generar claves y autenticar el usuario que establece la conexión mejorando la seguridad.
- *SSH-Agent o SSH-Add:* herramienta utilizada para autenticar de forma más sencilla, sin necesidad de estar escribiendo continuamente la contraseña utilizada como clave.
- *SSH-Keyscan:* herramienta para escanear una lista de clientes y recopila sus llaves públicas.

OpenSSH puede autenticar a los usuarios mediante diferentes métodos, los cuales pueden ser autenticación con contraseña, con llave pública, teclado interactivo, el cual es un mecanismo de pregunta-respuesta, aunque también puede hacer uso de otros más fuertes tales como: tokens[6], Kerberos o GSSAPI.

Igualmente cuenta con la autenticación basada en *host*, que es una versión segura de las relaciones de confianza utilizadas en *rlogin*, lo que puede incluir el uso del sistema de autenticación BSD (*auth bsd*) o librerías PAM. Sin embargo, en ocasiones el uso de estas librerías tienen efectos secundarios ya que tienen que ser utilizadas con privilegios de *root,* lo cual podría comprometer la seguridad. Aunque las versiones de *OpenSSH* después de la 3.7 del 16 de septiembre de 2003 permiten que las librerías PAM puedan ser desactivadas mientras las mismas se están ejecutando. Estas librerías pueden utilizar métodos que permiten realizar la autenticación por contraseña una sola vez, los más conocidos son: *OTPW*, *S/KEY*, *OPIE*.

[5]BSD: Es una licencia de software libre que permite el uso de código de fuente en software libre.

[6] Token o Componente Léxico: Es una cadena de caracteres que tiene un significado coherente en cierto lenguaje de programación.

En el sistema operativo *OpenBSD*, se utiliza *systrace*[7] para *sandboxing*[8] pero al igual que la mayoría de los servicios nativos de *OpenBSD*, *OpenSSH* también utiliza un usuario dedicado al demonio, conocido como (*sshd)* el cual lleva a cabo la política de seguridad de *OpenBSD* de mínimos privilegios [8].

1.2.1 Selección del servidor

A continuación se recogen en una tabla comparativa las características que justifican la selección del servidor más eficiente para el desarrollo de la solución propuesta.

Tabla 1: Comparación entre servidores

Características	Dropbear	OpenSSH	Tectia SSH
Licencias Libres	X	X	
Autenticación por Contraseña	X	X	X
Autenticación por llave pública	X	X	X
Autenticación basada en *host*		X	
Autenticación por *Kerberos* o *GSSAPI*		X	
Separación de Privilegios	X	X	
Multiplataforma	X	X	X
Transferencia segura de archivos	X	X	X
Exportar aplicaciones gráficas	X	X	
Reenvío de puertos	X	X	

La tabla comparativa muestra que los servidores Dropbear y OpenSSH abarcan la mayoría de los elementos fundamentales que se toman en cuenta para seleccionar el servidor correcto a utilizar en el desarrollo del módulo. Con el objetivo principal de garantizar la seguridad en las conexiones se elige el servidor OpenSSH, por permitir diferentes vías de autenticación y la posibilidad de restringir los usuarios que se conectan a través de este servicio.

[7] Systrace: Es una herramienta que analiza el rendimiento de las aplicaciones a través de la captura y visualización de los tiempos de ejecución de los procesos.

[8] Sandboxing: Mecanismo de seguridad para aislar aplicaciones o programas del resto del sistema.

1.3 Herramientas de Administración de SSH

Actualmente existen aplicaciones que permiten la administración de servicios telemáticos. Siguiendo el hilo de la investigación es necesario centrar el estudio en cómo estas herramientas administran el servicio SSH, con el fin de acatar aquellas características que puedan ser útiles para el desarrollo del módulo en cuestión y desechar los aspectos negativos. Además, de ser necesario estudiar si algunos de sus módulos pueden ser integrados a HMAST. En la concepción de HMAST se realizó un análisis de las principales herramientas de este tipo como son Yast2 y Webmin, partiendo de dicho estudio se describen a continuación estas herramientas, haciendo énfasis principalmente en el proceso de administración del servicio SSH.

1.3.1 Módulo SSH en Webmin

Esta herramienta permite la administración del servicio SSH a través de los parámetros del fichero de configuración. Para esto, la página principal mostrará 6 secciones, donde cada una es un formulario que contiene los campos con los parámetros a configurar en cada categoría, como la autenticación, red, control de acceso, configuraciones de llave, opciones para el cliente, y el fichero de configuración; que permite al administrador configurar tanto el servidor como el cliente. Cuenta con todos los parámetros de configuración necesarios para garantizar mayor seguridad en las conexiones a los usuarios del sistema.

Una de las principales desventajas que presenta es que la ayuda para el módulo SSH no está muy explícita, ya que no explican ningún parámetro del fichero de configuración, ni siquiera los que son muy usados, esto trae como consecuencia que no se tenga un buen entendimiento de los mismos. Otro de los problemas que presenta es que se sobrescribe el fichero de configuración al realizarse algún cambio en la interfaz gráfica. Además, al estar desarrollado en el lenguaje Perl, no puede ser integrado a la herramienta HMAST.

1.3.2 Módulo de SSH de Yast2

El módulo SSH no se activa como en herramientas similares, sino que tiene que ser instalado y desinstalado en caso de ser necesario. La instalación del módulo no significa que se instale el servicio, pues esa opción la brinda cuando se quiere configurar el mismo.

Para configurar el servidor SSH en la herramienta Yast2 se selecciona la sección Configuración de Servicios de Red utilizándose *sshd*.

Esta herramienta permite:

- Utilizar varios puertos de escucha.
- Enviar datos a través de TCP.
- Levantar la interfaz gráfica.
- Utilizar la autenticación de llave pública con cifrado RSA.
- Enumerar los métodos de cifrado con los que cuenta.
- Dar la posibilidad al usuario de decidir qué versión del protocolo SSH quiere usar.

Presenta problemas cuando se van a configurar los parámetros en el fichero, ya que estos se sobrescriben cuando se realiza algún cambio. Además, es una herramienta de escritorio lo que trae como consecuencia que no pueda ser integrada a HMAST. Aunque el mayor problema que presenta es que las últimas versiones de Yast2 ya no cuentan con la posibilidad de instalar el servicio SSH.

El análisis de Yast2 y Webmin fue de gran ayuda para definir los requisitos funcionales que no pueden faltar en el nuevo módulo a implementar para la herramienta HMAST. Esto permite llegar a la conclusión de que ninguno de los módulos SSH con que cuenta estas herramientas puede ser integrado a HMAST.

1.4 Descripción de HMAST

Para poder integrar la solución al software HMAST, permitiendo así la administración del servicio SSH2 es necesario conocer la arquitectura y funcionalidades de esta herramienta. Además, se debe tener en cuenta las características que debe tener un módulo de HMAST.

1.4.1 Arquitectura

En la herramienta HMAST se propone el diseño de una arquitectura N-Capas orientada al dominio, distribuida con cinco paquetes. Los cuales interactúan a través de interfaces y utilizando inyección de dependencias[9] [12].

La capa Presentación se encarga de presentar a través de una interfaz de usuario (IU) los conceptos de negocio, facilita la utilización de dichos procesos, además de informar sobre la situación de los procesos de negocio e implementar las reglas de validación de dicha interfaz. Para la implementación de esta capa es utilizado el módulo MVC de *Spring*.

La capa Aplicación tiene asignada la responsabilidad de hacer las llamadas a los servicios de las capas inferiores (Dominio). Los servicios que publica (servicios de aplicación) son los responsables de adaptar la información que le llega a los requisitos de los servicios de dominio. En esta capa también se ubican operaciones de trazas, seguridad, envío de correos electrónicos, cuando no forman parte estricta del negocio.

La capa Dominio es la encargada de implementar la lógica de dominio (reglas de negocio), siendo responsable de las validaciones. Define las interfaces de persistencia a datos (contratos de repositorio) pero no las implementa. Además, constituye el hilo conductor de la aplicación, sus componentes solo dependen de la Capa de Infraestructura Transversal, pero no están ligados a tecnologías específicas.

La capa Persistencia contiene el código necesario para persistir los datos. Sus principales componentes son los repositorios, los cuales son clases que implementan los contratos de repositorios definidos en la capa de Dominio.

La capa Infraestructura Transversal: tiene el objetivo de promover la reutilización de código, contiene las operaciones de seguridad, *logging*, monitoreo del sistema, mecanismos de persistencia reutilizables, validadores genéricos y las operaciones que se puedan llamar desde otras capas.

[9] Inyección de dependencia: es un patrón de diseño orientado a objetos, en el que se suministran objetos a una clase en lugar de crearlos ella misma.

1.4.2 Funcionalidades que ofrece

- Gestión de servidores lógicos: Permite la adición, edición y eliminación de los datos de un servidor lógico. Además, posibilita la conexión remota y desconexión a un servidor seleccionado.
- Gestión de servicios telemáticos asociados a un servidor lógico: Permite la adición, edición y eliminación de los datos de un módulo, así como activación y desactivación de los mismos.
- Gestión de las variables de configuración asociadas a un servidor lógico: Permite cargar y salvar las variables de configuración de los servicios telemáticos en un servidor lógico determinado (ficheros de configuración, nombre de módulos, demonios).

1.4.3 Consideraciones para implementar un módulo para HMAST

Para implementar un módulo que se desee integrar en HMAST es necesario conocer los siguientes aspectos [13]:

- La lógica de Aplicación no deberá incluir ninguna lógica del dominio, solo tareas de coordinación relativas a requerimientos técnicos de la aplicación, como conversiones de formatos de datos de entrada a entidades del Dominio, llamadas a componentes de Infraestructura para que realicen tareas complementarias.
- Se garantizará que no se envíen hacia y desde la capa de Presentación objetos de Dominio, en su lugar deben viajar Objetos de Transferencia de Datos (*DTO, Data Object Transfer*).
- Las clases de servicios deben ser las únicas responsables (vías de acceso) de acceder a los repositorios; no se puede implementar código de persistencia a datos en la capa de Dominio.
- Solo se puede acceder a la información almacenada en los servidores haciendo uso de los repositorios.
- Es necesario que todo el código reutilizable por más de un repositorio se ponga a disposición de todos en la capa de Infraestructura Transversal.

1.5 Lenguajes de programación

Un lenguaje de programación es una técnica estándar de comunicación diseñada para expresar procesos que puedan ser ejecutados en una computadora. Está conformado por un conjunto de reglas sintácticas y semánticas que definen un programa informático. Un lenguaje de programación permite a un programador especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados y transmitidos, además de qué acciones debe tomar bajo una variada gama de circunstancias [14].

1.5.1 Java

Dentro de los lenguajes de programación más conocidos se encuentra Java el cual se relaciona con C++ influenciando así muchas de las características orientadas a objetos que presenta. Este lenguaje carece de punteros y mediante su biblioteca de clases estándar proporciona todas las estructuras contenedoras clásicas. Proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores. Soportando la sincronización de múltiples hilos de ejecución a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Tiene cuatro niveles de empaquetamiento: variables y funciones, al igual que los lenguajes anteriores y otros dos propios de él, denominados: clases y paquetes [15].

1.6 Marco 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 fácil de mantener [16].

Por último, un *framework* facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. El marco de trabajo definido para el desarrollo de la aplicación fue:

1.6.1 Spring

El *Framework* de *Spring* proporciona un modelo integral de programación y configuración para aplicaciones modernas empresariales basadas en Java, para cualquier tipo de plataforma de despliegue.

Está compuesto por aplicaciones web de tipo Modelo Vista Controlador (MVC) y el marco de servicios web *Restful*[10].

Es un *framework* de código abierto utilizado para el desarrollo de aplicaciones para la plataforma Java, aunque también existe una versión para la plataforma .NET. Ofrece mucha libertad a los desarrolladores y soluciones muy bien documentadas. Sus características principales son las inyecciones de dependencias y la programación orientada a aspectos.

La inyección de dependencia tiene como objetivo lograr un bajo acoplamiento entre los objetos de la aplicación. Con este patrón de diseño, los objetos no crean o buscan sus dependencias sino que estas son dadas al objeto. El contenedor (*Context*) es el encargado de realizar este trabajo al momento de instanciar el objeto. Se invierte la responsabilidad en cuanto a la manera en que un objeto obtiene la referencia a otro. De esta manera los objetos conocen sus dependencias por su interfaz. Así la dependencia puede ser intercambiada por distintas implementaciones a través del contenedor.

La otra característica relevante es la Programación Orientada a Aspectos (AOP), que es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos. Gracias a la AOP se pueden capturar los diferentes conceptos que componen una aplicación en entidades bien definidas, de manera apropiada en cada uno de los casos y eliminando las dependencias entre cada uno de los módulos. De esta forma se consigue razonar mejor sobre los conceptos, se elimina la dispersión del código y las implementaciones resultan más comprensibles, adaptables y reusables [17].

1.7 Metodología de desarrollo

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar determinado software.

Principalmente se encarga de elaborar estrategias de desarrollo de software que promuevan prácticas adaptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente.

[10]Restful: Es un estilo arquitectónico que cuenta con un conjunto de restricciones aplicadas a los diferentes componentes.

1.7.1 SXP

Es una metodología ágil caracterizada por tener las buenas prácticas de las metodologías SCRUM y XP. Con el uso de SCRUM para la planificación de los proyectos, SXP logra una eficiente gestión de proyectos de software, y con la toma de las mejores prácticas de XP proporciona una mejor calidad en el producto a entregar.

SXP consta de cuatro fases principales: definición, desarrollo, entrega y por último mantenimiento; en cada una de estas, se realizan actividades que generan artefactos para documentar todo el proceso de desarrollo. Esta metodología ayuda a que trabajen todos juntos en equipo, en la misma dirección y con claridad en los objetivos, además de ayudar al líder del proyecto a tener un mejor control sobre el progreso del trabajo.

Dicha metodología será usada como guía para realizar la modelación del módulo a implementar por las ventajas anteriormente mencionadas, además de ser la utilizada en el proyecto SIMAYS [18].

1.8 Entornos de Desarrollo Utilizado (IDE)

Un Entorno de Desarrollo Integrado (IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Puede utilizarse en un único lenguaje de programación o para varios de estos. Además, existen entornos de desarrollo en los que se pueden utilizar varios lenguajes de programación. Un IDE consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Permite utilizar el lenguaje de programación de forma interactiva sin necesidad de trabajarlos con archivos de texto [19].

1.8.1 Netbeans

Netbeans es un Entorno de Desarrollo Integrado (IDE) gratuito, de código abierto, desarrollado principalmente para el lenguaje de programación Java. Dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis, entre otros. Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida como son: el soporte de Java o soporte para el sistema de control de versiones, soporte para JDK8, JDK7, Java EE 7 e incluye mejoras en HTML5 y JavaFX2. Contiene todos los módulos necesarios para el desarrollo de

aplicaciones Java en una sola descarga, aunque permite crear aplicaciones con Python, identificar errores y depurarlos [20].

1.9 Herramienta CASE para el modelado

Las herramientas CASE son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, documentación o detección de errores, entre otras [21]. A continuación se describe la herramienta a utilizar en el desarrollo de la presente investigación.

1.9.1 Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos [22].

1.10 Otras tecnologías

Además de las herramientas anteriormente mencionadas se utiliza RapidSVN, para mantener el control de versiones de ficheros, Augeas para el trabajo con ficheros y Péncil para el diseño de prototipos de interfaz gráfica.

1.10.1 RapidSVN

RapidSVN es un cliente de interfaz gráfica, distribuida bajo la licencia GPL, para la comunicación con servidores *Subversion*[11]. Provee el mantenimiento de las diferentes versiones de ficheros, a través de una

[11]Subversion: es una herramienta de control de versiones basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros.

interfaz sencilla e intuitiva, y se encuentra disponible para plataformas Windows, Linux, MAC OS X y Solaris [23].

1.10.2 Augeas

Augeas es una herramienta de edición de configuración, analiza los archivos de configuración en sus formatos nativos y los transforma en un árbol. Los cambios de configuración se hacen manipulando este árbol y guardándolo de nuevo en los archivos de configuración de origen. Se utiliza para manipular los ficheros de configuración de forma segura [24].

1.10.3 Péncil

Péncil es una herramienta gratuita y de código abierto, utilizada para diseñar gráficos y animarlos en dos dimensiones. Su interfaz gráfica dispone de los elementos básicos para crear dibujos y personalizarlos al estilo deseado. Se caracteriza por poseer soporte para dibujo de diagramas y exportar los dibujos a diferentes formatos de salida. Permite el vínculo entre páginas, debido a que los elementos de un dibujo se pueden vincular a una página específica en el mismo documento [25].

1.11 Conclusiones del Capítulo

En el desarrollo del presente capítulo fueron analizados los principales servidores SSH existentes en entornos de código abierto, seleccionando el servidor OpenSSH como más indicado para la administración del servicio en la herramienta HMAST. Además, el estudio realizado a las herramientas Yast y Webmin, las cuales ofrecen el servicio SSH a través del servidor seleccionado, permitió obtener aspectos positivos que contribuyeron en la concepción del módulo a desarrollar. La definición de las características que posee HMAST permitió definir los requisitos que debe tener un módulo para integrarse a la herramienta.

Capítulo 2: Propuesta del Diseño de la Solución

Una vez definido OpenSSH como servidor a administrar y detallados los requisitos que se necesitan en HMAST, en el presente capítulo se definirán las características y funcionalidades que tendrá el sistema, las tareas de ingenierías a realizar para darle cumplimiento a las mismas, así como la arquitectura y patrones de diseño a emplear.

2.1 Propuesta del Módulo

Se propone el desarrollo de un módulo para la administración del servicio SSH2, que permita realizar la administración del mismo en la herramienta HMAST, de forma que se asegure la facilidad de uso a los administradores del sistema. Como servidor a administrar se seleccionó OpenSSH, por tanto la propuesta será estructurada basándose en su funcionamiento. Además, brindará la posibilidad de realizar configuraciones para definir las diferentes formas de autenticación, con el objetivo de garantizar un adecuado nivel de seguridad.

2.2 Modelo de Dominio

La creación de un modelo del dominio posibilita identificar conceptos asociados con el entorno en que se desarrolla la solución y mostrar las relaciones entre ellos. El modelo de dominio de la *Ilustración 1* muestra las relaciones que existen entre los principales conceptos que componen el módulo de administración del servicio OpenSSH, destacándose los aspectos organizativos que deben ser tomados en cuenta para su desarrollo.

Este modelo está compuesto por el objeto HMAST el cual es el responsable de administrar servidores y contener un módulo que administre OpenSSH, el cual se encarga de implementar el protocolo SSH2. Para realizar la conexión con los servidores es necesario tener un *host* que haga función de cliente y se conecte a través del protocolo SSH2.

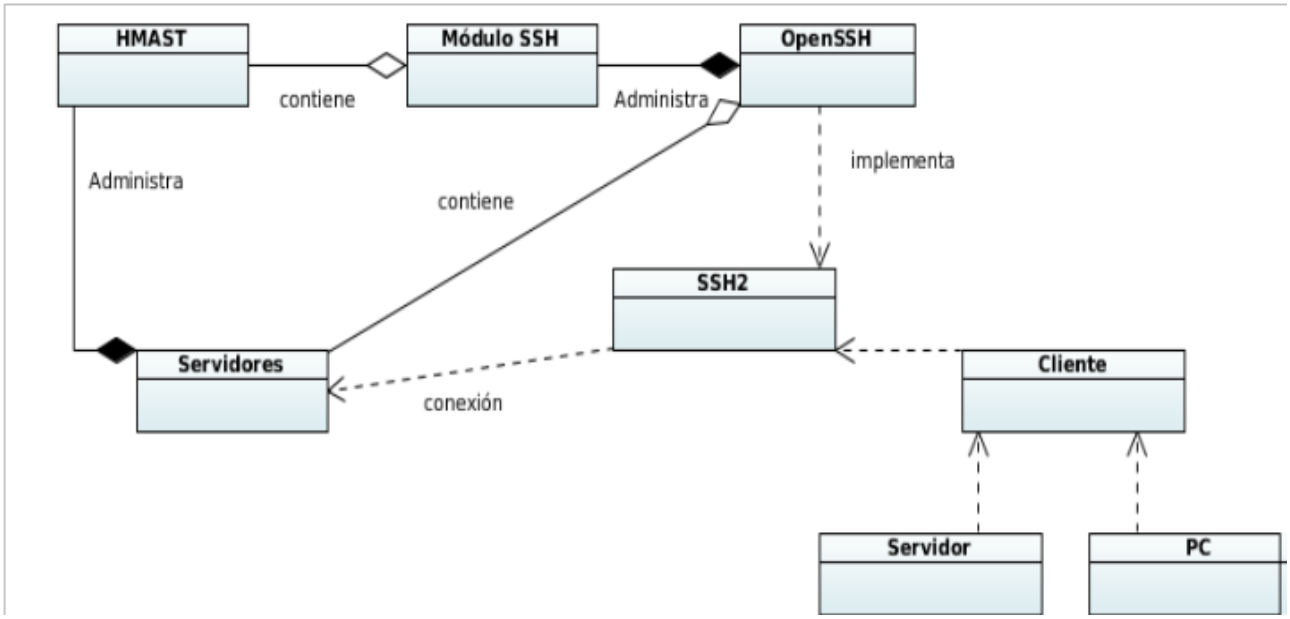


Ilustración 1: Modelo de dominio

2.2.1 Conceptos Asociados al modelo de dominio.

- **HMAST**: Herramienta para la Migración y Administración de los Servicios Telemáticos.
- **Módulo SSH**: permite a la herramienta HMAST administrar el servicio SSH.
- **OpenSSH**: servidor encargado de implementar el servicio SSH.
- **SSH**: es el protocolo de seguridad encargado de enviar y recibir información cifrada.
- **Servidores**: computadoras que forman parte de una red y que proveen servicios a otros ordenadores, que reciben el nombre de clientes.
- **Clientes**: se conectan a través de la red con el servidor y acceden a la información.
- **PC**: entorno de trabajo a través del cual se realizará la conexión.

2.3 Requisitos del Sistema

En el desarrollo del módulo guiado por la metodología SXP, se generaron artefactos como son: la Lista de Reserva del Producto, las Historias de Usuario y las Tareas de Ingeniería.

2.3.1 Lista de Reserva del Producto

La Lista de Reserva del Producto (LRP) contiene los requisitos agrupados y ordenados según su prioridad, definiendo así la planificación del trabajo a realizar. Durante el proceso de desarrollo de software (entre iteraciones) esta puede ser modificada y mejorada con nuevos requisitos, lo que garantiza una mejor calidad del producto final. A continuación se muestran los requisitos funcionales que quedaron definidos en la LRP.

Requisitos Funcionales.

Prioridad Alta

RF.1.Mostrar autenticación por contraseña.

RF.2.Modificar autenticación por contraseña.

RF.3.Mostrar autenticación por contraseña vacía.

RF.4.Modificar autenticación por contraseña vacía.

RF.5.Mostrar autenticación de *root*.

RF.6.Modificar autenticación de *root*.

RF.7.Mostrar autenticación RSA.

RF.8.Modificar autenticación RSA.

RF.9.Mostrar Modo Estricto.

RF.10.Modificar Modo Estricto.

RF.11.Mostrar autenticación de llave pública.

RF.12.Modificar autenticación de llave pública.

RF.13.Mostrar fichero de llaves públicas.

RF.14.Modificar fichero de llaves públicas.

RF.15.Mostrar las llaves públicas.

RF.16.Eliminar las llaves públicas.

RF.17.Generar llaves públicas y privada.

RF.18.Mostrar tiempo de autenticación.

RF.19.Modificar tiempo de autenticación.

RF.20.Mostrar usuarios del sistema.

RF.21.Mostrar usuarios que están denegados a conectarse por SSH.

RF.22.Denegar usuarios a conectarse por SSH.

RF.23.Mostrar grupos del sistema.

RF.24.Mostrar grupos que están denegados a conectarse por SSH.

RF.25.Denegar grupos del sistema.

RF.26.Mostrar dirección de escucha.

RF.27.Adicionar dirección de escucha.

RF.28.Eliminar dirección de escucha.

RF.29.Mostrar direcciones que están permitidas a conectarse por SSH.

RF.30.Adicionar direcciones que están permitidas conectarse por SSH.

RF.31.Eliminar direcciones permitidas a conectarse por SSH.

RF.32.Mostrar puertos.

RF.33.Modificar puertos.

Prioridad Media

RF.34.Mostrar exportación de las X11.

RF.35.Modificar exportación de las X11.

RF.36.Mostrar autenticación basada en *host*.

RF.37.Modificar autenticación basada en *host*.

RF.38.Mostrar usuarios que son ignorados en las conexiones por *host*.

RF.39.Modificar usuarios que son ignorados en las conexiones por *host*.

RF.40.Mostrar Autenticación Pregunta-Respuesta.

RF.41.Modificar Autenticación Pregunta-Respuesta.

RF.42.Mostrar usar *Login*.

RF.43.Modificar usar *Login*.

RF.44.Mostrar intervalo de regeneración de llave.

RF.45.Modificar intervalo de regeneración de llave.

RF.46.Mostrar bits del servidor.

RF.47.Modificar bits del servidor.

RF.48.Mostrar tipo de registro.

RF.49.Modificar tipo de registro.

RF.50.Mostrar nivel de registro.

RF.51.Modificar nivel de registro.

RF.52.Mostrar separación de privilegios.

RF.53.Modificar separación de privilegios.

RF.54.Mostrar cantidad de conexiones simultáneas.

RF.55.Modificar cantidad de conexiones simultáneas.

RF.56.Mostrar autenticación por *kerberos*.

RF.57.Modificar autenticación por *kerberos*.

RF.58.Mostrar *kerberos AFS*.

RF.59.Modificar *kerberos AFS*.

RF.60.Mostrar mecanismo local adicional de autenticación.

RF.61.Modificar mecanismo local adicional de autenticación.

RF.62.Mostrar limpieza de entradas.

RF.63.Modificar limpieza de entradas.

RF.64.Mostrar autenticación por *GSSAPI*.

RF.65.Modificar autenticación por *GSSAPI*.

RF.66.Mostrar limpieza de credenciales por *GSSAPI.*

RF.67.Modificar limpieza de credenciales por *GSSAPI*.

RF.68.Mostrar mensajes por inactividad.

RF.69.Modificar mensajes por inactividad.

RF.70.Mostrar dirección del *banner*.

RF.71.Modificar dirección del *banner*.

RF.72.Mostrar *banner.*

RF.73.Modificar *banner*.

RF.74.Mostrar el estado del servicio.

RF.75.Modificar el estado del servicio.

Requisitos No Funcionales

Diseño

RNF.1.Usar como lenguaje de programación Java y el *framework* Spring.

RNF.2.Utilizar Netbeans como entorno de desarrollo integrado.

RNF.3.Utilizar HTML5 para la interfaz de la aplicación.

Software

RNF.4.Disponer para el SO GNU/Linux del paquete *openssh-server*.

Interfaz Gráfica

RNF.5.Crear interfaz sencilla e intuitiva desde la cual se acceda a todas las funcionalidades del módulo.

2.3.2 Historias de Usuario

Las Historias de Usuario (HU) son la técnica utilizada en la metodología SXP para especificar las funcionalidades que serán añadidas al sistema, son asignadas al desarrollador encargado de la programación con un número de horas de desarrollo estimado. 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. A continuación se presentarán las HU definidas, para un mejor entendimiento del módulo a implementar.

Tabla 2: HU Estado del servicio

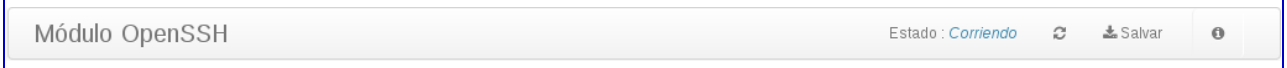
Historia de Usuario	
Número:HMAST_SSH_1	**Nombre Historia de Usuario:** Reiniciar el servicio SSH.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Eva Lianet Ferrer Ruano	**Iteración Asignada:** 1
Prioridad en Negocio: Media	**Puntos Estimados:** 0.4 semana
Riesgo en Desarrollo: Medio	**Puntos Reales:** 0.4 semana
Descripción: Muestra el estado y permite reiniciar el servicio SSH.	
Observaciones: Para reiniciar el servicio se utilizará el comando *service ssh restart***,** el cual reiniciará el sistema con las nuevas configuraciones determinadas por el usuario. Para conocer el estado del servicio se utilizará el comando service ssh status.	
Prototipo de interfaz: Estado del módulo.  Ilustración 2: PI Estado del Servicio	

Tabla 3: HU Gestionar Autenticación

Historia de Usuario	
Número: HMAST_SSH_2	**Nombre Historia de Usuario:** Gestionar autenticación de OpenSSH.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Eva Lianet Ferrer Ruano.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 4 semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 4 semanas
Descripción: Permite la configuración de los parámetros más importantes de autenticación del SSH.	
Observaciones: Todos los cambios que se apliquen se realizarán mediante la edición del fichero	

/etc/ssh/sshd_config.

Mostrar autenticación por contraseña: La aplicación mostrará un campo con la opción de autenticar por contraseña los túneles, la cual se mostrará habilitada o deshabilitada. El valor que se mostrará es el de la directiva *PassworAuthentication*.

Modificar autenticación por contraseña: Se permite habilitar o deshabilitar la autenticación por contraseña, la cual puede tomar valores de *yes* o *no*. Además, es muy importante habilitar esta opción para garantizar la seguridad en el servidor. Para permitir la autenticación por contraseña se mantendrá la directiva de la siguiente manera: *PassworAuthentication yes*.

Mostrar autenticación por contraseña vacía: La aplicación mostrará un campo con la opción de permitir contraseñas vacías, la cual se mostrará habilitada o deshabilitada. Se mostrará el valor de la directiva *PermitEmptyPasswords*.

Modificar autenticación por contraseña vacía: Puede tomar valores de *yes* o *no*, permitiendo al usuario conectarse sin tener que poner contraseña para autenticarse; por problemas de seguridad esta opción no es recomendada. Se le recomienda al usuario especificar esta opción de la siguiente manera *PermitEmptyPasswords no*.

Mostrar autenticación de root: La aplicación mostrará un campo con la opción de permitir a un usuario obtener los privilegios de administrador, la cual se mostrará habilitada o deshabilitada. Se mostrará el valor de la directiva *PermitRootLogin*.

Modificar la autenticación de root: Tomará valores de *yes* o *no*, aunque por seguridad esta opción no se recomienda ya que un usuario cualquiera podría obtener los permisos de administrador. Por esto se le recomienda al usuario configurar la directiva de la siguiente manera *PermitRootLogin no*.

Mostrar modo estricto: Especifica si debe revisar el directorio inicial antes de aceptar la conexión, la directiva de la siguiente manera *StrictModes*.

Modificar modo estricto: Esta opción tomará los valores de *yes* o *no,* inicialmente se encuentra de la siguiente manera *StrictModes* yes.

Mostrar autenticación por rsa: La aplicación le dará la posibilidad al usuario de generar un par de llaves a través del método RSA, la cuales podrán ser utilizadas para conectarse. Se mostrará el valor que tenga la directiva *RSAAuthentication*.

Modificar autenticación por rsa: Esta opción tomará los valores de *yes* o *no*, en caso de marcar *yes* se generarán inicialmente llaves utilizando el método RSA. La directiva que se modifica es *PubkeyAuthentication*.

Mostrar autenticación con llave pública: La aplicación le dará al usuario la posibilidad de conectarse utilizando la autenticación de llave pública y privada. Esta directiva se mostrará habilitada o deshabilitada. Se muestra el valor que tenga la directiva *PubkeyAuthentication*.

Modificar autenticación con llave pública: Esta opción tomará los valores de *yes* o *no*. La directiva que se modifica es *PubkeyAuthentication*.

Mostrar fichero de llaves públicas: La aplicación mostrará la dirección donde se almacenaran las llaves públicas del servidor, por defecto es %h/.ssh/authorized_keys. El valor que se muestra es el de la directiva *AuthorizedKeysFile*.

Modificar fichero de Llaves públicas: La aplicación le dará al usuario la posibilidad de guardar las llaves públicas de otros servidores en la carpeta que él desee, siempre que tenga permisos para acceder a los mismos. Se modifica el valor de la directiva *AuthorizedKeysFile*.

Mostrar las llaves públicas: La aplicación mostrará todas las llaves públicas generadas. Las llaves las mostrará del directorio establecido para su almacenaje. Por defecto es en el directorio home del usuario, por el cual está conectado. Por defecto es ~/.ssh/known_hosts.

Eliminar las llaves públicas: La aplicación permitirá eliminar las llaves que el usuario no desee mantener utilizando el comando sed -i 8d ~/.ssh/known_hosts, donde en **d** se especifica la línea en la que se encuentra la llave que se desea eliminar. Este comando tiene que ser ejecutado por el usuario que quiere eliminar las llaves. Se verifica que el número de la línea introducida es menos que la cantidad de líneas que tiene el fichero.

Generación de llave pública y privada: Para generar ambas llaves se utiliza el comando ssh-keygen -b 1024 -t dsa, donde la opción –b especifica el tamaño de la llave generada y –t el tipo de encriptación, el cual puede ser también (ECDSA o RSA). En caso de usarse RSA podrá tomar valores como: mínimo de 768 bits y por defecto de 2048. Si se utiliza ECDSA podrá tomar un tamaño de llave de 256, 384 o 521 bits y DSA 1024 específicamente, aunque puede tomar un tamaño de hasta 2048. Se le recomienda al usuario por problemas de seguridad, trabajar con el tipo de encriptación RSA, el cual es más seguro y no debe especificar el largo ya que por defecto genera llaves de un tamaño de 2048 bits. En caso de utilizar cualquier otro tipo de encriptación se recomienda poner un tamaño mayor o igual que 1024. Cuando la llave sea generada el programa preguntará por una contraseña, este valor se mostrará en un campo de texto y el usuario tendrá la posibilidad de dejar este parámetro en blanco o escribir un texto de una longitud de 10-30 caracteres. En caso de que luego quiera cambiar la contraseña para esa llave utilizará la opción –p. En el momento en el que las llaves se generen, el usuario tendrá la posibilidad de protegerlas modificando los permisos.

Mostrar tiempo de autenticación: La aplicación mostrará el tiempo que tendrá el usuario para autenticarse. Se mostrará el valor que tenga la directiva *LoginGraceTime*.

Modificar tiempo de autenticación: Para modificar la directiva se recomienda utilizar un tiempo pequeño por lo que se especificaría de la siguiente manera: *LoginGraceTime* 120. Este tiempo será en segundos y se mostrará el valor que tenga la directiva, el cual puede tomar valores reales, mayores o iguales que cero y menores que 2000000000.

Mostrar usuarios del sistema: La aplicación mostrará un listado con los usuarios del sistema. Los usuarios que se muestran tendrán por defecto habilitados los permisos para conectarse por SSH al servidor. Estos usuarios se mostrarán utilizando el comando *getent passwd*.

Mostrar usuarios que están denegados a conectarse por SSH: Se muestran los usuarios que están en la directiva *DenyUsers*.

Denegar usuarios a conectarse por SSH: Para denegar un usuario a conectarse por SSH, solo se deberá marcar y eliminar de la lista de usuarios que se muestra, así no tendrá permisos de conectarse al servidor por SSH, pero no será eliminado del sistema. Se escribirá en el fichero de configuración de la siguiente manera: *DenyUsers* Paloma.

Mostrar grupos del sistema: La aplicación mostrará un listado con los grupos que se encuentran en el sistema. Los grupos que se muestran con el comando *cat* /etc/group, los cuales tendrán por defecto habilitados los permisos para conectarse por SSH al servidor.

Mostrar grupos que están denegados a conectarse por SSH: Se muestran los grupos que están en la directiva *Denygroups*.

Denegar grupos del sistema: Para denegar un grupo a conectarse por SSH, solo se deberá marcar y eliminar de la lista de grupos que se muestran, el cual no tendrá permisos de conectarse al servidor por SSH, pero no será eliminado del sistema. Se escribirá en el fichero de configuración de la siguiente manera *DenyGroups* Paloma.

Para que las configuraciones tengan efecto se le informará al usuario que se debe reiniciar el servicio y se le dará la posibilidad de reiniciar.

Prototipo de interfaz: Se muestra el prototipo Autenticación General, los demás se encuentran en el Anexo A; ilustraciones 9, 10, 11 y 12.

Ilustración 3: PI Autenticación General

Tabla 4: HU Gestionar Conexiones

Historia de Usuario	
Número: HMAST_SSH_3	**Nombre Historia de Usuario:** Gestionar las conexiones de OpenSSH.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Eva Lianet Ferrer Ruano.	**Iteración Asignada:** 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 1.2 semana
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1.2 semana
Descripción: Permite la configuración de los parámetros de conexión del OpenSSH.	
Observaciones: Todos los cambios que se apliquen se realizarán mediante la edición del fichero **/etc/ssh/sshd_config.** **Mostrar direcciones de escucha**: Se muestran todas las direcciones de escucha a través de la directiva *ListenAddress*. **Adicionar dirección de escucha:** El sistema adicionará las nuevas direcciones utilizando la directiva *ListenAddress*, escribiendo las direcciones de la siguiente manera. *ListenAddress* 10.53.3.63 Según la cantidad de tarjetas de red, serán la cantidad de *listenAddres* que se podrán escribir. Además, de poder ser especificado el puerto por el cual va a escuchar esa subred siempre que se haya escrito en la directiva *Port*, por ejemplo: *ListenAddress* 10.53.3.71:3245 Para permitir todas las direcciones se debe poner la directiva con el valor *ListenAddress* 0.0.0.0. **Eliminar dirección de escucha:** Para eliminar una dirección hay que eliminar la directiva tiene el IP seleccionado. Si se quiere eliminar todas las direcciones se introduce la directiva con el siguiente valor: *ListenAddress* 255.255.255.255. **Mostrar direcciones que están permitidas a conectarse por SSH:** Se muestran los valores que tengan las directivas *sshd* del fichero de */etc/hosts.allow.* **Adicionar direcciones que están permitidas conectarse por SSH:** En caso de querer adicionar un host	

se debe editar el archivo */etc/hosts.allow* como usuario *root* y agregar el *host* al final:
sshd: IP, donde IP es el IP versión 4 de la máquina a la que se le va a dar permisos para conectarse mediante SSH.

Eliminar direcciones permitidas a conectarse por SSH: Para eliminar una dirección de la lista de direcciones permitidas hay que eliminarlo de la lista de *host* permitidos en el fichero de */etc/hosts.allow.*

Mostrar puertos: La aplicación le dará al usuario la posibilidad de especificar el puerto por el cual va a escuchar, esta directiva se mostrará con el puerto definido por defecto en este caso el 22, y permitirá al administrador cambiarlo. Se mostrará el valor que tenga la directiva *Port*.

Modificar puerto: Determina el puerto de escucha del servicio SSH el cual en la mayoría de los casos es el 22, por lo que se le recomienda al usuario poner uno diferente. Puede especificarse un número de puerto entre 0 y 65535, aunque debe validarse que no esté siendo usado por otro servicio utilizando el comando *netstat -tln*. El valor se modificará en la directiva *Port*, quedando la directiva seguida del puerto introducido.

Para que los cambios tengan efectos se le informará al usuario que debe reiniciar el servicio y se alertará que los cambios realizados pueden causar que se desconecte el servidor.

Prototipo de interfaz: Se muestra el prototipo Direcciones de Escucha, el otro se encuentran en el Anexo A; ilustración 11.

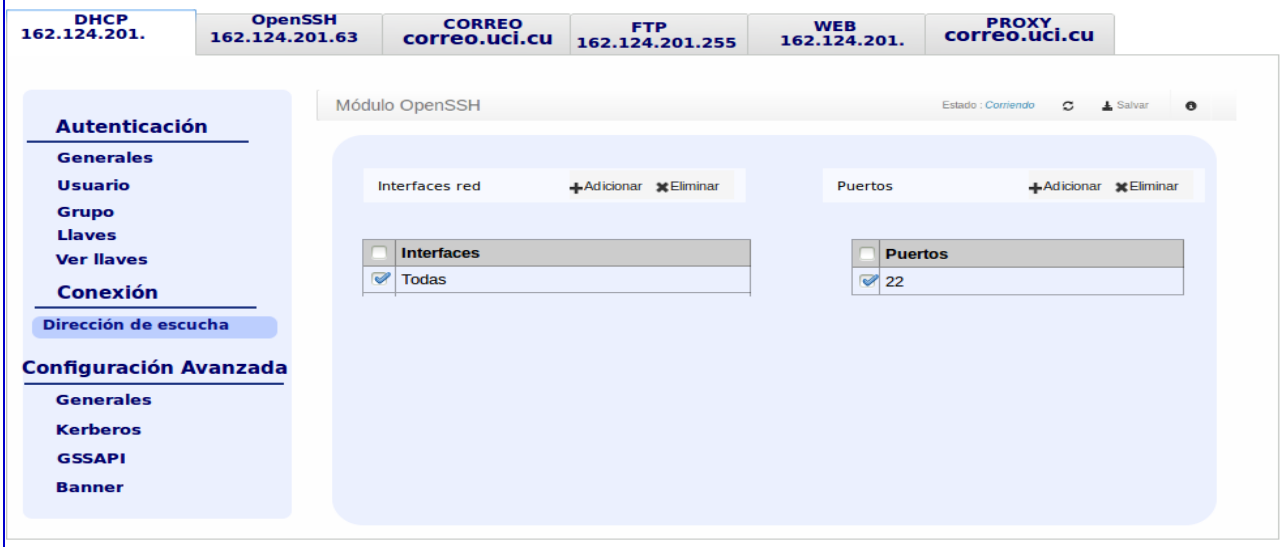


Ilustración 4: PI Direcciones de Escucha

Tabla 5: HU Gestionar Configuración Avanzada

Historia de Usuario	
Número:HMAST_SSH_4	**Nombre Historia de Usuario:** Gestionar las configuraciones avanzadas del OpenSSH.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Eva Lianet Ferrer Ruano.	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2.1 semanas
Riesgo en Desarrollo: Muy Alto	**Puntos Reales:** 2.1 semanas
Descripción: Permite realizar configuraciones que son menos usadas en un servidor SSH y que su modificación requiere de mayor conocimiento sobre SSH.	
Observaciones: Todos los cambios que se apliquen se realizarán mediante la edición del fichero **/etc/ssh/sshd_config.** **Mostrar exportación de X11**: La aplicación permitirá la opción de ejecutar las aplicaciones gráficas del servidor, esta directiva se mostrará habilitada o deshabilitada. Se muestra el valor de la directiva *X11Forwarding*. **Modificar la exportación de X11**: Se modifica la directiva *X11Forwarding*, la cual tomará valores de *yes* o *no*. **Mostrar la autenticación basada en host**: El fichero *host* es donde cada usuario puede tener en su directorio de *autenticación*, en él se especifican las máquinas y usuarios que pueden acceder al servidor sin validación de *password*. Es lo mismo que utilizaba la primera versión de SSH pero esta se establecía con el nombre de *IgnoreRhosts,* mientras que en su segunda versión la directiva cambia llamándose de la siguiente manera: *HostbasedAuthentication*. **Modificar la autenticación basada en host:** Se modifica la directiva *HostbasedAuthentication*, la cual podrá tomar valores *yes* o *no*, pero por problemas de seguridad se recomienda mantenerlo en *no*. **Mostrar usuarios que son ignorados en las conexiones por host:** En caso de habilitar host basado en autenticación, se habilitará la directiva *IgnoreUserKnownHosts*, se realizará utilizando las llaves públicas y privadas a través del método RSA, como se hacía en la versión anterior del protocolo.	

Modificar los usuarios que son ignorados en las conexiones por host: La directiva podrá tomar valores *yes* o *no*, pero por defecto se encuentra habilitada.

Mostrar Autenticación por Pregunta-Respuesta: Se encarga de preguntar y que respondan para que se establezca una conexión. Se muestra en la directiva con el nombre de *ChallengeResponseAuthentication*.

Modificar Autenticación por Pregunta-Respuesta: Esta directiva puede tomar valores de *yes* o *no*, pero por defecto se muestra de la siguiente manera: *ChallengeResponseAuthentication no*.

Mostrar usar login: Es utilizado para múltiples secciones de *login*; se muestra en la directiva con el nombre *UseLogin*.

Modificar usar login: Puede tomar valores de *yes* o *no*, pero por defecto se encuentra la directiva de la siguiente manera: *UseLogin no*.

Mostrar intervalo de regeneración de llave: Especifica cuántos segundos deberá esperar el servidor antes de regenerar su clave automáticamente, se muestra bajo la directiva *KeyRegenerationInterval*.

Modificar intervalo de regeneración de llave: Puede tomar valores de enteros desde cero hasta 2000000000, pero por defecto se encuentra 3600.

Mostrar bits del servidor: Indica cuántos bits hay que usar en la clave del servidor. Estos bits son usados cuando el demonio empieza a generar su clave RSA.

Modificar bits del servidor: Puede tomar cualquier valor inicialmente la directiva se encuentra de la siguiente manera: *ServerKeyBits* 768.

Mostrar tipo de registro: Indican el origen de un mensaje, pueden ser mensajes de los demonios del sistema, de procesos de usuario, del sistema de correo electrónico, entre otros. Por defecto el valor de la directiva es *SyslogFacility*.

Modificar tipo de registro: Puede tomar valores como *DAEMON*, *USER*, *AUTH*, *LOCAL0*, *LOCAL1*, *LOCAL2*, *LOCAL3*, *LOCAL4*, *LOCAL5*, *LOCAL6*, *LOCAL7*. Se muestra inicialmente de la siguiente manera:

SyslogFacility AUTH.

Mostrar el nivel de los logs: La aplicación permitirá definir el nivel de detalles con el que se registrarán los logs, mostrando el valor que tenga la directiva. Se mostrará el valor que tenga la directiva *LogLevel*.

Modificar el nivel de los logs: Se mostrarán los diferentes niveles de *logs*, donde el usuario podrá seleccionar cuál es el que se desea según los requisitos del servidor. Los valores posibles son: QUIET, FATAL, ERROR, *INFO*, *VERBOSE*, *DEBUG*, *DEBUG1*, *DEBUG2* y *DEBUG3*. Los valores predeterminados son *INFO*, (*DEBUG* y *DEBUG1*) los cuales son equivalentes; *DEBUG2* y *DEBUG3,* que especifican los niveles más altos de depuración. La directiva podrá tomar cualquiera de estos valores. La directiva que se modifica es *LogLevel.*

Mostrar separación de privilegios: La aplicación mostrará la opción habilitar la separación de privilegios de administración de los usuarios que se conecten a los servidores a través del SSH. Esta directiva se mostrará habilitada o deshabilitada. Se muestra el valor que tenga la directiva *UsePrivilegeSeparation***.**

Modificar separación de privilegios: Podrá tomar valores de *yes* o *no*. Se le recomienda al usuario mantener la directiva de la siguiente manera: *UsePrivilegeSeparation yes*, para evitar la elevación de privilegios.

Mostrar cantidad de conexiones simultáneas: La aplicación mostrará un campo que indicará cuántas conexiones se van a permitir simultáneamente en el servidor. Se muestra el valor que tenga la directiva *MaxStartups***.**

Modificar cantidad de conexiones simultáneas: La directiva inicialmente se mostrará de la siguiente manera *MaxStartups* 10 pero puede ser modificada según las características del servidor donde se vaya a administrar la aplicación.

Mostrar autenticación por *kerberos*: La aplicación mostrará un campo que indicará la autenticación por *kerberos*, la cual se mostrará a través de la directiva *KerberosAuthentication*.

Modificar autenticación por *kerberos*: Esta opción tomará valores de *yes* o *no*, y especifica si la contraseña proporcionada por el usuario para la autenticación por contraseña será válida a través del

kerberos KDC. Para utilizar esta opción el servidor necesita un *servtab Kerberos* que permite la verificación de la identidad del *KDC*. Inicialmente la directiva se mostrará de la siguiente manera *KerberosAuthentication no*.

Mostrar mecanismo local adicional de autenticación: La aplicación mostrará un campo, el cual permitirá validar a través de cualquier mecanismo local adicional si la autenticación de la contraseña a través de k*erberos* falla. Esta opción podrá estar habilitada o deshabilitada.

Modificar mecanismo local adicional de autenticación: Esta opción se ejecutará a través de la directiva *KerberosOrLocalPasswd,* la cual podrá tomar valores de *yes* o *no*.

Mostrar limpieza de entradas: La aplicación mostrará esta opción para especificar si se destruye automáticamente el archivo de las entradas del usuario al cerrar la sesión. Esta opción podrá estar habilitada o deshabilitada.

Modificar limpieza de entradas: Esta opción se ejecutará a través de la directiva *KerberosTicketCleanup* la cual podrá tomar valores de *yes* o *no*.

Mostrar Kerberos AFS: Cuando se usa Kerberos con el "*Andrew File System*" (AFS), OpenSSH podrá acceder al directorio *home* antes de que se haya completado la autentificación.

Modificar Kerberos AFS: Esta opción se ejecutará a través de la directiva *KerberosGetAFSToken* la cual podrá tomar valores de *yes* o *no*, inicialmente se encuentra en no.

Mostrar autenticación por GSSAPI: La aplicación mostrará esta opción para especificar si se permite la autenticación de usuarios basada en GSSAPI. Tenga en cuenta que esta opción sólo se aplica a la versión 2 del protocolo SSH. Esta opción podrá estar habilitada o deshabilitada.

Modificar autenticación por GSSAPI: Esta opción se ejecutará a través de la directiva *GSSAPIAuthentication* la cual podrá tomar valores de *yes* o *no*.

Mostrar limpieza de credenciales por GSSAPI: Especifica si se destruirán automáticamente las credenciales del usuario al cerrar la sesión. Esta opción podrá estar habilitada o deshabilitada.

Modificar limpieza de credenciales por GSSAPI: Esta opción se ejecutará a través de la directiva *GSSAPICleanupCredentials* la cual podrá tomar valores de *yes* o *no*.

Mostrar Mensajes por Inactividad: Se envía un mensaje por inactividad, cuando el servidor lleve un tiempo sin recibir respuesta, se mostrará con la directiva *TCPKeepAlive*.
Modificar Mensajes por Inactividad: Esta opción podrá tomar valores de *yes* o *no*, inicialmente se encuentra de la siguiente manera: *TCPKeepAlive yes*.

Mostrar dirección del banner: El sistema mostrará un campo con la dirección donde se encuentra el mensaje del banner. Se muestra la directiva con el nombre de *Banner*, el cual inicialmente está deshabilitado.
Modificar dirección del banner: El usuario podrá cambiar la dirección, dando *click* al botón de examinar. La directiva inicialmente se muestra de esta forma *Banner* /etc/issue.net

Mostrar banner: El banner tendrá por defecto un mensaje definido por el servidor. Se muestra el texto que esté dentro de la dirección mencionada anteriormente.

Modificar banner: El usuario podrá cambiar el mensaje del *banner* en el momento que desee. Las modificaciones se realizan en el fichero /etc/issue.net o en el que se defina en la directiva antes mencionada
.
Luego de haber realizado todas las configuraciones necesarias se deberá reiniciar el servicio con el comando /etc/init.d/ssh restart.

Prototipo de interfaz Se muestra el prototipo Configuraciones Avanzadas Generales, los demás se encuentran en el Anexo A; ilustraciones 14, 15 y 16.

Ilustración 5: PI Configuraciones Avanzadas Generales

2.4 Arquitectura del módulo

El proceso de diseño de la arquitectura tiene que decidir qué funcionalidades son las más importantes a desarrollar. A partir de esta decisión se selecciona el tipo de aplicación y el estilo arquitectural, y se toman las decisiones importantes sobre seguridad y rendimiento que afectan al conjunto del sistema. El diseño de la arquitectura decide cuáles son los componentes más básicos del sistema y cómo se relacionan entre ellos para implementar la funcionalidad [26].

Con el objetivo de mantener una homogeneidad entre los componentes del sistema, se define la utilización de la arquitectura establecida para HMAST ver (*Ilustración 2*), incluyendo en esta la distribución del módulo para la administración del servicio OpenSSH con sus responsabilidades específicas. Además, se representa el flujo de información entre las diferentes capas del módulo, indicado por las flechas representadas.

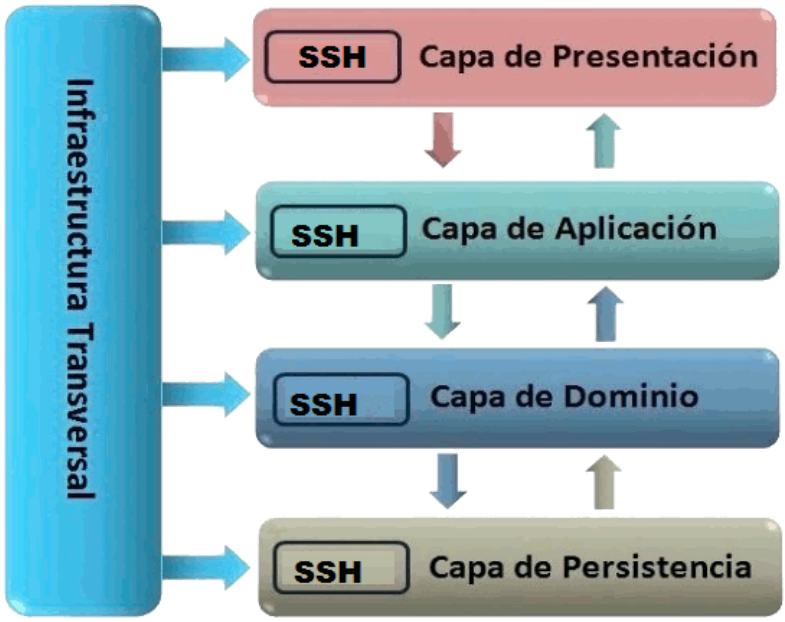


Ilustración 6: Arquitectura

2.5 Diagrama de Paquetes

Para la representación del módulo basada en la organización de paquetes y sus elementos, se diseñó el diagrama de paquetes tomando como marco de referencia la arquitectura propuesta. El diagrama de paquetes muestra la forma en que un sistema está dividido en agrupaciones lógicas describiendo las dependencias entre las mismas. Los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema.

Existe un paquete representando cada capa y dentro de este se encuentra el paquete *openssh* que contiene todo lo referente al módulo. Las funcionalidades de todo el módulo están dentro del paquete *persistens* en *repository*.

El diagrama de la *Ilustración 7: Diagrama de paquetes*, representa la distribución de los paquetes y las relaciones entre ellos. Dicho diagrama está conformado por diferentes paquetes, entre ellos se encuentra el de Presentación (*presentation*). Este es el responsable de presentar al usuario los conceptos de

negocio mediante una interfaz de usuario (IU), que facilite la explotación de dichos procesos e informe sobre la situación de los procesos de negocio a través de las reglas de validación.

Otro paquete es el de Aplicación (*application*) el cual permite la comunicación con el de presentación. Aplicación contiene el paquete DTO (*Data Object Transfer*), el cual almacena clases que representan la información que llega desde el paquete de presentación.

El paquete de Dominio (*domain*) contiene los subpaquetes Entidades (*entities*), Servicio (*service*) y Contratos de Repositorios (*repositoryscontrats*). En Entidades se encuentran todas las clases necesarias para representar las funcionalidades del sistema. El subpaquete Servicio define métodos que son accedidos desde aplicación, y se encarga de realizar las validaciones de los datos antes de realizar las operaciones en el repositorio.

En el subpaquete Contratos de Repositorios (*repositoryContracts*) se encuentra la interfaz *IOpenSSHRepository*, la cual tiene definido los métodos que son accedidos desde el subpaquete Servicio y son implementados por la clase *OpenSSHServerRepository* encontrada en el paquete de persistencia (*persistens*). Esta es la responsable de cargar y salvar la información en los repositorios.

El paquete Infraestructura Transversal (*infrastructureCrosscutting*) es donde se almacena todo el código reutilizable de la Aplicación.

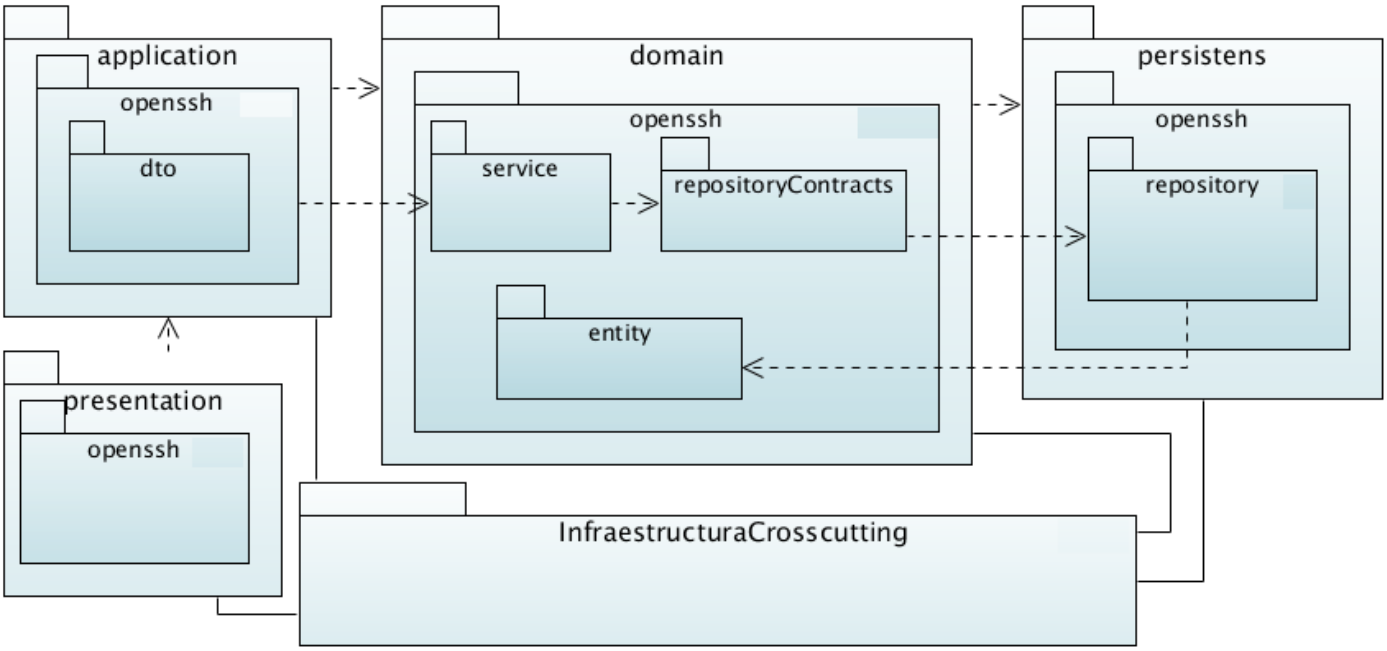


Ilustración 7: Diagrama de paquetes

2.6 Diagrama de clases

Los diagramas de clases muestran las diferentes clases que componen un sistema y cómo se relacionan entre ellas. Estos diagramas son conocidos como estáticos, porque muestran las clases, junto con sus métodos y atributos, así como las relaciones estáticas entre ellas. Su propósito principal es representar los objetos fundamentales del sistema, es decir los que percibe el usuario con los cuales espera realizar la acción deseada [27].

Para una mejor comprensión del módulo a implementar se muestra la *Ilustración 8: Diagrama de clases*, el cual está conformado por trece clases entrelazadas a través de las relaciones de composición.

La clase *AuthenticationParameters* es la encargada de gestionar toda la información referente a la autenticación del usuario. Es la que contiene las vías más importantes de autenticación como son: llaves públicas, usuario y contraseña. Además, es la encargada de permitir y denegar los usuarios y grupos que se van a conectar a través del servicio SSH. Esta clase va a contener un listado con todas las llaves públicas de los servidores, por lo cual va a estar compuesta por un objeto de la clase *KeyContainer* que es la que contiene esta lista.

KeyContainer es la encargada de englobar un listado de llaves y la dirección donde se van a guardar las mismas. Esta clase está compuesta por la clase *Key* que es la encargada de tener los parámetros para generar estas llaves.

La clase *Connections* tiene las funcionalidades referentes a direcciones de escucha, gestionar *host* y los puertos de escucha. Esta clase no está compuesta de ninguna otra, trayendo como consecuencia que tenga dentro todas sus funcionalidades.

AdvancedSettings es donde van a estar las funcionalidades del fichero menos utilizadas. La misma va a estar compuesta por cuatro clases: *Kerberos*, *GSSAPI, Banner* y *G*eneral*.* Cada una va a tener funcionalidades específicas, las cuales serán utilizadas por *AdvancedSettigns.*

Todas las clases anteriormente mencionadas van a ser controladas por la clase *OpenSSHServer*, la cual cuenta en sus atributos con un objeto de cada una de estas clases.

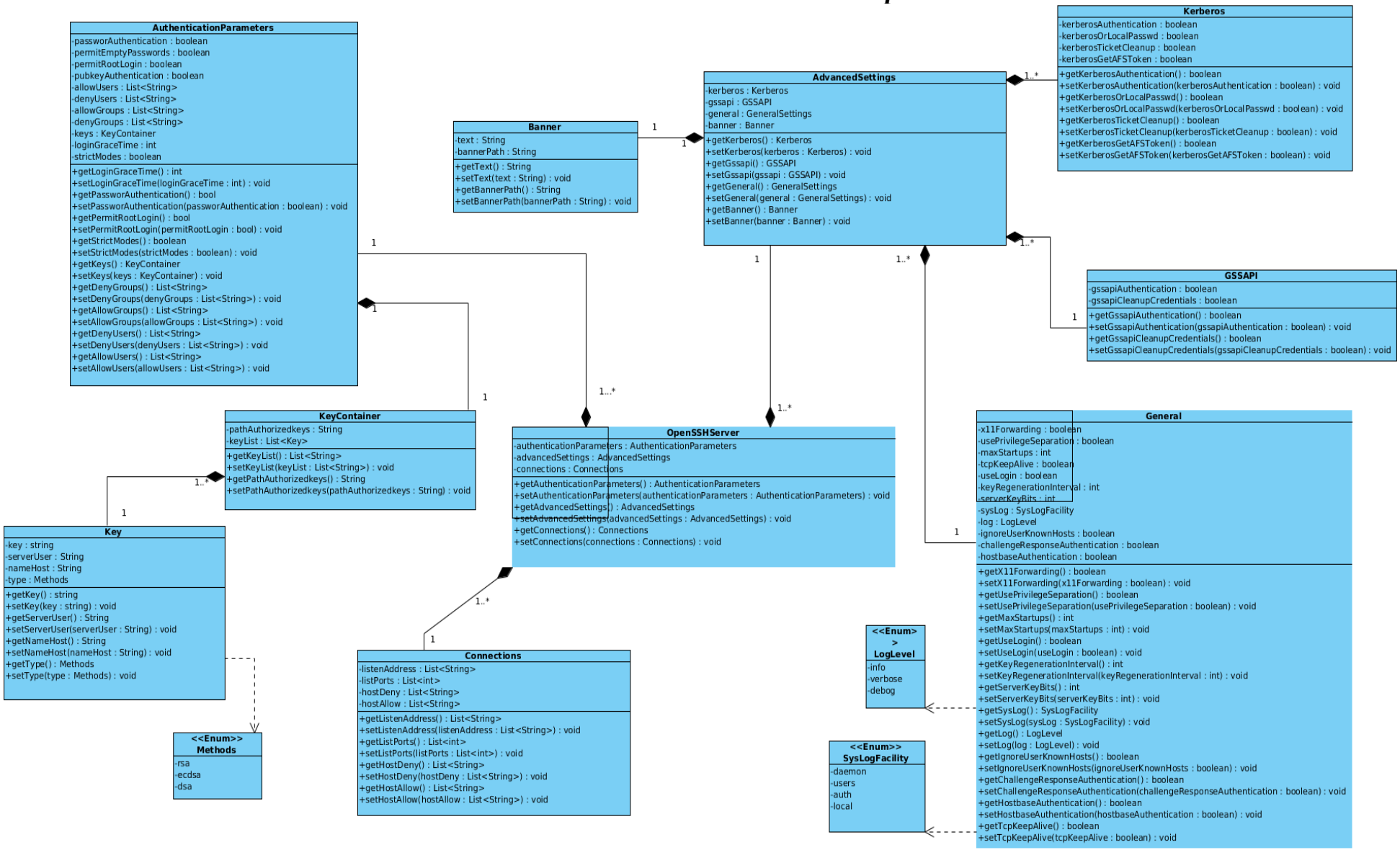


Ilustración 8: Diagrama de clases

2.7 Patrones de Diseño

Un patrón se define como “una solución a un problema de diseño que aparece con frecuencia”. Los patrones de diseño son el soporte de las soluciones a problemas comunes que surgen en el desarrollo de software. Son catalogados además, como soluciones simples y elegantes a problemas específicos del diseño orientado a objetos. En el diseño del módulo se tuvieron en cuenta los patrones generales de software para asignar responsabilidades conocidos en inglés con el nombre de *General Resposibility Asignment Software Patterns* (en lo adelante GRASP) y los patrones conocidos con el nombre de *The Gang of Four* (en lo adelante GOF) [28].

Patrones GRASP

Son utilizados para la asignación de responsabilidades, aunque se considera que más que patrones propiamente dichos, son una serie de buenas prácticas para el diseño de software. A continuación, se describen los patrones de este tipo que son utilizados para el diseño de la solución propuesta:

Experto: Es un patrón que se usa al asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos. Con el uso de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto posibilita un bajo acoplamiento, lo que favorece la creación de sistemas más robustos y de fácil mantenimiento. Con la utilización de este patrón, se hizo posible definir dónde colocar en cada clase las funcionalidades que necesitan de esa información, dicha clase sería el experto en información.

En el código se evidencia en cada una de las entidades, por ejemplo la entidad conexión, ya que va a contener todas las responsabilidades a la hora de conectarse a un servidor; dentro de ella se definen desde dónde se pude conectar el usuario, ya sea desde cualquier interfaz o una específica.

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental del mismo, es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Este patrón se utilizó para identificar qué clase A debe crear elementos de una clase B, apoyándose en que la clase A debería: contener, agregar, registrar, utilizar y tener los datos de inicialización de la clase B.

En el módulo, la utilización de este patrón se evidencia cuando las clases presentes en la capa Aplicación crean instancias de las entidades para la conversión de los objetos DTO a objetos del dominio. Además, brinda soporte a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras. Acoplamiento bajo significa que una clase no depende de muchas otras y una clase con alto o fuerte acoplamiento recurre a muchas otras. El uso de este patrón permite que las clases no se afecten por cambios de otros componentes, haciendo posible que sean fáciles de entender y de reutilizar.

Un ejemplo del bajo acoplamiento se ve también en la clase conexión ya que ella no depende de ninguna otra para responder cada una de sus responsabilidades.

El uso de este patrón trae beneficios como:

- Estimular la asignación de responsabilidades de modo que su colocación no incremente tanto el acoplamiento, como para producir resultados negativos propios de un alto acoplamiento.
- No se afectan por cambios de otros componentes.
- Son fáciles de entender y de reutilizar.

Alta cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Se hizo necesario utilizar este patrón en el software en cuestión, con el fin de controlar la complejidad de cada clase utilizada para mantener un buen comportamiento de las mismas; por esta razón, las que se identificaron con una amplia cantidad de funcionalidades se dividieron en otras clases siguiendo el propósito de distribuir de forma equitativa el peso de la complejidad y manteniendo además, la coherencia entre ellas.

Patrones GOF

Los patrones de diseño, conocidos como GOF se clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. A continuación, se detalla el que ha sido utilizado en el diseño de la solución.

Patrón Solitario (*Singleton*): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Se usa debido a la necesidad de trabajar con el mismo objeto en distintos momentos.

2.8 Conclusiones del capítulo

A través de la realización de este capítulo se concluye que:

- Una vez identificadas las principales funcionalidades que brindará el módulo a implementar, quedaron definidos 75 requisitos funcionales, los cuales fueron agrupados en 4 Historias de Usuarios y 5 requisitos no funcionales.
- Con el empleo de la metodología SXP fueron explicados los artefactos generados como son: Modelo de Dominio, Lista de Reserva del Producto e Historias de Usuarios.
- Partiendo de la arquitectura propuesta por HMAST fue diseñado el Diagrama de Paquetes. El cual ayudó a describir el empleo de los patrones de diseño y es flexible ante cualquier cambio en la lógica del negocio.

Capítulo 3: Implementación y prueba del módulo

En el presente capítulo se definen las iteraciones y en cada iteración se prueban cada una de las HU. Además, se describen las Tareas de Ingeniería propuestas para las Historias de Usuarios más relevantes, y los estándares de código a utilizar en la implementación del módulo. Para evitar efectos colaterales no deseados a la hora de realizar modificaciones, se realizan pruebas las cuales se dividen en dos grupos: pruebas de aceptación destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida, diseñada por el cliente final y pruebas unitarias, encargadas de verificar el código y diseñadas por los programadores. El objetivo que se persigue con la elaboración de este capítulo es documentar el desarrollo de la solución propuesta garantizando su calidad.

3.1 Planificación de desarrollo

Luego de ser analizadas las HU por parte del cliente, queda estimado el tiempo y esfuerzo dedicado para desarrollar cada una de ellas, realizándose la planificación de las etapas de implementación del sistema. Este plan concentra las HU por iteraciones, definiendo cuáles serán desarrolladas en cada iteración del proceso de implementación. Teniendo en cuenta lo anteriormente planteado se decide implementar el sistema en dos iteraciones, las cuales son descritas a continuación:

- Iteración 1: La primera iteración tiene como objetivo dar cumplimiento a las HU relacionadas con la configuración básica del servidor SSH2 (HMAST_SSH_1, HMAST_SSH_2, HMAST_SSH_3).
- Iteración 2: La segunda iteración tiene como finalidad dar cumplimiento a las HU relacionadas con las configuraciones avanzadas del servidor SSH2 (HMAST_SSH_4).

3.2 Tareas de Ingeniería

Las tareas de ingeniería son las que guían el trabajo de las iteraciones, las cuales se realizan para especificar las acciones llevadas a cabo por los programadores en cada HU, ya que estas no ofrecen el nivel de detalle requerido para saber qué implementar [29]. Según el Plan de Releases las HU se agruparon en dos iteraciones. A continuación se muestra las tareas de ingeniería derivadas de cada HU.

Tabla 6: TI Reiniciar el servicio

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** HMAST_SSH_1
Nombre Tarea: Probar el comando para reiniciar el servicio SSH	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 4/02/2014	**Fecha Fin:** 5/02/2014
Programador Responsable: Eva Lianet Ferrer Ruano	
Descripción: Ejecutar el siguiente comando y ver las posibles salida que este pueda tener en dependencia de los valores que tenga los parámetros del fichero de configuración. Reiniciar: service ssh restart	

Tabla 7: TI Parámetros de autenticación

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** HMAST_SSH_2
Nombre Tarea: Probar parámetros de configuración del SSH relacionados con la autenticación del mismo.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 6/02/2014	**Fecha Fin:** 7/02/2014
Programador Responsable: Eva Lianet Ferrer Ruano	
Descripción: Cambiar los valores de los siguientes parámetros de configuración y reiniciar el servicio para ver si los mismos no interfieren en la correcta ejecución del servicio. Parámetros a modificar: *PassworAuthentication*, *PermitEmptyPasswords*, *StrictModes*, *RSAAuthentication, PubkeyAuthentication*, %h/.ssh/authorized_keys, *AuthorizedKeysFile*, ~/.ssh/known_hosts, *LoginGraceTime, getent passwd DenyGroups*, *DenyUsers*.	

Tabla 8: TI Generar llave

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** HMAST_SSH_2
Nombre Tarea: Probar comando para la generación de llaves públicas y privadas de SSH.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 8/02/2014	**Fecha Fin:** 9/02/2014
Programador Responsable: Eva Lianet Ferrer Ruano	
Descripción: Crear llaves mediante la herramienta *keygen* descrito en la HU y cambiar los parámetros de configuración con el fin de estudiar las diferentes salidas del fichero en cada caso.	

Tabla 9: TI Conexión

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** HMAST_SSH_3
Nombre Tarea: Estudio de los comandos relacionados con las conexiones mediante SSH.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 10/02/2014	**Fecha Fin:** 11/02/2014
Programador Responsable: Eva Lianet Ferrer Ruano	
Descripción: Hacer cambios en los parámetros relacionados con las conexiones SSH para probar que el servicio se mantenga funcionando y estudiar su formato en el fichero de configuración. Las directivas que se modifican son: *ListenAddress*, en el fichero */etc/hosts.deny* la de *sshd:ip*, y *port*.	

Tabla 10: TI Configuraciones avanzadas

Tarea de Ingeniería	
Número Tarea:	**Número Historia de Usuario:** HMAST_SSH_4
Nombre Tarea: Probar los parámetros que forman parte de las configuraciones avanzadas de SSH.	

Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 12/02/2014	**Fecha Fin:** 13/02/2014
Programador Responsable: Eva Lianet Ferrer Ruano	
Descripción: Hacer cambios en los parámetros relacionados con las conexiones SSH para probar que el servicio se mantenga funcionando y estudiar su formato en el fichero de configuración. Las directivas a modificar son: *X11Forwarding, HostbasedAuthentication, IgnoreUserKnownHosts, ChallengeResponseAuthentication, UseLogin, KeyRegenerationInterval, ServerKeyBits, SyslogFacility, LogLevel, UsePrivilegeSeparation, MaxStartups, KerberosAuthentication, KerberosOrLocalPasswd, KerberosTicketCleanup, KerberosGetAFSToken, GSSAPIAuthentication, GSSAPICleanupCredentials, TCPKeepAlive, Banner, /etc/issue.net*	

3.3 Estándar de Codificación

Los estándares de codificación son utilizados para institucionalizar buenas prácticas, haciendo recomendaciones de diseño para lograr mayores niveles de calidad en la elaboración del producto. La definición de estos estándares se encuentra en el expediente de proyecto de HMAST.

3.3.1 Pautas Generales

- La primera letra de los nombres de las clases debe comenzar con mayúscula y el resto se pondrá con minúscula, cuando sea un nombre compuesto se utilizará la notación Pascal[12].

  Ejemplo: *Connections*

- Para nombrar las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se utiliza la notación CamelCase[13].

  Ejemplo: *addUser*

[12] Notación Pascal: especifica que la primera letra de cada palabra utilizada debe estar en mayúsculas.

[13] Notación CamelCase: CamelCase es la práctica de escribir frases o palabras compuestas eliminando los espacios y poniendo en mayúscula la primera letra de cada palabra.

- Para nombrar variables se escribe la primera palabra con minúscula, si es un nombre compuesto se utilizará la notación CamelCase.

  Ejemplo: *opensshRepository*

- Estándares para nombrar componentes

  Todos los paquetes comienzan con cu.uci.hmast.xxx.yyy.zzz.kkk

  xxx →presentation, application, domain, persistence.

  yyy →nombre del módulo (ssh, dhcp3, bind9).

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

  kkk → clases o subpaquetes.

  Ejemplo: cu.uci.hmast.domain.ssh.entitys.Authenticacion

3.4 Pruebas

Las pruebas permiten aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. Las propuestas por SXP son conocidas como pruebas de aceptación destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final. Además, se realizan pruebas unitarias las cuales se encargan de verificar el código y son diseñada por los programadores, sin embargo la metodología usada no requiere que sean plasmadas en el documento.

3.4.1 Pruebas de Aceptación

Las pruebas de aceptación (PA) también conocidas como pruebas del cliente, son utilizadas para probar que los requisitos han sido implementados de forma correcta al final de cada iteración y por lo tanto comprueban que las funcionalidades del sistema se encuentran en relación con las Historias de Usuario definidas [30]. A continuación se muestra una relación de las más importantes las demás se encuentran en el expediente del proyecto.

Tabla 11: Caso de prueba 1

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST-SSH_1-1	**Nombre Historia de Usuario:** Gestionar estado del servicio SSH.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Probar que el servicio SSH se reinicia.	
Condiciones de Ejecución: Estar conectado a un servidor OpenSSH.	
Entrada / Pasos de ejecución: 1. Abrir el módulo de OpenSSH. 2. Seleccionar la opción de reiniciar el servicio SSH.	
Resultado Esperado: El servicio SSH se reinicia y muestra su estado.	
Evaluación de la Prueba: Satisfactoria	

Tabla 12: Caso de prueba 2

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST-SSH_2-2	**Nombre Historia de Usuario:** Gestionar autenticación de OpenSSH.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se muestra el valor de la directiva *PassworAuthentication*.	
Condiciones de Ejecución: Estar conectado a un servidor OpenSSH. Tener la directiva *PassworAuthentication* configurada.	
Entrada / Pasos de ejecución: 1. Abrir el módulo OpenSSH. 2. Verificar que el valor que tiene la directiva en el fichero es la misma que la mostrada en la interfaz.	
Resultado Esperado: Coincide el valor de la interfaz con el del fichero de SSH.	
Evaluación de la Prueba: Satisfactoria	

Tabla 13: Caso de prueba 3

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST-SSH_2-3	**Nombre Historia de Usuario:** Gestionar autenticación de OpenSSH.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se modifica el valor de la directiva *PassworAuthentication*.	
Condiciones de Ejecución: Estar conectado a un servidor OpenSSH. Tener la directiva *PassworAuthentication* configurada.	
Entrada / Pasos de ejecución: 1. Abrir el módulo OpenSSH. 2. Activar la directiva mediante la interfaz. 3. Verificar que la directiva toma el valor “no” en el fichero.	
Resultado Esperado: El valor de la directiva en el fichero es “no”.	
Evaluación de la Prueba: Satisfactoria	

Tabla 14: Caso de prueba 4

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST-SSH_2-4	**Nombre Historia de Usuario:** Gestionar autenticación de OpenSSH.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se muestra el valor de la directiva *PermitEmptyPasswords*.	
Condiciones de Ejecución: Estar conectado a un servidor OpenSSH. Tener la directiva *PermitEmptyPasswords* configurada.	
Entrada / Pasos de ejecución: 1. Abrir el módulo OpenSSH. 2. Verificar que el valor que tiene la directiva en el fichero es la misma que la mostrada en la interfaz.	
Resultado Esperado: Coincide el valor de la interfaz con el del fichero de SSH.	
Evaluación de la Prueba: Satisfactoria	

Tabla 15: Caso de prueba 5

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST-SSH_2-5	**Nombre Historia de Usuario:** Gestionar autenticación de OpenSSH.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se modifica el valor de la directiva *PermitEmptyPasswords*.	
Condiciones de Ejecución: Estar conectado a un servidor OpenSSH. Tener la directiva *PermitEmptyPasswords* configurada.	
Entrada / Pasos de ejecución: 1. Abrir el módulo OpenSSH. 2. Activar la directiva mediante la interfaz. 3. Verificar que la directiva toma el valor “yes” en el fichero.	
Resultado Esperado: El valor de la directiva en el fichero es “yes”.	
Evaluación de la Prueba: Satisfactoria	

Tabla 16: Caso de prueba 6

Caso de Prueba de Aceptación	
Código Caso de Prueba: HMAST-SSH_2-6	**Nombre Historia de Usuario:** Gestionar autenticación de OpenSSH.
Nombre de la persona que realiza la prueba: Reidiel Castillo Arbelo	
Descripción de la Prueba: Verificar que se modifica el valor de la directiva *PermitEmptyPasswords*.	
Condiciones de Ejecución: Estar conectado a un servidor OpenSSH. Tener la directiva *PermitEmptyPasswords* configurada.	
Entrada / Pasos de ejecución: 1. Abrir el módulo OpenSSH. 2. Activar la directiva mediante la interfaz. 3. Verificar que la directiva toma el valor “no” en el fichero.	
Resultado Esperado: El valor de la directiva en el fichero es “no”.	
Evaluación de la Prueba: Satisfactoria	

Al terminar cada iteración de desarrollo se realizaron las pruebas correspondientes a las HU implementadas con el objetivo de detectar los posibles errores y corregirlos en el menor tiempo posible. En la primera iteración se ejecutaron 70 PA de las cuales 10 no fueron satisfactorias. Para la segunda iteración se realizaron 50 PA además de las pendientes de la iteración anterior, detectándose solo 5 no conformidades las cuales fueron corregidas y revisadas posteriormente con el cliente.

También se realizaron pruebas por parte del Jefe de proyecto y del Jefe de departamento con el objetivo verificar el correcto funcionamiento del mismo, resultando todas las pruebas positivas y quedando como constancia el acta de aceptación emitida por el jefe de departamento.

3.5 Conclusiones del Capítulo

A través de la realización de este capítulo se concluye que:

- A partir de las pruebas realizadas al módulo en el presente capítulo se logró verificar el correcto funcionamiento del mismo. La técnica de prueba utilizada permitió validar y verificar el funcionamiento del módulo a través de la realización de pruebas de aceptación. Los resultados obtenidos a partir de la ejecución de los casos de pruebas demuestran que el producto final cuenta con la calidad deseada y esperada por el cliente.

Conclusiones

Con la culminación de la presente investigación se puede concluir que:

- OpenSSH es el servidor más adecuado para la administración del servicio SSH en la herramienta HMAST, ya que garantiza la seguridad de las conexiones, permite diferentes vías de autenticación y brinda la posibilidad de restringir los usuarios que se conectan a través de este servicio.

- El módulo desarrollado está basado en las funcionalidades obtenidas del estudio de las herramientas Yast2 y Webmin, dotando a la herramienta HMAST de funcionalidades que facilitan la administración del servicio SSH por parte de los administradores de red, a través de un diseño que se integra a la arquitectura de HMAST.

- El módulo implementado cumple con la totalidad de los requisitos funcionales definidos por el cliente, lo que se pudo constatar con el resultado satisfactorio de las pruebas de aceptación realizadas.

Recomendaciones

Como recomendación de esta investigación se plantea:

- Agregar funcionalidades que permitan la monitorización en tiempo real de los usuarios que están conectados mediante SSH al servidor.

Referencias bibliográficas

[1] Hernández Sampieri, Roberto; Fernández Collado, Carlos y Lucio Batista, Pilar. Metodología de la Investigación. México, McGraw-Hill, 2006. 760 p. ISBN: 970-10-5753-8.

[2] BERNERS-LEE, Tim. Weaving the Web. Siglo XXI de España, 2000. 237 p. ISBN: 8432310409

[3] Jorba Esteve, Josep y Suppi Boldrito, Remo. Administración Avanzada de GNU/Linux. Universidad de Cataluña, Marzo 2004. 472 p. ISBN: 84-9788-116-8

[4] Kirch, Olaf. Guía de Administración de Redes con Linux. Traducción Proyecto LuCAS, Mayo 1999. 382 p.

[5] Definición de telnet - Qué es, Significado y Concepto. [En línea]. [Consultado el: 4 de Noviembre de 2013]. Disponible en: http://definicion.de/telnet/

[6] Tatu Ylonen, Finland | LinkedIn. [En línea]. [Consultado el: 31 de Enero de 2014]. Disponible en: http://fi.linkedin.com/in/tatuylonen

[7] Protocolos. [En línea]. [Consultado el: 31 de Enero de 2014]. Disponible en: http://es.kioskea.net/contents/275-protocolos

[8] DWIVEDI, HIMANSHU. Implementing SSH. Strategies for Optimizing the Secure Shell. Malm. Wiley Publishing, 2004. ISBN: 0-471-45880-5

[9] The MIT License (MIT) | Open Source Initiative. [En línea]. [Consultado el: 31 de Enero de 2014]. Disponible en: http://opensource.org/licenses/MIT

[10] Dropbear SSH. [En línea]. [Consultado el: 30 de Enero de 2014]. Disponible en: https://matt.ucc.asn.au/dropbear/dropbear.html

[11] Tectia SSH Server. [En línea]. [Consultado el: 31 de Enero de 2014]. Disponible en: http://www.ssh.com/index.php/products/tectia-ssh-server.html

[12] LLORENTE DE LA TORRE, CÉSAR y ZORRILLA CASTRO, UNAI y RAMOS BARROS, MIGUEL ANGEL JAVIER CALVARRO. Guía de Arquitectura N-Capas orientada al Dominio [En línea]. Microsoft_109.aspx., 2010. Disponible en: http://www.campusmvp.com/catalogo/Product-Gu%EF%BF%BDa-de-Arquitectura-N-Capas-orientada-al-ISBN 978-84-936696-3-8.

[13] Pérez Tasé, Alexander. Módulo de administración del servicio Proxy para HMAST. Universidad de las Ciencias Informáticas. Habana, Junio 2013. 78 p.

[14] KATHY SIERRA Y BERT BATES. Java Head First. 2da Edición. 677 p.

[15] Schildt, Herbert. Java 2, Manual de referencia. Madrid, McGraw-Hill, 2001. 453 p. ISBN: 0-07-213084-9.

[16] González Rodríguez, Leover Armando. Alternativas para el desarrollo de Aplicaciones Web. Junio 2011.

[17] CRAIG WALLS. Spring in Action [En línea]. MEAP Edition. 2010. [Consultado el: 13 de Febrero de 2014]. Habilitado desde: http://www.manning-sandbox.com/forum.jspa?forumID=573

[18] PEÑALVER ROMERO, GLADYS MARSI. Metodología ágil para proyectos de software libre. Ingeniera Informática. Habana: Universidad de las Ciencias Informáticas, 2008.80

[19] Entorno de Desarrollo Integrado (IDE). | fergarciac. [En línea]. [Consultado el: 12 de Febrero de 2014]. Habilitado desde: http://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/

[20] NetBeans IDE - Overview. [En línea]. [Consultado el: 31 de Enero de 2014]. Habilitado desde: https://netbeans.org/features/index.html

[21] Simple Tools for Software Modeling -OR- It's "Use the Simplest Tool" not "Use Simple Tools."[En línea]. [Consultado el: 3 de Abril de 2014]. http://www.agilemodeling.com/essays/simpleTools.htm#SelectingCASE

[22] UML, BPMN and Enterprise Architecture Tool for Software Development. [En línea]. [Consultado el: 3 de Abril de 2014]. Habilitado desde: http://www.visual-paradigm.com/

[23] RapidSVN. [En línea]. [Consultado el: 31 de Enero de 2014] Habilitado desde: http://www.rapidsvn.org/

[24] Tain Domínguez, Yuliette, y Valdivia Genó, Román Miguel. Augeas, propuesta tecnológica para la gestión de archivos de configuración de sistemas GNU/Linux. Mayo 2009. Pencil Project. [En línea]. [Consultado el: 3 de Abril de 2014]. Habilitado desde: http://pencil.evolus.vn/

[25] Pencil Project, 2013. [En línea] [Consultado el: 20 de febrero de 2014]. Habilitado desde: http://pencil.evolus.vn/

[26] ZORRILLA CASTRO, CÉSAR, RAMOS BARROSO, MIGUEL ANGEL y CALVARRO NELSO, Javier. Guía de Arquitectura N-Capas orientada al Dominio. 2010. ISBN 978- 84-936696-38.

[27] Berzal, Fernando. Relaciones entre clases: Diagrama de clases UML.

[28] Larman Craig. UML y Patrones, Introducción al Análisis y Diseño Orientado a Objetos.507

[29] Peñalver Romero, Gladys Marsi. MA-GMPR-UR2. Metodología ágil para proyectos de software libre. Junio 2008.

[30] Pruebas de Software. [En línea] [Consultado el: 31 de Enero de 2014] Gestión de calidad y Pruebas de Software. 2005. Habilitado desde: http://www.pruebasdesoftware.com/pruebadeaceptacion.htm

Glosario de Términos

A

Algoritmos de Encriptación: Un algoritmo criptográfico es una función matemática utilizada en los procesos de encriptación y desencriptación. El cual trabaja en combinación con una llave (un número, palabra, frase, o contraseña) para encriptar y desencriptar los datos. Su objetivo es hacer tan difícil como sea posible el desencriptar los datos sin utilizar la llave.

C

Clientes: Un cliente realiza peticiones a otro programa y es el servidor quien da la respuesta. El cliente tiene un papel activo en las comunicaciones, ya que es quien inicia las solicitudes o peticiones. Espera y recibe la respuesta del servidor. Puede conectarse a varios servidores a la vez y es el que interactúa con el usuario final mediante una interfaz gráfica.

Composición: Las composiciones son asociaciones que representan acumulaciones muy fuertes. Esto significa que las composiciones forman relaciones completas, pero dichas relaciones son tan fuertes que las partes no pueden existir por sí mismas. Únicamente existen como parte del conjunto, y si este es destruido las partes también lo son.

F

Framework: Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

FTP: Siglas en inglés *File Transfer Protocol*, en español Protocolo de Transferencia de Archivos, es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP (*Transmission Control Protocol*), basado en la arquitectura cliente-servidor.

I

IPv6: Protocolo de Internet versión 6, es una versión del protocolo *Internet Protocol* (IP), diseñada para reemplazar a Internet Protocol version 4 (IPv4).

N

P

Patrones GRASP: describen los principales fundamentos del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

S

Servidor: Un servidor es un proveedor de recursos, desempeñando un papel pasivo en las comunicaciones. Tras la recepción de una solicitud son los encargados de procesarla y enviar una respuesta al cliente. Puede por lo general aceptar conexiones de un gran número de clientes, y por lo general no interactúan con los usuarios finales.

U

UML (*Unified Modeling Language*): Lenguaje Unificado de Modelado es un lenguaje de modelado de sistemas de software.

Usuario: persona que tiene una cuenta en una determinada computadora por medio de la cual puede acceder a los recursos y servicios que ofrece una red.

Anexo A: Prototipos de Interfaz.

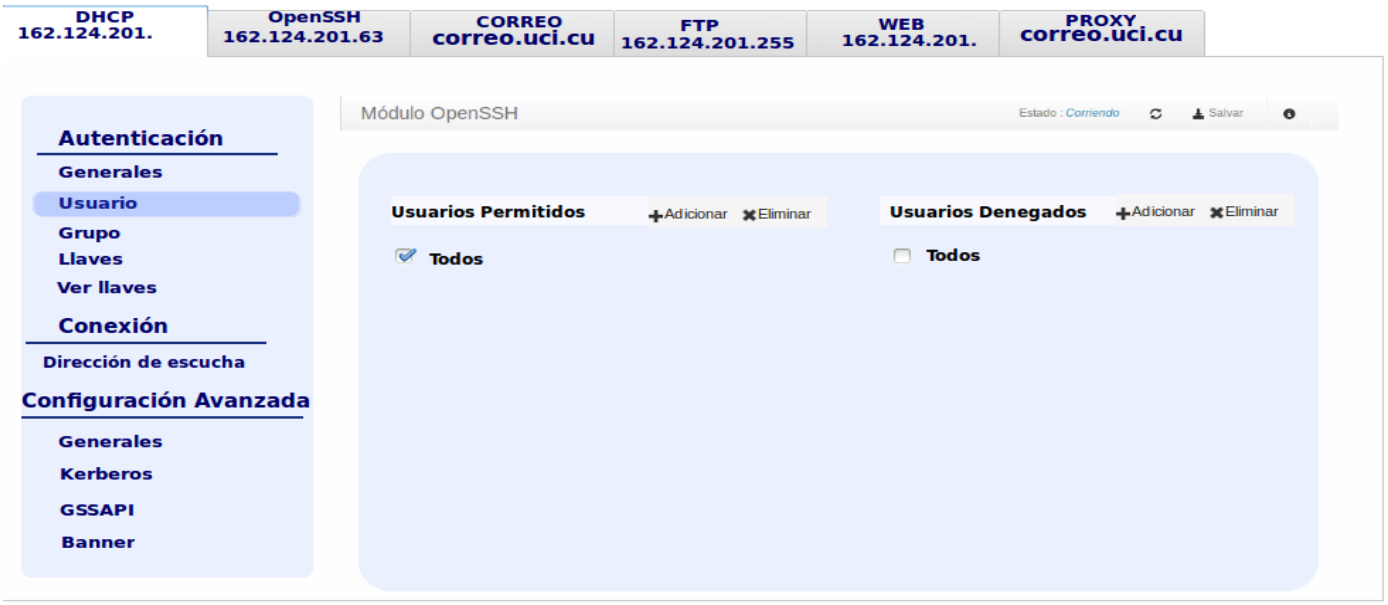


Ilustración 9: PI Autenticación de Usuarios.

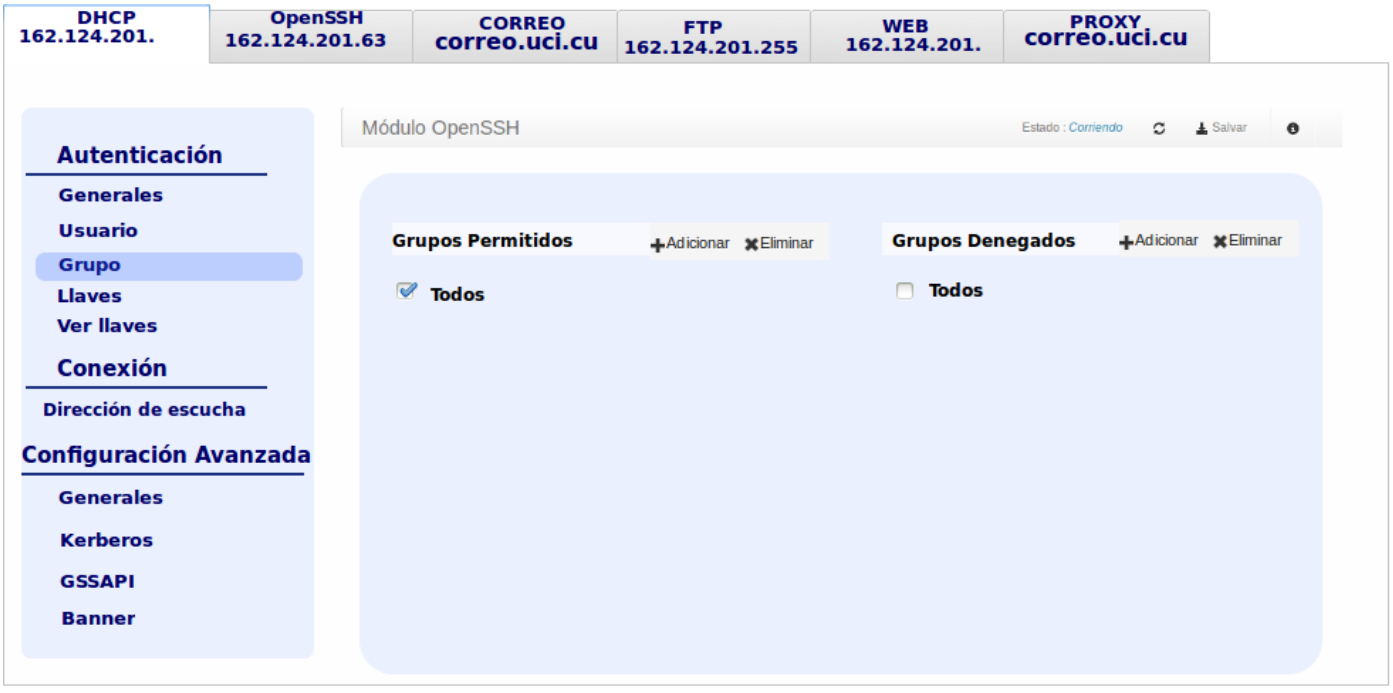


Ilustración 10: PI Autenticación de Grupos.

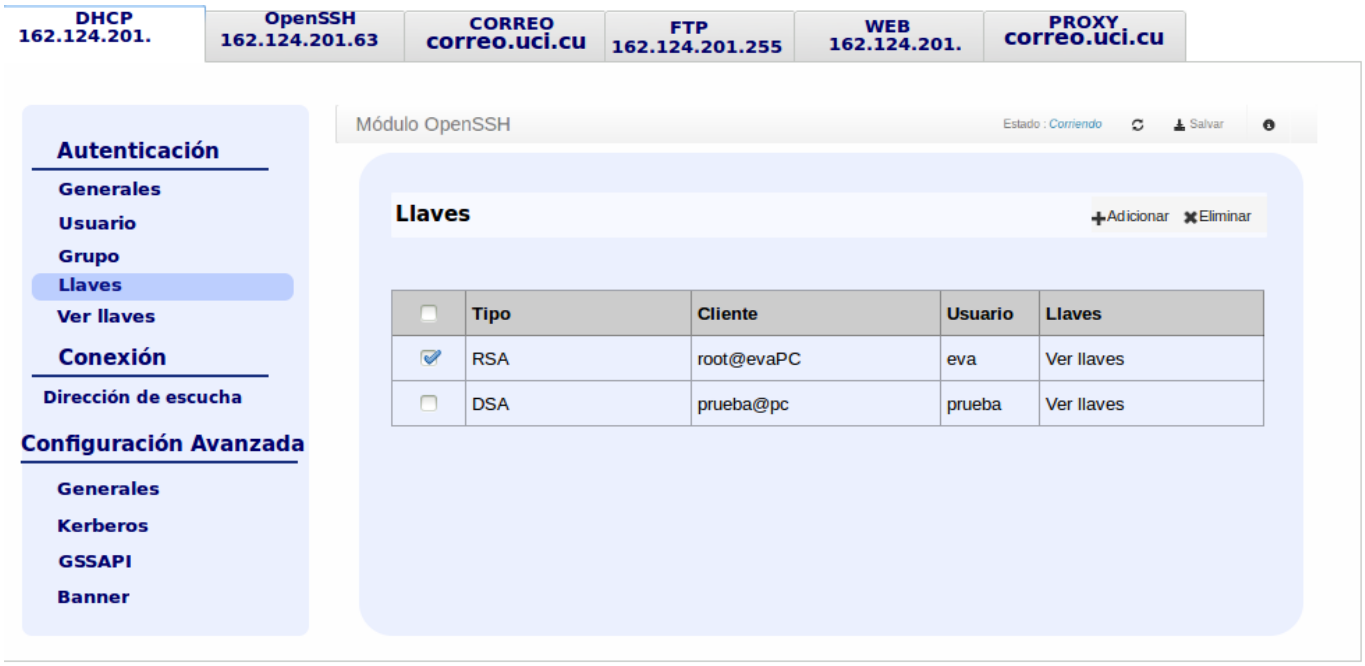


Ilustración 11: PI Autenticación de llave.

Ilustración 12: PI Llave Pública.

Ilustración 13: PI Seleccionar Interfaces de Red.

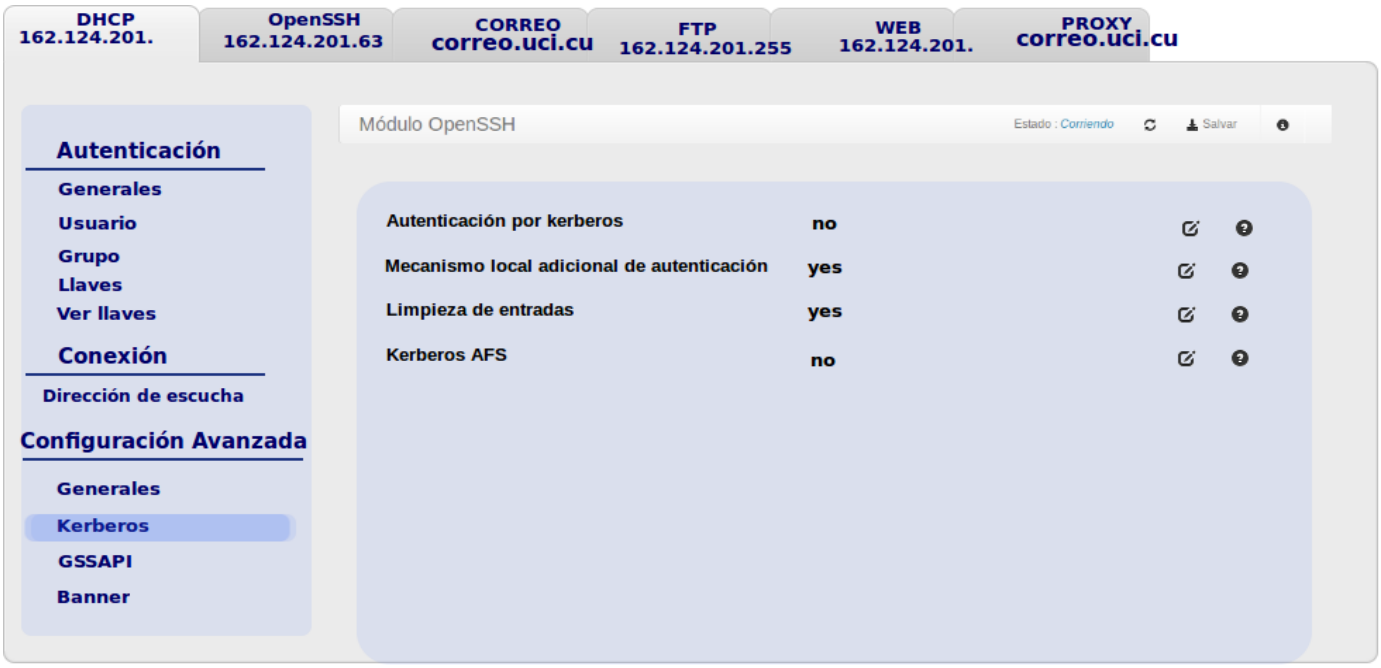


Ilustración 14: PI Configuraciones Avanzadas Kerberos.

Ilustración 15: PI Configuraciones Avanzadas GSSAPI.

Ilustración 16: PI Configuraciones Generales Banner.