Universidad de las Ciencias Informáticas

Facultad 1

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

Título: Servicio antivirus para la infraestructura del servicio de correo de la Herramienta para la Migración y Administración de Servicios Telemáticos

Autor: Yordán Herrera Cordova.

Tutores: Ing. Yadiel Pérez Villazón.

La Habana, junio de 2017

Declaración de Autoría

Declaro por este medio que yo Yordán Herrera Cordova, con carné de identidad 93092301946 soy el autor principal del trabajo de diploma titulado “Servicio antivirus para la infraestructura del servicio de correo de la Herramienta para la Migración y Administración de Servicios Telemáticos” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo. Para que así conste se firma la presente, a los ___ días del mes de _____ del año 2017.

_____

Firma del autor

_____

Firma del tutor

Ing. Yadiel Pérez Villazón

Resumen

El servicio de antivirus como parte de una infraestructura de correo electrónico es fundamental. La propagación de virus a través de adjuntos trae consigo la vulnerabilidad del servicio, pues pueden ocurrir fallas de seguridad en el tráfico de datos, el acceso a los mismos por un tercero y, además, se puede ver afectado en determinado momento la disponibilidad del servicio. El presente trabajo se centró en el objetivo de desarrollar un componente que permita el escaneo y detección de virus en tiempo real, así como su integración con el módulo de correo electrónico en la Herramienta para la Migración y Administración de Servicios Telemáticos. Para ello se analizan los servicios antivirus existentes, seleccionando el antivirus ClamAv como el software más completo de los que fueron analizados. Se emplearon los métodos Histórico-Lógico y Analítico-Sintético como técnica de recopilación de información. Se documentaron las diferentes tecnologías a usar y la metodología de desarrollo de software Proceso Unificado Ágil en su variación para la Universidad de las Ciencias Informáticas, como guía del proceso. Finalmente se obtuvo un componente que permite la instalación y administración de ClamAv por parte de los administradores del sistema, ya sea en uno o varios servidores de forma remota.

Palabras Clave: administración, ClamAv, migración, remota, servidor.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Los servicios telemáticos son aquellos que, utilizando como soporte servicios básicos, permiten el intercambio de información entre terminales con protocolos establecidos para sistemas de interconexión abiertos (1). Ejemplos de ellos lo constituyen dhcp[1], proxy[2], bacula[3] y correo.

El servicio de correo electrónico permite el intercambio de mensajes a través de sistemas de comunicación electrónicos. Los mensajes de correo electrónico posibilitan el envío, además de texto, de cualquier tipo de documento digital (imágenes, videos, audios, entre otros).

En el año 2016, el sitio de noticias Expansión expuso los resultados de una encuesta asociado al uso de correo electrónico y redes sociales. Dicha encuesta concluía que la mayor parte del mundo está interconectado debido al correo electrónico y redes sociales como Facebook y Twitter. De ello, el 96% de las personas de todo el mundo que está conectada online, envía y recibe correos electrónicos (2). En el caso de Cuba específicamente, el servicio de correo electrónico es utilizado usualmente como herramienta de trabajo en casi el total de las empresas que cuenten con una infraestructura de redes.

Dentro de las principales ventajas del correo electrónico se encuentran:

- Rapidez, un documento enviado por correo electrónico llega casi instantáneamente al destinatario.
- El costo es muy bajo, no importa la distancia ni la ubicación a donde se envíe. Solo se necesita contar con una conexión a internet y una cuenta gratuita de correo electrónico.
- No hay que desgastarse haciendo copias, puede ser enviado a muchas personas a la vez.
- Permite enviar todo tipo de archivos, video, sonido, imágenes, cuadros estadísticos y ejecutables.

El servicio de correo es resultado de diversas tecnologías, las cuales se integran para formar una infraestructura que permite el funcionamiento del mismo. Un servicio de correo debe tener un agente de transferencia de correo (MTA), agente de entrega de correo (MDA), herramientas para el filtrado de contenido, ya sea filtrado de *spam* o filtrado de virus.

Cuba se encuentra inmersa en un proceso de informatización de la sociedad, priorizando sectores como la salud, la educación y la cultura. Además, como consecuencia del acuerdo 084 del Consejo de

[1] DHCP: Servicio telemático de asignación dinámica de direcciones ip.
[2] Proxy: Servicio telemático intermediario en las peticiones de recursos que realiza un cliente a otro servidor.
[3] Bacula: Servicio telemático capaz de cubrir las necesidades de respaldo de equipos bajo redes IP.

Ministros, se encuentra en un proceso de migración paulatina y progresiva a software libre. En su mayoría, la infraestructura tecnológica de los Organismos de Administración Central del Estado (OACE) del país se encuentra sobre plataformas privativas, principalmente de la familia Microsoft Windows.

Con el objetivo de apoyar el proceso de migración que lleva a cabo el país es concebida la Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST). Recientemente fue liberada su versión 2.0, con un total de 5 módulos: dhcp, apache2, mysql, bacula y ssh. Para la próxima versión de HMAST se pretende incluir el módulo correspondiente al servicio de correo. Como parte del desarrollo de este módulo están definidas las tecnologías Postfix como MTA, Dovecot como MDA, Amavis-new como herramienta de filtrado de contenido y Spamassassin para el filtrado de correos *spam*. Para que el módulo de correo de HMAST sea más completo, debería contar con un componente para la gestión de escaneo y detección de virus en tiempo real.

El escaneo y detección de virus dentro de la infraestructura de un servicio de correo provee diversas ventajas. Existe una gran diversidad de virus que se propagan mediante archivos adjuntos. Es preciso contar con antivirus actualizados para escanear cada archivo adjunto, y preferiblemente no abrir correos electrónicos cuando son de dudosa procedencia. Esta situación trae consigo la vulnerabilidad del servicio, pues pueden ocurrir fallas de seguridad en el tráfico de datos, el acceso a los mismos por un tercero y, además, se puede ver afectado en determinado momento la disponibilidad del servicio.

Además de lo explicado anteriormente, según establece la Oficina de Seguridad de Redes Informáticas (OSRI), en los OACE, los servidores y aplicaciones informáticas que lo requieran deben incluir un sistema de detección de virus actualizado y correctamente funcional.

Para dar solución a la situación problemática anteriormente mencionada se presenta como **problema científico:** ¿Cómo garantizar el escaneo y detección de virus en tiempo real en la infraestructura de correo electrónico desde HMAST?

Como **objetivo general** del trabajo se plantea: desarrollar un componente que permita la administración del antivirus y su integración al módulo de correo electrónico de HMAST.

El **objeto de estudio** del presente trabajo se centra en el proceso de administración del servicio de antivirus en el correo electrónico en plataformas libres.

Como **objetivos específicos** se establecen los siguientes:

- Caracterizar la infraestructura del módulo de correo electrónico de HMAST.
- Sistematizar el estado del arte de las tecnologías de servicios antivirus y su integración con la

infraestructura de correo electrónico.

- Diseñar un componente para HMAST que permita la administración del servicio antivirus desde el módulo del servicio de correo.
- Implementar la solución propuesta.
- Realizar pruebas a la solución implementada.

Para el cumplimiento de los objetivos específicos se identificaron las siguientes **tareas de investigación:**

- Caracterización de la arquitectura y características del módulo de correo electrónico de HMAST.
- Caracterización y análisis de bibliografía especializada de los antivirus para servicios de correo electrónico sobre plataformas GNU/Linux[4].
- Realización del análisis y diseño del componente para la administración del servicio de antivirus para su integración en la infraestructura del servicio de correo electrónico de HMAST.
- Implementación del componente para la gestión del servicio de antivirus en la infraestructura del módulo de correo electrónico de HMAST.
- Realización de pruebas funcionales y de integración a la solución implementada.

Se plantea como **idea a defender:**

La creación de un componente que permita administrar el servicio de antivirus en el módulo correo electrónico desde HMAST, sustentado en la selección adecuada de un antivirus, asegurará el escaneo y detección de virus en la infraestructura de correo electrónico.

La investigación se sustenta en el empleo de los **métodos científicos** siguientes:

El análisis histórico - lógico: Se utiliza para determinar los antecedentes históricos relacionados con las herramientas para la gestión del servicio de antivirus, así como su administración en el servicio de correo electrónico y las aplicaciones que lo componen.

Analítico – sintético: Se utiliza en la investigación de los servicios de antivirus existentes para servicios de correo electrónico sobre plataformas GNU/Linux y los componentes que estos utilizan. Contribuye a escoger el más apropiado para utilizarlo en el desarrollo de la aplicación propuesta.

Resultados esperados:

- Un componente para la gestión del servicio de antivirus desde el módulo de correo electrónico de la herramienta HMAST.

[4] GNU/Linux: Sistema operativo, más conocido como Linux.

- Una infraestructura que cuente con un mecanismo de escaneo y detección en tiempo real de virus.

El presente documento se compone por introducción, tres capítulos, conclusiones generales, bibliografía, referencias bibliográficas, el glosario de términos, donde se explican los vocablos de difícil comprensión que se han empleado en la elaboración de esta investigación, y los anexos. La estructura de los capítulos se define a continuación:

Capítulo 1: Marco teórico referencial asociado al componente del servicio de antivirus en la infraestructura del servicio de correo. Se realiza un estudio de HMAST y de su módulo de correo electrónico para conocer sus características y recopilar información acerca de cómo implementar un nuevo componente para el mismo. Se definen los conceptos asociados y se analizan los principales antivirus para servicios de correo electrónico sobre plataformas GNU/Linux, seleccionando, de acuerdo a las condiciones de las empresas cubanas, el más adecuado para la solución a implementar.

Capítulo 2: Análisis y diseño del componente para la gestión del servicio de antivirus en el módulo de correo electrónico. Se presenta la propuesta de solución al problema planteado, son definidos los requisitos funcionales, no funcionales y de integración. Por último, en este capítulo se muestran las historias de usuario y se detalla la arquitectura del sistema.

Capítulo 3: Implementación del componente para la gestión del servicio de antivirus desde el módulo de correo electrónico. Se realizan las pruebas y comprobaciones necesarias para certificar que el componente desarrollado cumple con los requisitos planteados y se adapta al ambiente tecnológico de las instituciones cubanas.

Capítulo 1: Fundamentación teórica.

En el presente capítulo se realiza un estudio de las herramientas existentes para el escaneo y detección de virus en el servicio de correo electrónico sobre plataformas GNU/Linux, a fin de elegir la más adecuada para integrar al módulo de correo. Se caracteriza el módulo de correo de HMAST, así como su arquitectura, y se exponen las consideraciones a tener en cuenta para la integración del componente al mismo. Además, se detallan las herramientas y tecnologías a utilizar para el desarrollo del componente.

1.1 Conceptos asociados al servicio de antivirus en el correo.

Para una mejor comprensión del tema se definen los términos empleados en la investigación.

Antivirus.

Un antivirus es un programa informático que tiene el propósito de detectar y eliminar virus y otros programas perjudiciales en un sistema informático, antes o después de que ingresen al sistema (3).

Malware.

Malware es la abreviatura de "*Malicious software",* término que engloba todo tipo de programa o código informático malicioso cuya función es dañar un sistema o causar un mal funcionamiento. Dentro de este grupo se pueden encontrar términos como: virus, troyanos, gusanos, entre otros (4).

Anti-malware.

El anti-malware es un tipo de programa diseñado para prevenir, detectar y remediar software malicioso en los dispositivos informáticos individuales y sistemas. Los términos antivirus y anti-malware se utilizan a menudo como sinónimos ya que los virus informáticos son un tipo específico de malware. Se denomina anti-malware a aquel software que evita la infiltración en el sistema y el daño (5).

Virus.

Los virus son programas informáticos que tienen como objetivo alterar el funcionamiento del computador, sin que el usuario lo detecte. Estos, por lo general, infectan otros archivos del sistema con la intención de modificarlos para destruir de manera intencionada archivos o datos almacenados en tu computador

(6).

Troyano.

En informática, se denomina caballo de Troya, o troyano, a un software malicioso que se presenta al usuario como un programa aparentemente legítimo e inofensivo, pero que, al ejecutarlo, le brinda a un atacante acceso remoto al equipo infectado (7).

Gusano.

Un gusano informático es similar a un virus por su diseño, y es considerado una subclase de virus. Los gusanos informáticos se propagan de ordenador a ordenador, pero a diferencia de un virus, tienen la capacidad de propagarse sin la ayuda de una persona (8).

Listas grises.

Una lista gris o *greylist* es una técnica para el control de mensajes *spam*[5]. Es un método de defensa que bloquea la mayoría de los correos *spam* que se reciben en un servicio de correo electrónico (9).

Listas blancas.

La lista blanca de *spam* permite especificar remitentes de correo en los que siempre se confía. Los mensajes procedentes de dichos remitentes no van a ser analizados por ninguna de las protecciones antispam (9).

Listas negras.

La lista negra de *spam* permite especificar remitentes de correo que se consideran peligrosos. Los mensajes procedentes de dichos remitentes se van a clasificar siempre como *spam* (9).

Una vez analizados los conceptos relacionados con antivirus en el servicio de correo, se procede a estudiar el módulo de correo de HMAST.

[5] Spam: mensaje basura.

1.2 Descripción del módulo de correo de HMAST:

1.2.1 Tecnologías asociadas al módulo

El servicio de correo electrónico está compuesto por la unión de varias tecnologías que posibilitan la comunicación entre los usuarios. En específico el módulo de correo de HMAST está compuesto por seis tecnologías (10):

Postfix

Postfix es un servicio de correo electrónico libre (licenciado bajo IBM *Public License*) para el enrutamiento y envío de correo electrónico a través del protocolo SMTP[6] (11).

Amavisd-new

Amavisd-new es una interfaz entre el MTA y las herramientas que chequean contenido como aplicaciones antivirus y *spam*. Está escrito en Perl, lo que garantiza la facilidad de mantenimiento (12).

- Se ejecuta como demonio en el sistema, iniciándose automáticamente.
- Se ejecuta con un usuario propio, garantizando un mayor nivel de seguridad en el sistema.
- Implementa chequeo de errores, informando con mensajes de notificación las fallas ocurridas. Dispone de un mecanismo de funcionamiento a prueba de errores.
- Dispone de soporte para más de 40 antivirus, brindando la posibilidad de ejecución paralela de los mismos.
- Soporta de forma nativa SMTP y LMTP[7].
- Presenta mecanismos contra ataques de denegación de servicios (DOS).

Dovecot

Dovecot es un MDA. Tiene la función de recuperar los mensajes del servicio de correo electrónico remoto

[6] SMTP: Es un protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos.
[7] LMTP: Protocolo de transporte local de correo.

y copiarlos en el programa de correo del usuario mediante los protocolos IMAP[8] y POP3[9] (13).

SpamAssassin

SpamAssassin es el programa de filtrado *spam* que permite identificar y tratar correos no deseados que reciben los usuarios mediante el análisis de cabecera, texto, listas de bloqueo DNS[10] y bases de datos de filtrado colaborativo (14).

Fetchmail

Fetchmail se integra al servicio de correo con el objetivo de permitir la recuperación remota de correos. Es compatible con todos los protocolos de control remoto electrónico que se utilizan actualmente en Internet: POP2, POP3, RPOP, APOP, KPOP, todas las versiones de IMAP, ETRN y ODMR. Incluso puede soportar IPv6 e IPSec. Fetchmail recupera los correos remotos y los reenvía a través de SMTP[11] (15).

1.2.2 Integración de las tecnologías

Cuando un correo llega por el protocolo SMTP a través del puerto 25, Postfix acepta la conexión, lo lee y hace algunas comprobaciones básicas como chequear si el remitente se encuentra en la lista negra; autentica el usuario contra la base de datos local o el directorio activo en dependencia del tipo de autenticación, verificando que el usuario sea válido en el sistema, si no se confía en el sistema remoto se le aplica la política de listas grises. En esta etapa Postfix puede rechazar el correo electrónico o aceptarlo. Una vez que Postfix acepta el correo envía el mensaje a través del protocolo SMTP por puerto TCP 10024 a la interfaz de Amavisd-new, la cual realiza el análisis del contenido y entrega el mensaje a SpamAssassin para realizar el filtrado antispam. Después de estas comprobaciones, Amavisd-new devuelve el correo a Postfix por SMTP a través del puerto TCP 10025. Postfix envía posteriormente el correo a Dovecot. Opcionalmente esta herramienta puede aplicar filtros por usuario de manera que los mensajes se puedan almacenar en determinadas carpetas automáticamente si se desea. Finalmente

[8] IMAP: Protocolo de acceso a mensajes de internet.
[9] POP3: Protocolo de acceso a mensajes de internet.
[10] DNS: Sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como Internet o una red privada.
[11] SMTP: Protocolo para transferencia simple de correo.

escribe el correo en el disco duro en formato maildir (16) (ver Figura 1).

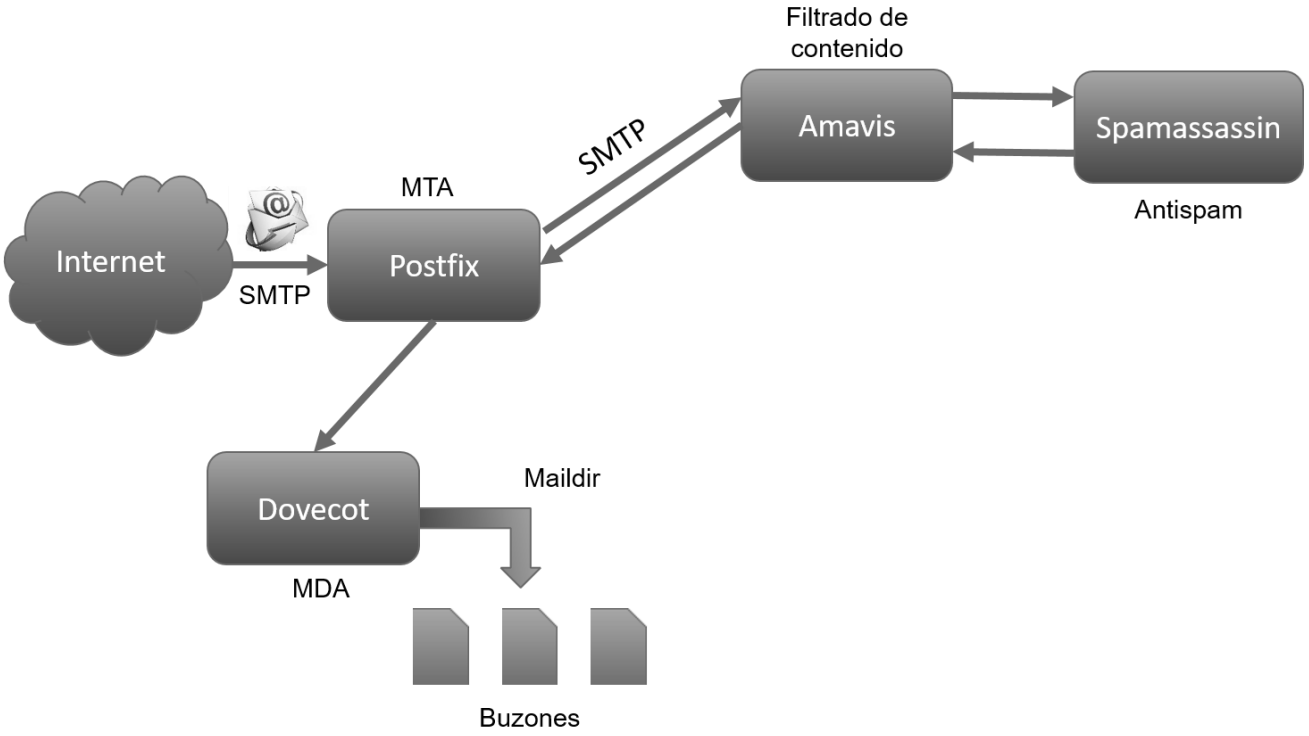


Figura 1. Integración de las tecnologías.

1.2.3 Arquitectura

El módulo posee el estilo arquitectónico N-Capas orientada al dominio que propone la herramienta HMAST, incluyendo en esta la distribución del módulo de correo entre las diferentes capas según las responsabilidades que estas establezcan (ver Figura 1) (16).

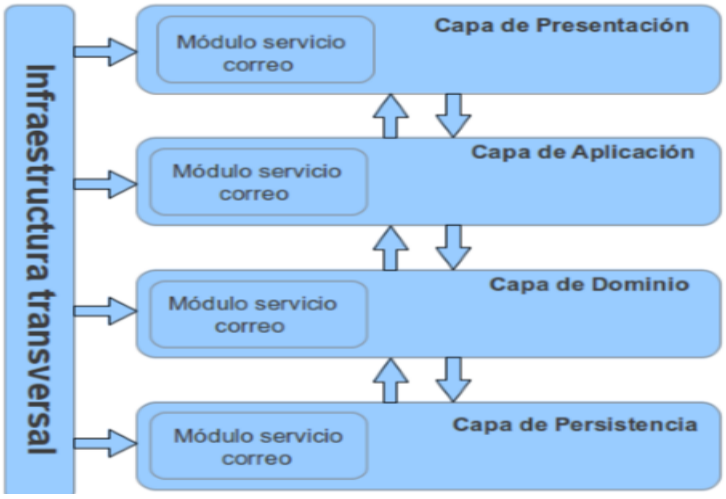


Figura 2. Vista de arquitectura lógica del módulo de correo electrónico de la herramienta HMAST.

Dentro de las principales funcionalidades que actualmente posee el módulo se encuentran:

- Gestionar cuenta de correo.
- Gestionar dominios virtuales.
- Gestionar configuración del MTA.
- Gestionar configuración del MDA.
- Gestionar listas negras.
- Gestionar listas blancas.
- Gestionar restricciones de adjunto.

Para el desarrollo del módulo de correo electrónico se utilizaron las tecnologías que se mencionan a continuación: lenguaje de programación Java, como marco de trabajo Spring, Visual Paradigm para el modelado UML, IntelliJ IDEA como IDE de programación y Apache Tomcat como servidor web. Definido ya el funcionamiento del módulo de correo de HMAST y las tecnologías utilizadas para su desarrollo, se hace necesario el estudio y selección del antivirus a emplear.

1.3 Selección del Antivirus

Entre los antivirus existentes para servicios de correo electrónico sobre plataformas GNU/Linux, se encuentran entre los más populares ClamAv, *Kaspersky Security for Linux Mail Server y Sophos Server Protection* (17). A continuación, se expone un resumen de sus principales características:

ClamAv

ClamAv es un motor antivirus de código abierto para la detección de troyanos, virus, malware y otras amenazas maliciosas. Incluye un escáner demonio multihilo, las utilidades de línea de comandos para el análisis de archivos de la demanda y las actualizaciones automáticas (18). Características más notables:

- Escáner de línea de comandos.
- Base de datos de virus actualizada varias veces al día.
- El soporte integrado para todos los formatos de archivos de correo estándar.
- El soporte integrado para varios formatos de archivo.
- Soporte integrado para archivos ejecutables portátiles empaquetados.
- El soporte integrado para los formatos de documentos populares.
- Licencia libre / código abierto.

Kaspersky Security for Linux Mail Server

Kaspersky Security for Linux Mail Server diseñado para integrarse con una gran variedad de sistemas de correo basados en Linux. Incluye tecnologías basadas en la nube lo cual permite lograr un bajo índice de falsos positivos (19).

Características más notables:

- Motor anti-malware con prevención de ataques dirigidos.
- Funciones anti-malware a varios niveles.
- Anti-spam y filtrado de contenidos.

- Integración con una gran variedad de sistemas de correo basados en Linux
- Fácil instalación y mantenimiento.
- Licencia privativa.

Sophos Server Protection

Sophos Server Protection está diseñado para proteger las aplicaciones y los datos en cualquier organización sin importar donde se encuentren almacenados, ya sea en servidores físicos, servidores virtuales, en la nube o en las instalaciones. Protección de Sophos Server integra listas blancas de aplicaciones de servidor las cuales protegen contra amenazas conocidas y nuevas, junto con un sistema HIPS (Sistema de prevención de intrusiones basado en el Host) que proporciona una gestión sencilla y centralizada de la seguridad de todos los servidores (20).

Características más notables:

- Sistema de control de dispositivos, el cual brinda protección contra robos de datos a través de memorias USB.
- Control de aplicaciones, el cual limita las aplicaciones que pueden ejecutarse, esto reduce aún más la superficie expuesta a ataques.
- Bloqueo de servidor, esto permite que se ejecuten solo determinadas aplicaciones y archivos asociados o de confianza.
- Escaneado exhaustivo anti-malware antes del bloqueo, para que sus servidores se bloqueen en un estado seguro.
- Licencia privativa.

Tabla 1. Comparación entre antivirus.

Aspectos a medir	Kaspersky	ClamAv	Sophos Server Protection
Tipo de licencia	Privativa	Libre / Código abierto	Privativa

Motor heurístico	Alto	Medio	Medio
Rendimiento	Alto	Alto	Alto
Consumo de recursos	Alto	Bajo	Bajo
Efectividad en el análisis	Alto	Alto	Alto
Integración con Amavisd-new	No	Si	No

Para comparar las herramientas de antivirus se tuvieron en cuenta varios aspectos a medir, los cuales estuvieron basados en el método QSOS (Calificación y Selección de Software Open Source), el cual fue diseñado para evaluar el software libre de manera objetiva, centrado en torno a los criterios definidos para estimar los riesgos asumidos por el usuario. La propuesta de software antivirus para el servicio de correo de HMAST es la herramienta ClamAv. Esta propuesta se sustenta en los argumentos de que la herramienta antes mencionada tiene licencia GPL, lo cual es un elemento primordial para las instituciones del país. Además, cuenta con características de poco consumo de recursos, un rasgo importante teniendo en cuenta que la mayoría de las empresas cubanas no cuentan con un hardware de alto rendimiento, así como una correcta integración con la herramienta Amavisd-new.

1.3.1 ClamAv

El proyecto ClamAv Antivirus fue fundado en el año 2001 por Tomasz Kojm. Actualmente tiene una implantación superior a los 500 000 servidores en todo el mundo. ClamAv nació como un proyecto *opensource* que pretende identificar y bloquear virus en el sistema. El primer objetivo de ClamAv fue combatir el correo electrónico *malware*. Como consecuencia de ello, ClamAv se está usando en un número elevado de servidores de correo.

El objetivo primario de ClamAv es la consecución de un conjunto de herramientas que identifiquen y bloqueen el *malware* proveniente del correo electrónico. Uno de los puntos fundamentales en este tipo de *software* es la rápida localización e inclusión en la herramienta de los nuevos virus encontrados y escaneados. Esto se consigue gracias a la colaboración de los miles de usuarios que usan ClamAv, y a

sitios como Virustotal.com que proporcionan los virus escaneados.

Otra pieza clave de ClamAv es el soporte de desarrolladores que posee en todo el mundo; esta red de desarrolladores global posibilita una rápida reacción ante cualquier evidencia de un nuevo virus.

El proyecto ClamAv se desarrolla gracias a una red de contribuidores (proporcionan *patches*, información de *bugs*, soporte técnico y documentación). Por otro lado, existe una serie de personas e instituciones que colaboran con donaciones a la realización del proyecto. Existe un comité de dirección que supervisa y coordina el proyecto (21).

Esta herramienta presenta requisitos para ser utilizada sobre plataformas GNU/Linux en cualquiera de sus versiones.

1.3.2 Configuración de ClamAv

El antivirus ClamAv se instala a través de la consola de Linux, en cualquier sistema operativo basado en GNU/Linux, en la ruta */etc/clamav.* El archivo de configuración principal es *clamd.conf* y se puede encontrar en la ruta: */etc/clamav/*clamd.conf. En este fichero se encuentran las directivas asociadas a la configuración general del antivirus, como son la cantidad de procesos de trabajo que utiliza el antivirus y máxima profundidad a escanear.

A continuación, se describen las directivas de mayor criticidad para la investigación.

Tabla 2. Directivas del fichero clamd.conf.

Directivas	Valores admisibles	Descripción
ScanMail	*true(*habilitado*)/false(*deshabilitado*)*	Permite el escaneo de los archivos de correo.
LogVerbose	*true(*habilitado*)/false(*deshabilitado*)*	Permite habilitar los *logs* detallados.
ScanArchive	*true(*habilitado*)/false(*deshabilitado*)*	Permite el escaneo de los archivos comprimidos.

ScanPE	*true(*habilitado*)/false(*deshabilitado*)*	Permite el escaneo de los archivos ejecutables.
ScanHTML	*true(*habilitado*)/false(*deshabilitado*)*	Permite el escaneo de los documentos HTML.
ScanOLE2	*true(*habilitado*)/false(*deshabilitado*)*	Permite el escaneo de los documentos de Microsoft Office y archivos msi.
ScanPDF	*true(*habilitado*)/false(*deshabilitado*)*	Permite el escaneo de los documentos PDF.
ScanELF	*true(*habilitado*)/false(*deshabilitado*)*	Permite el escaneo de los documentos ELF.
ScanXMLDOCS	*true(*habilitado*)/false(*deshabilitado*)*	Permite el escaneo de los documentos XMLDOCS.
ScanSWF	*true(*habilitado*)/false(*deshabilitado*)*	Permite el escaneo de los documentos SWF.
ScanHWP3	*true(*habilitado*)/false(*deshabilitado*)*	Permite el escaneo de los documentos HWP3.
MaxDirectoryRecursion	*true(*habilitado*)/false(*deshabilitado*)*	Especifica la máxima profundidad a escanear.
MaxThreads	*true(*habilitado*)/false(*deshabilitado*)*	Especifica la cantidad de procesos de trabajo que utiliza el antivirus.
DatabaseDirectory	*true(*habilitado*)/false(*deshabilitado*)*	Especifica donde se guardan las actualizaciones del antivirus.
LogFile	*true(*habilitado*)/false(*deshabilitado*)*	Especifica donde se guardan los *logs* detallados.

Para lograr la actualización, ClamAv posee una herramienta denominada Freshclam, la cual es responsable de descargar las actualizaciones. Freshclam tiene la finalidad de descargar las actualizaciones desde los servidores oficiales del antivirus.

1.4 Tecnologías para desarrollar el componente.

Para el correcto desarrollo y una correcta integración es necesario la utilización de las tecnologías usadas en el desarrollo del módulo de correo, las cuales se muestran a continuación.

A continuación, se describen las herramientas que apoyarán el desarrollo del componente para la solución.

Lenguaje de programación Java.

Java fue diseñado como un lenguaje orientado a objetos, los cuales agrupan en estructuras encapsuladas tanto sus datos como las funcionalidades. Proporciona una colección de clases para su uso en aplicaciones de red que permiten abrir *sockets* y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas. Es interpretado y compilado a la vez. Proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Soporta sincronización de múltiples hilos de ejecución a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas (9).

Framework Spring.

Es un *framework* de código abierto, ofrece un modelo de programación y configuración completa para las modernas aplicaciones empresariales basadas en Java, funciona en cualquier tipo de plataforma de despliegue. Entre sus características principales se encuentran la inyección de dependencias, permitiendo un bajo acoplamiento entre los objetos de la aplicación y Programación Orientada a Aspectos[12] (22).

[12] Programación Orientada a Aspectos: Es un paradigma de programación.

IntelliJ IDEA.

IntelliJ IDEA es un IDE para Java y otros lenguajes, se puede descargar sin costo alguno y está abierto a las contribuciones de los miembros de la comunidad. Dentro de sus más notables características incluye:

- Inteligencia profunda.
- Editor de código inteligente.
- Refactorizaciones, inspecciones e intenciones de código.
- Integración de *frameworks* de prueba.
- Soporte de herramientas de construcción.
- Integración de sistemas de control de versiones populares (23).

Visual Paradigm.

Visual Paradigm es una herramienta para desarrollo de aplicaciones utilizando modelado UML para ingenieros de software, analistas y arquitectos que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Es una herramienta colaborativa, pues soporta múltiples usuarios trabajando sobre el mismo proyecto. Con el uso de esta se pueden dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta también ofrece:

- Navegación intuitiva entre la escritura del código y su visualización.
- Generador de informes en formato PDF/HTML.
- Ambiente visualmente superior de modelado (24).

Apache Tomcat.

Apache Tomcat es un servidor web con una implementación de código abierto y participativo (25).

1.5 Metodología de desarrollo de software.

La metodología AUP en su variación para la UCI define 4 fases, 9 roles y 4 escenarios. Las fases de AUP-UCI son: inicio, elaboración, construcción y transición. Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación. En la segunda fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Una vez concluida la segunda fase el sistema es desarrollado y probado al completo en el ambiente de desarrollo. Por último, el sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación, y finalmente se despliega en los sistemas de producción.

Los roles definidos por esta metodología son: administrador de proyecto, ingeniero de procesos, desarrollado, administrador de base de datos, modelador ágil, administrador de la configuración, *stakeholder*, administrador de pruebas y probador. Además. Para el desarrollo del proyecto se emplea el escenario número 4, el cual indica que proyectos que no modelen negocio solo pueden modelar el sistema con HU (14).

Conclusiones parciales.

La revisión de la bibliografía especializada permitió identificar que las herramientas existentes para el escaneo y detección de virus en servicio de correo electrónico sobre plataformas GNU/Linux. Se identificó ClamaAV como la más adecuada, su estudio propició la comprensión de las características, funcionamiento, configuraciones y administración de la misma. El estudio del módulo de correo de HMAST precisó que, para la integración del componente, se debe emplear la metodología de desarrollo de software AUP en su variación para la UCI y las herramientas: Spring, IntelliJ IDEA, Visual Paradigm, Git.

Capítulo 2: Diseño y análisis del componente.

En el presente capítulo se describe la propuesta del software a desarrollar teniendo en cuenta las características del módulo de correo electrónico. Se presenta la propuesta de solución al problema planteado, son definidos los requisitos funcionales, no funcionales y de integración. Por último, en este capítulo se muestran las historias de usuario y se detalla la arquitectura del sistema.

2.1 Propuesta de solución.

A continuación, se muestra una descripción de cómo se integra la solución que se propone a la infraestructura del módulo de correo de HMAST.

Cuando un correo llega por el protocolo SMTP a través del puerto 25, Postfix acepta la conexión, lo lee y hace algunas comprobaciones básicas como chequear si el remitente se encuentra en la lista negra. Luego, autentica el usuario contra la base de datos local o el directorio activo en dependencia del tipo de autenticación, verificando que el usuario sea válido en el sistema, si no se confía en el sistema remoto se le aplica la política de listas grises. En esta etapa Postfix puede rechazar el correo electrónico o aceptarlo.

Posteriormente envía el mensaje a través del protocolo SMTP por puerto TCP 10024 a la interfaz de Amavis, la cual realiza el análisis del contenido. Esta entrega el mensaje a SpamAssassin y a ClamAv para realizar el filtrado antispam y antivirus respectivamente. ClamAv, para la actualización de sus firmas, se conecta a través del protocolo HTTP al servidor de descargas. Después de estas comprobaciones, Amavis devuelve el correo a Postfix por SMTP a través del puerto TCP 10025. Postfix envía posteriormente el correo a Dovecot. Opcionalmente esta herramienta puede aplicar filtros por usuario, de manera que los mensajes se puedan almacenar en determinadas carpetas automáticamente si se desea. Finalmente escribe el correo en el disco duro en formato maildir.

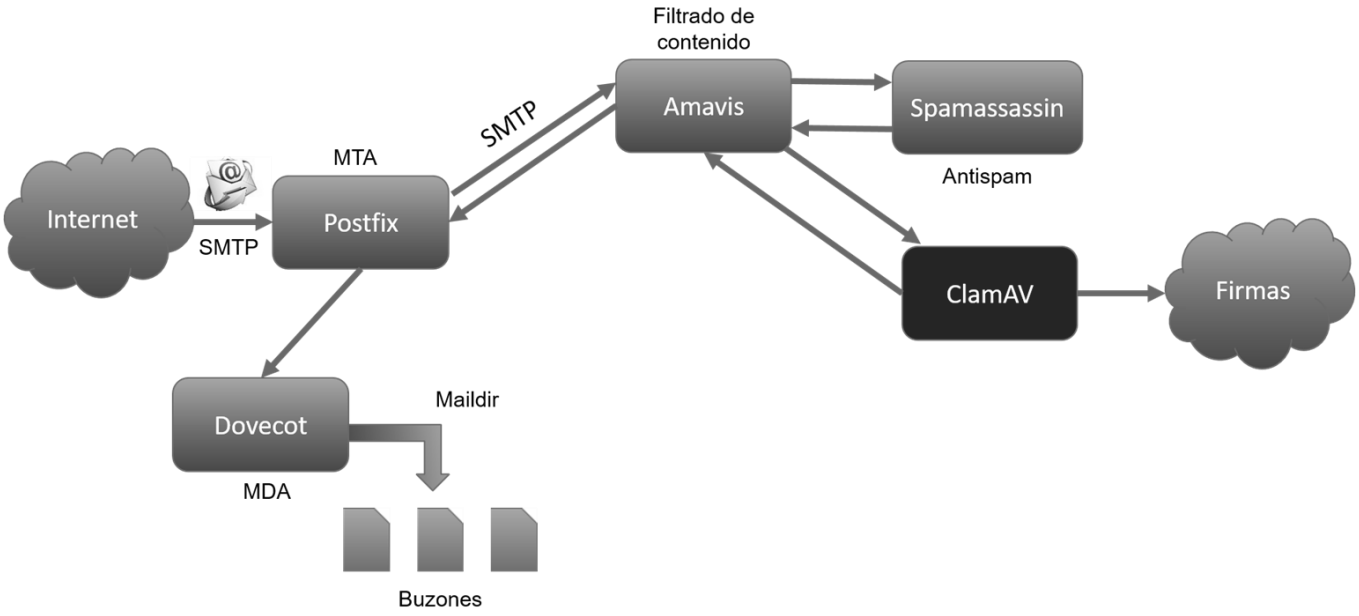


Figura 3. Propuesta del sistema a desarrollar.

El presente trabajo propone un componente que integra el antivirus ClamAv a la infraestructura de servicio de correo de HMAST. Además, permite al administrador, a través de una interfaz intuitiva poder establecer los detalles de actualización, configurar las acciones a tomar para los correos infestados, así como ajustar las propiedades del antivirus.

2.1.1 Mecanismo de actualización de ClamAv.

El mecanismo de actualización establecido de forma predeterminada por ClamAv, o sea Freshclam, no puede ser incorporado como parte de la solución por restricciones de acceso a los servidores oficiales de ClamAv. Ante este problema surge la necesidad de buscar una nueva alternativa para la correcta actualización del antivirus, esta alternativa consiste en la creación de un *script* en el directorio */etc/clamav/update.sh* con el fin de permitir la descarga de las actualizaciones. Dicho *script* establece la ejecución de la descarga definiendo los parámetros requeridos, dígase servidor de descarga, puerto, usuario, contraseña, url del servidor de actualizaciones y el directorio destino de actualización.

El *script* por sí mismo no garantiza que se descarguen las actualizaciones de forma automática, lo cual

es un elemento fundamental. Se propone la creación de una nueva tarea automática utilizando el servicio cron[13]. Se crea un fichero en */etc/cron.d/clamav* especificando minuto, hora, día del mes, mes, día de la semana, usuario y comando. Por defecto se propone la siguiente configuración 0 12 * * * root sh */etc/clamav/update.sh*.

2.1.1 Artefactos Generados

El desarrollo de la propuesta de solución es guiado por la metodología AUP-UCI. No se realiza el modelado del negocio y las funcionalidades se describen en un documento de Especificación de Requisitos de Software, todo esto corresponde al escenario 4 que establece esta metodología. Por lo que el principal artefacto generado en el proceso de desarrollo lo constituyen las Historias de Usuario, en las cuales los requisitos son encapsulados.

2.2 Especificación de requisitos

El Glosario de Terminología Estándar de Ingeniería de Software define al requisito como una condición que necesita un usuario para resolver un problema o lograr un objetivo. También como una capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. Estos se clasifican en funcionales y no funcionales (26).

Los requisitos funcionales describen las funciones que el software debe ejecutar, con el objetivo de establecer un entendimiento común entre el usuario y el proyecto de software. La siguiente tabla muestra el listado de los requisitos funcionales del módulo, clasificados según su prioridad en alta, media y baja.

Tabla 3. Requisitos funcionales.

No.	Nombre del requisito funcional	Descripción	Prioridad
RF 1	Instalar servicio.	Permite la instalación del servicio a través de un comando en consola.	Media

[13] Cron: Servicio por defecto en sistemas GNU/Linux para la gestión de tareas automáticas.

RF 2	Desinstalar servicio.	Permite la desinstalación del servicio a través de un comando en consola.	Media
RF 3	Iniciar servicio.	Permite iniciar el servicio a través de un comando en consola.	Media
RF 4	Detener servicio.	Permite detener el servicio a través de un comando en consola.	Media
RF 5	Reiniciar servicio.	Permite reiniciar el servicio a través de un comando en consola.	Media
RF 6	Mostrar configuración avanzada.	Permite mostrar los parámetros de configuración avanzada del antivirus.	Media
RF 7	Modificar configuración avanzada.	Permite modificar los parámetros de configuración avanzada del antivirus.	Media
RF 8	Mostrar configuración básica.	Permite mostrar los parámetros de configuración básica del antivirus.	Media
RF 9	Modificar configuración básica.	Permite modificar los parámetros de configuración básica del antivirus.	Media
RF 10	Mostrar configuración de actualización.	Permite mostrar los parámetros para la actualización del servicio Antivirus.	Media
RF 11	Modificar configuración de actualización.	Permite modificar los parámetros para la actualización del servicio Antivirus.	Media
RF 12	Mostrar configuración de actualización automática.	Permite mostrar los parámetros de configuración básica del antivirus.	Media
RF 13	Modificar configuración de actualización automática.	Permite modificar los parámetros de configuración básica del antivirus.	Media
RF 14	Mostrar ficheros que están en cuarentena.	Permite mediante esta opción mostrar los archivos que están en cuarentena.	Media
RF 15	Restaurar ficheros que están en cuarentena.	Permite mediante esta opción restaurar los archivos que están en cuarentena.	Media

RF 16	Eliminar ficheros que están en cuarentena.	Permite mediante esta opción eliminar los archivos que están en cuarentena.	Media
RF 17	Mostrar acción cuando un correo tiene virus.	Permite mediante esta opción mostrar acción cuando un correo tiene virus.	Alta
RF 18	Modificar acción cuando un correo tiene virus.	Permite mediante esta opción establecer una nueva acción cuando un correo tiene virus.	Alta
RF 19	Notificar amenaza de virus.	Permite mediante esta opción habilitar envió de un correo de notificación al remitente, destinatario y al administrador.	Baja

Los requisitos no funcionales se refieren a cualidades que imponen restricciones en el diseño y la implementación (27). La metodología AUP-UCI propone una taxonomía partiendo de la ISO 25010, donde se asocian estos requisitos a atributos de calidad. Diversos requisitos no funcionales son heredados de la herramienta HMAST, puesto que condicionan el funcionamiento del módulo para lograr una correcta integración con el sistema base. A continuación, se muestra el listado de los requisitos no funcionales.

Tabla 4. Listado de requisitos no funcionales.

No	Nombre del requisito no funcional	Atributo de calidad	Descripción
1	Emplear como lenguaje de programación Java.	Funcionalidad.	Estas son restricciones heredadas de HMAST, el módulo debe cumplirlas para poder integrarse correctamente.
2	El módulo se ejecutará sobre el sistema operativo GNU/Linux Nova Server.		
3	Disponer en el sistema operativo GNU/Linux de los siguientes paquetes: libjna-java, openjdk-8-jdk		
4	Proteger información y los datos, para que personas o sistemas desautorizados no puedan leer o modificar los mismos, y las personas	Seguridad (Acceso restringido.	Todos los datos manejados por el módulo estarán encriptados; tanto los que se transfieren entre la estación cliente y el

	o sistemas autorizados tenga el acceso a ellos.		servidor HMAST como los que se almacenan temporalmente en el servidor.
5	Permitir al usuario aprender su aplicación.	Usabilidad.	Internacionalizar la información que se muestra, en los idiomas español e inglés.
6	El módulo debe mantener un nivel de ejecución o desempeño especificado en caso de fallos del software o de infracción de su interfaz especificada.	Confiabilidad (Tolerancia a fallos).	Ante el fallo de una funcionalidad del sistema el resto de las funcionalidades que no dependen de esta, deberán seguir funcionando.

2.2.2 Historias de Usuario.

Las Historias de Usuario especifican las tareas que debe realizar el sistema, lo que equivale a los casos de uso en el Proceso Unificado. Son escritas en lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto. Guían la construcción de las pruebas de aceptación y son utilizadas para estimar tiempos de desarrollo (28). A continuación, se muestran algunas de las Historias de Usuario, las demás se pueden encontrar en los anexos.

Tabla 5. HU Instalar el servicio de antivirus.

Numero: HU_1	**Nombre del Requisito**: Instalar el servicio de antivirus.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Alta	**Tiempo Estimado:** 20
Riego en Desarrollo: Alto	**Tiempo Real**: 15
Descripción: El sistema debe permitir al usuario la instalación del servicio de antivirus ClamAv. La funcionalidad comienza cuando el usuario accede al sistema, si el servicio no está instalado, puede instalarlo.	
Observaciones: Se ejecuta en la PC servidora el comando ***apt install clamav-daemon –y***.	

Prototipo elemental de la interfaz gráfica de usuario:

Tabla 6. HU Desinstalar el servicio antivirus.

Numero: HU_2	**Nombre del Requisito**: Desinstalar el servicio antivirus.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo Estimado:** 20
Riego en Desarrollo: Alto	**Tiempo Real**: 15
Descripción: El sistema debe permitir al usuario desinstalar el servicio de antivirus ClamAv. La funcionalidad comienza cuando el usuario accede al sistema y en el listado de módulos, presiona la opción Desinstalar en el módulo Antivirus para el correo.	
Observaciones: Se ejecuta en la PC servidora el comando ***apt purge –y clamav*.***	
Prototipo elemental de la interfaz gráfica de usuario:	

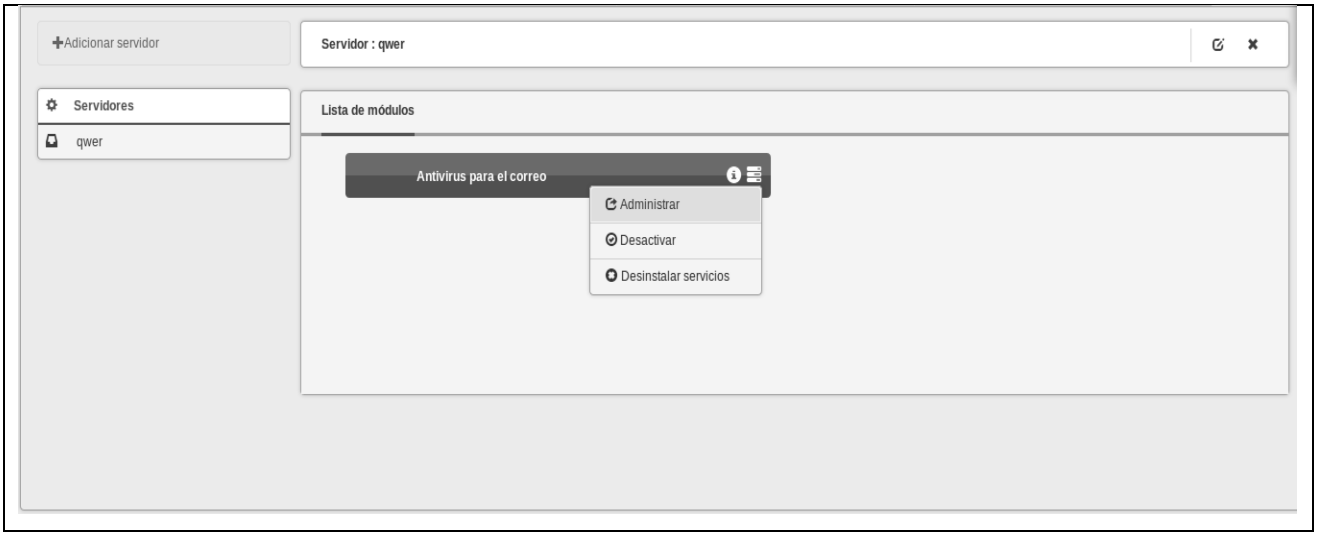


Tabla 7. HU Iniciar servicio.

Numero: HU_3	**Nombre del Requisito**: Iniciar servicio.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo Estimado:** 10
Riego en Desarrollo: Alto	**Tiempo Real**: 10
Descripción: El sistema debe permitir al usuario iniciar el servicio ClamAv. La funcionalidad comienza cuando el usuario accede al sistema y presiona el botón Iniciar.	
Observaciones: Se ejecuta en la PC servidora el comando ***service clamav-daemon start***.	
Prototipo elemental de la interfaz gráfica de usuario:	

Tabla 8. HU Detener servicio.

Numero: HU_4	**Nombre del Requisito**: Detener servicio.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo Estimado:** 10
Riego en Desarrollo: Alto	**Tiempo Real**: 10

Descripción: El sistema debe permitir al usuario detener el servicio ClamAv. La funcionalidad comienza cuando el usuario accede al sistema y presiona el botón Detener.
Observaciones: Se ejecuta en la PC servidora el comando ***service clamav-daemon stop***.
Prototipo elemental de la interfaz gráfica de usuario: 

Tabla 9. HU Reiniciar servicio.

Numero: HU_5	**Nombre del Requisito**: Reiniciar servicio.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo Estimado:** 10
Riego en Desarrollo: Alto	**Tiempo Real**: 10
Descripción: El sistema debe permitir al usuario reiniciar el servicio ClamAv. La funcionalidad comienza cuando el usuario accede al sistema y presiona el botón Reiniciar.	
Observaciones: Se ejecuta en la PC servidora el comando ***service clamav-daemon restart***.	
Prototipo elemental de la interfaz gráfica de usuario: 	

Tabla 10. HU Mostrar configuración avanzada.

Numero: HU_6	**Nombre del Requisito**: Mostrar configuración avanzada.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo Estimado:** 20
Riego en Desarrollo: Alto	**Tiempo Real**: 20
Descripción: El sistema debe permitir al usuario mostrar los parámetros avanzados del Antivirus. La funcionalidad comienza cuando el usuario accede al sistema y selecciona en el menú la opción Avanzada dentro de Configuración.	

El sistema debe mostrar una interfaz con los datos referentes a las configuraciones avanzadas del componente.

- Mostrar el estado del escaneo de correo: La aplicación permitirá mostrar el estado (habilitado o deshabilitado) del escaneo de los mensajes de correo mediante la directiva ScanMail.
- Mostrar el estado de los *logs* detallados: La aplicación permitirá mostrar el estado (habilitado o deshabilitado) de los *logs* detallados mediante la directiva LogVerbose.
- Mostrar el estado del escaneo de los archivos comprimidos: La aplicación permitirá mostrar el estado (habilitado o deshabilitado) del escaneo de los archivos comprimidos mediante la directiva ScanArchive.
- Mostrar el estado del escaneo de los archivos ejecutables: La aplicación permitirá mostrar el estado (habilitado o deshabilitado) del escaneo de los archivos ejecutables mediante la directiva ScanPE.
- Mostrar el estado del escaneo de los documentos HTML: La aplicación permitirá mostrar el estado (habilitado o deshabilitado) del escaneo de los documentos HTML mediante la directiva ScanHTML.
- Mostrar el estado del escaneo de los documentos PDF: La aplicación permitirá mostrar el estado (habilitado o deshabilitado) del escaneo de los documentos PDF mediante la directiva ScanPDF.
- Mostrar el estado del escaneo de los documentos de Microsoft Office y archivos msi: La aplicación permitirá mostrar el estado (habilitado o deshabilitado) del escaneo de los documentos de Microsoft Office y archivos msi mediante la directiva ScanOLE2.
- Mostrar el estado del escaneo de los documentos XMLDOCS: La aplicación permitirá mostrar el estado (habilitado o deshabilitado) del escaneo de los documentos XMLDOCS mediante la directiva ScanXMLDOCS.
- Mostrar el estado del escaneo de los documentos SWF: La aplicación permitirá mostrar el estado (habilitado o deshabilitado) del escaneo de los documentos SWF mediante la directiva ScanSWF.
- Mostrar el estado del escaneo de los documentos ELF: La aplicación permitirá mostrar el estado (habilitado o deshabilitado) del escaneo de los documentos ELF mediante la directiva ScanELF.

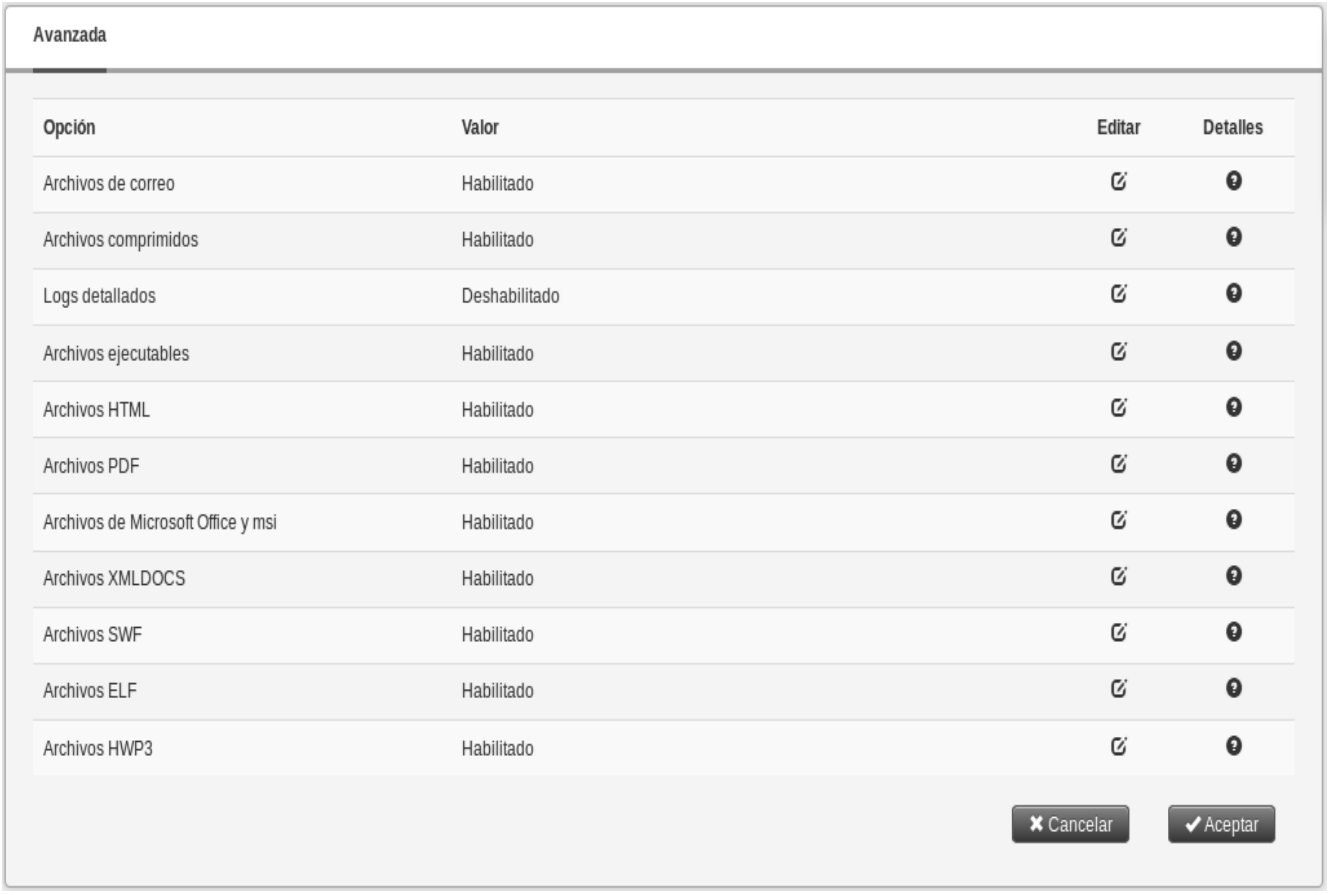
➢ Mostrar el estado del escaneo de los documentos HWP3: La aplicación permitirá mostrar el estado (habilitado o deshabilitado) del escaneo de los documentos HWP3 mediante la directiva ScanHWP3.
Observaciones: Estas directivas se encuentran en el fichero de *clamd.conf* ubicado en el directorio */etc/clamav/clamd.conf.*
Prototipo elemental de la interfaz gráfica de usuario: 

Tabla 11. HU Modificar configuración avanzada.

Numero: HU_7	**Nombre del Requisito**: Modificar configuración avanzada.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1

Prioridad: Media	**Tiempo Estimado:** 20
Riego en Desarrollo:	**Tiempo Real**: 20

Descripción: El sistema debe permitir al usuario modificar los parámetros avanzados del Antivirus.

La funcionalidad comienza cuando el usuario accede al sistema y selecciona en el menú la opción Avanzada dentro de Configuración y presionar el campo Editar (*checkbox*).

El sistema debe mostrar una interfaz con los datos referentes a las configuraciones avanzadas del componente.

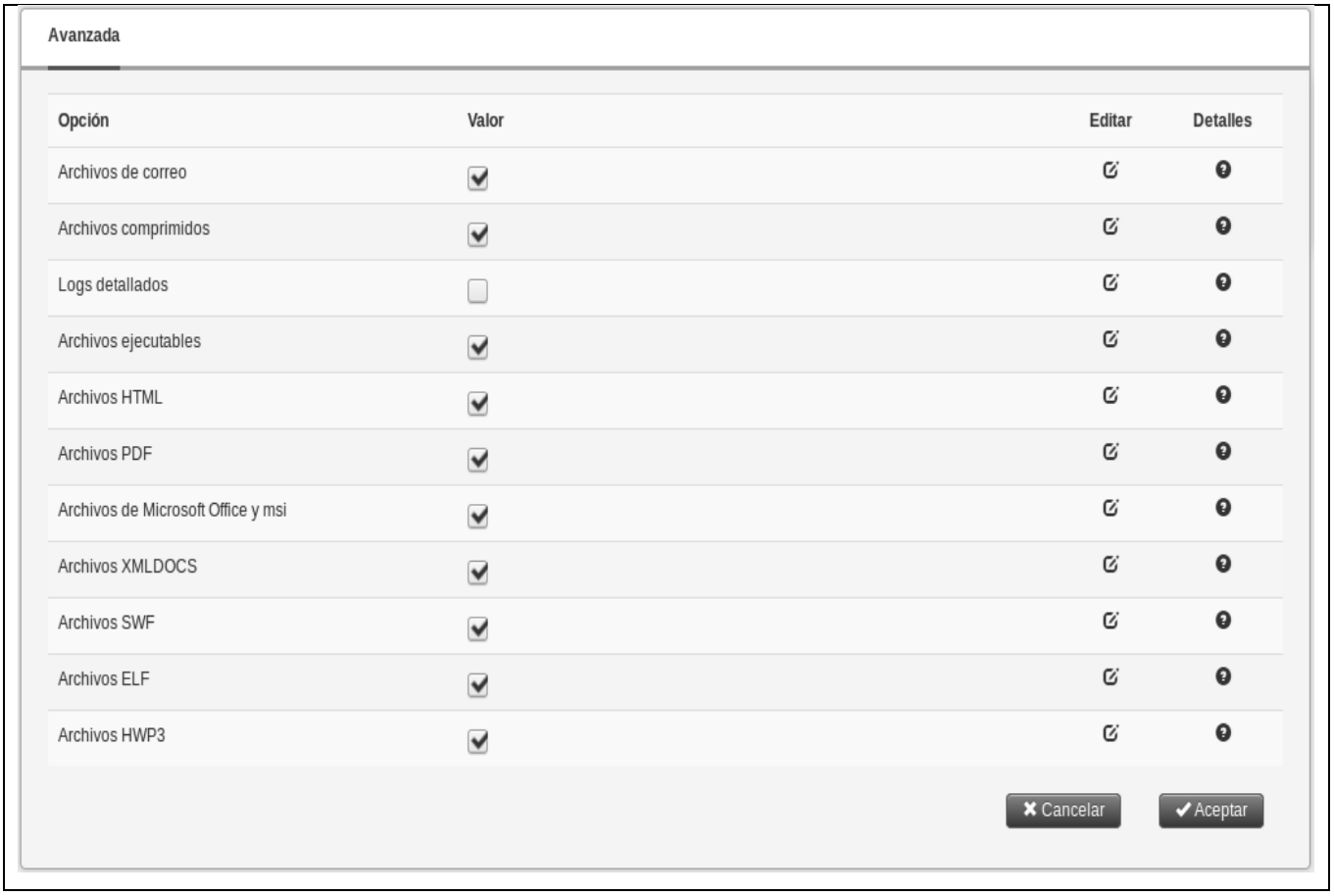
- Modificar el estado del escaneo de correo: La aplicación permitirá modificar el estado (habilitado o deshabilitado) del escaneo de los mensajes de correo mediante la directiva ScanMail.
- Modificar el estado de los *logs* detallados: La aplicación permitirá modificar el estado (habilitado o deshabilitado) de los *logs* detallados mediante la directiva LogVerbose.
- Modificar el estado del escaneo de los archivos comprimidos: La aplicación permitirá modificar el estado (habilitado o deshabilitado) del escaneo de los archivos comprimidos mediante la directiva ScanArchive.
- Modificar el estado del escaneo de los archivos ejecutables: La aplicación permitirá modificar el estado (habilitado o deshabilitado) del escaneo de los archivos ejecutables mediante la directiva ScanPE.
- Modificar el estado del escaneo de los documentos HTML: La aplicación permitirá modificar el estado (habilitado o deshabilitado) del escaneo de los documentos HTML mediante la directiva ScanHTML.
- Modificar el estado del escaneo de los documentos PDF: La aplicación permitirá modificar el estado (habilitado o deshabilitado) del escaneo de los documentos PDF mediante la directiva ScanPDF.
- Modificar el estado del escaneo de los documentos de Microsoft Office y archivos msi: La aplicación permitirá modificar el estado (habilitado o deshabilitado) del escaneo de los documentos de Microsoft Office y archivos msi mediante la directiva ScanOLE2.
- Modificar el estado del escaneo de los documentos XMLDOCS: La aplicación permitirá modificar el estado (habilitado o deshabilitado) del escaneo de los documentos XMLDOCS mediante la directiva ScanXMLDOCS.

- Modificar el estado del escaneo de los documentos SWF: La aplicación permitirá modificar el estado (habilitado o deshabilitado) del escaneo de los documentos SWF mediante la directiva ScanSWF.
- Modificar el estado del escaneo de los documentos ELF: La aplicación permitirá modificar el estado (habilitado o deshabilitado) del escaneo de los documentos ELF mediante la directiva ScanELF.
- Mostrar estado del escaneo de los documentos HWP3: La aplicación permitirá modificar el estado (habilitado o deshabilitado) del escaneo de los documentos HWP3 mediante la directiva ScanHWP3.

Además, contará con cuatro botones uno Aceptar para guardar localmente los cambios realizados, el botón Cancelar para cancelar las modificaciones que se han realizado, Aplicar para aplicar los cambios guardados localmente en el servidor y Descartar para descartar los cambios que fueron guardados localmente.

Observaciones: Todas estas directivas a modificar se encuentran en el fichero de *clamd.conf* ubicado en el directorio */etc/clamav/clamd.conf.*

Prototipo elemental de la interfaz gráfica de usuario:



2.3 Arquitectura del software

El diseño de la arquitectura decide cuál es la funcionalidad más importante a desarrollar. Define cuáles son los componentes más básicos del sistema y cómo se relacionan entre ellos para implementar la funcionalidad. Se determina como arquitectura del módulo la establecida para HMAST, como se muestra en la Figura 3, con el fin de lograr la consistencia con los componentes del sistema. Se emplea una arquitectura N-Capas orientada al Dominio, la cual tiene como objetivo estructurar de forma clara la complejidad de una aplicación empresarial basada en las diferentes capas de la arquitectura siguiendo el patrón N-Capas y las tendencias de arquitecturas orientada al dominio (29). El módulo de ClamAv tiene sus funcionalidades representadas en cada una de las capas de dicha arquitectura y forma parte del módulo de correo como una extensión del mismo.

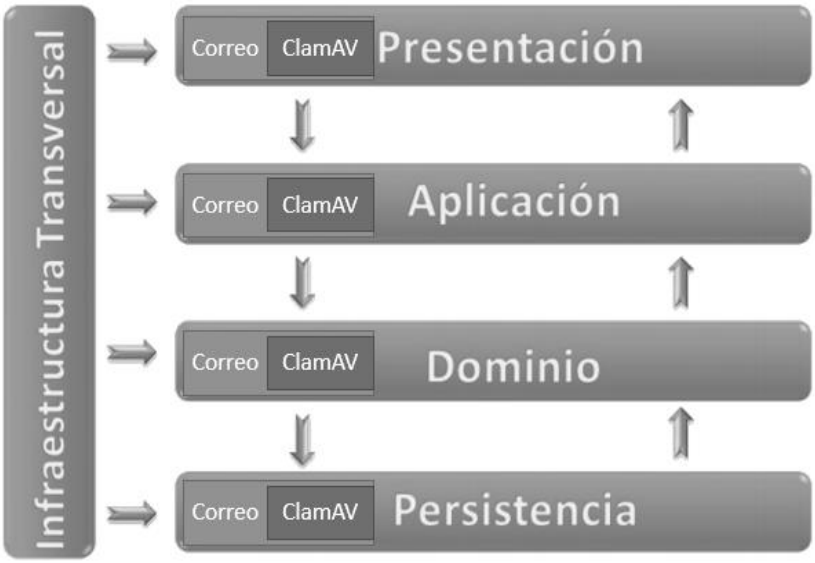


Figura 4. Arquitectura del módulo.

2.4 Diagrama de Paquetes

El diagrama de paquetes muestra las agrupaciones lógicas en que está dividido el sistema, así como las dependencias entre dichas agrupaciones. Se diseñó el diagrama de paquetes tomando como marco de referencia la arquitectura propuesta; el módulo ClamAv está representado en cada uno de los paquetes y contiene todas las clases pertenecientes al módulo.

El diagrama de la Figura 4 representa la distribución de los paquetes y las relaciones entre ellos con las diferentes capas de la arquitectura del módulo, este proceso se describe a continuación:

En la capa de Presentación está el paquete *clamavmail* que contiene la clase controladora *ClamavMail*, que se encarga de implementar los métodos para el envío y recepción de datos para cada una de las vistas del módulo. En la capa de Aplicación, se encuentra el paquete *clamavmail* que contiene la clase *IClamavMailAppService* donde se definen los métodos que se llamarán desde la capa de Presentación y la clase *ClamavMailAppService* que es la encargada de la implementación de los métodos de la interfaz mencionada anteriormente. En *clamavmail* se encuentra también el paquete *dto*, que contiene los DTO, los cuales son enviados desde y hacia la capa de Presentación, que depende de la interfaz

IClamavMailAppService. La clase encargada de convertir los DTO en entidades y viceversa es *ClamavMailAppService* en la capa de Aplicación.

La capa de Dominio se relaciona con la capa de Aplicación a través de una relación de agregación con la clase *IClamavMailAppService* mediante la inyección de dependencias proporcionada por el framework Spring. Dentro de esta capa se encuentra el paquete *clamavmail* que contiene los paquetes *service*, *repository* y *entities.* En *service* se define la interfaz *IClamavMailService* y la clase *NginxService* que es donde se implementan los métodos definidos en dicha interfaz, además, tiene una relación de agregación con la clase *ClamavMailService* mediante la inyección de dependencias. En *repository* se definen los contratos de repositorio en la interfaz *IClamavMailRepository*. En *entities* se definen las entidades del módulo, que representan los objetos del dominio.

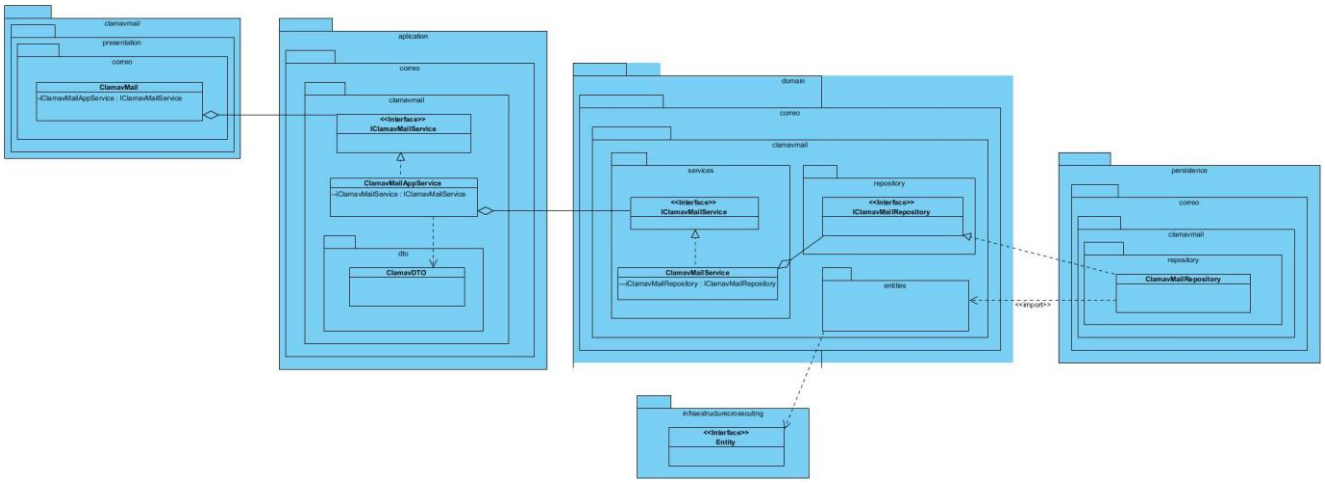


Figura 5. Diagrama de paquetes del módulo.

2.4.1 Diagrama de clases de las entidades.

La Figura 6 representa el diagrama de clases de las entidades pertenecientes al paquete *entities* de la capa de Dominio, donde se encuentran las entidades siguientes: Clamav, AdvancedConfiguration, VirusConfiguration, BasicConfiguration, UpdateConfiguration.

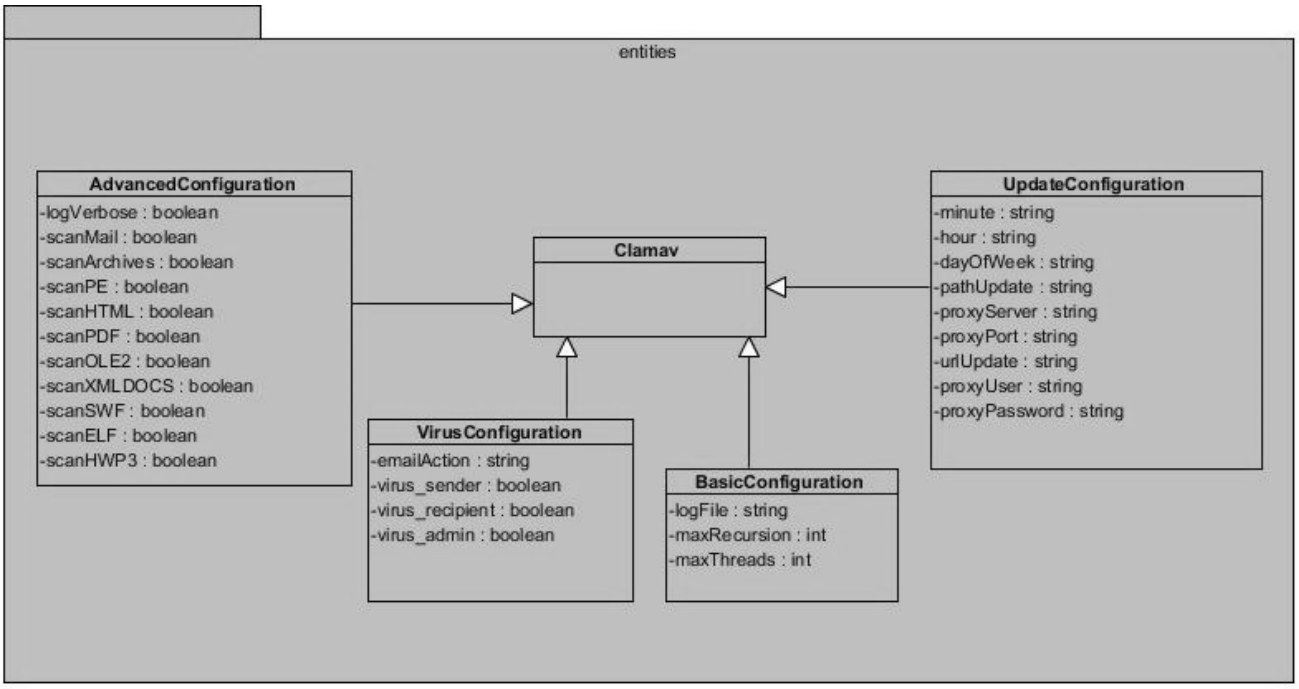


Figura 6. Diagrama de clases de las entidades.

2.5 Patrones de Diseño

Los patrones de diseño son el soporte de las soluciones a problemas comunes que surgen en el desarrollo de software. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas. En el modelo de diseño del módulo se aplican patrones de asignación de responsabilidad con el nombre de General Responsibility Assignment *Software Patterns* (GRASP) y los patrones conocidos con el nombre de *The Gang of Four* (GOF) (30).

2.5.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales para la asignación de responsabilidades a objetos, expresado en forma de patrones. A continuación se describen los patrones que son utilizados para el diseño de la solución propuesta (30).

- **Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos. Se utiliza en la capa de Aplicación en la clase *ClamavMailAppService* al realizarse la conversión de

clases de tipo DTO a entidades, pues se crean instancias de las entidades debido a que las clases DTO contienen los datos de inicialización de estas. Además, se utiliza en la conversión de entidades a DTO.

- **Experto:** Se utiliza para la asignación de responsabilidades relacionadas con la obtención de información. Este patrón se manifiesta en todas las capas, pues cada una tiene un objetivo y una responsabilidad asignada, siendo cada capa la encargada de implementarla.
- **Bajo Acoplamiento:** Consiste en asignar responsabilidades de modo que el acoplamiento permanezca bajo. El mecanismo de inyección de dependencias proporcionado por el *framework* Spring, permite el uso de este patrón. Además, permite la reutilización de las clases y que no sean afectadas por cambios que se realicen en otros componentes. Este patrón se emplea en las distintas capas del módulo mediante el uso de interfaces que relacionan una capa con otra de forma que dichas relaciones no se establezcan directamente hacia las clases. Las conexiones se realizan a través del mecanismo de inyección de dependencias. Esto se evidencia en el paquete *repositorycontrats*, donde se define la interfaz *IClamavMailRepository*, y la implementación de sus métodos es realizada por la clase *ClamavMailRepository* en la capa de Persistencia.
- **Alta Cohesión:** La cohesión es una medida del grado de focalización de las responsabilidades de una clase. Permite que las clases sean fáciles de entender, mantener y reutilizar. Se manifiesta en las validaciones en cada una de las entidades específicamente en el método *Validate*.

2.5.2 Patrones GOF

Los patrones GoF se pueden categorizar en tres grupos teniendo en cuenta 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): Es un patrón de tipo creacional con el objetivo de garantizar la existencia de una única instancia para una clase y posibilitar el acceso global a dicha instancia. Se usa debido a la necesidad de trabajar con el mismo objeto en distintos momentos. Su empleo se evidencia cuando se realiza una conexión SSH a un servidor, pues con una única instancia del objeto conexión se realizan las operaciones sobre este.

Conclusiones parciales

El estudio realizado permitió identificar las características y funcionalidades del componente propuesto, obteniendo un total de 6 requisitos no funcionales y 19 funcionales, llegando a un acuerdo sobre las capacidades y cualidades con las que el software debe cumplir. Estos últimos fueron encapsulados en historias de usuario. Se asumió como arquitectura la N-Capas orientada al Dominio de HMAST, lo cual permitió mantener la homogeneidad del módulo con la herramienta base. Además, la correcta definición de los patrones de arquitectura y los de diseño, permitieron que se obtuviera una base del sistema capaz de soportar posteriores cambios en los requisitos. El estudio realizado en este capítulo ha facilitado un entendimiento de la dinámica y estructura de la aplicación, contribuyendo a una mejor comprensión del problema que el módulo a desarrollar debe resolver. Se sentaron las bases para las restantes fases del proceso.

Capítulo 3: Implementación y prueba.

Las pruebas de software implican la realización de actividades que conllevan a identificar y documentar posibles fallos durante y después del proceso de desarrollo del software. En el presente capítulo se expone el estándar de codificación empleado para el desarrollo del componente, con el objetivo de mantener la uniformidad con la codificación de los restantes módulos integrados a la herramienta. Se establece la estrategia de pruebas para verificar la calidad del módulo implementado y se documenta la realización de las pruebas seleccionadas.

3.1 Estándar de codificación

Los estándares de codificación mejoran la legibilidad del código, facilitan el mantenimiento dado que permite que cualquier programador tenga una comprensión rápida y pueda dar mantenimiento la aplicación (31). En el componente se emplea el estándar de codificación de HMAST, así como las pautas definidas en el mismo.

- **Asignación de nombres.** Emplear descriptores en inglés. Evitar nombres largos y que difieran en una letra o en el uso de mayúsculas. Para nombrar las funciones y variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se utiliza la notación CamelCase[14].
- **Ficheros de código fuente:** Cada fichero contiene una única clase o interfaz. Si hay una clase privada o una interfaz asociada a una clase pública se puede poner en el mismo fichero. La clase pública debe ser la primera.
- **Indentación:** La unidad de identación de bloques de sentencias son 4 espacios.
- **Comentarios:** Los comentarios deben añadir claridad al código. Deben contar el por qué y no el cómo. Deben ser concisos.
- **Declaraciones:** Se debe declarar cada variable en una línea distinta, de esta forma cada variable

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

se puede comentar por separado.

- **Continuidad de las líneas largas:** Cuando una sentencia no quepa en una única línea se debe fraccionar después de una coma, después de un operador y alinear la nueva línea con el comienzo de la expresión al mismo nivel de la línea anterior.
- **Longitud de la línea:** Limitar todas las líneas a un máximo de 120 caracteres.
- **Nombres de componentes:** Todos los paquetes comienzan con cu.uci.hmast.xxx.yyy.zzz.kkk
  - xxx →presentation, application, domain, persistence.
  - yyy →nombre del módulo (cron).
  - zzz→elementos que pueden contener los componentes verticales (entitys, repositorys).
  - kkk → clases o subpaquetes.

3.2 Estrategia de pruebas.

La estrategia de pruebas de software que se emplea propone pruebas de alto y bajo nivel que incluyen todos los requisitos del componente implementado. Estas pruebas tienen un enfoque incremental que incluyen las pruebas de aceptación, integración y de sistema, las cuales son descritas en la Figura 7.

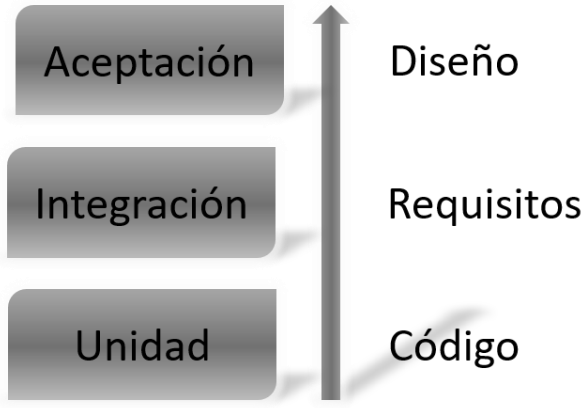


Figura 7. Estrategia de pruebas.

3.2.1 Prueba de unidad

Las pruebas de unidad son un conjunto de procedimientos de pruebas locales cuyo objetivo es comprobar el correcto funcionamiento de los métodos implementados en cada una de las clases que de conjunto integran el componente. Para acometer dicha actividad se ha realizado la prueba de camino básico cuyo objetivo es detectar durante su ejecución que cada sentencia del programa se ejecute por lo menos una vez, garantizado por el caso de prueba que se diseña previamente.

Tabla 12. Método prueba de camino básico: Código fuente etiquetado.

1	public Clamav loadServiceConfiguration(LogicalServer server) throws InterruptedException,ENotFoundUrlRemoto,NoSuchAlgorithmException, IOException, JSchException, EInvalidEntity, SftpException {
2	IdentityGenerator id = new IdentityGenerator();
3	UUID id_ = id.getNewUUID();
4	String baseDir = this.createRootFolderConfigurationModuleIfNotExist(server).getPath();
5	this.createPrincipalFileConfigIfNotExist(server);
6	String fileConfig=baseDir+serversAdministrationService.getConfigurationValueByKey (server.getId(), canonicalName, ”rootFolderConfiguration”)+ serversAdministrationService. getConfigurationValueByKey(server.getId(),canonicalName,”nameFilePrincipalConfig”);
7	File fileConfig_ = new File(fileConfig);
8	boolean logVerbose = false;
9	boolean scanMail = false;
10	boolean scanArchive = false;
11	if (fileConfig_.exists()
12	&& fileConfig_.isFile()) {
13	FileReader file = new FileReader(fileConfig_);
14	BufferedReader br = new BufferedReader(file);
15	LinkedList<String> lineList = new LinkedList<>();
16	String linea;
17	while ((linea = br.readLine()) != null) {

```
18              lineList.add(linea);
19          }
20  for (int i = 0; i < lineList.size(); i++) {
21      String line = lineList.get(i);
22      if (line.contains("LogVerbose")) {
23          String logVerbose_ = line.split(" ")[1];
24          if (logVerbose_ != null
25          && logVerbose_.toString().equals("true")) {
26           logVerbose=true;
27          }
28      }
29      if (line.contains("ScanMail")) {
30           String ScanMail = line.split(" ")[1];
31           if (ScanMail != null
32           && ScanMail.toString().equals("true")) {
33               scanMail=true;
34           }
35      }
36      if (line.contains("ScanArchive")) {
37           String ScanArchive = line.split(" ")[1];
38           if (ScanArchive != null
39           && ScanArchive.toString().equals("true")) {
40               scanArchive = true;
41           }
42      }
43  }
44  }
```

45	AdvancedConfiguration advancedConfiguration = new AdvancedConfiguration(id_, logVerbose, scanMail, scanArchive, scanPE, scanHTML, scanPDF, scanOLE2, scanXMLDOCS, scanSWF, scanELF,scanHWP3);
46	BasicConfiguration basicConfiguration = new BasicConfiguration(logFile, maxRecursion, maxThreads);
47	Clamav clamav = new Clamav(advancedConfiguration, basicConfiguration, null);
48	return clamav;
49	}

Luego de haber etiquetado el código fuente se procede a representarlo en un grafo de flujo, ver Figura 8 cada nodo del grafo corresponde a una o más sentencias de código.

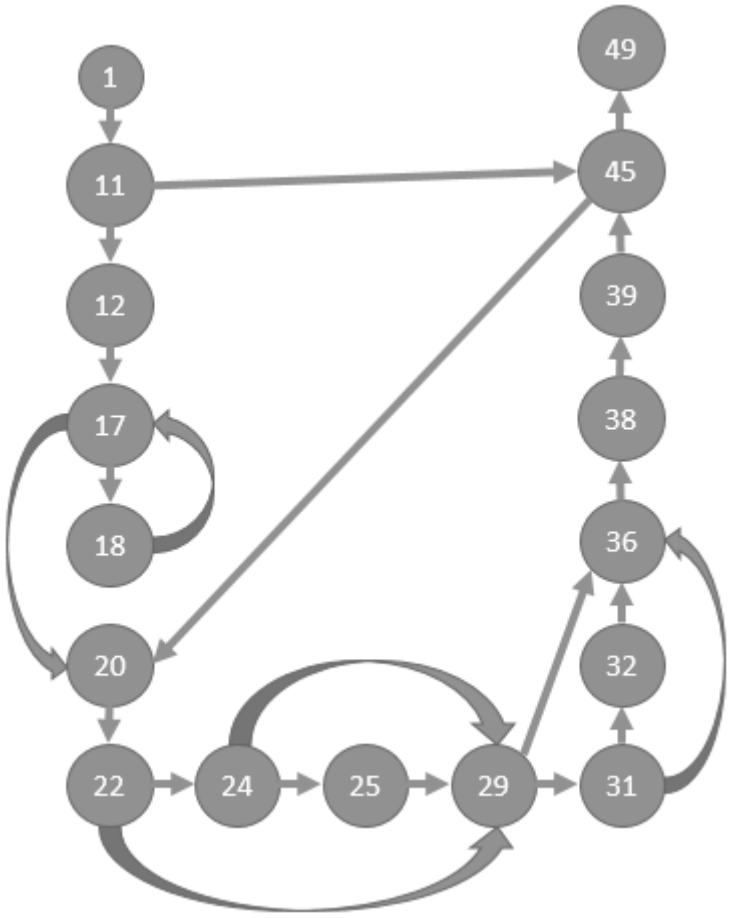


Figura 8. Prueba de camino básico: Grafo de flujo.

Una vez obtenido el grafo se procede a calcular la complejidad ciclomática **V(G)**; la complejidad ciclomática indica el número máximo de caminos independientes. Para calcular la misma se hace uso de la siguiente fórmula:

V(G) = A – N + 2, donde A es el número de aristas y N es el número de nodos.

V(G) = 23 – 17 + 2 = 8.

Otra fórmula para calcular la complejidad ciclomática es:

V(G) = P + 1, donde P es el número de nodos predicados, se denomina nodo predicado a cada nodo que representa una condición.

V(G) = 7 + 1 = 8.

Luego se determina el conjunto de caminos independientes del grafo, se detectaron 8 caminos, cualquier otro camino no será un camino independiente ya que simplemente es una combinación de caminos ya especificados.

- Camino 1: 1-11-45-49.
- Camino 2: 1-11-12-17-20-22-24-25-29-31-32-36-38-39-45-49.
- Camino 3: 1-11-12-17-18-17-20-22-24-25-29-31-32-36-38-39-45-49.
- Camino 4: 1-11-12-17-18-17-20-22-29-31-32-36-38-39-45-49.
- Camino 5: 1-11-12-17-18-17-20-22-24-25-29-36-38-39-45-49.
- Camino 6: 1-11-12-17-18-17-20-22-24-25-29-31-36-38-39-45-49.
- Camino 7: 1-11-12-17-18-17-20-22-24-25-29-31-32-36-38-39-45-20-22-24-25-29-31-32-36-38-39-45-49.
- Camino 8: 1-11-12-17-18-17-20-22-24-25-29-31-32-36-38-39-45-49.

3.2.3 Prueba de integración

Las pruebas de integración describen el comportamiento del componente analizado previamente por las pruebas unitarias, enfocándose en la integración entre las unidades involucradas (clases) en dicho proceso. Asumiendo la arquitectura por capas empleada y que el módulo debe integrarse a una herramienta base, se emplea una estrategia de integración ascendente, donde empieza la construcción y las pruebas a los componentes de los niveles más bajos de la estructura del programa hacia arriba.

En el contexto de una estrategia de prueba de integración ascendente, se realiza la prueba de regresión, que permite volver a ejecutar un subconjunto de pruebas que se han llevado a cabo anteriormente para asegurarse de que los cambios no han propagado efectos colaterales no deseados (32).

3.2.4 Prueba de aceptación.

Las pruebas de aceptación buscan asegurar el correcto funcionamiento del software y que los requisitos satisfagan la necesidad del usuario con la calidad esperada. Las mismas se realizan mediante un proceso minucioso de preguntas que den cumplimiento al objetivo propuesto. A continuación, se muestran algunos casos de prueba de aceptación, el resto se encuentran en los anexos.

Tabla 13. Caso de prueba 1

Caso de prueba de aceptación	
Código de caso de prueba: HU_1	**Nombre de la HU:** Instalar el servicio de antivirus.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Probar que el servicio antivirus se instala.	
Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe adicionar un servidor lógico y activar el módulo Antivirus para el correo.	
Entrada/Pasos de ejecución: 1. Dar clic en el botón 'Instalar ClamAv'.	
Resultado esperado: 1. Se instala el servicio antivirus.	
Evaluación de la prueba: *Satisfactoria.*	

Tabla 14. Caso de prueba 2

Caso de prueba de aceptación	
Código de caso de prueba: HU_3	**Nombre de la HU:** Iniciar servicio.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Probar que el servicio antivirus se puede iniciar.	

Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe estar conectado a un servidor lógico y activar el módulo. El servicio antivirus debe estar instalado.
Entrada/Pasos de ejecución: 1. Dar clic en el botón ▸.
Resultado esperado: 1. El servicio se inicia correctamente. 2. El estado se actualiza.
Evaluación de la prueba: *Satisfactoria*.

Tabla 15. Caso de prueba 3

Caso de prueba de aceptación	
Código de caso de prueba: HU_4	**Nombre de la HU:** Detener servicio.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Probar que el servicio antivirus se detiene.	
Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe estar conectado a un servidor lógico y activar el módulo. El servicio antivirus debe estar instalado e iniciado.	
Entrada/Pasos de ejecución: 1. Dar clic en el botón ■.	
Resultado esperado: 1. El servicio se detiene correctamente. 2. El estado se actualiza.	
Evaluación de la prueba: *Satisfactoria*.	

Tabla 16. Caso de prueba 4

Caso de prueba de aceptación	
Código de caso de prueba: HU_5	**Nombre de la HU:** Reiniciar servicio.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Probar que el servicio antivirus se reinicia.	

Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe estar conectado a un servidor lógico y activar el módulo. El servicio antivirus debe estar instalado e iniciado.
Entrada/Pasos de ejecución: 1. Dar clic en el botón ♋.
Resultado esperado: 1. El servicio se detiene correctamente. 2. El estado se actualiza. 3. Se cargan los nuevos valores de configuración.
Evaluación de la prueba: *Satisfactoria*.

Los resultados obtenidos para cada uno de los casos de prueba efectuados fueron satisfactorios, propiciando que, al finalizar estas pruebas, el cliente fue capaz de verificar el cumplimiento de los objetivos planteados.

Como resultado a las pruebas realizadas al componente se realizaron 4 iteraciones. La primera iteración arrojó 16 no conformidades de las cuales 9 fueron resueltas; en una segunda se arrojaron 7 no conformidades sin resolver, de las cuales se resolvieron 5; en la tercera iteración quedaron 2 no conformidades las cuales fueron resueltas en su totalidad. En la iteración número 4 no se detectaron no conformidades.

Conclusiones Parciales

El uso del estándar de codificación definido para HMAST permitió desarrollar un código reutilizable y fácil de entender. La selección de una estrategia de pruebas con un enfoque incremental propició comprobar el módulo en cada una de sus partes y las relaciones entre ellas. Con la realización de las pruebas se verificó que todas las instrucciones del módulo se ejecutan al menos una vez, que los componentes se integran correctamente y se validó que el módulo se ajusta al sistema y cumple con las exigencias del cliente.

Conclusiones Generales

El desarrollo del presente trabajo de diploma permitió desarrollar un componente para la administración del servicio antivirus desde el módulo de correo de HMAST, dando cumplimiento a los objetivos trazados, destacándose de manera general los siguientes aspectos:

- El estudio de la infraestructura módulo de correo electrónico de HMAST permitió identificar las tecnologías que integra, así como la comprensión de la arquitectura del mismo.
- El estudio de antivirus existentes permitió identificar a ClamAv como la solución más adecuada para su administración desde el módulo de correo de HMAST.
- La correcta utilización de las herramientas, lenguajes y tecnologías descritas, hicieron posible la obtención de un diseño e implementación acertados para el módulo desarrollado, proceso que estuvo guiado por la metodología AUP-UCI.
- El establecimiento de AUP-UCI como metodología de desarrollo, así el uso de las tecnologías y herramientas seleccionadas, permitieron analizar, describir e implementar el componente, materializando así una solución acorde a los requerimientos del cliente.
- La implementación del componente, luego de ser validada a partir de las pruebas definidas permitió obtener un componente que una vez integrado al sistema base, está apto para ser usado en un entorno real de producción.

Recomendaciones

Se recomienda para próximas versiones de la solución:

- Mostrar estadísticas de la cantidad de mensajes escaneados, así como datos de virus encontrados.
- Mostrar detalles de la última actualización realizada, así como la fecha y estado de su ejecución.
- Extender el uso de otros antivirus en la infraestructura de correo.

Anexo

Tabla 17. HU Mostrar configuración básica.

Numero: HU_8	**Nombre del Requisito**: Mostrar configuración básica.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo Estimado:** 20
Riego en Desarrollo: Alto	**Tiempo Real**: 20
Descripción: El sistema debe permitir al usuario mostrar los parámetros básicos del Antivirus. La funcionalidad comienza cuando el usuario accede al sistema y selecciona en el menú lateral la opción Básica dentro de Configuración. El sistema debe mostrar una interfaz con los datos referentes a las configuraciones básicas del Antivirus. ➢ Mostrar dirección donde serán almacenados los *logs*: La aplicación permitirá mostrar la ruta donde serán almacenados los *logs* en el pc servidor mediante la directiva LogFile. ➢ Mostrar la máxima profundidad a escanear: La aplicación permitirá mostrar la máxima profundidad a escanear dentro de los archivos adjuntos mediante la directiva MaxRecursion. ➢ Mostrar el máximo número de subprocesos: La aplicación permitirá mostrar el máximo número de subprocesos que podrá utilizar ClamAv mediante la directiva MaxThreads.	
Observaciones: Estas directivas se encuentran en el fichero de *clamd.conf* ubicado en el directorio */etc/clamav/clamd.conf.*	
Prototipo elemental de la interfaz gráfica de usuario:	

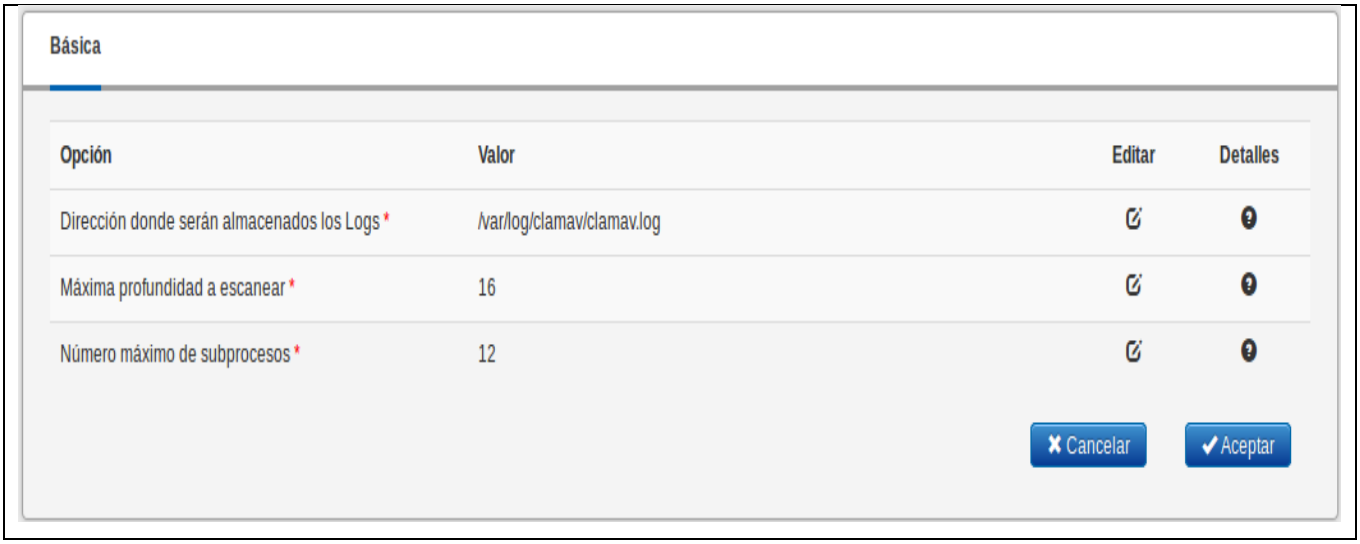


Tabla 18. HU Editar configuración básica.

Numero: HU_9	**Nombre del Requisito**: Editar configuración básica.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo Estimado:** 20
Riego en Desarrollo: Alto	**Tiempo Real**: 20

Descripción: El sistema debe permitir al usuario editar los parámetros básicos del Antivirus. La funcionalidad comienza cuando el usuario accede al sistema y selecciona en el menú lateral la opción Básica dentro de Configuración, y da clic en la opción editar.

El sistema debe mostrar una interfaz con los datos referentes a las configuraciones básicas del Antivirus, y permitir la edición de los mismos:

- Modificar dirección donde serán almacenados los *logs*: La aplicación permitirá modificar la ruta donde serán almacenados los *logs* en el pc servidor mediante un *fileChooser* modificando la directiva LogFile.
- Modificar la máxima profundidad a escanear: La aplicación permitirá modificar la máxima profundidad a escanear dentro de los archivos adjuntos mediante un *spinner* obteniendo como el mayor número a introducir el 20 y el mínimo 1, modificando la directiva MaxRecursion.

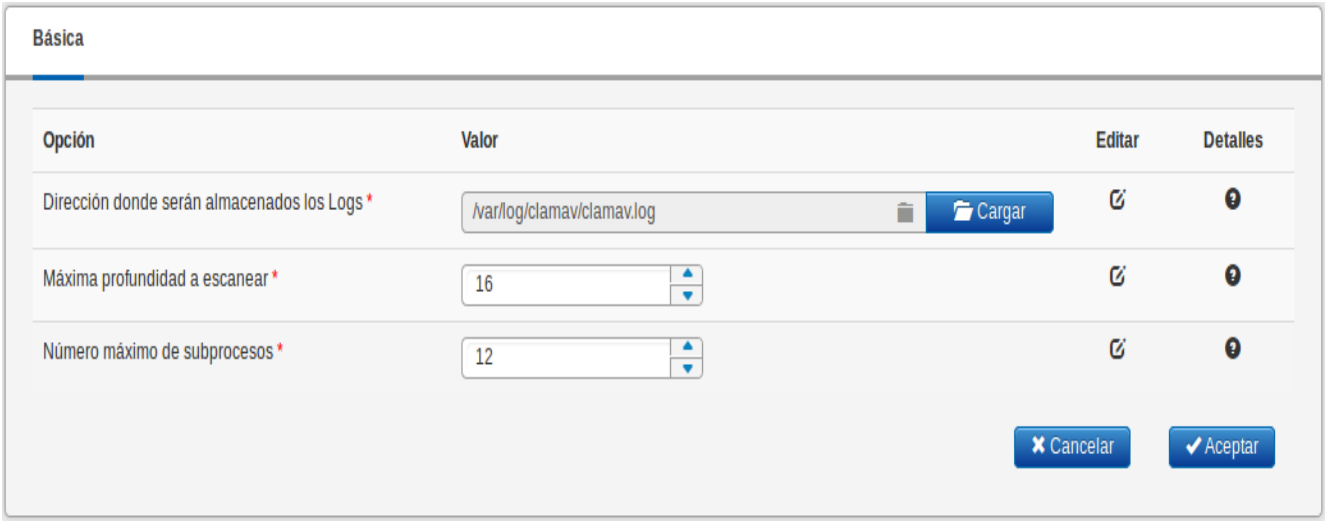
➢ Modificar el máximo número de subprocesos: La aplicación permitirá modificar el máximo número de subprocesos que podrá utilizar ClamAv mediante un *spinner* obteniendo como el mayor número a introducir el 20 y el mínimo 1, modificando la directiva MaxThreads.
Observaciones: Estas directivas se encuentran en el fichero de *clamd.conf* ubicado en el directorio */etc/clamav/clamd.conf.*
Prototipo elemental de la interfaz gráfica de usuario: 

Tabla 19. HU Mostrar configuración de actualización.

Numero: HU_10	**Nombre del Requisito**: Mostrar configuración de actualización.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo Estimado:** 20
Riego en Desarrollo: Alto	**Tiempo Real**: 20
Descripción: El sistema debe permitir al usuario mostrar los parámetros de actualización del Antivirus. La funcionalidad comienza cuando el usuario accede al sistema y selecciona en el menú lateral la opción Parámetros de actualización dentro de Configurar actualización. El sistema debe mostrar una interfaz con los datos referentes a las configuraciones de actualización del Antivirus.	

- Mostrar dirección donde serán almacenados las actualizaciones: La aplicación permitirá mostrar la ruta donde serán almacenados las actualizaciones en el pc servidor mediante la directiva databaseDirectory.
- Mostrar origen desde donde serán descargadas las actualizaciones: La aplicación permitirá mostrar la ruta desde donde serán descargadas las actualizaciones mediante la directiva URLMAIN.
- Mostrar servidor proxy el cual será utilizado para descargar las actualizaciones: La aplicación permitirá mostrar servidor proxy el cual será utilizado para descargar las actualizaciones mediante la directiva PROXY_SERVER.
- Mostrar puerto del servidor proxy el cual será utilizado para descargar las actualizaciones: La aplicación permitirá mostrar el puerto del servidor proxy el cual será utilizado para descargar las actualizaciones mediante la directiva PROXY_PORT.
- Mostrar usuario del servidor proxy el cual será utilizado para descargar las actualizaciones: La aplicación permitirá mostrar el usuario del servidor proxy el cual será utilizado para descargar las actualizaciones mediante la directiva PROXY_USER.
- Mostrar contraseña utilizada por el servidor proxy la cual será utilizada para descargar las actualizaciones: La aplicación permitirá mostrar la contraseña utilizada por el servidor proxy la cual será utilizada para descargar las actualizaciones mediante la directiva PROXY_PASWORD.

Observaciones: Estas directivas se encuentran en el fichero de *update.sh* ubicado en el directorio */etc/clamav/update.sh.*

Prototipo elemental de la interfaz gráfica de usuario:

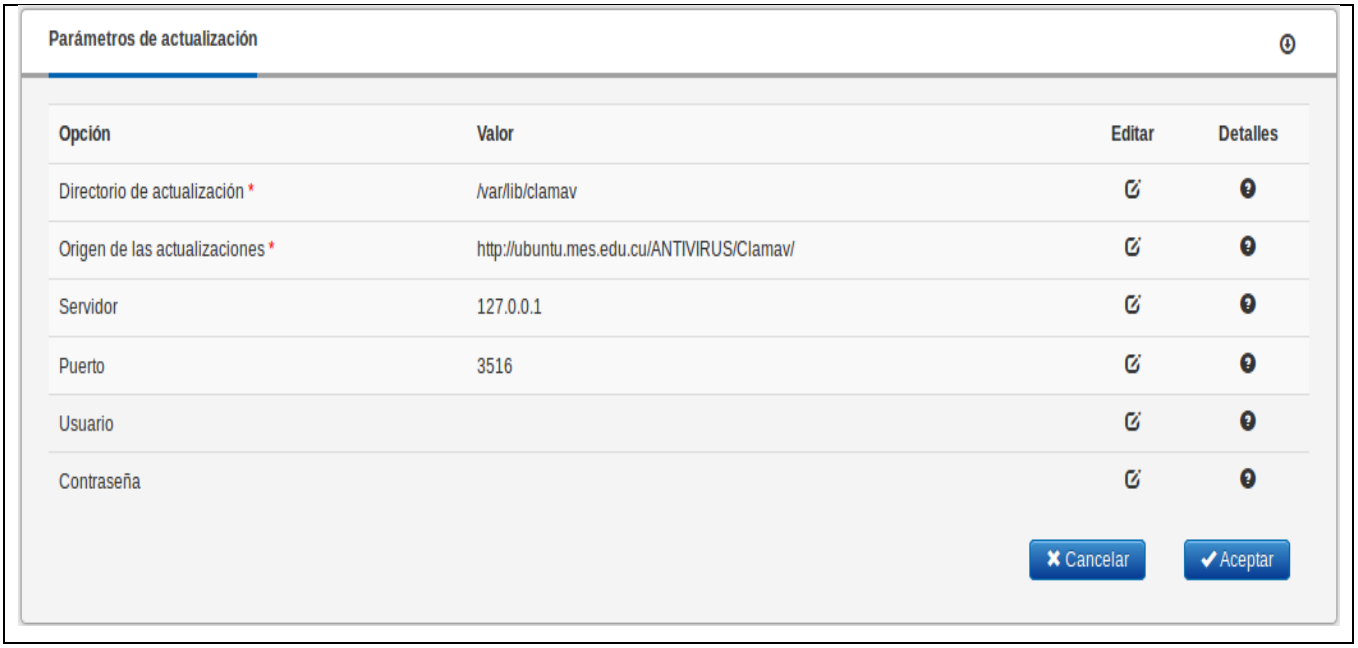


Tabla 20. HU Modificar configuración de actualización.

Numero: HU_11	**Nombre del Requisito**: Modificar configuración de actualización.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo Estimado:** 20
Riego en Desarrollo: Alto	**Tiempo Real**: 20

Descripción: El sistema debe permitir al usuario modificar los parámetros de actualización del Antivirus. La funcionalidad comienza cuando el usuario accede al sistema y selecciona en el menú lateral la opción Parámetros de actualización dentro de Configurar actualización y presionar el campo Editar.

El sistema debe mostrar una interfaz con los datos referentes a las configuraciones de actualización del Antivirus.

- Modificar dirección donde serán almacenados las actualizaciones: La aplicación permitirá modificar la ruta donde serán almacenados las actualizaciones en el pc servidor mediante un *fileChooser* modificando la directiva databaseDirectory.

- Modificar origen desde donde serán descargadas las actualizaciones: La aplicación permitirá modificar la ruta desde donde serán descargadas las actualizaciones mediante un *input* modificando la directiva URLMAIN, solo permite direcciones URL.
- Modificar servidor proxy el cual será utilizado para descargar las actualizaciones: La aplicación permitirá modificar servidor proxy el cual será utilizado para descargar las actualizaciones mediante un *input* modificando la directiva PROXY_SERVER, en caso de que el administrador no desee utilizar ninguno dejar este campo en blanco, solo permite IP en su versión ipv4 o nombre del servidor.
- Modificar puerto del servidor proxy el cual será utilizado para descargar las actualizaciones: La aplicación permitirá modificar el puerto del servidor proxy el cual será utilizado para descargar las actualizaciones mediante un *spiner* modificando la directiva PROXY_PORT.
- Modificar usuario del servidor proxy el cual será utilizado para descargar las actualizaciones: La aplicación permitirá modificar el usuario del servidor proxy el cual será utilizado para descargar las actualizaciones mediante un *input* modificando la directiva PROXY_USER, en caso de que el administrador no desee utilizar ninguno dejar este campo en blanco.
- Modificar contraseña utilizada por el servidor proxy la cual será utilizada para descargar las actualizaciones: La aplicación permitirá modificar la contraseña utilizada por el servidor proxy la cual será utilizada para descargar las actualizaciones mediante un *input* modificando la directiva PROXY_PASWORD.

Observaciones: Estas directivas se encuentran en el fichero de *update.sh* ubicado en el directorio */etc/clamav/update.sh.*

Prototipo elemental de la interfaz gráfica de usuario:

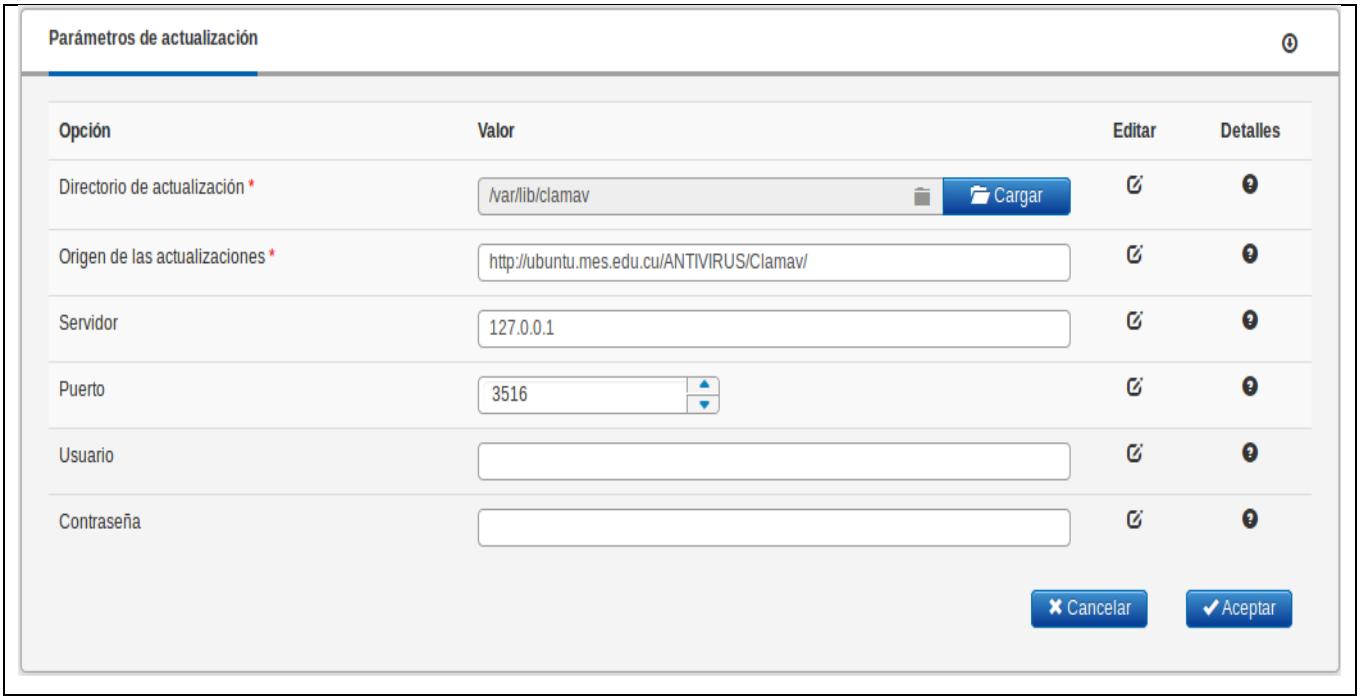


Tabla 21. HU Mostrar configuración de actualización automática.

Numero: HU_12	**Nombre del Requisito**: Mostrar configuración de actualización automática.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo Estimado:** 20
Riego en Desarrollo: Alto	**Tiempo Real**: 20

Descripción: El sistema debe permitir al usuario mostrar los parámetros de actualización automática del Antivirus. La funcionalidad comienza cuando el usuario accede al sistema y selecciona en el menú lateral la opción Actualización automática dentro de Configurar actualización.

El sistema debe mostrar una interfaz con los datos referentes a las configuraciones actualización automática del Antivirus.

- Mostrar hora en la cual se realizará la actualización automática: La aplicación permitirá mostrar la hora elegida para la actualización automática.
- Mostrar minuto en el cual se realizará la actualización automática: La aplicación permitirá mostrar el minuto elegido para la actualización automática.

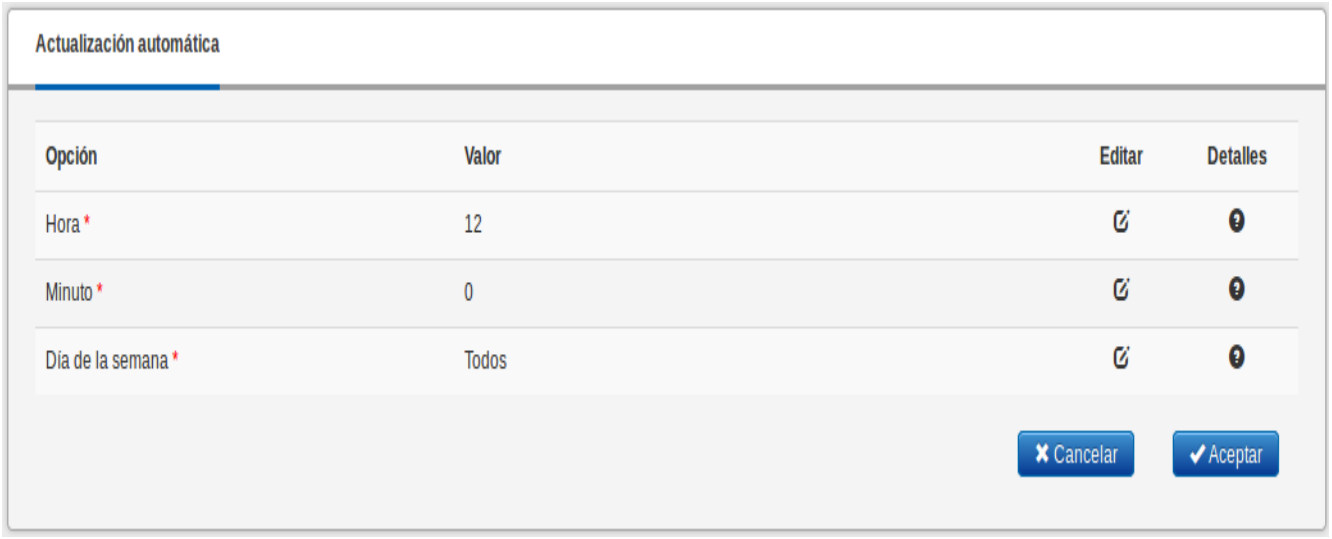
➢ Mostrar el día de la semana elegido para realizar actualización automática: La aplicación permitirá mostrar el día de la semana elegido para realizar actualización automática.
Observaciones: Estas directivas se encuentran en el fichero de *update.sh* ubicado en el directorio */etc/cron.d/clamav.*
Prototipo elemental de la interfaz gráfica de usuario: 

Tabla 22. HU Modificar configuración de actualización automática.

Numero: HU_13	**Nombre del Requisito**: Modificar configuración de actualización automática.
Programador: Yordán Herrera Cordova.	**Iteración Asignada**: 1
Prioridad: Media	**Tiempo Estimado:** 20
Riego en Desarrollo: Alto	**Tiempo Real**: 20
Descripción: El sistema debe permitir al usuario modificar los parámetros de actualización automática del Antivirus. La funcionalidad comienza cuando el usuario accede al sistema y selecciona en el menú lateral la opción Actualización automática dentro de Configurar actualización. El sistema debe mostrar una interfaz con los datos referentes a las configuraciones actualización automática del Antivirus.	

➢ Modificar hora en la cual se realizará la actualización automática: La aplicación permitirá modificar la hora elegida para la actualización automática, este valor podrá ser editado dentro del rango menor que 24 y mayor igual que 0. ➢ Modificar minuto en el cual se realizará la actualización automática: La aplicación permitirá modificar el minuto elegido para la actualización automática, este valor podrá ser editado dentro del rango menor que 60 y mayor igual que 0. ➢ Modificar el día de la semana elegido para realizar actualización automática: La aplicación permitirá modificar el día de la semana elegido para realizar actualización automática.
Observaciones: Estas directivas se encuentran en el fichero de *update.sh* ubicado en el directorio */etc/cron.d/clamav.*
Prototipo elemental de la interfaz gráfica de usuario:

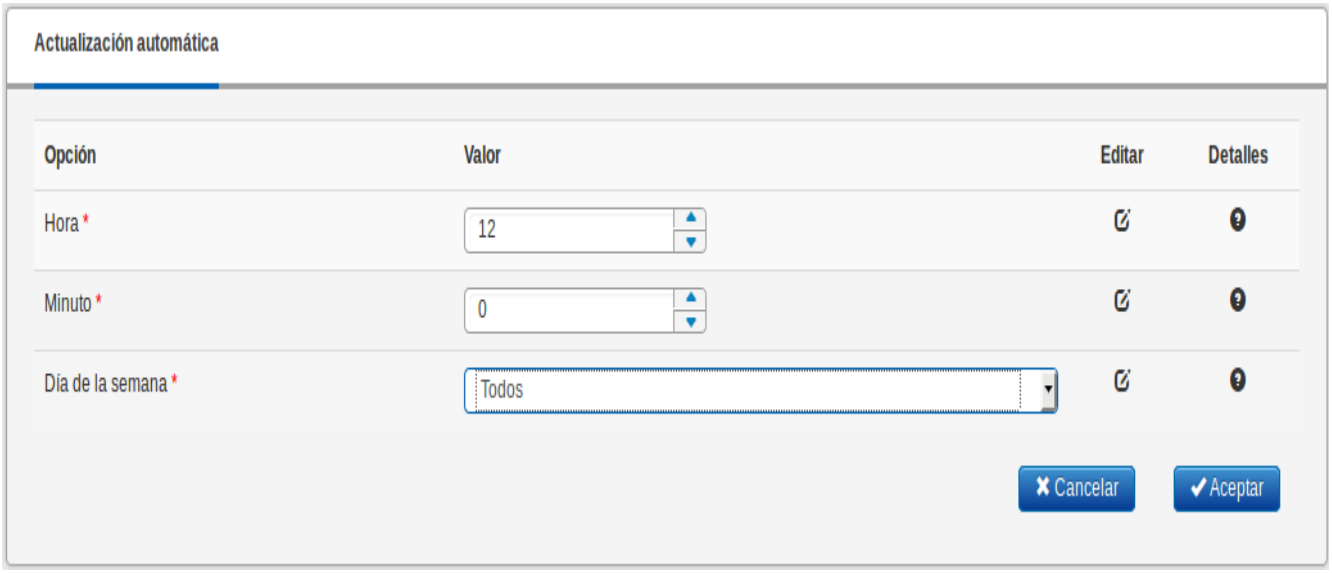


Tabla 23. Caso de prueba 5.

Caso de prueba de aceptación	
Código de caso de prueba: HU_6	**Nombre de la HU:** Mostrar configuración avanzada.
Nombre de la persona que realiza la prueba:	
Descripción de la prueba: Probar que el servicio muestra la configuración avanzada.	

Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe estar conectado a un servidor lógico y activar el módulo. El servicio antivirus debe estar instalado e iniciado.
Entrada/Pasos de ejecución: 1. Dar clic en el panel lateral en la opción Avanzada dentro de Configuración.
Resultado esperado: 1. El sistema muestra una interfaz con el valor actual de las directivas del fichero *clamd.conf.* • ScanMail: Escaneo de correo, el valor por defecto es deshabilitado. • LogVerbose: *Logs* detallados, el valor por defecto es deshabilitado. • ScanArchive: Escaneo de archivos comprimidos, el valor por defecto es deshabilitado. • ScanPE: Escaneo de archivos ejecutables, el valor por defecto es deshabilitado. • ScanHTML: Escaneo de archivos HTML, el valor por defecto es deshabilitado. • ScanPDF: Escaneo de archivos PDF, el valor por defecto es deshabilitado. • ScanOLE2: Escaneo de archivos de Microsoft Office y msi, el valor por defecto es deshabilitado. • ScanXMLDOCS: Escaneo de archivos XMLDOCS, el valor por defecto es deshabilitado. • ScanSWF: Escaneo de archivos SWF, el valor por defecto es deshabilitado. • ScanELF: Escaneo de archivos ELF, el valor por defecto es deshabilitado. • ScanHWP3: Escaneo de archivos HWP3, el valor por defecto es deshabilitado.
Evaluación de la prueba: *Satisfactoria*.

Tabla 24. Caso de prueba 6.

Caso de prueba de aceptación	
Código de caso de prueba: HU_7	**Nombre de la HU:** Modificar configuración avanzada.
Nombre de la persona que realiza la prueba:	
Descripción de la prueba: Probar que el servicio edita la configuración avanzada.	

Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe estar conectado a un servidor lógico y activar el módulo. El servicio antivirus debe estar instalado e iniciado.

Entrada/Pasos de ejecución:

1. Dar clic en el panel lateral en la opción Avanzada dentro de Configuración.
2. Seleccionar la opción Editar de las directivas.
3. Marcar o desmarcar los *checkbox* correspondientes a las directivas.
4. Seleccionar la opción Aceptar de la interfaz.

Resultado esperado:

1. El sistema muestra una interfaz con el valor actual de las directivas del fichero *clamd.conf.*
    - ScanMail: Escaneo de correo, el valor por defecto es deshabilitado.
    - LogVerbose: *Logs* detallados, el valor por defecto es deshabilitado.
    - ScanArchive: Escaneo de archivos comprimidos, el valor por defecto es deshabilitado.
    - ScanPE: Escaneo de archivos ejecutables, el valor por defecto es deshabilitado.
    - ScanHTML: Escaneo de archivos HTML, el valor por defecto es deshabilitado.
    - ScanPDF: Escaneo de archivos PDF, el valor por defecto es deshabilitado.
    - ScanOLE2: Escaneo de archivos de Microsoft Office y msi, el valor por defecto es deshabilitado.
    - ScanXMLDOCS: Escaneo de archivos XMLDOCS, el valor por defecto es deshabilitado.
    - ScanSWF: Escaneo de archivos SWF, el valor por defecto es deshabilitado.
    - ScanELF: Escaneo de archivos ELF, el valor por defecto es deshabilitado.
    - ScanHWP3: Escaneo de archivos HWP3, el valor por defecto es deshabilitado.
2. Después de aceptar los cambios realizados, se actualiza la interfaz con los nuevos valores.
3. El sistema muestra el mensaje “Las configuraciones han sido editadas correctamente”.

Evaluación de la prueba: *Satisfactoria*.

Tabla 25. Caso de prueba 7.

Caso de prueba de aceptación	
Código de caso de prueba: HU_8	**Nombre de la HU:** Mostrar configuración básica.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Probar que el servicio muestra la configuración básica.	
Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe estar conectado a un servidor lógico y activar el módulo. El servicio antivirus debe estar instalado e iniciado.	
Entrada/Pasos de ejecución: 1. Dar clic en el panel lateral en la opción Básica dentro de Configuración.	
Resultado esperado: 1. El sistema muestra una interfaz con el valor actual de las directivas del fichero *clamd.conf.* • LogFile: Dirección donde serán almacenados los *logs*. • MaxRecursion: Máxima profundidad a escanear. • MaxThreads: Número máximo de subprocesos.	
Evaluación de la prueba: *Satisfactoria*.	

Tabla 26. Caso de prueba 8.

Caso de prueba de aceptación	
Código de caso de prueba: HU_9	**Nombre de la HU:** Editar configuración básica.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Probar que el servicio edita la configuración básica.	
Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe estar conectado a un servidor lógico y activar el módulo. El servicio antivirus debe estar instalado e iniciado.	
Entrada/Pasos de ejecución: 1. Dar clic en el panel lateral en la opción Básica dentro de Configuración. 2. Seleccionar la opción Editar de las directivas.	

3. Seleccionar la opción Aceptar de la interfaz.
Resultado esperado: 1. El sistema muestra una interfaz con el valor actual de las directivas del fichero *clamd.conf.* • LogFile: Dirección donde serán almacenados los *logs*. • MaxRecursion: Máxima profundidad a escanear. • MaxThreads: Número máximo de subprocesos. 2. Después de aceptar los cambios realizados, se actualiza la interfaz con los nuevos valores. 3. El sistema muestra el mensaje “Las configuraciones han sido editadas correctamente”.
Evaluación de la prueba: *Satisfactoria*.

Tabla 27. Caso de prueba 9.

Caso de prueba de aceptación	
Código de caso de prueba: HU_10	**Nombre de la HU:** Mostrar configuración de actualización.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Probar que el servicio muestra la configuración avanzada.	
Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe estar conectado a un servidor lógico y activar el módulo. El servicio antivirus debe estar instalado e iniciado.	
Entrada/Pasos de ejecución: 1. Dar clic en el panel lateral en la opción Parámetros de actualización dentro de Configurar actualización.	
Resultado esperado: 1. El sistema muestra una interfaz con el valor actual de las directivas del fichero *update.sh.* • databaseDirectory: Directorio de actualización. • URLMAIN: Origen de las actualizaciones. • PROXY_SERVER: Servidor. • PROXY_PORT: Puerto. • PROXY_USER: Usuario.	

• PROXY_PASWORD: Contraseña.
Evaluación de la prueba: *Satisfactoria*.

Tabla 28. Caso de prueba 10.

Caso de prueba de aceptación	
Código de caso de prueba: HU_11	**Nombre de la HU:** Modificar configuración de actualización.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Probar que el servicio edita la configuración de actualización.	
Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe estar conectado a un servidor lógico y activar el módulo. El servicio antivirus debe estar instalado e iniciado.	
Entrada/Pasos de ejecución: 1. Dar clic en el panel lateral en la opción Parámetros de actualización dentro de Configurar actualización. 2. Seleccionar la opción Editar de las directivas. 3. Seleccionar la opción Aceptar de la interfaz.	
Resultado esperado: 1. El sistema muestra una interfaz con el valor actual de las directivas del fichero *update.sh.* • databaseDirectory: Directorio de actualización. • URLMAIN: Origen de las actualizaciones. El campo admite direcciones URL. • PROXY_SERVER: Servidor. El campo admite direcciones IP en su versión ipv4 y nombre del servidor. • PROXY_PORT: Puerto. • PROXY_USER: Usuario. El campo se puede dejar vacío. • PROXY_PASWORD: Contraseña. El campo se puede dejar vacío solo si el de la contraseña lo está. 2. Después de aceptar los cambios realizados, se actualiza la interfaz con los nuevos valores. 3. Al introducir datos incorrectos los campos se ponen en rojo.	

4. El sistema muestra el mensaje “Las configuraciones han sido editadas correctamente”.
Evaluación de la prueba: *Satisfactoria*.

Tabla 29. Caso de prueba 11.

Caso de prueba de aceptación	
Código de caso de prueba: HU_12	**Nombre de la HU:** Mostrar configuración de actualización automática.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la prueba: Probar que el servicio muestra la configuración de actualización automática.	
Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe estar conectado a un servidor lógico y activar el módulo. El servicio antivirus debe estar instalado e iniciado.	
Entrada/Pasos de ejecución: 1. Dar clic en el panel lateral en la opción Actualización automática dentro de Configurar actualización.	
Resultado esperado: 1. El sistema muestra una interfaz con el valor actual de las directivas del fichero *clamav.* • Hora • Minuto • Día de la semana.	
Evaluación de la prueba: *Satisfactoria*.	

Tabla 30. Caso de prueba 12.

Caso de prueba de aceptación	
Código de caso de prueba: HU_13	**Nombre de la HU:** Modificar configuración de actualización automática.
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	

Descripción de la prueba: Probar que el servicio edita la configuración de actualización automática.
Condiciones de ejecución: El usuario debe de estar autenticado en el sistema HMAST. Debe estar conectado a un servidor lógico y activar el módulo. El servicio antivirus debe estar instalado e iniciado.
Entrada/Pasos de ejecución: 1. Dar clic en el panel lateral en la opción Actualización automática dentro de Configurar actualización. 2. Seleccionar la opción Editar de las directivas. 3. Seleccionar la opción Aceptar de la interfaz.
Resultado esperado: 2. El sistema muestra una interfaz con el valor actual de las directivas del fichero *update.sh.* 2. El sistema muestra una interfaz con el valor actual de las directivas del fichero *clamav.* • Hora • Minuto • Día de la semana 5. Después de aceptar los cambios realizados, se actualiza la interfaz con los nuevos valores. 6. Al introducir datos incorrectos los campos se ponen en rojo. 7. El sistema muestra el mensaje “Las configuraciones han sido editadas correctamente”.
Evaluación de la prueba: *Satisfactoria.*

Universidad de las Ciencias Informáticas

Acta de aceptación de productos de trabajo

ACTA DE ACEPTACIÓN DE PRODUCTOS DE TRABAJO

En cumplimiento del **Convenio de colaboración** establecido entre el **Centro de Software Libre (CESOL)** y el estudiante **Yordán Herrera Cordova** de la Facultad 1 de la Universidad de las Ciencias Informáticas y en función de la ejecución del proyecto: **Servicio antivirus para la infraestructura del servicio de correo de la Herramienta para la Migración y Administración de Servicios Telemáticos**, se hace entrega de los productos que se relacionan a continuación:

- Componente de Antivirus para el módulo de correo de HMAST

La parte Cliente, luego de haber revisado los productos de trabajos relacionados anteriormente procede a firmar la aceptación de los mismos en total conformidad.

Entrega	**Recibe**
Nombre y apellidos: Yordán Herrera Cordova	**Nombre y apellidos:** Yoandy Pérez Villazón
Cargo: Estudiante Facultad 1	**Cargo:** Director del Centro de Soluciones Libres
Firma:	**Firma:**

Fecha: 31/05/2017

Figura 9. Acta de aceptación.

Referencias Bibliográficas:

1. Servicios telemáticos (DL. 1900-90) - Ministerio de Tecnologías de la Información y las Comunicaciones. [En línea] [Citado el: 04 de 05 de 2017.] http://www.mintic.gov.co/portal/604/w3-article-5737.html.

2. Correo electrónico y redes sociales dominan internet, según encuesta. [En línea] 27 de 03 de 2016. [Citado el: 02 de 02 de 2017.] http://expansion.mx/tecnologia/2012/03/27/correo-electronico-y-redes-sociales-dominan-internet-segun-encuesta.

3. Definición de antivirus — Definicion.de. [En línea] [Citado el: 27 de 12 de 2016.] http://definicion.de/antivirus/.

4. ¿Qué son los Malwares? | InfoSpyware. [En línea] 2013. [Citado el: 06 de 12 de 2016.] https://www.infospyware.com/articulos/que-son-los-malwares/.

5. www.pandasecurity.com. [En línea] 4 de octubre de 2016. http://www.pandasecurity.com/spain/enterprise/solutions/security-appliances/anti-malware.htm.

6. *Virus informáticos y antivirus - ¿Que es un virus informático?* [En línea] 2016-12-29 19:19:43. [Citado el: 29 de 12 de 2016.] https://www.gcfaprendelibre.org/tecnologia/curso/virus_informaticos_y_antivirus/los_virus_informaticos/1.do.

7. Los Troyanos Informáticos - ¿Qué son los Troyanos Informáticos? - Concepto de Troyanos Informáticos. [En línea] [Citado el: 17 de 12 de 2016.] http://www.seguridadpc.net/troyanos.htm.

8. Qué es un worm o gusano informático - Definición de worm o gusano. [En línea] [Citado el: 17 de 12 de 2016.] http://www.masadelante.com/faqs/que-es-un-gusano.

9. J. R. Méndez, F. F. Riverola, F. D. Gómez, J. M. C. Rodríguez. *Sistemas inteligentes para la detección y filtrado de correo spam: una revisión.* 2007.

10. Y. Pérez Villazón, Y. Pérez Villazón. *Módulo de administración de correo electrónico para HMAST.* 2015.

11. Kyle D. Dent, O'Reilly. *Postfix: The Definitive Guide.* 2003.

12. amavisd-new. [En línea] [Citado el: 05 de 12 de 2016.] https://amavis.org/.

13. Dovecot. [En línea] [Citado el: 20 de 02 de 2017.] https://www.dovecot.org/.

14. Schwartz, Alan. *SpamAssassin.* s.l. : "O'Reilly Media, Inc.", 2004.

15. Fetchmail. [En línea] 04 de 07 de 2014. [Citado el: 20 de 02 de 2017.] http://www.fetchmail.info/.

16. Castañeda Valdés, Jesús, Viera Hernández, Amaury y Pérez Villazón, Yadiel. *Perfeccionamiento del módulo de correo de HMAST.* 2015.

17. 5 of the Best Antivirus Programs for Ubuntu - Make Tech Easier. [En línea] [Citado el: 15 de 12 de 2016.] https://www.maketecheasier.com/ubuntu-antivirus-programs/.

18. ClamavNet. [En línea] [Citado el: 05 de 03 de 2017.] https://www.clamav.net.

19. Security for Linux Mail Server | Kaspersky Lab ES. [En línea] [Citado el: 05 de 03 de 2017.] https://www.kaspersky.es/small-to-medium-business-security/linux-mail-server.

20. Antivirus para servidores para AIX, Hyper-V y entornos virtuales | Protección para vShield sin agente. [En línea] [Citado el: 05 de 03 de 2017.] https://www.sophos.com/es-es/products/server-security.aspx.

21. ClamAv Antivirus. [En línea] [Citado el: 02 de 03 de 2017.] https://www.ecured.cu/ClamAv_Antivirus.

22. Rod Johnson, Juergen Hoeller , Keith Donald , Colin Sampaleanu , Rob Harrop , Thomas Risberg , Alef Arendsen , Darren Davison , Dmitriy Kopylenko , Mark Pollack , Thierry Templier , Erwin Vervaet , Portia Tung , Ben Hale , Adrian Colyer , John Lewis. *Spring Framework Reference Documentation.* 2016.

23. IntelliJ IDEA: the Java IDE for Professional Developers. [En línea] [Citado el: 03 de 02 de 2017.] https://www.jetbrains.com/idea/.

24. Visual Paradigm para UML. [En línea] [Citado el: 02 de 02 de 2017.] http://www.software.com.ar/p/visual-paradigm-para-uml.

25. Apache Tomcat® - Welcome! [En línea] [Citado el: 03 de 02 de 2017.] http://tomcat.apache.org/.

26. Requisitos de Software - EcuRed. [En línea] [Citado el: 21 de 04 de 2017.] https://www.ecured.cu/Requisitos_de_Software.

27. *PRÁCTICA RECOMENDADA PARA LAS ESPECIFICACIONES DE REQUISITOS DEL SOFTWARE.* 2008.

28. Vera, Yosel. *MÓDULO DE HMAST PARA LA ADMINISTRACIÓN Y MIGRACIÓN HACIA SAMBA4 DEL SERVICIO DIRECTORIO ACTIVO.* 2016.

29. OpenLibra | Guía de Arquitectura N-Capas DDD .NET 4.0. [En línea] 09 de 05 de 2011. [Citado el: 05 de 03 de 2017.] https://openlibra.com/es/book/guia-de-arquitectura-n-capas-ddd-net-4-0.

30. Larman, Craig. *UML y Patrones, 2da Edición.*

31. Flower. Java Foundations: Java - Estándares de programación. [En línea] 02 de 07 de 2010. [Citado el: 27 de 01 de 2017.] http://javafoundations.blogspot.com/2010/07/java-estandares-de-programacion.html.

32. Pressman, Roger S. *Software Engineering.* 2009.

33. Microsoft. msdn.microsoft.com. [En línea] [Citado el: 19 de marzo de 2017.] http://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx..

34. Pressman, Roger. S. *Ingeniería de software. Un enfoque práctico.* Quinta .

35. Tecnónología y Synergix. [En línea] 15 de marzo de 2008. [Citado el: 23 de marzo de 2017.] https://synergix.wordpress.com/2008/03/15/definimos-pruebas-de-unidad-como/.

36. Ana M. del Carmen García Oterino . Javierganzas,com. [En línea] http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html.

37. PÉREZ VILLAZÓN, YASIEL y PÉREZ VILLAZÓN, YADIEL. *Módulo de administración de correo electrónico para HMAST .* Trabajo de Diploma.

38. Libro Electrónico: Virus y Antivirus Seguridad Informática. [En línea] 2012. [Citado el: 1 de 12 de 2016.] https://www.eaepublishing.com/catalog/details//store/es/book/978-3-8484-7057-0/libro-electr%C3%B3nico:-virus-yantivirus-seguridad-inform%C3%A1tica.

39. Miyares, Yoel. *Administración del servicio antivirus desde la Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST).* 2014.

40. GAVIRIA, BEATRIZ FLORIAN. *TÉCNICAS DE PRUEBAS DE SOFTWARE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN.* UNIVERSIDAD DEL VALLE. s.l. : SEMESTRE 2013A , SEMESTRE 2013A . Clase.