

Universidad de las Ciencias Informáticas  
Facultad 1

Título:

Administración del servicio antivirus desde la Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST).

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

Autor:

Yoel Miyares Tamayo

Tutores:

Ing. Yoandy Pérez Villazón

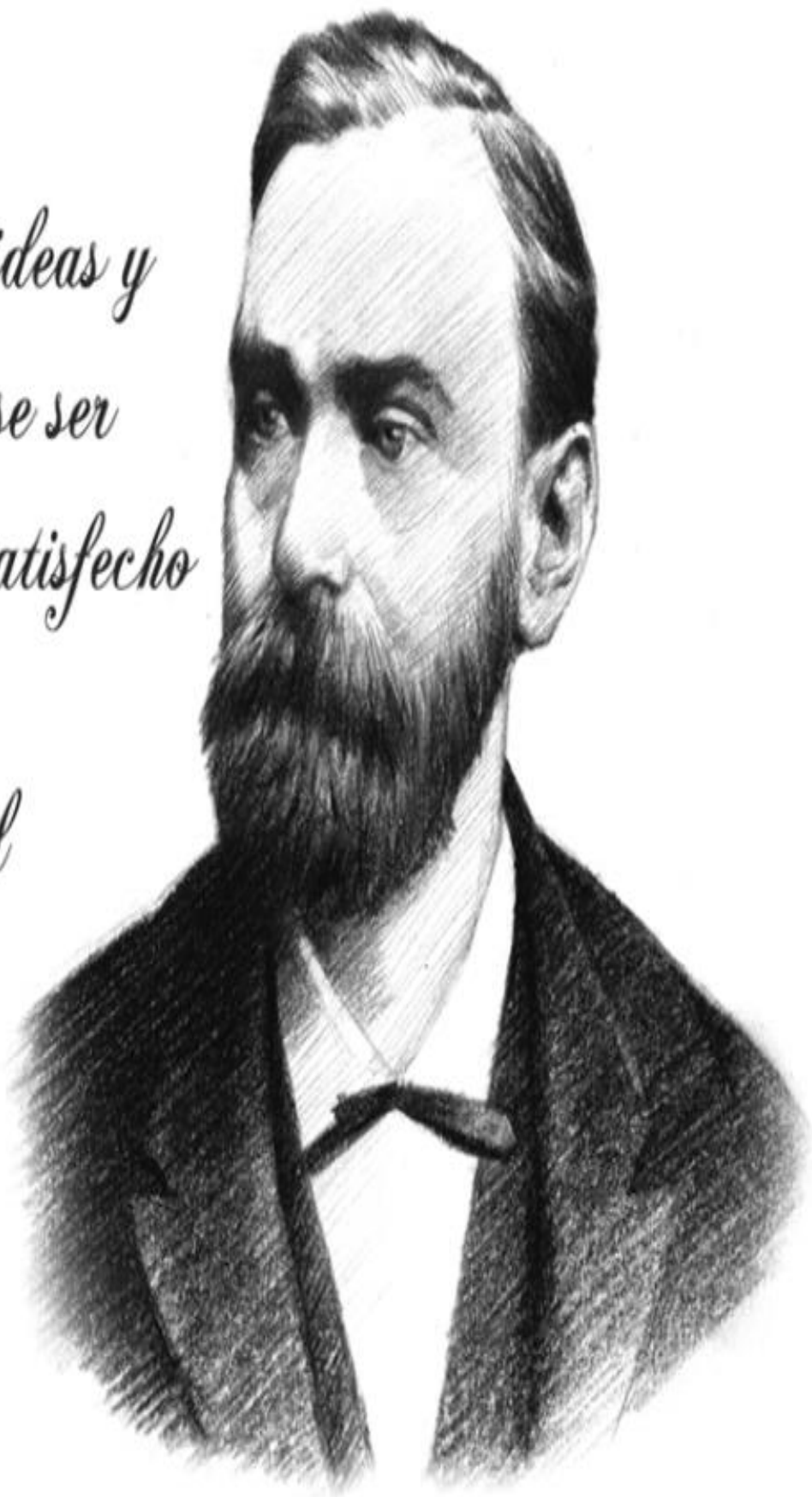
Ing. Nelio Véliz Pedraza

La Habana

Junio 2014

*Si tuviera mil ideas y  
sólo una resultase ser  
buena, estaría satisfecho*

*Alfred Nobel*  
(1833 - 1896)



### Agradecimientos

*Primero que todo le quiero agradecer a **mis padres**, porque sin su apoyo no estaría donde estoy ni sería la persona que soy hoy.*

*A **ti mami** por ser mi ejemplo a seguir, por darme fuerzas para salir adelante y brindarme todo tu amor y cariño, gracias por confiar en mí, te quiero mucho.*

*A **ti papá** por consentirme y quererme tanto, por brindarme todo tu apoyo siempre, te quiero con la vida.*

*A **mis abuelos** Blanca y Elio, que aunque no están conmigo físicamente siempre seguirán presentes en mi mente y mi corazón.*

*A **mis abuelos** Juan y Ángela por todo el cariño que me han dado.*

*A **toda mi familia** que de una forma u otra siempre me dio ánimos en todos los sentidos, a mis primos Yosmany, Maslice.*

*A **Yileni** por demostrarme que el amor es capaz de cualquier cosa, sin ella no hubiese tenido fuerzas para alcanzar muchas cosas en estos cinco años, gracias por todos los momentos compartidos, gracias por soportar todas mis malcriadeces, siempre te voy a querer y nunca me voy a olvidar de ti mi carita de pan.*

*Al **equipo de HMAST** por apoyarnos en todo momento y en especial a la rata del Flaco.*

*A todas esas personas que de una forma u otra, en el transcurso de estos cinco años han vivido momentos tristes y alegres a mi lado. A los que me ayudaron en la realización de este trabajo. A todos, gracias por haber contribuido en la formación de la persona que soy hoy.*

**Dedicatoria**

*A mis padres por brindarme todo su amor incondicional y por ser las personas más importantes de mi vida.*

*A mi niño que es el regalo más grande que la vida me dio.*

*A mi tío de la vida, Osmany y a mi primo Riky por todos sus consejos*

**Declaración de autoría**

Declaro por este medio que\_\_\_\_, con carné de identidad estudiante de la Universidad de las Ciencias Informáticas (UCI), soy el autor principal de la tesis de pregrado Administración del servicio antivirus desde la Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST) y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio; así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmamos la presente declaración jurada de autoría en La Habana a los días del mes de\_\_\_\_del año\_\_.

\_\_\_\_\_

Yoel Miyares Tamayo

\_\_\_\_\_

Ing. Yoandy Pérez Villazón

\_\_\_\_\_

Ing. Nelio Véliz Pedraza

## **Resumen**

En la Universidad de las Ciencias Informáticas al frente del proceso de migración a software libre y plataformas de código abierto se encuentra el Departamento de Servicios Integrales en Migración Asesoría y Soporte (SIMAYS), perteneciente al Centro de Software Libre (CESOL). En este departamento, como apoyo a la migración se están desarrollando aplicaciones como la Herramienta de Migración y Administración de Servicios Telemáticos (HMAST). HMAST permite administrar los servidores de forma remota, contemplando las funcionalidades necesarias para administrar los usuarios, las tareas programadas y los servicios. Sin embargo, en esta herramienta también se hace necesario la administración del servicio antivirus. El presente trabajo de diploma tiene el propósito de desarrollar un módulo para HMAST que administre el servicio antivirus de forma que facilite su uso por parte de los administradores del sistema en las diferentes empresas e instituciones cubanas. Para ello se analizan los servicios antivirus existentes en entornos de código abierto, seleccionando él antivirus ClamAv como el software más completo de los que fueron analizados. Además, se documentan las tecnologías, herramientas, lenguajes a utilizar en la construcción del módulo y la definición de los elementos necesarios para el exitoso desarrollo del mismo. Como resultado más importante se destaca la creación de un software capaz de llevar a cabo una serie de procesos encaminados a lograr un correcto funcionamiento del servicio antivirus, tales como realizar configuraciones generales del mismo para su correcto funcionamiento, así como las diferentes vías para realizar la actualización de antivirus; permitiendo analizar diferentes documentos y directorios y brindando la posibilidad de observar el resultado del mismo.

**Palabras claves:** administración, ClamAv, migración, plataformas libres, servicios telemáticos.

**Índice**

Introducción .....	1
Capítulo 1: Marco teórico referencial para el módulo antivirus .....	5
1.1 Conceptos Fundamentales .....	5
1.2 Funcionamiento de los antivirus .....	6
1.3 Motores de búsqueda de los antivirus.....	7
1.4 Análisis de la Resolución 127 del 2007 del MIC.....	7
1.5 Selección del Antivirus .....	8
1.6 Herramientas que administran el servicio antivirus .....	10
1.6.1 Zentyal.....	10
1.6.2 ClamTk .....	12
1.7 Descripción de HMAST .....	13
1.7.1 Funcionalidades que ofrece.....	15
1.7.2 Consideraciones para implementar un módulo para HMAST .....	15
1.8 Tecnologías asociadas al desarrollo del módulo de antivirus. ....	16
1.8.1 Lenguajes de Programación Utilizados para el desarrollo del módulo .....	16
1.8.2 Marco de trabajo.....	17
1.9 Metodología y herramientas utilizadas de desarrollo .....	18
1.9.1 Metodología SXP.....	18
1.10 Herramientas y tecnologías a emplear.....	19
1.10.1 Entornos de Desarrollo Utilizado (IDE) .....	19
1.11 Herramienta CASE para el modelado .....	20
1.11.1 Sistema de Control de Versiones .....	20
1.11.2 Herramienta de diseño gráfico Pencil.....	21
Capítulo 2: Análisis y diseño del módulo para HMAST.....	23
2.1 Propuesta del módulo a desarrollar.....	23

---

2.2 Requisitos del sistema .....	24
2.3 Arquitectura del software.....	39
2.4 Diagrama de paquetes .....	41
2.5 Diagrama de clases .....	42
2.6 Patrones de diseño.....	44
2.7 Conclusiones parciales .....	46
Capítulo 3: Desarrollo y prueba de la solución propuesta.....	47
3.1 Planificación de la implementación .....	47
3.2 Tareas de Ingeniería .....	47
3.3 Estándares de codificación .....	50
3.4 Pruebas de software .....	51
3.4.1 Diseño de casos de prueba .....	51
3.5 Conclusiones Parciales .....	57
Conclusiones .....	58
Recomendaciones .....	59
Referencias bibliográficas.....	60
Bibliografía.....	63
Glosario de términos.....	67
Anexos.....	69



**Índice de figuras**

Figura 1: Arquitectura del sistema. ....	15
Figura 2: Modelo de dominio. ....	24
Figura 3: Arquitectura del módulo. ....	40
Figura 4: Diagrama de paquetes. ....	42
Figura 5: Diagrama de clases. ....	44
Figura 6: Interfaz de instalación del antivirus. ....	69
Figura 7: Interfaz gestionar estado del antivirus ....	69
Figura 8: Interfaz de la configuración general del antivirus. ....	69
Figura 9: Interfaz del escaneo del antivirus. ....	70
Figura 10: Interfaz del resumen de los análisis realizados. ....	70
Figura 11: Interfaz de los ficheros en cuarentena. ....	71
Figura 12: Interfaz de las tareas planificadas. ....	71
Figura 13: Interfaz de las opciones generales para la actualización. ....	72
Figura 14: Interfaz de la actualización mediante proxy. ....	72
Figura 15: Interfaz de la actualización manual. ....	73
Figura 16: Interfaz de la actualización automática. ....	73

**Índice de tablas**

Tabla 1: Comparación entre antivirus .....	10
Tabla 2: Requisitos funcionales del módulo a implementar. ....	29
Tabla 3: Descripción de los requisitos no funcionales. ....	30
Tabla 4: HU Instalación del servicio antivirus. ....	31
Tabla 5: HU Gestionar estado del servicio.....	32
Tabla 6: HU Configuraciones generales del servicio antivirus. ....	34
Tabla 7: HU Configurar actualizaciones del servicio antivirus. ....	35
Tabla 8: HU Configurar actualizaciones del servicio antivirus mediante proxy. ....	37
Tabla 9: HU Actualización de forma manual.....	38
Tabla 10: HU Actualización de forma automática. ....	39
Tabla 11: TI Instalación del servicio antivirus. ....	48
Tabla 12: TI Gestionar acciones sobre el servicio antivirus. ....	48
Tabla 13: TI Escanear ficheros con el antivirus. ....	49
Tabla 14: TI Crear fichero para cuarentena. ....	49
Tabla 15: TI Ver ficheros que están en cuarentena. ....	49
Tabla 16: TI Eliminar ficheros infectados. ....	50
Tabla 17: Caso de prueba 1. ....	52
Tabla 18: Caso de prueba 2. ....	53
Tabla 19: Caso de prueba 3. ....	53
Tabla 20: Caso de prueba 4. ....	53
Tabla 21: Caso de prueba 5. ....	54
Tabla 22: Caso de prueba 6. ....	54
Tabla 24: Caso de prueba 8. ....	54
Tabla 25: Caso de prueba 9. ....	55
Tabla 26: Caso de prueba 10. ....	55
Tabla 27: Caso de prueba 11. ....	56
Tabla 28: Caso de prueba 12. ....	56
Tabla 29: Caso de prueba 13. ....	56

## **Introducción**

Cuba ha identificado desde muy temprano la necesidad e importancia de lograr una cultura digital como una de las características para alcanzar la eficiencia y competitividad en sus diferentes esferas y procesos. Para darle cumplimiento a dicho objetivo, se ha propuesto el dominio e introducción en la práctica social de las Tecnologías de la Información y las Comunicaciones (TIC). Es a partir de 1996 que se dan los primeros pasos en cuanto a la informatización de la sociedad, definida como el proceso de utilización ordenada y masiva de las TIC para satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad [1].

Este proceso ha ido encaminado y enfocado a los pilares fundamentales del proceso revolucionario: la educación, la salud y la cultura. No se puede lograr una utilización masiva de las TIC sin haber elevado la calidad de la educación, instrumentado un proceso de educación continua y ampliando la cultura general de la población sobre estas tecnologías. Como expresara el compañero Fidel Castro Ruz el 15 de enero de 1960, en un discurso ante la Sociedad Espeleológica: “El futuro de nuestra Patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento...” [2].

Con el fin de asegurar la soberanía tecnológica, impulsar la innovación nacional y facilitar la inclusión digital, actualmente el país se encuentra inmerso en un proceso paulatino de migración paulatina hacia Software Libre y aplicaciones de código abierto, pues es real el hecho de que constituye una necesidad, debido a lo que representa para la sociedad cubana en los aspectos políticos, económicos, sociales y tecnológicos. Persiguiendo el propósito de apoyar la migración en el país, en la Universidad de las Ciencias Informáticas (UCI), fue creado el Centro de Software Libre (CESOL), el cual cuenta con el Departamento de Servicios Integrales en Migración, Asesoría y Soporte (SIMAYS). La misión del mismo va dirigida a tres líneas fundamentales: el desarrollo de estrategias, guías y planes de migración; las herramientas de soporte al proceso de migración y la migración de los servicios telemáticos y aplicaciones de escritorio.

Como parte de la estrategia de migración, desde el año 2011 SIMAYS se dio a la tarea de desarrollar la Herramienta de Migración y Administración de Servicios Telemáticos (HMAST), la cual surge debido a la necesidad que tenían algunos de los Organismos de la Administración Central del Estado (OACE), de contar con una herramienta que se ajustara a sus necesidades a la hora de migrar desde diferentes versiones de Windows Server hacia plataformas libres, los servicios telemáticos que estos brindan, así como la administración de los mismos. HMAST está

compuesta por un sistema base que permite administrar los servidores, el cual posee las funcionalidades necesarias para la administración de usuarios, tareas programadas y servicios.

A pesar de la existencia de HMAST, todavía resulta un reto llevar a cabo el proceso de migración y administración de los servicios telemáticos de forma exitosa haciendo uso de la misma, debido a que está en su primera versión y solo tiene desarrollados dos módulos: uno para administrar las conexiones mediante el protocolo SSH y otro para realizar la administración del servicio DHCP, sin embargo, en estas empresas una vez migradas, se hace necesario la administración del servicio antivirus, teniendo en cuenta que éste es protagonista del proceso de seguridad en las aplicaciones.

Para dar solución a la situación problemática anteriormente mencionada se presenta como **problema científico**: ¿cómo realizar la administración del servicio antivirus desde HMAST, de modo que facilite su uso por parte de los administradores del sistema?

Por lo que se establece como **objetivo general** desarrollar un módulo para la administración del servicio antivirus en HMAST que facilite su uso y configuración, haciendo uso de tecnologías de código abierto.

Para dar cumplimiento al objetivo general se determinaron los siguientes **objetivos específicos**:

- ✓ Sistematizar sobre las herramientas de código abierto para la administración del servicio antivirus.
- ✓ Analizar y diseñar el módulo para la administración del servicio antivirus.
- ✓ Implementar las funcionalidades del módulo definido.
- ✓ Probar las funcionalidades del módulo obtenido.

Para dar cumplimiento a los objetivos general y específicos se identificaron las siguientes **tareas de investigación**:

- ✓ Revisión de la literatura especializada sobre el software de código abierto que brinda el servicio antivirus.
- ✓ Estudio de herramientas de código abierto que realizan la administración del servicio antivirus.
- ✓ Especificación de los requisitos funcionales y no funcionales para el módulo a desarrollar.
- ✓ Definición de la arquitectura del módulo y patrones de diseño a utilizar.
- ✓ Desarrollo de las Historias de Usuario.

- ✓ Diseño y ejecución de los casos de pruebas.

El **objeto de estudio de la investigación** se centra en la administración del servicio antivirus, enmarcando el **campo de acción** en la administración del servicio antivirus en entornos de código abierto.

En el presente trabajo de diploma se plantea como **idea a defender** que el desarrollo de un módulo de administración del servicio antivirus para HMAST garantizará la administración del servicio antivirus en las instituciones cubanas, facilitando su uso y configuración, haciendo uso de tecnologías de código abierto.

### Métodos Utilizados

Con el objetivo de obtener los conocimientos a partir de los datos de la práctica y de la teoría precedente y elaborar una estrategia general para enfrentar el problema que se investiga, se utilizan los siguientes métodos del diseño metodológico de la investigación científica.

- ✓ Analítico-Sintético: se utiliza en el estudio de diferentes fuentes bibliográficas para extraer los elementos más significativos de los diferentes tipos de antivirus que existen para la administración y configuración del servicio antivirus y un estudio del funcionamiento y las configuraciones necesarias de este servicio, facilitando la implementación del módulo propuesto.
- ✓ Histórico-lógico: se utiliza para analizar la trayectoria y evolución de los antivirus en servidores y determinar las razones que fundamentan la necesidad de seleccionar uno de ellos.

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

**Capítulo 1: Marco teórico referencial para el módulo antivirus.** Se realiza un estudio de HMAST para conocer sus características y recopilar información acerca de cómo implementar un módulo para la misma. Se definen los conceptos asociados y se analizan los principales antivirus utilizados para el sistema operativo GNU/Linux, seleccionándose de acuerdo a las condiciones de las empresas cubanas, el más adecuado para la solución a implementar.

**Capítulo 2: Análisis y diseño del módulo para HMAST.** Se presenta la propuesta de solución al problema planteado, son definidos los requisitos funcionales y no funcionales del módulo. Posteriormente se realiza la descripción de las Historias de Usuario y es elaborado el Plan de Iteraciones, el cual define las funcionalidades que serán desarrolladas en cada iteración del proceso de implementación. Por último, en este capítulo se detalla la arquitectura del módulo en cuestión y se describen los patrones utilizados en el diseño del software.

**Capítulo 3: Desarrollo y prueba de la solución propuesta.** Se implementa la solución propuesta, además, se explican las clases principales del sistema. Así mismo se confecciona el plan de pruebas y la ejecución de las pruebas necesarias para verificar que las funcionalidades desarrolladas dan cumplimiento a los requisitos planteados.

## **Capítulo 1: Marco teórico referencial para el módulo antivirus**

En el marco del estudio del estado del arte, en el presente capítulo se hará referencia a conceptos relacionados con los servicios antivirus y específicamente los de código abierto. Como resultado del estudio se seleccionará el servicio antivirus más indicado a emplear para el desarrollo del módulo.

### **1.1 Conceptos Fundamentales**

**Antivirus:** Se denomina antivirus a un software utilizado para eliminar programas elaborados con intención destructiva, surgieron como una solución a la proliferación de software malicioso cuando el uso de computadoras personales comenzó a masificarse y con ello surgió todo un nuevo mercado [3].

El autor considera que un antivirus es un programa informático que sirve para detectar, prevenir y eliminar virus informáticos en los distintos soportes de almacenamiento de datos, como el disco duro o un disquete.

**Antivirus de ficheros:** la característica primordial de cualquier antivirus actual es detectar la mayor cantidad de amenazas informáticas que puedan afectar un ordenador y bloquearlas antes de que la misma pueda infectar un equipo, o poder eliminarla tras la infección [4].

**Antivirus de correos:** el correo electrónico es el método de propagación más utilizado por los virus, por lo que protegerlo es imperativo. La protección antivirus a nivel de servidores de correo electrónico puede ser utilizada en distintos niveles, pero siempre con el mismo objetivo: detener los virus antes de que ingresen a la red y lleguen a los usuarios. Por ello, es importante que un antivirus para servidores de correo electrónico cuente con una protección activa capaz de detectar amenazas desconocidas, ya que de esta manera provee mayor seguridad.

**Spam:** se denomina Spam (correo basura o no deseado) al correo electrónico no solicitado que se envía por internet de forma masiva [5].

**Vacuna:** se denomina vacuna a un programa que instalado residente en la memoria, actúa como filtro de los programas que son ejecutados, abiertos para ser leídos o copiados, en tiempo real.

**Los antivirus pueden ser clasificados en [6]:**

**Antivirus Preventores:** Los programas que previenen la infección, quedan residentes en la memoria de la computadora todo el tiempo y monitorizan algunas funciones del sistema.

**Antivirus Identificadores:** Estos productos antivirus identifican programas malignos específicos que infectan al sistema. Los mismos trabajan con las características de un programa maligno o sus variantes, o exploran el sistema buscando cadenas (secuencias de *bytes*) de códigos particulares o patrones característicos de los mismos para identificarlos.

**Antivirus Descontaminadores:** Sus características son similares a los productos identificadores, con la diferencia que su principal función es descontaminar a un sistema que ha sido infectado, eliminando los programas malignos y retornando el sistema a su estado original.

### **1.2 Funcionamiento de los antivirus**

Los antivirus cuentan generalmente con una lista de virus conocidos y formas de reconocerlos llamadas firmas<sup>1</sup>. Un antivirus se encarga de comparar esas firmas con los archivos enviados y recibidos. Usan otra forma de detección de virus llamada detección pro-activa: su funcionamiento básico consiste en detectar los virus, no comparándolos con una base de datos sino detectándolos por sus comportamientos. El análisis de la información se produce de muy diversas maneras dependiendo de dónde provenga [5].

Ejemplo del principio de funcionamiento de un antivirus:

El ejemplo consiste en que la información que está en el "sistema origen" debe llegar al "sistema destino". El sistema origen podría ser una memoria USB y el sistema destino el disco duro del ordenador.

El funcionamiento del mecanismo de interceptación de la información varía en función de su implantación en sistemas operativos, en aplicaciones o bien de la necesidad de mecanismos especiales.

Una vez analizada la información, por el método que sea, si se ha detectado cualquier peligro, se llevan a cabo dos acciones:

1. Devolver la información limpia al mecanismo de interceptación que, a su vez, la devolverá al sistema para que siga su curso hasta el destino final.
2. Emitir una alarma a la interfaz del usuario. En un antivirus para una estación de trabajo puede ser un mensaje mostrado en pantalla, pero en una solución para servidores la alarma puede

---

<sup>1</sup> **Firmas o Firmas antivirus:** son vacunas que comparan las firmas de archivos sospechosos para saber si están infectados.



consistir en un mensaje de correo electrónico, un mensaje a la red interna, una entrada en un informe de actividad o una comunicación de algún tipo a la herramienta de gestión del antivirus.

### **1.3 Motores de búsqueda de los antivirus**

Independientemente de cómo se haya obtenido la información a analizar, entra en acción la parte más importante de un antivirus: el motor de búsqueda. Este motor se encarga de buscar virus en la información que ha sido interceptada y, si procede, desinfectarla.

Esta búsqueda de información se lleva a cabo de dos maneras:

- ✓ comparar la información recibida con una base de datos de virus (las llamadas "firmas de virus"). Si coincide la información con los patrones previamente conocidos mediante las firmas, se concluye que el fichero está infectado por un virus [5].
- ✓ averiguar si lo que se está analizando puede ser peligroso sin saber previamente si es un virus o no, esto se llama método heurístico. Para ello se analiza cómo se comporta la información [7] y se compara con una lista de patrones de comportamientos peligrosos.

Por ejemplo, si se encuentra que un fichero tiene capacidad de formatear un disco duro el antivirus puede avisar al usuario. Quizá no sea un virus, sino un nuevo sistema de formateo que el usuario está instalando en el sistema; sin embargo, la acción de por sí, es peligrosa. Es el usuario, ante la alerta que le da el antivirus el que debe decidir si elimina el peligro o no.

### **1.4 Análisis de la Resolución 127 del 2007 del MIC**

Los avances alcanzados en los últimos años en la informatización de los procesos de la sociedad y el impulso orientado por la dirección del país al desarrollo acelerado de programas que multipliquen dichos logros, requieren la adopción de medidas que garanticen un adecuado nivel de seguridad para su protección y ordenamiento.

En Cuba, la Resolución 127 del año 2007 aprobada por el Ministerio de la Informática y las Comunicaciones, puso en vigor el Reglamento para las tecnologías de la información, donde a través de 100 artículos, regula el funcionamiento del sistema informático nacional.

La Resolución 127 tiene como objetivo, establecer los requerimientos que rigen la seguridad de las tecnologías de la información y garantizar un respaldo legal que responda a las condiciones y necesidades del proceso de informatización del país [8].

El capítulo 3 que lleva como nombre “Empleo conveniente y seguro de las tecnologías de la información”, está compuesto por 8 secciones las cuales tienen sus particularidades. La sección 6 trata sobre la “Seguridad ante programas malignos”. Brindarle especial atención en esta sección al artículo 50 que plantea: “En cada entidad se implementarán los controles y procedimientos para protegerse contra virus y otros programas dañinos que puedan afectar los sistemas en explotación, así como para impedir su generalización. Para la protección contra virus se utilizarán los programas antivirus de producción nacional u otros autorizados oficialmente para su uso en el país, debidamente actualizados”.

### **1.5 Selección del Antivirus**

Actualmente existe una amplia gama de antivirus que pueden funcionar en GNU/Linux, aunque es uno de los sistemas operativos más seguro, ya éste cuenta con la existencia de varios antivirus. Partiendo de un estudio previo realizado por el equipo de migración de servicios telemáticos del departamento SIMAYS, a continuación se realiza un estudio para determinar cuál satisface las necesidades del cliente, además de estar acorde con la resolución 127 del MIC.

#### **SavUnix Milter**

*SavUnix Milter*: producto para integrar el antivirus SavUnix con servidores de correo que soporten el protocolo *Milter*. Para su instalación y correcto funcionamiento el programa requiere tener instalado y actualizado el antivirus SavUnix. El producto escanea y analiza en tiempo real los mensajes que procesan los servidores de correo electrónico, es capaz de detectar programas malignos que se ejecuten en Microsoft Windows y GNU/Linux. Entre sus principales deficiencias se encuentra la imposibilidad de utilizar una herramienta de comunicación con algún MTA [9].

#### **Kaspersky Security for Linux Mail Server**

*Kaspersky Security for Linux Mail Server* ofrece funciones esenciales de seguridad del correo electrónico, incluyendo *antimalware* filtrado de contenido, en un paquete fácil de administrar. Al combinar los motores *antimalware* y *antispam* de Kaspersky Lab, además del poder de la nube, *Kaspersky Security for Linux Mail Server* proporciona detección de *malware* y spam mejorada con una baja tasa de falsos positivos<sup>2</sup>.

---

<sup>2</sup> Falso positivo ocurre cuando el programa de antivirus detecta virus no reales.

Las características más recientes de combate contra el spam incluyen el filtrado por reputación y el servicio de actualizaciones antispam forzadas para el filtrado del spam hora cero. Además la nueva tecnología *ZetaShield* protege a las empresas de los ataques dirigidos y el *spam* día cero [10].

### Características más notables son:

- ✓ Detección de *spam* mejorada con menos falsos positivos.
- ✓ Detección *antimalware* y hora cero mejorada.
- ✓ Alta utilidad y alto rendimiento.
- ✓ Flexible y fácil de ejecutar y monitorear.
- ✓ Fácil de integrar.

### ClamAv

ClamAv es un antivirus de código abierto, licenciado bajo GNU/GPL, diseñado para la detección de troyanos, virus, *malware* y otras amenazas maliciosas, es el estándar de facto para escanear mensajes de correo. Además, proporciona un alto rendimiento como demonio<sup>3</sup> de escaneo multihilo, dispone de utilidades de línea de comandos para el análisis bajo demanda de archivos y una herramienta inteligente para las actualizaciones automáticas de firmas. El núcleo ClamAv presenta librerías que ofrecen numerosos mecanismos de detección de virus en archivos [11].

La aplicación también cuenta con un Militer<sup>4</sup> interfaz para *Sendmail* y análisis bajo demanda. Tiene soporte para *Zip*, *RAR*, *Tar*, *Gzip*, *Bzip2*, *OLE2*, *Gabinete*, *CHM*, *BinHex*, *ELF* ejecutables y portables (PE). Los archivos comprimidos con *UPX*, *FSG*, *Petite*, *Nspack*, *WWPack32*, *MEW*, *Upack* y ofuscado con *SUE*, *Y0da Cryptor*. También es compatible con muchos formatos de documentos, incluyendo *Microsoft Office*, HTML, formato de texto enriquecido (RTF) y *Portable Document Format* (PDF).

A continuación se presenta una tabla comparativa entre las herramientas antes descritas teniendo en cuenta los aspectos que se describen a continuación:

Aspectos a Medir	Kaspersky	SavUnix Militer	ClamAv
Tipo de licencia	Privativa	Privativa	Libre / Código abierto

<sup>3</sup> Tipo especial de proceso informático no interactivo, que se ejecuta en segundo plano.

<sup>4</sup> Filtro de correo

## Capítulo 1: Marco teórico referencial para el módulo antivirus

Motor heurístico	Alto	Alto	Alto
Rendimiento	Alto	Medio	Alto
Consumo de recursos	Alto	Bajo	Bajo
Velocidad de escaneo	Alta	Media	Alta
Efectividad en el análisis	Alta	Media	Media

Tabla 1: Comparación entre antivirus

Luego del estudio de las tecnologías de antivirus antes mencionadas y teniendo en cuenta las características del entorno que se propone, la propuesta de software antivirus para el desarrollo del módulo de HMAST es la herramienta ClamAv. La 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 recurso, un rasgo importante teniendo en cuenta que la mayoría de las empresas cubanas no cuentan con un *hardware* de alto rendimiento.

Aunque se analiza la resolución 127 del MIC que establece que se hará uso de un antivirus de producción nacional y por las características que posee la herramienta a la cual se va integrar el módulo, no es seleccionado el antivirus *SavUnix Militer*. Además, la Organización de Seguridad de Redes Informática (OSRI) plantea que se puede hacer uso de cualquier antivirus para ser utilizados en las diferentes instituciones del país, siempre y cuando este actualizado.

### 1.6 Herramientas que administran el servicio antivirus

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

#### 1.6.1 Zentyal

El desarrollo de Zentyal se inició en el año 2004 con el nombre de eBox Platform<sup>5</sup> y actualmente es una solución consolidada de reconocido prestigio que integra más de 30 herramientas de código abierto para la

<sup>5</sup> **eBox Platform:** Servidor de linux para PYMES, creado en el 2004 por eBox Technologies S.L.

administración de sistemas y redes en una sola tecnología. Está incluido en Ubuntu<sup>6</sup> desde el año 2007, en la actualidad tiene más de 1.000 descargas diarias y dispone de una comunidad activa de más de 5.000 miembros.

Zentyal se desarrolló con el objetivo de acercar GNU/Linux a las PYMES y permitirles aprovechar su potencial como servidor de empresa. Es la alternativa en código abierto a *Windows Small Business Server*, basado en la distribución Ubuntu. Este sistema permite a los informáticos de una institución, administrar todos los servicios telemáticos necesarios en su infraestructura de servidores tales como: acceso a Internet, seguridad de la red, compartición de recursos, comunicaciones, entre otros, de forma sencilla y a través de una única plataforma. Durante su desarrollo se hizo énfasis en la usabilidad, creando una interfaz intuitiva<sup>7</sup> que incluye únicamente aquellas funcionalidades de uso más frecuente, aunque también dispone de los medios necesarios para realizar toda clase de configuraciones más complejas.

Otra de las características importantes de este sistema es que todas sus funcionalidades, construidas a partir de una serie de aplicaciones en principio independientes, están estrechamente integradas entre sí, automatizando la mayoría de las tareas y ahorrando tiempo en la administración de sistemas. Teniendo en cuenta que el 42% de los fallos de seguridad y el 80% de los cortes de servicio en una empresa se deben a errores humanos en la configuración y administración de los mismos [12]. El resultado es una solución no sólo más sencilla de manejar, sino también más segura y fiable que mejora la seguridad y disponibilidad de los servicios en una empresa.

Este producto informático tiene un conjunto de funcionalidades entre los cuales se destacan los siguientes:

- ✓ Gestión de red: firewall, servidor DHCP, servidor NTP, servidor DNS, soporte para VPN, proxy HTTP, entre otros.
- ✓ Servidor de correo: POP3 e IMAP con SSL/TLS, filtro *antispam* y antivirus, *webmail*, etc.
- ✓ Comunicaciones: FTP, servidor VoIP, *Voicemail* y servidor *Jabber* de mensajería instantánea.
- ✓ Compartición de recursos y trabajo en grupo: servidor de archivos, servidor de impresión y *groupware* (agenda, contactos, etc.).

---

<sup>6</sup> **Ubuntu:** es una distribución de Linux desarrollada por Canonical y la comunidad orientada a ordenadores portátiles, de sobremesa y servidores: <http://www.ubuntu.com/>.

<sup>7</sup> **Interfaz Intuitiva:** Es aquella interfaz fácil de trabajar, donde el usuario final sabe que tiene que hacer en cada opción que se muestre, pues generalmente son comunes a todos los usuarios.

- ✓ Gestión centralizada de usuarios mediante LDAP, sincronización con Directorio de *Windows*, etc.
- ✓ Reportes y monitoreo del sistema, además del envío de notificaciones vía correo, RSS o Jabber.
- ✓ Respaldos de configuraciones y datos de manera remota.
- ✓ Autoridad de certificación.

El módulo de Zentyal para la administración del servicio antivirus presenta, como todos los demás módulos, una interfaz con un alto grado de usabilidad. Sin embargo este servidor no reconoce y sobre-escribe las configuraciones establecidas en el fichero principal de configuración de SavUnix.

### 1.6.2 ClamTk

ClamTk es una interfaz gráfica de usuario (GUI) que facilita el uso de ClamAV (*Clam AntiVirus*). Es un conjunto de herramientas antivirus de código abierto (GPL) para UNIX, diseñadas especialmente para análisis de correo electrónico, aunque también son utilizadas para escanear directorios y ficheros de un disco duro. Es además una interfaz gráfica para el conocido antivirus ClamAV, el cuál ha sido diseñado para escanear el correo que llegue por el servidor de correo en busca de posibles virus, pero también puede utilizarse para escanear ficheros individuales, o directorios enteros del disco duro. Debido a que el uso de la línea de comandos no es muy cómodo para todos, se dispone de este *frontend* que facilita enormemente la labor de actualizar la base de datos de los virus, como para escanear el disco duro [13].

El estudio realizado a esta herramienta ayudo a detectar una serie de requisitos funcionales que fueron tomados en cuenta a la hora de la implantación del módulo a integrar a HMAST entre los más destacados se encuentran:

#### ✓ **Analizar un archivo**

No siempre es necesario escanear todo el sistema o un directorio completo, tal vez se sospeche de un fichero en concreto o no se conozca su procedencia original, permite seleccionar un archivo y escanearlo para despejar dudas. Al encontrar un virus lo notifica y da a elegir entre enviarlo a la cuarentena o eliminarlo, con esta última opción hay que prestar especial atención ya que se puede originar una "falsa alarma" concepto que se explica más adelante.

#### ✓ **Analizar un directorio**

Para escanear varios ficheros de un mismo lugar, analizando un directorio, analiza todo el contenido de éste (sin incluir subdirectorios). Si se quiere profundizar en todo un árbol de directorios con un análisis recursivo, escanea todo un directorio al completo incluyendo subdirectorios.

### ✓ **Administrar historiales**

Una forma muy sencilla de administrar el historial de análisis, muestra una ventana con todos los *logs* ordenados por fecha, donde se puede ver cada *log*, eliminar uno en concreto o eliminarlos todos.

### ✓ **Cuarentena**

Una cuarentena para ficheros infectados, es una acción preventiva, ya que en ocasiones es posible que un fichero detectado como virus no lo sea realmente. Esto produce una falsa alarma, pudiendo restaurarlo al directorio donde se encontraba. Por supuesto, se podrá ver el estado de la cuarentena conociendo la cantidad de ficheros afectados por esta, restaurar ficheros al ser falsas alarmas o eliminarlos si ciertamente son virus.

Atendiendo a las características anteriores ClamTk pudiera ser una buena opción a integrar a HMAST, sin embargo no se utiliza debido a que está programada en GTK, la cual es una librería para el desarrollo de interfaz gráfica, basada fundamentalmente en el lenguaje de programación *Perl*. HMAST está desarrollada con el lenguaje *Spring* y *Java*, por lo que su interfaz gráfica no es compatible con esta versión del antivirus.

## **1.7 Descripción de HMAST**

HMAST es el sistema al cual será integrado la solución a desarrollar. Esta herramienta permite administrar los servidores de forma remota, contemplando las funcionalidades necesarias para administrar los usuarios, las tareas programadas y los servicios. A continuación se describen aspectos relevantes de la misma, tales como la arquitectura que utiliza, sus principales funcionalidades, los requisitos no funcionales con los que debe cumplir y por último, se exponen las consideraciones a tener en cuenta al implementar un módulo para dicha herramienta.

### **Arquitectura del sistema**

La arquitectura que presenta HMAST propone el diseño de una arquitectura N-Capas orientada al dominio, distribuida en cinco componentes o paquetes (ver Figura 1). La interacción entre los mismos se realiza a través de interfaces y utilizando inyección de dependencias.

**La capa de Presentación** es la encargada de presentar al usuario los conceptos de negocio mediante una interfaz de usuario (IU), facilitar la explotación de dichos procesos, informar sobre la situación de los procesos de negocio e implementación de las reglas de validación de dicha interfaz. Para la implementación de la misma se utiliza el módulo MVC de *Spring*.

**La capa de Aplicación** es responsable de realizar llamadas a servicios de capas inferiores (Dominio). Los servicios que publica (servicios de aplicación) tienen la responsabilidad, por ejemplo, de adaptar la información que le llega a los requerimientos de los servicios de dominio. En esta capa también se ubican operaciones de trazas, seguridad, envío de correos electrónicos, cuando no forman parte estricta del negocio.

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

**La capa de Persistencia** tiene asignada la responsabilidad de contener el código necesario para persistir los datos. Los principales componentes que contendrá la capa son los repositorios, que son clases que implementan los contratos de repositorios definidos en la capa de Dominio.

Las responsabilidades de la capa **Infraestructura Transversal** están dadas a promover la reutilización de código, contiene las operaciones de seguridad, *logging*, monitoreo del sistema, mecanismos de persistencia reutilizables, validadores genéricos y todas aquellas operaciones que se puedan llamar desde otras capas.



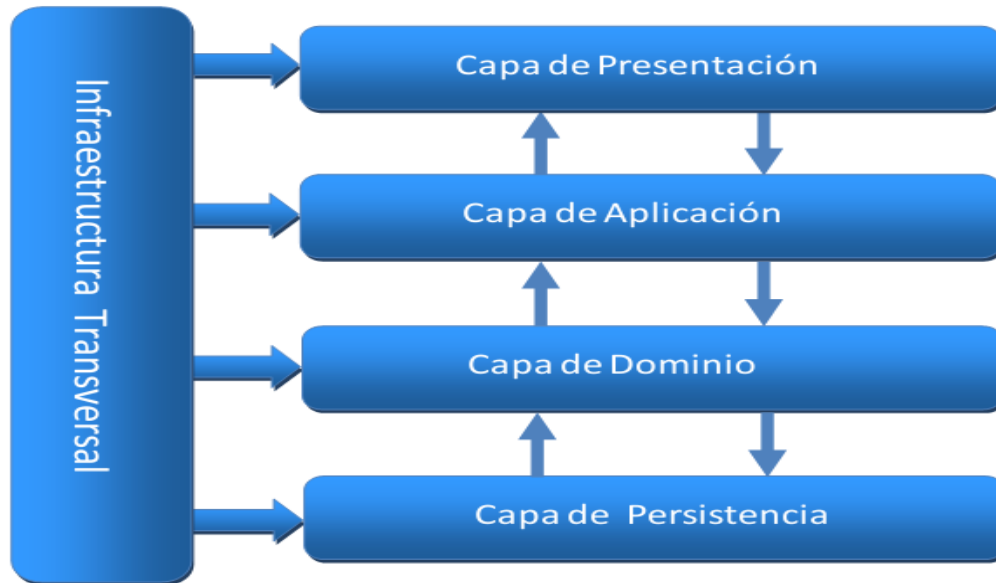


Figura 1: Arquitectura del sistema

### 1.7.1 Funcionalidades que ofrece

La herramienta cuenta con funcionalidades como son:

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

### 1.7.2 Consideraciones para implementar un módulo para HMAST

Para la implementación de un módulo que se desee integrar en HMAST se debe tener en cuenta que:

- ✓ La lógica de Aplicación no deberá incluir ninguna lógica del Dominio, solo tareas de coordinación relativas a requisitos técnicos de la aplicación, como conversiones de formatos de datos de entrada a entidades del Dominio, llamadas a componentes de infraestructura para que realicen tareas complementarias.
- ✓ Se garantizará que no se envíen hacia y desde la capa de Presentación objetos de Dominio, en su lugar deben viajar Objetos de Transferencia de Datos (*DTO, Data Object Transfer*).

- ✓ Las clases de servicios deben ser las únicas responsables (vías de acceso) de acceder a los repositorios, no se puede implementar código de persistencia a datos en la capa de Dominio.
- ✓ Solo se puede acceder a la información almacenada en los servidores haciendo uso de los repositorios.
- ✓ Es necesario que todo el código reutilizable por más de un repositorio se ponga a disposición de todos en la capa de Infraestructura Transversal.

### **1.8 Tecnologías asociadas al desarrollo del módulo de antivirus.**

Para el proceso de desarrollo del módulo de administración del servicio antivirus para HMAST se seleccionaron un conjunto de tecnologías y herramientas que ayudan a diseñar, modelar e implementar la solución propuesta, las cuales ayudan en gran medida, a obtener un resultado final de calidad. La correcta selección de las mismas permite que estas se adapten a las necesidades y características del desarrollador.

#### **1.8.1 Lenguajes de Programación Utilizados para el desarrollo del módulo**

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

#### **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. Sus características de memoria liberan a los programadores de una familia entera de errores. Soporta sincronización de múltiples hilos de ejecución a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas [15].

Entre sus características más importantes se encuentran:

- ✓ Multiplataforma: permite funcionar en diferentes sistemas operativos sin distinguir si es privativo o de distribución libre, lo único que necesita es que tenga instalada la máquina virtual de java.
- ✓ Seguro: por la utilización de punteros y realización de comprobaciones de seguridad.
- ✓ Fácil manipulación debido a su estructura orientada a objeto.

### 1.8.2 Marco de trabajo

Un marco de trabajo o *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener [16].

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

### Spring

El *Framework* de *Spring* proporciona un modelo integral de programación y configuración para aplicaciones modernas empresariales basadas en Java, para cualquier tipo de plataforma de despliegue. Está compuesto por aplicaciones web de tipo Modelo Vista Controlador (MVC) y el marco de servicios web *RESTful*<sup>8</sup>.

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

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

---

<sup>8</sup> *Restful*: Es un estilo arquitectónico que cuenta con un conjunto de restricciones aplicadas a los diferentes componentes.

interfaz. Así la dependencia puede ser intercambiada por distintas implementaciones a través del contenedor.

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

### **1.9 Metodología y herramientas utilizadas de desarrollo**

Las metodologías de desarrollo de software son el conjunto de procedimientos, técnicas, herramientas y un soporte documental, que ayuda a los desarrolladores a realizar nuevo software. La metodología define quién debe hacer qué, cuándo y cómo debe hacerlo para obtener los distintos productos parciales y finales. Se clasifican en dos tipos: tradicionales y ágiles.

Aunque el proceso de desarrollo de software incluye otros aspectos importantes y determinantes, es precisamente la metodología utilizada, el elemento rector a lo largo del mismo. Por tal motivo es que se debe analizar, de acuerdo a las características del software a desarrollar y del equipo de desarrollo, cuál es la más óptima. Por las ventajas que ofrecen las metodologías ágiles, se propone su uso para el desarrollo del trabajo de diploma, específicamente la metodología SXP. Además, esta es la definida por el proyecto en el desarrollo de sistemas similares.

#### **1.9.1 Metodología SXP**

SXP es un híbrido de metodologías ágiles que toma las mejores prácticas de las metodologías *SCRUM* y *XP* además de regirse por los lineamientos de calidad de la UCI. Con la aplicación de esta metodología se logra la gestión de un equipo de forma que se tengan siempre medidos los progresos y que el trabajo se realice de forma eficiente. Se caracteriza por una programación rápida o extrema, teniendo como parte del equipo al usuario final, siendo uno de los requisitos para llegar al éxito del proyecto.

Consta de 4 fases principales:

- ✓ Planificación-Definición donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.
- ✓ Desarrollo en la cual se realiza la implementación del sistema hasta que esté listo para ser entregado.
- ✓ Entrega que es responsable de la puesta en marcha
- ✓ Mantenimiento donde se realiza el soporte para el cliente.

Se caracteriza por ser una metodología iterativa e incremental, basada en Historias de Usuario (HU), con pequeñas mejoras unas tras otras, está atenta al cambio y permite que el equipo de programación se mantenga en una interacción frecuente con el cliente o usuario. Está indicada especialmente para proyectos de pequeños equipos de trabajo, con requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta a una entrega rápida de resultados y una alta flexibilidad. Permite que el trabajo se realice de forma unida, en la misma dirección, con un objetivo claro, siguiendo el avance de las tareas a realizar [18].

### **1.10 Herramientas y tecnologías a emplear**

Para el desarrollo del módulo propuesto se definió el uso de las siguientes herramientas y tecnologías, pues constituye un requisito debido a que HMAST está desarrollada con las mismas.

#### **1.10.1 Entornos de Desarrollo Utilizado (IDE)**

Un Entorno de Desarrollo Integrado (IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Puede estar pensado para su utilización con un único lenguaje de programación o para varios de estos. Además existen entornos de desarrollo en los que se pueden utilizar varios lenguajes de programación [19].

#### **NetBeans**

Netbeans es un Entorno de Desarrollo Integrado (IDE) gratuito de código abierto, permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software [20].

Es el principal competidor en el campo de código abierto frente a Eclipse. Tiene una interfaz clara e intuitiva. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Permite identificar errores, el debugger o depurador y cuenta con un aceptable completamiento de

código. Es un producto libre y gratuito sin restricciones de uso, fundado por *Sun Microsystems* en junio de 2000 y actualmente continúa siendo el patrocinador principal de los proyectos.

Para el desarrollo de la solución se utilizará NetBeans en su versión 7.4, debido a que ofrece un rendimiento significativamente mejorado con respecto a versiones anteriores y brinda una amplia documentación. Incluye características como el detector errores y posibles soluciones para los mismos, auto completado de código y ordenamiento de código, legible y fácil de entender. Así como mejoras en Java EE, Maven, C/C++ y PHP. Además, se integra con la herramienta de modelado como Visual Paradigm, brindado de esta forma una gran ventaja al programador.

### **1.11 Herramienta CASE para el modelado**

Las herramientas CASE (*Computer Aided Software Engineering*) son diversas aplicaciones informáticas que tienen el propósito de aumentar la productividad en el desarrollo de software, disminuyendo el costo de las mismas en términos de dinero y tiempo. Pueden contribuir en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el cálculo de costos, la compilación automática, el proceso de realizar un diseño del proyecto, la implementación de parte del código automáticamente con el diseño dado, la documentación o la detección de errores, entre otras [21].

#### **Visual Paradigm para UML**

Visual Paradigm para UML es una herramienta CASE, que soporta el ciclo de vida completo del desarrollo de software. Es potente y fácil de utilizar, permite el modelado visual UML propiciando la rápida construcción de aplicaciones de calidad. 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.

Esta herramienta se caracteriza por: soporte para varias versiones de UML, una versión gratuita y otra comercial, permitir la ingeniería inversa (código a modelo, código a diagrama), permitir la generación de código (modelo a código, diagrama a código), permitir exportar el diseño de la base de datos para varios gestores y ser un producto multiplataforma [22].

##### **1.11.1 Sistema de Control de Versiones**

Un sistema de control de versiones es una combinación de tecnologías y prácticas para seguir y controlar los cambios realizados en los ficheros del proyecto, en particular en el código fuente, en

la documentación y en las páginas web. Todo este proceso se realiza manteniendo una correcta gestión sobre las versiones de la información almacenada [23].

## **RapidSVN**

RapidSVN es un cliente de interfaz gráfica, distribuida bajo la licencia GPL, para la comunicación con servidores *Subversion*<sup>9</sup>. Está escrito en C++ y distribuido bajo licencia GPL. Este facilita el versionado de ficheros, desde una interfaz sencilla e intuitiva y se encuentra disponible para plataformas *Windows*, *Linux*, *MAC OS X* y *Solaris*. Es una herramienta rápida y eficiente.

### **Características de *RapidSVN*:**

- ✓ Simple: proporciona una interfaz fácil de usar para las funciones de *Subversion*.
- ✓ Eficiente: sencilla para los principiantes, pero lo suficientemente flexible como para aumentar la productividad de los usuarios de *Subversion* con experiencia.
- ✓ Portable: se ejecuta en cualquier plataforma en la que *Subversion* y *wxWidgets* puede ejecutar *Linux*, *Windows*, *Mac OS / X*, *Solaris*, etc.
- ✓ Rápido: completamente escrito en C++.

### **1.11.2 Herramienta de diseño gráfico Pencil**

Pencil es una herramienta de código abierto y gratuito, utilizada para diseñar gráficos y animarlos en dos dimensiones. Su interfaz gráfica dispone de los elementos básicos para crear dibujos y personalizarlos al estilo deseado. Se caracteriza por poseer soporte para dibujo de diagramas y exportar los dibujos a diferentes formatos de salida. Permite el vínculo entre páginas, debido a que los elementos de un dibujo se pueden vincular a una página específica en el mismo documento [24].

## **Conclusiones Parciales**

En el desarrollo del presente capítulo fueron analizados los principales servicios antivirus existentes para ser utilizados en entornos de código abierto, seleccionando el antivirus ClamAv como herramienta más indicada para la administración del servicio en HMAST.

Además, se realizó una breve descripción del funcionamiento y las principales características de HMAST, lo que permitió tomar como punto de partida dichas consideraciones para concebir la

---

<sup>9</sup> *Subversion: es una herramienta de control de versiones basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros.*

## ***Capítulo 1: Marco teórico referencial para el módulo antivirus***

---

solución al problema planteado. Por último, fueron detalladas cada una de las herramientas, lenguajes de programación y tecnologías que quedaron definidas para el desarrollo del sistema en general, las cuales lógicamente se aplican al módulo en cuestión. Todo este proceso está guiado por la metodología SXP, mediante la cual se logra gestionar de forma precisa el desarrollo del software garantizando la calidad del producto final.



## **Capítulo 2: Análisis y diseño del módulo para HMAST**

El presente capítulo se basa en describir la propuesta del software a desarrollar, tomando como punto de partida las características del sistema base HMAST. Además, son detalladas las funcionalidades que debe cumplir la aplicación, las cuales se especifican mediante las HU. También se define y describe la arquitectura y los patrones de diseño a utilizar en el desarrollo del módulo.

### **2.1 Propuesta del módulo a desarrollar.**

Con el objetivo de darle solución al problema planteado, el presente trabajo de diploma propone desarrollar un módulo para HMAST con las funcionalidades necesarias para administrar el servicio antivirus.

El módulo a desarrollar permitirá la administración del servicio antivirus, estableciendo las principales configuraciones para su correcto funcionamiento. Brinda la posibilidad de realizar configuraciones para la actualización del antivirus, permitiendo establecer el origen del directorio de la base de datos, dirección de los *logs* y el origen de las actualizaciones. Permite además la posibilidad de conectarse a las base de datos de la actualización a través de un proxy por lo que podrá introducir los parámetros para realizar esta acción entre los que se encuentran: servidor proxy, puerto, usuario y contraseña.

Otro elemento a destacar es la posibilidad de configurar la forma en qué desea realizar la actualización, la misma puede ser de forma manual o automática, además de poder establecer en que momento va actualizar de forma automática.

A continuación se muestra el modelo de dominio para comprender con mayor claridad la propuesta de solución planteada anteriormente.

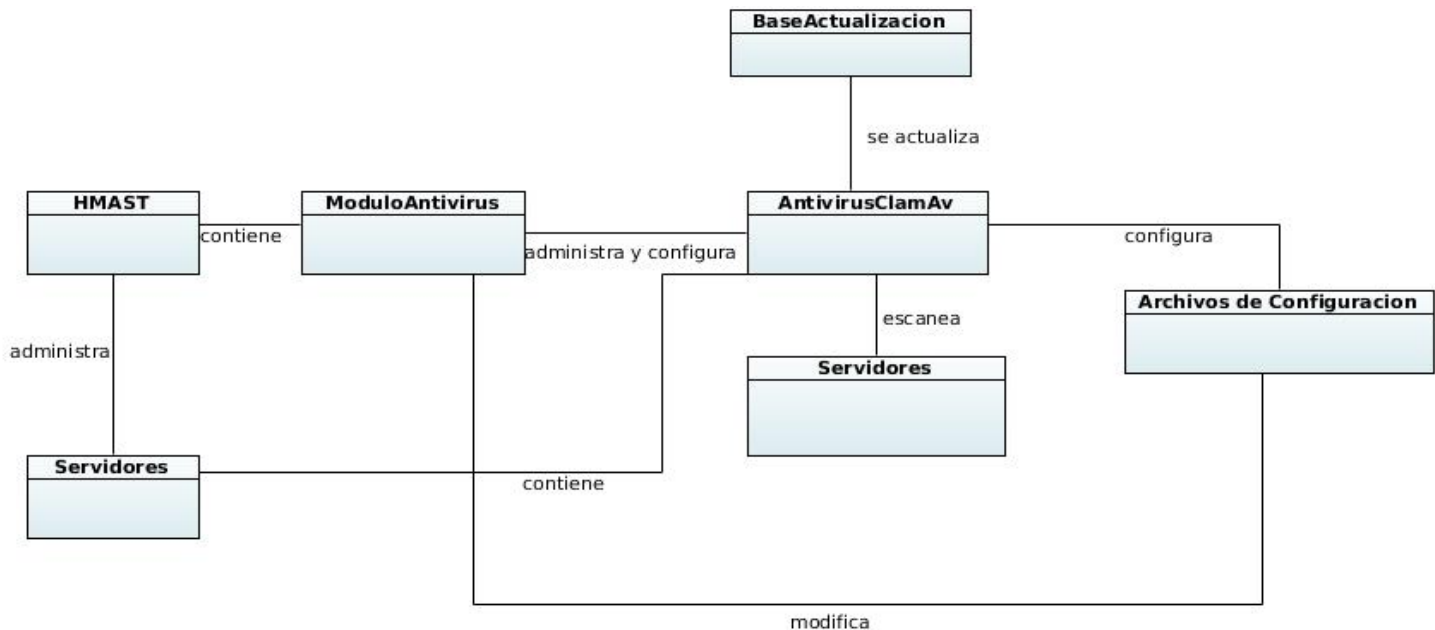


Figura 2: Modelo de dominio

## 2.2 Requisitos del sistema

Los requisitos se definen como las condiciones que el sistema debe cumplir o capacidades que debe tener, con el objetivo de establecer un entendimiento común entre el usuario y el proyecto de *software*. Los mismos se clasifican en funcionales y no funcionales. Los requisitos funcionales son las características o funcionalidades con las que cuenta el sistema [25]. Los requisitos no funcionales se definen como las cualidades o propiedades que el *software* debe tener. Debe pensarse en estas propiedades como las características que hacen un producto atractivo, usable, rápido y confiable. Generalmente están vinculados a los requisitos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse. Los clientes y usuarios pueden valorar las características no funcionales del producto, y así marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.

En el desarrollo del módulo se definen diferentes prioridades para los requisitos funcionales, considerando las exigencias del cliente y la importancia que posean para el funcionamiento del servicio en cuestión.

A continuación se expone la tabla que genera el artefacto Lista de Reserva del Producto (LRP) contenida en el expediente del proyecto del módulo a desarrollar, quedando detalladas las funcionalidades con las que debe cumplir, así como los requisitos no funcionales referentes a las categorías de diseño y software definidos para el proceso de desarrollo, debido a que las demás

clasificaciones como son seguridad, disponibilidad, rendimiento, hardware, usabilidad, soporte, entre otras, son manejadas por el sistema de forma global y quedaron descritas en el epígrafe si ciertamente son virus.

Atendiendo a las características anteriores ClamTk pudiera ser una buena opción a integrar a HMAST, sin embargo no se utiliza debido a que está programada en GTK, la cual es una librería para el desarrollo de interfaz gráfica, basada fundamentalmente en el lenguaje de programación *Perl*. HMAST está desarrollada con el lenguaje *Spring* y *Java*, por lo que su interfaz gráfica no es compatible con esta versión del antivirus.

1.7 Descripción de HMAST, se definieron 80 requisitos funcionales de ellos 54 son de prioridad alta pues con estos se garantiza el 50% de la realización del trabajo de diploma, 17 de prioridad media y 9 de prioridad baja.

A continuación se especifican los requisitos funcionales y no funcionales del módulo, así como su tiempo de duración. Los mismos están agrupados por su prioridad, se mostrarán en una tabla que se encuentra en el artefacto Lista de reserva del producto, en el que se reúnen todos los requerimientos.

Asignado a	Ítem *	Descripción	Estimación	Estimado por
		<i>Prioridad</i>	<i>Alta</i>	
Yoel Miyares Tamayo	1	Instalar el Servicio	0,1	Analista
Yoel Miyares Tamayo	2	Desinstalar el servicio	0,1	Analista
Yoel Miyares Tamayo	3	Iniciar servicio	0,1	Analista
Yoel Miyares Tamayo	4	Detener servicio	0,1	Analista
Yoel Miyares Tamayo	5	Reiniciar servicio	0,1	Analista
Yoel Miyares Tamayo	6	Mostrar dirección de los Logs	0,1	Analista
Yoel Miyares Tamayo	7	Modificar dirección de los Logs	0,1	Analista
Yoel Miyares Tamayo	8	Mostrar máxima profundidad a escanear	0,1	Analista
Yoel Miyares	9	Modificar máxima profundidad a	0,1	Analista

## Capítulo 2: Análisis y diseño del módulo para HMAST

Tamayo		escanear		
Yoel Miyares Tamayo	10	Mostrar número máximo de subprocesos	0,1	Analista
Yoel Miyares Tamayo	11	Modificar número máximo de subprocesos	0,1	Analista
Yoel Miyares Tamayo	12	Habilitar escaneo dispositivos extraíbles automáticamente	0,1	Analista
Yoel Miyares Tamayo	13	Deshabilitar escaneo dispositivos extraíbles automáticamente	0,1	Analista
Yoel Miyares Tamayo	14	Mostrar dirección de cuarentena	0,1	Analista
Yoel Miyares Tamayo	15	Modificar dirección de cuarentena	0,1	Analista
Yoel Miyares Tamayo	16	Habilitar escaneado de correo	0,1	Analista
Yoel Miyares Tamayo	17	Deshabilitar escaneado de correo	0,1	Analista
Yoel Miyares Tamayo	18	Habilitar escaneo de archivos Comprimidos	0,1	Analista
Yoel Miyares Tamayo	19	Deshabilitar escaneo de archivos Comprimidos	0,1	Analista
Yoel Miyares Tamayo	20	Habilitar escaneo de archivos Ejecutables	0,1	Analista
Yoel Miyares Tamayo	21	Deshabilitar escaneo de archivos Ejecutables	0,1	Analista
Yoel Miyares Tamayo	22	Habilitar escaneo de documentos HTML	0,1	Analista
Yoel Miyares Tamayo	23	Deshabilitar escaneo de documentos HTML	0,1	Analista
Yoel Miyares Tamayo	24	Mostrar directorio de las bases de Actualizaciones	0,1	Analista
Yoel Miyares Tamayo	25	Modificar directorio de las bases de Actualizaciones	0,1	Analista
Yoel Miyares Tamayo	26	Mostrar dirección de los logs de Actualizaciones	0,1	Analista
Yoel Miyares Tamayo	27	Modificar dirección de los logs de Actualizaciones	0,1	Analista
Yoel Miyares	28	Habilitar logs detallados	0,1	Analista

Tamayo				
Yoel Miyares Tamayo	29	Deshabilitar logs detallados	0,1	Analista
Yoel Miyares Tamayo	30	Mostrar origen de las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	31	Modificar origen de las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	32	Mostrar chequeos diarios de las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	33	Modificar chequeos diarios de las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	34	Mostrar servidor proxy para descargar las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	35	Modificar servidor proxy para descargar las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	36	Mostrar puerto para descargar las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	37	Modificar puerto para descargar las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	38	Mostrar usuario para descargar las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	39	Modificar usuario para descargar las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	40	Mostrar contraseña para descargar las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	41	Modificar contraseña para descargar las actualizaciones	0,1	Analista
Yoel Miyares Tamayo	42	Adicionar dirección donde está la actualización principal del antivirus	0,1	Analista
Yoel Miyares Tamayo	43	Mostrar dirección donde está la actualización principal del antivirus	0,1	Analista
Yoel Miyares Tamayo	44	Modificar dirección donde está la actualización principal del antivirus	0,1	Analista
Yoel Miyares Tamayo	45	Adicionar dirección donde está la actualización diaria del antivirus	0,1	Analista
Yoel Miyares Tamayo	46	Mostrar dirección de la actualización diaria del antivirus	0,1	Analista
Yoel Miyares	47	Modificar dirección de la actualización	0,1	Analista

## Capítulo 2: Análisis y diseño del módulo para HMAST

Tamayo		diaria del antivirus		
Yoel Miyares Tamayo	48	Mostrar día actualización automática	0,1	Analista
Yoel Miyares Tamayo	49	Modificar día actualización automática	0,1	Analista
Yoel Miyares Tamayo	50	Mostrar mes actualización automática	0,1	Analista
Yoel Miyares Tamayo	51	Modificar mes actualización automática	0,1	Analista
Yoel Miyares Tamayo	52	Adicionar hora para la actualización automática	0,1	Analista
Yoel Miyares Tamayo	53	Mostrar hora para la actualización automática	0,1	Analista
Yoel Miyares Tamayo	54	Modificar hora para la actualización automática	0,1	Analista
			<b>Media</b>	
Yoel Miyares Tamayo	55	Permitir el escaneo de un archivo	0,1	Analista
Yoel Miyares Tamayo	56	Permitir el escaneo de un directorio	0,1	Analista
Yoel Miyares Tamayo	57	Habilitar el análisis recursivo	0,1	Analista
Yoel Miyares Tamayo	58	Deshabilitar análisis recursivo	0,1	Analista
Yoel Miyares Tamayo	59	Adicionar dirección del fichero o directorio a analizar	0,1	Analista
Yoel Miyares Tamayo	60	Mostrar dirección del fichero o directorio a analizar	0,1	Analista
Yoel Miyares Tamayo	61	Modificar dirección del fichero o directorio a analizar	0,1	Analista
Yoel Miyares Tamayo	62	Mostrar día para realizar la tarea planificada	0,1	Analista
Yoel Miyares Tamayo	63	Modificar día para realizar la tarea planificada	0,1	Analista
Yoel Miyares Tamayo	64	Mostrar mes para realizar la tarea planificada	0,1	Analista
Yoel Miyares Tamayo	65	Modificar mes para realizar la tarea planificada	0,1	Analista

Yoel Miyares Tamayo	66	Adicionar hora para realizar la tarea planificada	0,1	Analista
Yoel Miyares Tamayo	67	Mostrar hora para realizar la tarea planificada	0,1	Analista
Yoel Miyares Tamayo	68	Modificar hora para realizar la tarea planificada	0,1	Analista
Yoel Miyares Tamayo	69	Mostrar ficheros que están en cuarentena	0,1	Analista
Yoel Miyares Tamayo	70	Restaurar ficheros que están en cuarentena	0,1	Analista
Yoel Miyares Tamayo	71	Eliminar ficheros que están en cuarentena	0,1	Analista
<b>Baja</b>				
Yoel Miyares Tamayo	72	Mostrar ficheros infectados	0,1	Analista
Yoel Miyares Tamayo	73	Eliminar ficheros infectados	0,1	Analista
Yoel Miyares Tamayo	74	Desinfectar ficheros infectados	0,1	Analista
Yoel Miyares Tamayo	75	Enviar a cuarentena ficheros infectados	0,1	Analista
Yoel Miyares Tamayo	76	Mostrar información del análisis realizado	0,1	Analista
Yoel Miyares Tamayo	77	Guardar los registros de los escaneos realizados	0,1	Analista
Yoel Miyares Tamayo	78	Mostrar los historiales de los análisis realizado	0,1	Analista
Yoel Miyares Tamayo	79	Mostrar contenido de un historial determinado	0,1	Analista
Yoel Miyares Tamayo	80	Eliminar los historiales de los análisis realizados	0,1	Analista

*Tabla 2: Requisitos funcionales del módulo a implementar.*

<b>Requisitos no Funcionales</b>	
<b>Software</b>	
✓	La aplicación se ejecutará en sistemas operativos libres (Nova, Ubuntu o Debian)
<b>Usabilidad</b>	
✓	El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.

<ul style="list-style-type: none"> <li>✓ Los mensajes y textos en las interfaces conjuntamente como los mensajes para interactuar con los usuarios, así como los mensajes de error, deberán ser en idioma castellano y tener una apariencia estándar.</li> <li>✓ El sistema debe tener acceso al menú general desde cualquiera de sus páginas.</li> <li>✓ El sistema funcionará de manera óptima en los navegadores Web, Mozilla Firefox 3.5.0 en adelante.</li> <li>✓ El sistema funcionará correctamente con dimensiones de 1024 x 768 pixeles en las ventanas de los navegadores Web.</li> </ul>
<b>Fiabilidad</b>
<ul style="list-style-type: none"> <li>✓ El sistema deberá estar disponible los 7 días de la semana durante 24 horas al día.</li> </ul>
<b>Eficiencia</b>
<ul style="list-style-type: none"> <li>✓ Tiempo de respuesta por transiciones: los tiempos de respuesta deberán ser reducidos al máximo.</li> </ul>
<b>Restricciones del Diseño</b>
<p>Se utilizará como:</p> <ul style="list-style-type: none"> <li>✓ Metodología Ágil de Desarrollo SXP.</li> <li>✓ Lenguaje de Modelado: UML.</li> <li>✓ Lenguaje de Programación: Java en su versión 7.4</li> <li>✓ Entorno Integrado de Desarrollo: Netbeans.</li> <li>✓ Framework de Desarrollo: Spring</li> <li>✓ Herramientas: Visual Paradigm, RapidSVN, Pencil.</li> </ul>
<b>Apariencia o Interfaz Externa</b>
<ul style="list-style-type: none"> <li>✓ El sistema debe implementar una interfaz Web con un diseño sencillo y debe comunicarse a través del protocolo de comunicación ssh.</li> </ul>
<b>Requisitos Legales, de Derecho de Autor y otros</b>
<ul style="list-style-type: none"> <li>✓ El sistema debe ser sometido a una evaluación y certificación por parte del cliente.</li> </ul>

*Tabla 3: Descripción de los requisitos no funcionales.*

## **Historias de Usuario (HU).**

En la metodología SXP, las HU constituyen la técnica utilizada con el propósito de especificar los requisitos del software, el equivalente a los casos de uso en el Proceso Unificado. Sirven de guía en la construcción de las pruebas de aceptación y son utilizadas para estimar tiempos de desarrollo. En este sentido, solo proveen detalles suficientes para hacer una estimación razonable del tiempo que llevará implementarlas [25].

A continuación se describen las HU que agrupan los Requisitos Funcionales de prioridad alta anteriormente descritos.

<b>Historia de Usuario</b>
----------------------------



<b>Número:</b> HMAST_Antivirus_1	<b>Nombre Historia de Usuario:</b> Gestionar instalación del servicio antivirus.						
<b>Modificación de Historia de Usuario Número:</b> Ninguna							
<b>Usuario:</b> Yoel Miyares Tamayo.	<b>Iteración Asignada:</b> 1						
<b>Prioridad en Negocio:</b> Muy Alta	<b>Puntos Estimados:</b> 0.2 semana						
<b>Riesgo en Desarrollo:</b> Muy Alto	<b>Puntos Reales:</b> 0.2 semana						
<b>Descripción:</b> Permitirá instalar y desinstalar el servicio antivirus.							
<p><b>Observaciones:</b> La aplicación le dará la opción al usuario de instalar y desinstalar el servicio.</p> <ul style="list-style-type: none"> <li>✓ <b>Instalar el servicio antivirus:</b> Debe verificar la existencia del servicio de antivirus en el servidor a través del comando <b>dpkg --get-selections clamav   grep install</b>, si no existe debe brindar la posibilidad de instalarlo. Para la instalación del servicio la aplicación utilizará el comando <b>apt-get install -y -f clamav</b>.</li> <li>✓ <b>Desinstalar el servicio antivirus:</b> Debe verificar la existencia del servicio de antivirus en el servidor con el comando <b>dpkg --get-selections clamav   grep install</b>, en caso de que esté instalado debe brindar la posibilidad de desinstalarlo. Para la desinstalación del servicio la aplicación utilizará el comando <b>apt-get remove --purge clamav</b>.</li> </ul>							
<b>Prototipo de interfaz:</b>							
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="text-align: center;">Antivirus 10.0.0.1</td> <td style="text-align: center;">DNS 10.0.0.2</td> <td style="text-align: center;">CORREO correo.uci.cl</td> <td style="text-align: center;">FTP 10.0.0.4</td> <td style="text-align: center;">WEB 10.0.0.5</td> <td style="text-align: center;">PROXY correo.uci.cl</td> </tr> </table> <div style="border: 1px solid #ccc; border-radius: 15px; padding: 20px; background-color: #e6f2ff; margin-top: 10px;"> <p style="text-align: center; color: #4f81bd;"><i>Para el funcionamiento del módulo es necesario instalar el servicio Antivirus</i></p> <div style="text-align: center; margin-top: 10px;"> <input type="button" value="Instalar ClamAv"/> </div> </div>		Antivirus 10.0.0.1	DNS 10.0.0.2	CORREO correo.uci.cl	FTP 10.0.0.4	WEB 10.0.0.5	PROXY correo.uci.cl
Antivirus 10.0.0.1	DNS 10.0.0.2	CORREO correo.uci.cl	FTP 10.0.0.4	WEB 10.0.0.5	PROXY correo.uci.cl		

Tabla 4: HU Instalación del servicio antivirus

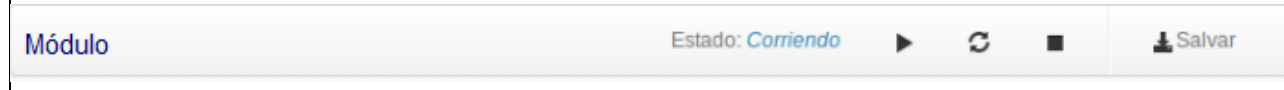
Historia de Usuario	
Número:HMAST_Antivirus_2	Nombre Historia de Usuario: Gestionar estado del servicio antivirus.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Yoel Miyares Tamayo.	Iteración Asignada: 1
Prioridad en Negocio: Muy Alta	Puntos Estimados: 0.4 semana
Riesgo en Desarrollo: Muy Alto	Puntos Reales: 0.4 semana
<p><b>Descripción:</b> Permitirá mostrar el estado en que se encuentra el servicio Antivirus, el cual puede ser Detenido o Ejecutándose. Además permitirá iniciarlo, reiniciarlo o detener el servicio.</p>	
<p><b>Observaciones:</b> Cuando se realice alguna acción en la parte superior dará la posibilidad de ver si el antivirus se encuentra detenido o si se está ejecutándose.</p> <ul style="list-style-type: none"> <li>✓ <b>Iniciar el servicio antivirus:</b> El servicio antivirus podrá ser iniciado utilizando el comando <b>service clamav-fresclam start</b>, el cual permitirá una vez instalado el servicio iniciarlo.</li> <li>✓ <b>Reiniciar el servicio antivirus:</b> Para reiniciar el servicio se utilizará el comando <b>service clamav-fresclam restart</b> el cual reiniciará el sistema con las nuevas configuraciones determinadas por el usuario.</li> <li>✓ <b>Detener el servicio antivirus:</b> El servicio podrá ser detenido utilizando el comando <b>service clamav-fresclam stop</b> esto hará que el servicio se detenga y puedan ser realizadas las configuraciones necesarias por el usuario.</li> </ul>	
<p><b>Prototipo de interfaz:</b></p> 	

Tabla 5: HU Gestionar estado del servicio

Historia de Usuario	
Número:HMAST_Antivirus_3	Nombre Historia de Usuario: Configurar parámetros generales del servicio antivirus.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Yoel Miyares Tamayo.	Iteración Asignada: 1
Prioridad en Negocio: Alta	Puntos Estimados: 2 semana
Riesgo en Desarrollo: Alto	Puntos Reales: 2 semana
<p><b>Descripción:</b> Permitirá configurar los parámetros más importantes del Antivirus.</p>	
<p><b>Observaciones:</b> El sistema mostrará una interfaz con los datos referentes a las configuraciones generales del módulo.</p>	

La interfaz mostrará uno los elementos más importantes a configurar para el correcto funcionamiento del módulo. Todas estas directivas a configurar se encuentran en el fichero de **clamd.conf** ubicado en el directorio **/etc/clamav/clamd.conf** permitiendo trabajar con las siguientes directivas:

- ✓ **Mostrar dirección de los Logs:** La aplicación permitirá mediante esta opción establecer la dirección donde serán almacenados los Logs del Antivirus a través de la directiva **LogFile**.
- ✓ **Mostrar la máxima profundidad a escanear:** La aplicación permitirá mediante esta opción permitirá establecer el nivel máximo de profundidad a la que el antivirus va a llegar a analizar mediante la directiva **MaxDirectoryRecursion**.
- ✓ **Mostrar número máximo de subprocesos a ejecutarse simultáneamente:** La aplicación permitirá mediante esta opción establecer la cantidad de subprocesos a ejecutar simultáneamente mediante la directiva **MaxThreads**.
- ✓ **Habilitar detección de dispositivos extraíbles:** La aplicación permitirá si se acepta esta opción, detectar automáticamente los dispositivos extraíbles que se conecten mediante la directiva **LogSyslog**.
- ✓ **Dirección de Cuarentena:** La aplicación permitirá mediante esta opción crear el directorio que será utilizado para la cuarentena de los virus o utilizar el que se le mostrará por defecto.
- ✓ **Habilitar el escanear de correo:** La aplicación permitirá si se acepta esta opción, escanear los mensajes de correo mediante la directiva **ScanMail**.
- ✓ **Habilitar el escaneo de archivos comprimidos:** La aplicación permitirá si se acepta esta opción, escanear los archivos que estén dentro de un comprimido mediante la directiva **ScanArchive**.
- ✓ **Habilitar el escaneo de los archivos ejecutables:** La aplicación permitirá si se acepta esta opción, escanear los archivos ejecutables mediante la directiva **ScanPE**.
- ✓ **Habilitar el escaneo de los documentos HTML:** La aplicación permitirá si se acepta esta opción, escanear los ficheros HTML mediante la directiva **ScanHTML**.

Además contará con dos botones uno **Enviar** para guardar los cambios realizados y el botón **Cancelar** para cancelar las modificaciones que se han realizados. Cuando se guardan los cambios se debe recargar el servicio de Antivirus.

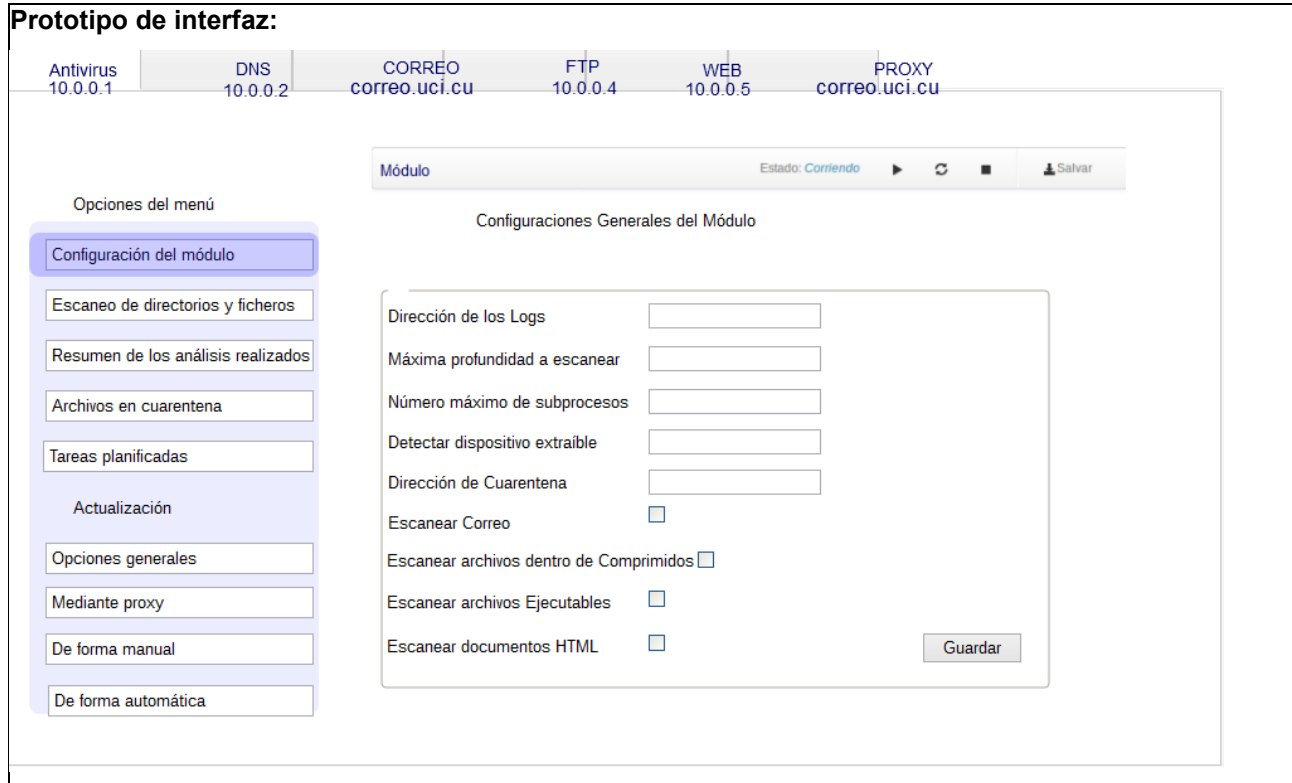


Tabla 6: HU Configuraciones generales del servicio antivirus

Historia de Usuario	
Número: HMAST_Antivirus_4	Nombre Historia de Usuario: Configurar parámetros para la actualización del servicio Antivirus.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Yoel Miyares Tamayo.	Iteración Asignada: 1
Prioridad en Negocio: Alta	Puntos Estimados: 2 semana
Riesgo en Desarrollo: Alto	Puntos Reales: 2 semana
<p><b>Descripción:</b> Permitirá configurar los parámetros más importantes a tener en cuenta a la hora de realizar la actualización del Antivirus.</p>	
<p><b>Observaciones:</b> La aplicación dará la posibilidad de realizar las configuraciones generales de las actualizaciones a través de una tabla.</p> <p>La tabla muestra los elementos más importantes a configurar para el correcto funcionamiento del módulo. Todas estas directivas a configurar se encuentran en el fichero de <b>freshclam.conf</b> ubicado en el directorio <b>/etc/clamav/freshclam.conf</b> permitiendo trabajar con las siguientes directivas:</p> <ul style="list-style-type: none"> <li>✓ <b>Directorio de las base de datos de las Actualizaciones:</b> La aplicación permitirá mediante esta opción establecer el directorio donde serán almacenadas las bases de las actualizaciones del Antivirus mediante la directiva <b>DatabaseDirectory</b>.</li> </ul>	

- ✓ **Dirección de los logs de Actualizaciones:** La aplicación permitirá mediante esta opción establecer donde serán almacenados los logs de Actualizaciones mediante la directiva **UpdateLogFile**.
- ✓ **Mostrar nivel de los Logs:** La aplicación permitirá si se acepta esta opción, que los historiales de las actualizaciones serán almacenados de manera detallada mediante la directiva **LogVerbose**.
- ✓ **Origen de las Actualizaciones:** La aplicación permitirá mediante esta opción establecer el origen de donde serán descargadas las actualizaciones mediante la directiva **DatabaseMirror**.
- ✓ **Mostrar chequeos diarios:** Cantidad de veces al día que el antivirus verifica si hay nuevas actualizaciones.

Permitirá mostrar la fecha y hora de la última actualización de forma resaltada y de otro color con el comando **date -r /var/lib/clamav/main.cvd**.

Además contará con un botón **Aceptar** para enviar los cambios realizados. Cuando se guardan los cambios se debe recargar el servicio de Antivirus y en caso contrario se cancelará lo realizado mediante el botón **Cancelar**. Cuando se guardan los cambios se debe recargar el servicio de antivirus.

**Prototipo de interfaz:**



Tabla 7: HU Configurar actualizaciones del servicio antivirus

Historia de Usuario	
Número:HMAST_Antivirus_5	Nombre Historia de Usuario: Configurar actualizaciones mediante proxy.
Modificación de Historia de Usuario Número: Ninguna	

<b>Usuario:</b> Yoel Miyares Tamayo.	<b>Iteración Asignada:</b> 1
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 2 semana
<b>Riesgo en Desarrollo:</b> Alto	<b>Puntos Reales:</b> 2 semana
<p><b>Descripción:</b> Permitirá configurar los parámetros más importantes a tener en cuenta a la hora de realizar la actualización del Antivirus.</p>	
<p><b>Observaciones:</b> La aplicación dará la posibilidad de realizar las configuraciones generales de las actualizaciones mediante proxy a través de una tabla</p> <p>La tabla dará la posibilidad de conectarse a las base de datos de la actualización a través de un proxy por lo que podrá introducir:</p> <ul style="list-style-type: none"> <li>✓ <b>Servidor proxy:</b> La aplicación permitirá mediante esta opción establecer la dirección del servidor donde se encuentra la actualización el cual es el valor que contribuye el parámetro HTTPProxyServer del fichero de <b>freshclam.conf</b> ubicado en el directorio <b>/etc/clamav/freshclam.conf</b>.</li> <li>✓ <b>Puerto:</b> La aplicación permitirá mediante esta opción establecer el puerto para acceder por el cual va a salir para actualizar el antivirus el cual corresponde al valor que contribuye el parámetro HTTPProxyPort del fichero de <b>freshclam.conf</b> ubicado en el directorio <b>/etc/clamav/freshclam.conf</b>.</li> <li>✓ <b>Usuario:</b> La aplicación permitirá mediante esta opción establecer el usuario para acceder al proxy para actualizar el antivirus el cual corresponde al valor que contribuye el parámetro HTTPProxyUsername del fichero de <b>freshclam.conf</b> ubicado en el directorio <b>/etc/clamav/freshclam.conf</b>.</li> <li>✓ <b>Contraseña:</b> La aplicación permitirá mediante esta opción establecer la contraseña para acceder al proxy para actualizar el antivirus el cual corresponde al valor que contribuye el parámetro HTTPProxyPassword del fichero de <b>freshclam.conf</b> ubicado en el directorio <b>/etc/clamav/freshclam.conf</b>.</li> </ul> <p>Todos estos parámetros hay que escribirlos en ese fichero si se desea configurar la conexión a través de proxy se debe <i>Especificar servidor proxy</i>. Se debe validar el nombre o ip del servidor entrado, el puerto que sea un entero y que el mismo se encuentre entre 0 y 65535, y el usuario, una cadena de letras y números.</p> <p>Además contará con un botón <b>Aceptar</b> para enviar los cambios realizados y en caso contrario aparecerá el botón <b>Cancelar</b> para cancelar los cambios realizados. Cuando se guardan los cambios se debe recargar el servicio de antivirus.</p>	

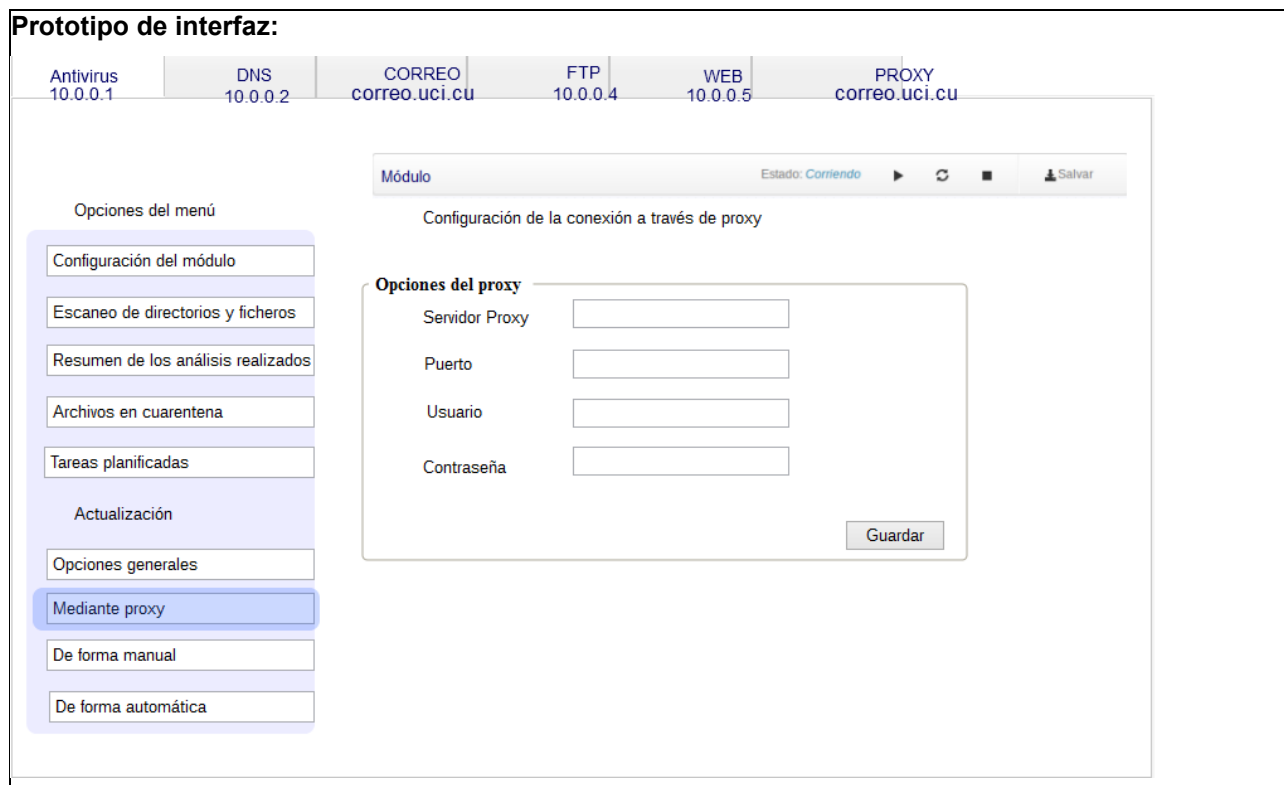


Tabla 8: HU Configurar actualizaciones del servicio antivirus mediante proxy

Historia de Usuario	
<b>Número:</b> HMAST_Antivirus_6	<b>Nombre Historia de Usuario:</b> Actualizar del antivirus de forma manual.
<b>Modificación de Historia de Usuario Número:</b> Ninguna	
<b>Usuario:</b> Yoel Miyares Tamayo.	<b>Iteración Asignada:</b> 1
<b>Prioridad en Negocio:</b> Alta	<b>Puntos Estimados:</b> 2 semana
<b>Riesgo en Desarrollo:</b> Alto	<b>Puntos Reales:</b> 2 semana
<b>Descripción:</b> Permitirá actualizar el antivirus de la forma manual configurando los parámetros más relevantes para realizar dicha acción.	
<b>Observaciones:</b> La aplicación mostrará una interfaz con los elementos necesarios para realizar la actualización de forma manual entre los que se encuentran: <b>Actualización principal (main.cvd):</b> se debe cargar el fichero de actualización main.cvd. <b>Actualización diaria (daily.cvd):</b> se debe cargar el fichero de actualización daily.cvd.	
Además contará con un botón <b>Enviar</b> para enviar los cambios realizados y en caso contrario aparecerá un botón <b>Cancelar</b> para cancelar la actualización del antivirus.	

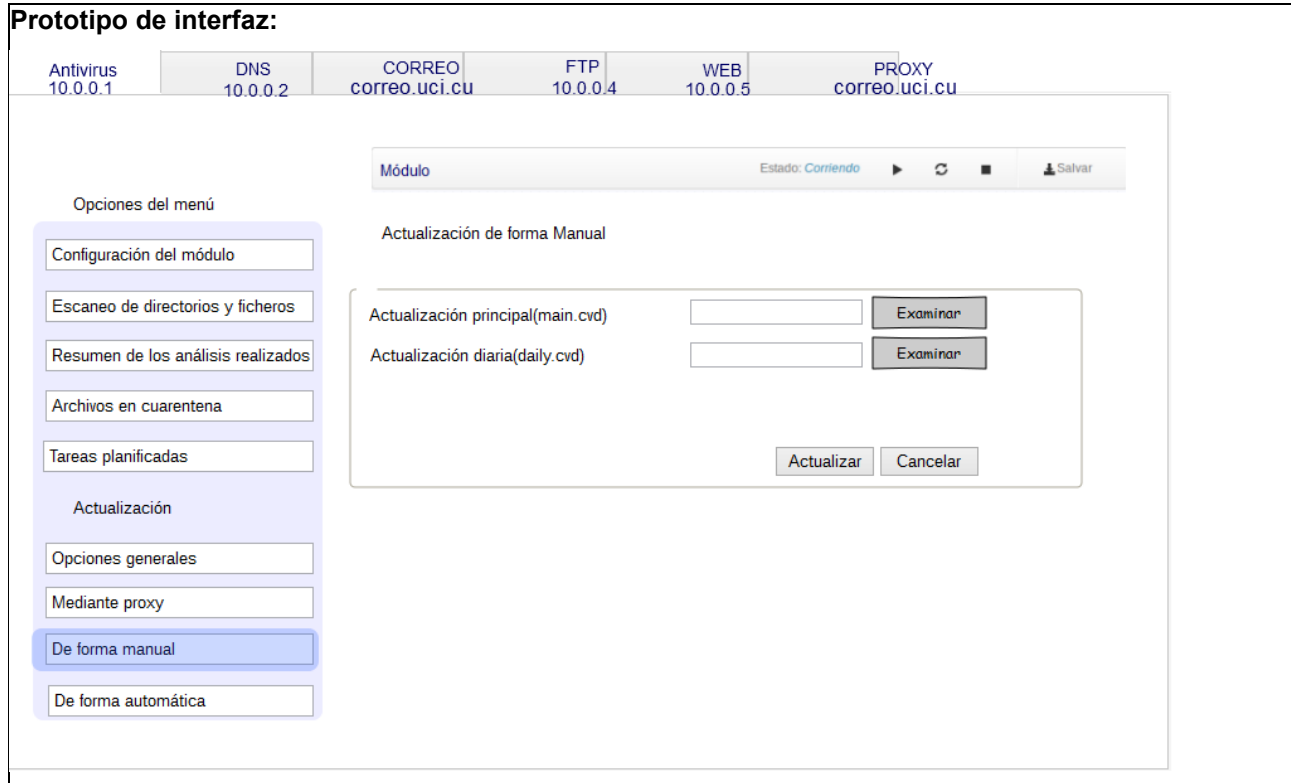


Tabla 9: HU Actualización de forma manual

Historia de Usuario	
Número:HMAST_Antivirus_7	Nombre Historia de Usuario: Actualización del antivirus de forma automática.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Yoel Miyares Tamayo.	Iteración Asignada: 1
Prioridad en Negocio: Alta	Puntos Estimados: 2 semanas
Riesgo en Desarrollo: Alto	Puntos Reales: 2 semanas
Descripción: Permitirá actualizar el antivirus de la forma automática configurando los parámetros más relevantes para realizar dicha acción.	
Observaciones: La aplicación mostrará una interfaz con los elementos necesarios para realizar la actualización de forma automática entre los que se encuentran: <ul style="list-style-type: none"> <li>✓ <b>Día de la semana:</b> Está opción brindará la posibilidad de especificar el horario en que se quiere actualizar el antivirus por lo que se debe escoger el día de la semana o especificar "Todos"</li> <li>✓ <b>Mes:</b> Está opción brindará la posibilidad de escoger el mes que el administrador desea actualizar el antivirus.</li> <li>✓ <b>Hora:</b> Está opción brindará la posibilidad de escoger la hora que el administrador desea actualizar el antivirus.</li> <li>✓ <b>Minuto:</b> Está opción brindará la posibilidad de escoger los minutos que el administrador desea</li> </ul>	



actualizar el antivirus.

Además contará con un botón **Enviar** para enviar los cambios realizados y en caso contrario aparecerá el botón **Cancelar** para cancelar los cambios realizados. Cuando se guardan los cambios se debe recargar el servicio de antivirus.

**Prototipo de interfaz:**

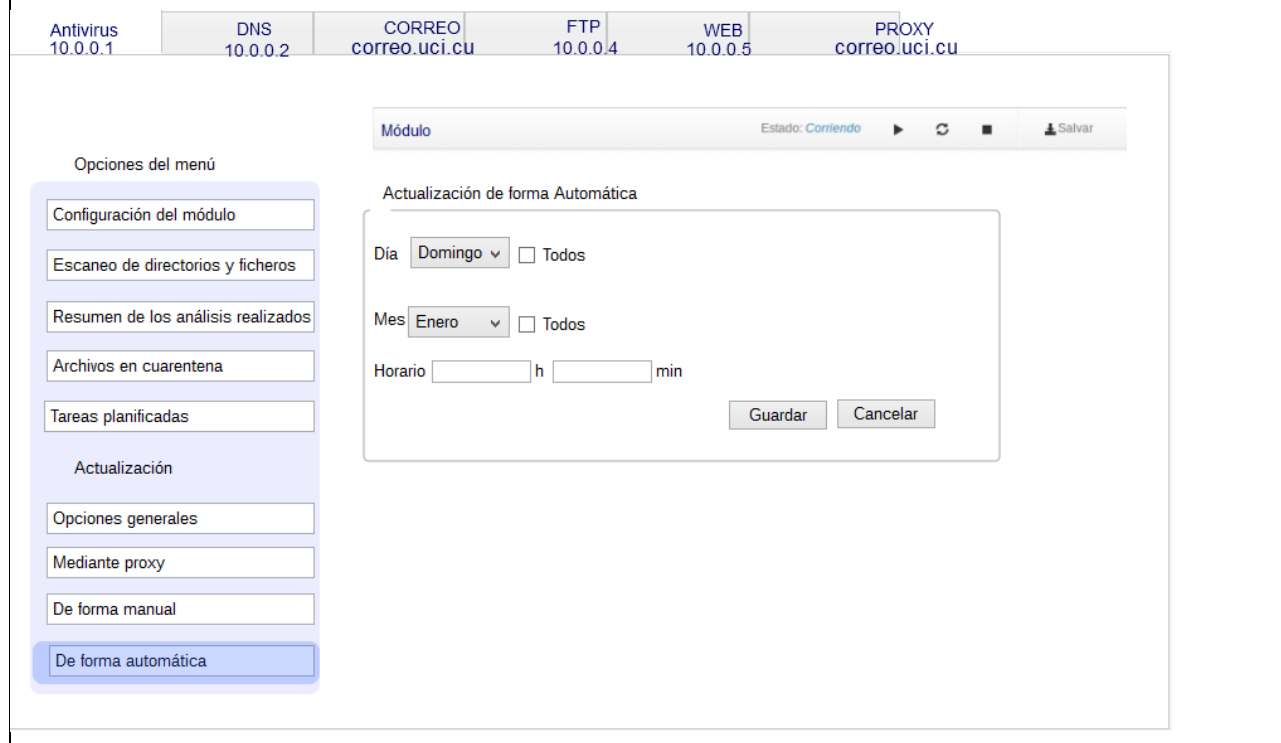


Tabla 10: HU Actualización de forma automática

### 2.3 Arquitectura del software

El diseño de la arquitectura de un sistema es el proceso por el cual se define una solución para los requisitos técnicos y operacionales del mismo. Este proceso define qué componentes forman el sistema, cómo se relacionan entre ellos, y cómo mediante su interacción llevan a cabo la funcionalidad especificada, cumpliendo con los criterios de calidad indicados como seguridad, disponibilidad, eficiencia o usabilidad [25].

#### Estilos Arquitectónicos

Un estilo arquitectónico se puede entender como un conjunto de principios que definen a alto nivel un aspecto de la aplicación. Un estilo arquitectónico está definido por un grupo de componentes, un conjunto de conexiones entre dichos componentes y restricciones sobre cómo se comunican dos componentes cualesquiera conectados. Los estilos arquitectónicos pueden organizarse según

el aspecto de la aplicación sobre el que se centran. Los principales aspectos son: comunicaciones, despliegue, dominio, interacción y estructura.

El sistema base del módulo a implementar define el uso del estilo arquitectónico orientado al dominio. A continuación se describe el funcionamiento del mismo.

### Estilo arquitectónico N-Capas Orientada al Dominio

La Arquitectura Orientada al Dominio no es solo un estilo arquitectura, es también una forma de afrontar los proyectos a nivel de trabajo del equipo de desarrollo, razón por la cual es importante el uso de una Arquitectura N-Capas Orientada al Dominio, especialmente en los casos donde el comportamiento del negocio a automatizar está sujeto a muchos cambios y evoluciones. El objetivo de esta arquitectura es 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 orientadas al dominio.

En el diseño de HMAST quedó definida la utilización de una arquitectura N-Capas orientada al Dominio. Teniendo en cuenta que la aplicación en cuestión es un módulo que será integrado a dicha herramienta, es necesario guiar el diseño por la arquitectura definida para la misma (ver Figura 3), incluyendo en este, la distribución del módulo para la administración del servicio antivirus en cada una de las capas, teniendo en cuenta las responsabilidades que cada una establezca.



Figura 3: Arquitectura del módulo

Una de las ventajas asociadas al uso de una Arquitectura N-Capas orientada al Dominio, es que facilita el mantenimiento de la aplicación, debido a que las responsabilidades están perfectamente distribuidas en las diferentes capas de la misma.

### 2.4 Diagrama de paquetes

Para la representación del módulo basada en la organización de paquetes y sus elementos, se diseñó el diagrama de paquetes tomando como marco de referencia la arquitectura propuesta. El diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas reflejando las dependencias entre las mismas. Dado que un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema [26] (ver Figura 4).

#### Sus elementos son:

Paquetes: Agrupación de elementos.

Dependencias: Indican que un elemento de un paquete requiere a otro paquete distinto.

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

El diagrama de la Figura 4 representa la distribución de los paquetes y las relaciones entre ellos. El cual está conformado por diferentes paquetes, los cuales se describen a continuación:

El paquete presentación (*presentation*) es el responsable de presentar al usuario los conceptos de negocio mediante una interfaz de usuario, facilitando la explotación de dichos procesos e informando sobre la situación de los procesos de negocio a través las reglas de validación.

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

El paquete dominio (*domain*) contiene los subpaquetes Entidades (*entities*) donde se encuentran todas las clases necesarias para representar las funcionalidades del sistema, Servicio (*service*) que define los métodos que son accedidos desde el paquete aplicación, implementados por la clase *ClamAvUpdateAppService*, quien es la encargada de realizar las validaciones de los datos antes de realizar las operaciones en el repositorio y los subpaquetes contratos de repositorios

(*repositoriescontrats*) donde se encuentra la interfaz *IClamAvUpdateRepository*, la cual tiene definidos los métodos que son accedidos desde el subpaquete Servicio y son implementados por la clase *ClamavUpdateRepository* encontrada en el paquete de persistencia (*persistens*). Esta es la responsable de cargar y salvar la información en los repositorios.

El paquete infraestructura transversal (*infrastructureCrosscutting*) almacena todo el código reutilizable de la aplicación.

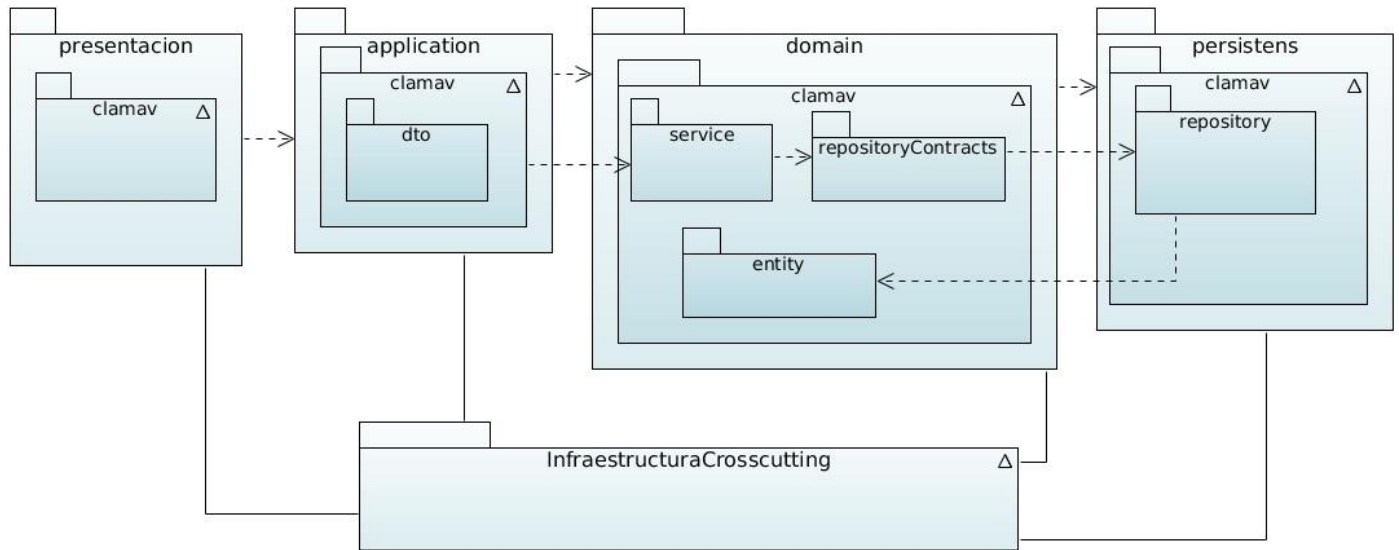


Figura 4: Diagrama de paquetes

## 2.5 Diagrama de clases

A continuación se presenta el diagrama de clases, para un mejor entendimiento del módulo a implementar. El cual está conformado por 8 clases las cuales están entrelazadas a través de las relaciones de composición.

La clase *GlobalConfiguration* es la encargada de gestionar toda la información referente al proceso de escaneo, especifica los diferentes archivos que se van a permitir para que estos sean escaneados; también permite establecer la dirección donde se guardarán los *logs* de los análisis realizados, indicando además que, en caso de que lo que se escanee sea un directorio, este análisis se realice de forma recursiva. Además permite especificar el número de subprocesos que pueden ser ejecutados al mismo tiempo, y contiene un objeto *taskList* que lista todas las tareas que se definan en la clase *Task*.

La clase *Scan* es la encargada de realizar el proceso de escanear, la misma está compuesta por 3 clases: *ScanResult*, *ScanSummary*, *scannedFile*. Cada una de ellas tiene funcionalidades específicas:

- ✓ *ScanResult* contiene dos objetos uno de tipo *infectedFileList* y el otro de tipo *scanSummary*: el primer objeto contiene una lista de todos los ficheros infectados que se muestran en la clase *scannedFile* y el otro muestra un resumen del análisis realizado con los parámetros definidos en la clase *ScanSummary*.
- ✓ *ScannedFile* contiene la dirección y el estado de los ficheros escaneados.
- ✓ *ScanSummary* contiene las funcionalidades de mostrar los directorios escaneados, el tiempo que se tardó durante el análisis, tamaño de los ficheros analizados y la cantidad de ficheros infectados.

La clase *Update* se encarga de gestionar la forma en que se va a actualizar el antivirus, mostrando así la dirección de la base de datos de las actualizaciones, permite establecer la hora, el día, y la dirección de donde se va actualizar el antivirus de forma automática. En caso de que la actualización sea de forma manual brinda la posibilidad de establecer la ruta donde se encuentra la misma. Contiene un objeto de tipo proxy para el caso que se determine descargar la clase *Update* de un sitio o de una página web, los datos del proxy son recogidos en la clase *Proxy*, donde se recogen parámetros como: puerto, usuario, contraseña y la máquina donde se encuentran las actualizaciones.

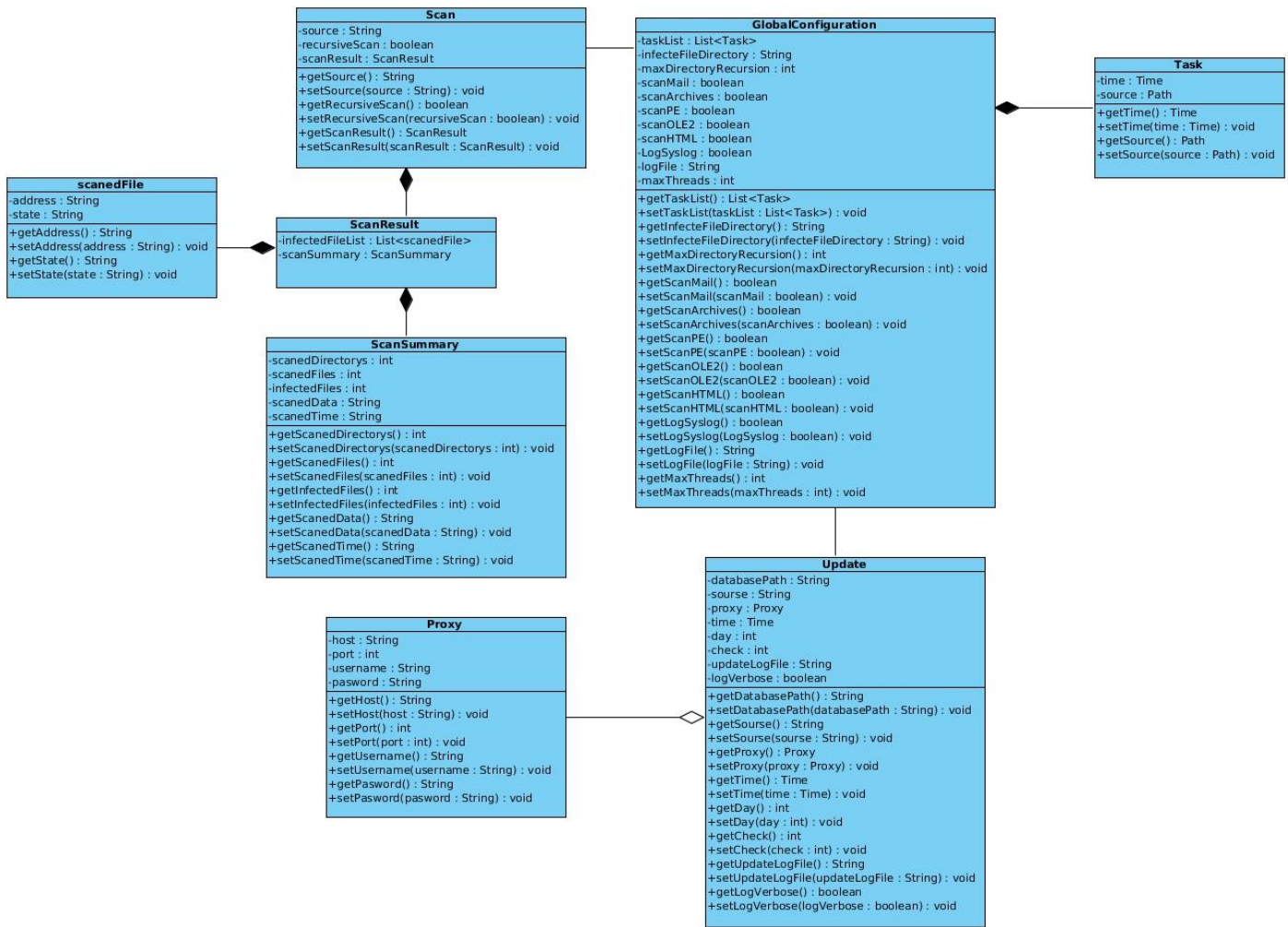


Figura 5: Diagrama de clases

## 2.6 Patrones de diseño

En la terminología de objetos, el patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de utilizarlo en circunstancias diversas. Muchos patrones ofrecen orientación sobre cómo asignar las responsabilidades a los objetos ante determinada categoría de problemas [27]. En el modelo de diseño del software se tuvieron en cuenta los patrones de asignación de responsabilidades GRASP y los patrones GOF, los cuales son descritos a continuación.

### Patrones GRASP

GRASP es un acrónimo que significa *General Responsibility Assignment Software Patterns*, del español Patrones Generales de Software para Asignar Responsabilidades. Los patrones GRASP describen los principios fundamentales para la asignación de responsabilidades a objetos,

expresado en forma de patrones [27]. A continuación, son descritos los patrones de este tipo que son utilizados para el diseño de la solución propuesta.

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

**Creador:** La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental del mismo, es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Este patrón se utilizó para la creación de los objetos y de los objetos que estas contengan como atributo en las diferentes capas del módulo lo cual se realiza a través de la inyección de dependencia<sup>10</sup>.

Un ejemplo en la aplicación es cuando la clase *ClamAvUpdateService* crea un objeto de tipo *ClamavUpdateRepository*.

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

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

---

<sup>10</sup> Inyección de dependencia: es un patrón de diseño orientado a objetos, en el que se suministran objetos a una clase en lugar de crearlos ella misma.

de distribuir de forma equitativa el peso de la complejidad, manteniendo además, la coherencia entre ellas.

Los patrones de alta cohesión y bajo acoplamiento se evidencian en el uso de la inyección de dependencia, la cual consiste en hacer que las clases del software sean independientes comunicándose únicamente a través de interfaces.

### Patrones GOF

GoF, acrónimo de *Gang o Four*, Pandilla de los Cuatro, en su traducción al español. Los patrones de diseño, conocidos como GoF se clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento [28]. A continuación es detallado el que fue utilizado en el diseño de la solución.

**Patrón Solitario (*Singleton*):** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Es usado debido a la necesidad de trabajar con el mismo objeto en distintos momentos. Este patrón se evidencia en la conexión a un servidor ya que con una única instancia de este objeto se realizan todas las operaciones sobre el servidor.

En la aplicación para establecer la conexión con el servidor que se desee administrar se hace una única instancia del objeto *SSHConection*, presente en el paquete *infrastructureCrosscutting*.

### 2.7 Conclusiones parciales

El módulo a implementar tuvo un total de 78 funcionalidades, las cuales están organizadas para conformar un total de 10 HU y 5 requisitos no funcionales. La descripción de las HU permitió un mejor entendimiento entre el programador y el cliente, llegando a un acuerdo sobre las capacidades y cualidades con las que el software debe cumplir. 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. Además se sentaron las bases para las restantes fases del proceso.



## **Capítulo 3: Desarrollo y prueba de la solución propuesta**

Los procesos de desarrollo de software implican la realización de una serie de actividades que garanticen la calidad del mismo, es por ello que primeramente se realiza la implementación y posterior a esta se define una etapa de pruebas con el objetivo de mostrar errores no detectados hasta entonces. En el presente capítulo se definen las Tareas de Ingeniería como apoyo al proceso de implementación y se describen las pruebas realizadas para la verificación de la calidad del módulo implementado.

### **3.1 Planificación de la implementación**

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

Iteración 1: La primera iteración tiene como objetivo dar cumplimiento a las HU de la 1 a la 6. Estas contemplan las funcionalidades más importantes, que responden a las características principales de un servicio antivirus.

Iteración 2: La segunda iteración tiene como objetivo dar cumplimiento a las HU de la 7 a la 12. Con la implementación de estas funcionalidades se completan las demás configuraciones básicas con las que debe contar el servidor.

### **3.2 Tareas de Ingeniería**

Las Tareas de Ingeniería se generan como artefactos en la metodología SXP, facilitando el entendimiento en el proceso de implementación debido a que se organizan del mismo modo que la definición de actividades asociadas a las HU. A continuación se definen las Tareas de Ingeniería correspondientes a cada una de las Historias de Usuario especificadas en el capítulo anterior.

En el desarrollo del módulo quedaron un total de 24 Tareas de Ingeniería de las cuales solamente se van a mencionar a continuación las que corresponden a las Historias de Usuarios anteriormente descritas en el Capítulo 2.

### Capítulo 3: Desarrollo y prueba de la solución propuesta

Tarea de Ingeniería	
<b>Número Tarea:</b> 1	<b>Número Historia de Usuario:</b> HMAST_ClamAv_1
<b>Nombre Tarea:</b> Estudiar comando para realizar la instalación del servidor.	
<b>Tipo de Tarea:</b> Desarrollo	<b>Puntos Estimados:</b> 0.2
<b>Fecha Inicio:</b> 4/02/2014	<b>Fecha Fin:</b> 5/02/2014
<b>Programador Responsable:</b> Yoel Miyares Tamayo	
<b>Descripción:</b> Se ejecuta el comando <b>apt-get install clamav</b> , el cual tiene como finalidad la instalación del servicio antivirus.	

Tabla 11: TI Instalación del servicio antivirus

Tarea de Ingeniería	
<b>Número Tarea:</b> 2	<b>Número Historia de Usuario:</b> HMAST_ClamAv_2
<b>Nombre Tarea:</b> Probar los comandos que permiten Iniciar, Detener, Recargar y Reiniciar el servicio.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 0.2
<b>Fecha Inicio:</b> 6/02/2014	<b>Fecha Fin:</b> 7/02/2014
<b>Programador Responsable:</b> Yoel Miyares Tamayo.	
<b>Descripción:</b> Los comandos que permiten gestionar el servicio son los siguientes: <ul style="list-style-type: none"> <li>- Iniciar: <b>service clamav-freshclam start</b></li> <li>- Detener: <b>service clamav-freshclam stop</b></li> <li>- Reiniciar: <b>service clamav-freshclam restart</b></li> </ul> Se ejecutan cada uno de estos comandos y se comprueban los resultados.	

Tabla 12: TI Gestionar acciones sobre el servicio antivirus

Tarea de Ingeniería	
<b>Número Tarea:</b> 3	<b>Número Historia de Usuario:</b> HMAST_ClamAv_3
<b>Nombre Tarea:</b> Probar los comandos que permite realizar el análisis con el servicio.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 0.2
<b>Fecha Inicio:</b> 8/02/2014	<b>Fecha Fin:</b> 9/02/2014

<b>Programador Responsable:</b> Yoel Miyares Tamayo.
<b>Descripción:</b> El comando que permite realizar el análisis con el servicio antivirus es el siguiente: Escanear: <b>clamscan “dirección donde se encuentra el archivo”</b> .

*Tabla 13: TI Escanear ficheros con el antivirus*

<b>Tarea de Ingeniería</b>	
<b>Número Tarea:</b> 4	<b>Número Historia de Usuario:</b> HMAST_ClamAv_4
<b>Nombre Tarea:</b> Estudiar el comandos que permite crear el fichero que sería utilizado como cuarentena.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 0.2
<b>Fecha Inicio:</b> 10/02/2014	<b>Fecha Fin:</b> 11/02/2014
<b>Programador Responsable:</b> Yoel Miyares Tamayo	
<b>Descripción:</b> El comando que permite crear el directorio que sería utilizado como cuarentena es:  Cuarentena: <b>mkdir -p /home/fulano/.clamav/viruses</b> .	

*Tabla 14: TI Crear fichero para cuarentena*

<b>Tarea de Ingeniería</b>	
<b>Número Tarea:</b> 12	<b>Número Historia de Usuario:</b> HMAST_ClamAv_4
<b>Nombre Tarea:</b> Estudiar el comandos que permite mover ficheros que han sido movidos a cuarentena.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 0.1
<b>Fecha Inicio:</b>	<b>Fecha Fin:</b>
<b>Programador Responsable:</b> Yoel Miyares Tamayo	
<b>Descripción:</b> El comando que permite mover el ficheros que han sido movidos a cuarentena es:  Mover: <b>clamscan --move=/home/fulano/.clamav/viruses \ -r “Dirección”</b>	

*Tabla 15: TI Ver ficheros que están en cuarentena*

<b>Tarea de Ingeniería</b>	
<b>Número Tarea:</b> 13	<b>Número Historia de Usuario:</b> HMAST_ClamAv_4
<b>Nombre Tarea:</b> Estudiar el comandos que permite eliminar los ficheros infectados.	
<b>Tipo de Tarea :</b> Desarrollo	<b>Puntos Estimados:</b> 0.1
<b>Fecha Inicio:</b>	<b>Fecha Fin:</b>
<b>Programador Responsable:</b> <i>Yoel Miyares Tamayo</i>	
<p><b>Descripción:</b> El comando que permite eliminar los ficheros infectados es:</p> <p>Eliminar: <b>clamscan --remove=yes \ -r “Dirección”</b></p>	

*Tabla 16: TI Eliminar ficheros infectados*

### **3.3 Estándares de codificación**

Un estándar de codificación completo comprende todos los aspectos de la generación de código, si bien los programadores deben implementar un estándar de forma prudente, este debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez [29].

#### **Estándar para nombrar las clases:**

Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto se pondrá con minúscula, cuando sea un nombre compuesto se utilizará la notación Pascal.

Ejemplo: *ClamavUpdateRepository*

#### **Estándar para nombrar las funciones:**

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se utiliza la notación Camel y con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: *loadServiceConfiguration*

#### **Estándar para nombrar las variables:**

El nombre de estas variables se escribe la primera palabra con minúscula, si es un nombre compuesto se utilizará notación Camel.

Ejemplo: *scannedFile*

### Estándar para nombrar los componentes:

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

`xxx` → *presentation, application, domain, persistence*.

`yyy` → nombre del módulo (*clamav, dhcp3, bind9*).

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

`kkk` → clases o subpaquetes.

Ejemplo: `cu.uci.hmast.domain.clamav.entitys.Update`

### 3.4 Pruebas de software

La realización de pruebas es la tarea de demostrar que un programa realiza las funciones para las cuales fue construido. Las pruebas no aseguran la ausencia de errores, sino que están orientadas a demostrar que existen defectos en el software. Estas permiten aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. Las propuestas por SXP son conocidas como pruebas de aceptación destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñadas por el cliente final. Además, se realizan pruebas unitarias las cuales se encargan de verificar el código y son diseñada por los programadores, sin embargo la metodología usada no requiere que sean plasmadas en el documento.

#### 3.4.1 Diseño de casos de prueba

El caso de prueba específica la forma de probar un sistema incluyendo las entradas, salidas y resultados esperados, así como bajo qué condiciones debe probarse el software. Para probar la calidad del módulo, fue realizada una primera etapa de pruebas correspondiente a la iteración inicial del Plan de iteraciones definido, en el cual se contemplan las HU críticas dentro de la aplicación, donde fueron identificadas diez no conformidades asociadas a errores de validación, las cuales fueron solucionadas de forma satisfactoria. Una vez culminada esta etapa, se procede a realizar una segunda fase de pruebas correspondiente a las HU definidas en la segunda iteración, identificándose dos no conformidades vinculadas con errores de sintaxis e igualmente fueron

solucionadas. El tipo de prueba que se utilizó para demostrar el correcto funcionamiento del módulo según el método seleccionado, fue las pruebas de aceptación.

### Pruebas de Aceptación

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

A continuación se describen los casos de pruebas, basados en el método de pruebas de caja negra, realizadas al módulo correspondiente a las HU mencionadas en el documento, las restantes pruebas se encuentran en el Expediente de Proyecto las cuales están contenidas dentro del artefacto que lleva por nombre Pruebas de Aceptación.


Caso de Prueba de Aceptación	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU1	<b>Nombre Historia de Usuario:</b> Gestionar instalación del servicio Antivirus.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	
<b>Descripción de la Prueba:</b> Probar que el servicio antivirus se instala.	
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus.	
<b>Entrada / Pasos de ejecución:</b> 1. Dar clic en el botón 'Instalar Clamav' 2. Lanzar una excepción cuando no se puede realizar la instalación.	
<b>Resultado Esperado:</b> Se instala el servicio antivirus.	
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>	

Tabla 17: Caso de prueba 1


Caso de Prueba de Aceptación	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU2	<b>Nombre Historia de Usuario:</b> Gestionar estado del servicio Antivirus.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	
<b>Descripción de la Prueba:</b> Probar que el servicio antivirus se puede iniciar.	
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus. El servicio antivirus está instalado.	
<b>Entrada / Pasos de ejecución:</b> 1. Abrir el servicio Antivirus 2. Dar clic en el botón ▶	

3. Lanzar una excepción cuando no se puede realizar la inicialización del servicio
<b>Resultado Esperado:</b> El servicio se inicia correctamente En la interfaz el estado se actualiza y pone 'Corriendo'
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>

*Tabla 18: Caso de prueba 2*

<b>Caso de Prueba de Aceptación</b>	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU2	<b>Nombre Historia de Usuario:</b> Gestionar estado del servicio Antivirus.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	
<b>Descripción de la Prueba:</b> Probar que el servicio antivirus se reinicia.	
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus. El servicio antivirus está instalado.	
<b>Entrada / Pasos de ejecución:</b> 1. Abrir el servicio Antivirus 2. Dar clic en el botón  3. Lanzar una excepción cuando no se puede reiniciar el servicio	
<b>Resultado Esperado:</b> Se carguen nuevos valores pero sin borrar los anteriores.	
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>	

*Tabla 19: Caso de prueba 3*

<b>Caso de Prueba de Aceptación</b>	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU2	<b>Nombre Historia de Usuario:</b> Gestionar estado del servicio Antivirus.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	
<b>Descripción de la Prueba:</b> Probar que el servicio antivirus se detiene.	
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus. El servicio antivirus está instalado.	
<b>Entrada / Pasos de ejecución:</b> 1. Abrir el servicio Antivirus 2. Dar clic en el botón  3. Lanzar una excepción cuando no se puede detener el servicio	
<b>Resultado Esperado:</b> Se detienen todas las acciones que se estén realizando con el servicio. En la interfaz el estado se actualiza y pone 'Detenido'	
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>	

*Tabla 20: Caso de prueba 4*

<b>Caso de Prueba de Aceptación</b>	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU3	<b>Nombre Historia de Usuario:</b> Configurar parámetros generales del servicio antivirus.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	

<b>Descripción de la Prueba:</b> Verificar si modifica la dirección de los logs correctamente.
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus. El servicio antivirus está instalado.
<b>Entrada / Pasos de ejecución:</b> 1. Dar clic en el botón 'Editar' 2. Se genera una ventana donde permitirá navegar por el sistema de fichero y escoger el deseado.
<b>Resultado Esperado:</b> La nueva dirección del log en la interfaz coincide con la del fichero de configuración del antivirus
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>

*Tabla 21: Caso de prueba 5*

<b>Caso de Prueba de Aceptación</b>	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU3	<b>Nombre Historia de Usuario:</b> Configurar parámetros generales del servicio antivirus.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	
<b>Descripción de la Prueba:</b> Verificar dirección de cuarentena.	
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus. El servicio antivirus está instalado.	
<b>Entrada / Pasos de ejecución:</b> 1. Dar clic en el botón 'Editar' 2. Se genera una ventana donde permitirá navegar por el sistema de fichero y escoger el deseado.	
<b>Resultado Esperado:</b> La nueva dirección del log en la interfaz coincide con la del fichero de configuración del antivirus.	
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>	

*Tabla 22: Caso de prueba 6*

<b>Caso de Prueba de Aceptación</b>	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU4	<b>Nombre Historia de Usuario:</b> Configurar parámetros para la actualización del servicio antivirus.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	
<b>Descripción de la Prueba:</b> Verificar la opción Dirección de los logs de las actualizaciones.	
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus. El servicio antivirus está instalado.	
<b>Entrada / Pasos de ejecución:</b> 1. Dar clic en el botón 'Editar' 2. Se genera una ventana donde permitirá navegar por el sistema de fichero y escoger el deseado	
<b>Resultado Esperado:</b> La dirección de los logs mostrado en la interfaz coincide con el del fichero de configuración del antivirus.	
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>	

*Tabla 23: Caso de prueba 7*



Caso de Prueba de Aceptación	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU4	<b>Nombre Historia de Usuario:</b> Configurar parámetros para la actualización del servicio antivirus.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	
<b>Descripción de la Prueba:</b> Verificar la opción Origen de las actualizaciones.	
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus. El servicio antivirus está instalado.	
<b>Entrada / Pasos de ejecución:</b> 1. Dar clic en el botón 'Editar' 2. Escribir la dirección del origen de las actualizaciones en el campo que se activa. 3. Lanzar una excepción cuando la dirección del origen de las actualizaciones es incorrecta.	
<b>Resultado Esperado:</b> La dirección origen mostrada en la interfaz coincide con el del fichero de configuración del antivirus	
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>	

*Tabla 24: Caso de prueba 8*

Caso de Prueba de Aceptación	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU5	<b>Nombre Historia de Usuario:</b> Configurar actualizaciones mediante proxy.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	
<b>Descripción de la Prueba:</b> Verificar la opción Servidor proxy.	
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus. El servicio antivirus está instalado.	
<b>Entrada / Pasos de ejecución:</b> 1. Dar clic en el botón 'Editar' 2. Escribir una URL o dirección IP en el campo que se activa. 3. Lanzar una excepción cuando la URL o la dirección IP es incorrecta.	
<b>Resultado Esperado:</b> El servidor proxy mostrado en la interfaz coincide con el del fichero de configuración del antivirus.	
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>	

*Tabla 25: Caso de prueba 9*

Caso de Prueba de Aceptación	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU5	<b>Nombre Historia de Usuario:</b> Configurar actualizaciones mediante proxy.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	
<b>Descripción de la Prueba:</b> Verificar el campo usuario.	
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus. El servicio antivirus está instalado.	

<b>Entrada / Pasos de ejecución:</b> 1. Dar clic en el botón 'Editar' 2. Escribir el nombre de usuario en el campo que se activa. 3. Lanzar una excepción cuando el nombre de usuario es incorrecto
<b>Resultado Esperado:</b> El nombre de usuario que se muestra en la interfaz coincide con el usuario registrado en el fichero de configuración del módulo.
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>

*Tabla 26: Caso de prueba 10*

<b>Caso de Prueba de Aceptación</b>	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU5	<b>Nombre Historia de Usuario:</b> Configurar actualizaciones mediante proxy.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	
<b>Descripción de la Prueba:</b> Verificar que la contraseña es correcta.	
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus. El servicio antivirus está instalado.	
<b>Entrada / Pasos de ejecución:</b> 1. Dar clic en el botón 'Editar' 2. Escribir la contraseña en el campo que se activa. 3. Lanzar una excepción cuando la contraseña es incorrecta.	
<b>Resultado Esperado:</b> La contraseña introducida se almacena en el fichero de configuración del antivirus.	
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>	

*Tabla 27: Caso de prueba 11*

<b>Caso de Prueba de Aceptación</b>	
<b>Código Caso de Prueba:</b> HMAST-Antivirus_HU6	<b>Nombre Historia de Usuario:</b> Actualización del antivirus de forma manual.
<b>Nombre de la persona que realiza la prueba:</b> Reidiel Castillo Arbelo	
<b>Descripción de la Prueba:</b> Verificar la actualización manual.	
<b>Condiciones de Ejecución:</b> Estar conectado a un servidor antivirus. El servicio antivirus está instalado.	
<b>Entrada / Pasos de ejecución:</b> 1. Dar clic en el botón 'Examinar' 2. Se genera una ventana donde permitirá navegar por el sistema de fichero y escoger el archivo a analizar.	
<b>Resultado Esperado:</b> Se analiza el fichero o directorio que se encuentra en la dirección seleccionada.	
<b>Evaluación de la Prueba:</b> <i>Satisfactoria</i>	

*Tabla 28: Caso de prueba 12*

**Pruebas realizadas por el cliente**

Con el fin de demostrar el cumplimiento del objetivo planteado y la satisfacción del cliente con el resultado obtenido, se realizaron pruebas para validar la calidad del módulo implementado. Las mismas fueron realizadas a través de una interfaz visual teniendo en cuenta que el módulo a implementar contempla interfaces donde se realizan llamadas a cada una de las funcionalidades implementadas y se evalúa el correcto funcionamiento.

#### **Resultados de las pruebas**

Como método de caja negra se utilizó la prueba de aceptación. Luego de aplicar las pruebas al finalizar la primera iteración de desarrollo, fueron detectadas 10 no conformidades relacionadas con errores de validación de entidades, las cuales fueron erradicadas en su totalidad para dar paso así a la segunda etapa de desarrollo. En esta etapa, se detectaron 2 no conformidades asociadas con errores de sintaxis. Un factor importante a tener en cuenta, es que durante el proceso de desarrollo fueron aplicadas pruebas a nivel de programador, lo cual permitió que en las pruebas finales de cada iteración de desarrollo, se obtuvieran un menor número de errores. Además, en las pruebas realizadas por el cliente se comprobó que el módulo cuenta con los requisitos establecidos por el mismo y por tanto, se encuentra listo para integrarse a HMAST.

#### **3.5 Conclusiones Parciales**

Con el desarrollo del capítulo, aplicando la metodología de desarrollo de software SXP se generaron las Tareas de Ingeniería, las cuales facilitaron el trabajo en el desarrollo de la solución, permitiendo un ahorro considerable de tiempo ya que indicaban el camino a seguir para darle cumplimiento a las Historias de Usuario. Una vez desarrolladas las funcionalidades del software, se procedió a la documentación y ejecución de las Pruebas de Aceptación basadas en el método de Caja Negra, con el objetivo de verificar la calidad del producto obteniéndose resultados satisfactorios, además de las Pruebas Unitarias las cuales corresponden al método de Caja Blanca. A partir de los resultados de estas pruebas se comprobó que el módulo se encuentra listo para ser usado en entornos reales.

## **Conclusiones**

El desarrollo del presente trabajo de diploma permitió elaborar el módulo para la Administración del servicio antivirus desde HMAST del Centro de Soluciones Libres de la Universidad de las Ciencias Informáticas, dando cumplimiento a los objetivos trazados, destacándose de manera general los siguientes aspectos:

- ✓ La aplicación antivirus ClamAv es la más adecuada para su administración desde 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 sobre la metodología SXP.
- ✓ El empleo de la arquitectura N-Capas orientada al Dominio permitió la reutilización de código e integración a HMAST.
- ✓ En los casos de prueba descritos y efectuados, se detallaron las pruebas a realizar sobre los escenarios seleccionados para comprobar el correcto funcionamiento de las funcionalidades del software. Todos los resultados arrojados por estas pruebas fueron satisfactorios, siendo la base para conocer que la aplicación, una vez integrado al sistema base, está apta para ser usada en un entorno real de producción.

## **Recomendaciones**

Como recomendación de esta investigación se plantea:

- ✓ Agregar funcionalidades que permitan el análisis en tiempo real del servicio antivirus Clamav.

## **Referencias bibliográficas**

- [1] Anón. Cubaminrex...Cuba en la Cumbre Mundial sobre la Sociedad de la Inform. [citado 5 octubre2012]. Available from world wide web:<[http://www.cubaminrex.cu/sociedad\\_informacion/cuba\\_si/Informatizacion.htm](http://www.cubaminrex.cu/sociedad_informacion/cuba_si/Informatizacion.htm)>.
- [2] Castro Ruz, Fidel. Discurso pronunciado por el Comandante Fidel Castro Ruz, Primer Ministro del Gobierno Revolucionario, en el acto celebrado por la Sociedad Espeleológica de Cuba, en la Academia de Ciencias, el 15 de enero de 1960. enero 1960.[ Accessed 19 April 2014]. Available from world wide web: <<http://www.cuba.cu/gobierno/discursos/1960/esp/f150160e.html>>.
- [3] Definición de Antivirus. [Accessed 26 April 2014]. Available from world wide web: <http://definicion.mx/antivirus/#ixzz32oVtollJ>
- [4] Tipos de malware y otras amenazas informáticas « Xpress Hosting Blog. [online]. [Accessed 3 April 2014]. Available from: <http://blog.xpress.com.mx/2013/05/tipos-de-malware-y-otras-amenazas-informaticas/>
- [5] Funcionamiento de un programa antivirus. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://www.vsantivirus.com/fdc-funcionamiento-antivirus.htm>
- [6] Informática para la Administración (La Valverde : D): Antivirus, clasificación de antivirus y tipos de antivirus, virus y tipos de virus. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://thalyvalverde.blogspot.com/2012/01/antivirus.html>
- [7] Security in Computing - Charles P. Pfleeger, Shari Lawrence Pfleeger - Google Libros. [online]. [Accessed 1 April 2014]. Available from: [http://www.google.com/cu/books?hl=es&lr=&id=O3VB-zspJo4C&oi=fnd&pg=PR19&dq=Security+in+Computing&ots=pQ6wOloG2I&sig=kMwpVY8PeoyOtqCoWINmcNa7aAM&redir\\_esc=y#v=onepage&q=Security%20in%20Computing&f=false](http://www.google.com/cu/books?hl=es&lr=&id=O3VB-zspJo4C&oi=fnd&pg=PR19&dq=Security+in+Computing&ots=pQ6wOloG2I&sig=kMwpVY8PeoyOtqCoWINmcNa7aAM&redir_esc=y#v=onepage&q=Security%20in%20Computing&f=false)
- [8] - Res 127 2007 MIC Reglamento de Seguridad Informatica.pdf [online]. [Accessed 1 April 2014]. Available from: <http://seguridad.uci.cu/sites/default/files/Res%20127%202007%20MIC%20Reglamento%20de%20Seguridad%20Informatica.pdf>
- [9] Copyright© Segurmática 2001-2012.SAVUnix Milter. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://www.segurmatica.cu/laboratorio/lab1.jsp>

- [10] Mail Server Security | Antivirus & Spam Filter | Kaspersky Lab. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://www.kaspersky.com/business-security/mail-server>
- [11] Clam AntiVirus. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://www.clamav.net/lang/en/>
- [12] Anon. Las PYMES y las TICs. [10 Mayo 2014]. Disponible en la Web: <http://doc.zentyal.org/es/presentation.html>
- [13] ClamTk Virus Scanner. [online]. 18 February 2014. [Accessed 10 February 2014]. Available from: <http://www.clamav.net/lang/en/>
- [14] Lenguaje de programación / definición de Lenguaje de programación - Un lenguaje de programación es una técnica estándar de comunicaci. [online]. 19 March 2014. [Accessed 19 March 2014]. Available from: [http://www.diclib.com/lenguaje%20de%20programaci%C3%B3n/show/es/es\\_wiki\\_10/74813](http://www.diclib.com/lenguaje%20de%20programaci%C3%B3n/show/es/es_wiki_10/74813)
- [15] KATHY SIERRA Y BERT BATES. Java Head First. 2da Edición. [no date]. 677
- [16] CRAIG WALLS. Spring in Action [online]. MEAP Edition. 2010. [Accessed 13 February 2014]. Available from: <http://www.manning-sandbox.com/forum.jspa?forumID=573>
- [17] CRAIG WALLS. Spring in Action [online]. MEAP Edition. 2010. [Accessed 13 February 2014]. Available from: <http://www.manning-sandbox.com/forum.jspa?forumID=573>.
- [18] GLADYS MARSÍ PEÑALVER ROMERO. Metodología ágil para proyectos de software libre. Ingeniera Informática. Habana: Universidad de las Ciencias Informáticas, 2008.80
- [19] Entorno de Desarrollo Integrado (IDE). | fergarcia. [online]. [Accessed 12 February 2014]. Available from: <http://fergarcia.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/>
- [20] NetBeans IDE - Overview. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <https://netbeans.org/features/index.html#o6>
- [21] Simple Tools for Software Modeling -OR- It's "Use the Simplest Tool" not "Use Simple Tools." [online]. [Accessed 3 April 2014]. Available from: <http://www.agilemodeling.com/essays/simpleTools.htm#SelectingCASE>

- [22] UML, BPMN and Enterprise Architecture Tool for Software Development. [online]. [Accessed 3 April 2014]. Available from: <http://www.visual-paradigm.com/>
- [23] RapidSVN. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://www.rapidsvn.org/>
- [24] Pencil Project, 2013. <http://pencil.evolus.vn/>
- [25] CÉSAR DE LA TORRE LLORENTE, UNAI ZORRILLA CASTRO, MIGUEL ANGEL RAMOS BARROS, JAVIER CALVARRO NELSON. Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0 [online]. 2014. [Accessed 10 February 2014]. ISBN 978-84-936696-3-8. Available from: [http://www.campusmvp.com/catalogo/Product-Gu%EF%BF%BDA-de-Arquitectura-N-Capas-orientada-al-Dominio-con-.NET-4.0---Microsoft\\_109.aspx](http://www.campusmvp.com/catalogo/Product-Gu%EF%BF%BDA-de-Arquitectura-N-Capas-orientada-al-Dominio-con-.NET-4.0---Microsoft_109.aspx)
- [26] UML - Lenguaje de Modelado Unificado. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://kuainasi.ciens.ucv.ve/ads2010-2/uml/index.html#>
- [27] CRG LARMAN. UML y Patrones, Introducción al análisis y diseño orientado a objetos [online]. 2014. [Accessed 19 March 2014]. ISBN 970-17-0261-1. Available from: [http://www.freownloadmanager.org/es/downloads/Paradigma\\_Visual\\_para\\_UML\\_\(M%C3%8D\)\\_14720\\_pAI/](http://www.freownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_pAI/)
- [28] Patrones GoF. [online]. 19 March 2014. [Accessed 19 March 2014]. Available from: <http://geektheplanet.net/5462/patrones-gof.xhtml>
- [29] Revisiones de código y estándares de codificación. [online]. [Accessed 9 April 2014]. Available from: [http://msdn.microsoft.com/es-es/library/aa291591\(v=vs.71\).aspx](http://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx)
- [30] Pruebas de Software. [En línea] [Consultado el: 31 de Enero de 2014] Gestión de calidad y Pruebas de Software. 2005. Habilitado desde: <http://www.pruebasdesoftware.com/pruebadeacceptacion.htm>



**Bibliografía**

1. Tipos de malware y otras amenazas informáticas « Xpress Hosting Blog. [online]. [Accessed 3 April 2014]. Available from: <http://blog.xpress.com.mx/2013/05/tipos-de-malware-y-otras-amenazas-informaticas/>
2. Funcionamiento de un programa antivirus. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://www.vsantivirus.com/fdc-funcionamiento-antivirus.htm>
3. Copyright© Segurmática 2001-2012.SAVUnix Milter. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://www.segurmatica.cu/laboratorio/lab1.jsp>
4. Informática para la Administración (La Valverde :D): Antivirus, clasificación de antivirus y tipos de antivirus, virus y tipos de virus. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://thalyvalverde.blogspot.com/2012/01/antivirus.html>
5. Security in Computing - Charles P. Pfleeger, Shari Lawrence Pfleeger - Google Libros. [online]. [Accessed 1 April 2014]. Available from: [http://www.google.com/cu/books?hl=es&lr=&id=O3VB-zspJo4C&oi=fnd&pg=PR19&dq=Security+in+Computing&ots=pQ6wOloG2l&sig=kMwpVY8PeoyOtqCoWINmcNa7aAM&redir\\_esc=y#v=onepage&q=Security%20in%20Computing&f=false](http://www.google.com/cu/books?hl=es&lr=&id=O3VB-zspJo4C&oi=fnd&pg=PR19&dq=Security+in+Computing&ots=pQ6wOloG2l&sig=kMwpVY8PeoyOtqCoWINmcNa7aAM&redir_esc=y#v=onepage&q=Security%20in%20Computing&f=false)
6. Is Stuxnet the “best” malware ever? | Kaspersky Lab US. [online]. 21 March 2014. [Accessed 21 March 2014]. Available from: <http://usa.kaspersky.com/about-us/press-center/in-the-news/stuxnet-best-malware-ever>
7. Kaspersky Lab: Same Countries Behind Stuxnet And Flame Malware. [online]. [Accessed 1 April 2014]. Available from: <http://www.forbes.com/sites/kenrapoza/2012/06/11/kaspersky-lab-same-countries-behind-stuxnet-and-flame-malware/>
8. Endpoint, Cloud, Mobile & Virtual Security Solutions | Symantec. [online]. 24 March 2014. [Accessed 24 March 2014]. Available from: <http://www.symantec.com/index.jsp>
9. Siemens: Stuxnet worm hit industrial systems - Computerworld. [online]. [Accessed 1 April 2014]. Available from: [http://www.computerworld.com/s/article/9185419/Siemens\\_Stuxnet\\_worm\\_hit\\_industrial\\_systems](http://www.computerworld.com/s/article/9185419/Siemens_Stuxnet_worm_hit_industrial_systems)

10. The Flame: Questions and Answers - Securelist. [online]. [Accessed 1 April 2014]. Available from: [http://www.securelist.com/en/blog/208193522/The\\_Flame\\_Questions\\_and\\_Answers](http://www.securelist.com/en/blog/208193522/The_Flame_Questions_and_Answers)
11. La Jornada: En 2013 se crearon al menos 50 millones de virus informáticos. [online]. [Accessed 3 April 2014]. Available from: <http://www.jornada.unam.mx/2013/12/30/economia/021n1eco>
12. - Res 127 2007 MIC Reglamento de Seguridad Informatica.pdf [online]. [Accessed 1 April 2014]. Available from: <http://seguridad.uci.cu/sites/default/files/Res%20127%202007%20MIC%20Reglamento%20de%20Seguridad%20Informatica.pdf>
13. Mail Server Security | Antivirus & Spam Filter | Kaspersky Lab. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://www.kaspersky.com/business-security/mail-server>
14. Clam AntiVirus. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://www.clamav.net/lang/en/>
15. Lenguaje de programación / definición de Lenguaje de programación - Un lenguaje de programación es una técnica estándar de comunicaci. [online]. 19 March 2014. [Accessed 19 March 2014]. Available from: [http://www.diclib.com/lenguaje%20de%20programaci%C3%B3n/show/es/es\\_wiki\\_10/74813](http://www.diclib.com/lenguaje%20de%20programaci%C3%B3n/show/es/es_wiki_10/74813)
16. Características del lenguaje Java. [online]. [Accessed 16 March 2014]. Available from: <http://www.iec.csic.es/criptonomicon/java/quesjava.html>
17. ¿Qué es un Framework? Global Mentoring - Cursos Java Online |. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://globalmentoring.com.mx/cursos-java/java-frameworks/que-es-un-framework/>
18. Spring Framework. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://projects.spring.io/spring-framework/>
19. PEÑALVER ROMERO, GLADYS MARSI. Metodología ágil para proyectos de software libre. UCI, 2008.
20. NetBeans IDE - Overview. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <https://netbeans.org/features/index.html#o6>

21. Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm for UML (ME)) por Visual Paradigm International Ltd. - reporte y descarga. [online]. 19 March 2014. [Accessed 19 March 2014]. Available from: [http://www.freownloadmanager.org/es/downloads/Paradigma\\_Visual\\_para\\_UML\\_\(M%C3%8D\)\\_14720\\_p/](http://www.freownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/)
22. Control de Versiones. [online]. 19 March 2014. [Accessed 19 March 2014]. Available from: <http://producingoss.com/es/vc.html>
23. RapidSVN. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://www.rapidsvn.org/>
24. Programación Orientada a Aspectos - Fernando Córdoba. [online]. [Accessed 1 April 2014]. Available from: <http://www.slideshare.net/2008PA2Info3/programacin-orientada-a-aspectos-fernando-crdoaba>
25. PEÑALVER ROMERO, GLADYS MARSÍ. Metodología ágil para proyectos de software libre. UCI, 2008. Página 9
26. CÉSAR DE LA TORRE LLORENTE, UNAI ZORRILLA CASTRO, MIGUEL ANGEL RAMOS BARROS, JAVIER CALVARRO NELSON. Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0 [online]. 2014. [Accessed 10 February 2014]. ISBN 978-84-936696-3-8. Available from: [http://www.campusmvp.com/catalogo/Product-Gu%EF%BF%BDA-de-Arquitectura-N-Capas-orientada-al-Dominio-con-.NET-4.0---Microsoft\\_109.aspx](http://www.campusmvp.com/catalogo/Product-Gu%EF%BF%BDA-de-Arquitectura-N-Capas-orientada-al-Dominio-con-.NET-4.0---Microsoft_109.aspx)
27. UML - Lenguaje de Modelado Unificado. [online]. 10 February 2014. [Accessed 10 February 2014]. Available from: <http://kuainasi.ciens.ucv.ve/adsi2010-2/uml/index.html#>
28. CRAIG LARMAN. UML y Patrones, Introducción al análisis y diseño orientado a objetos [online]. 2014. [Accessed 19 March 2014]. ISBN 970-17-0261-1. Available from: [http://www.freownloadmanager.org/es/downloads/Paradigma\\_Visual\\_para\\_UML\\_\(M%C3%8D\)\\_14720\\_p/](http://www.freownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/)
29. Patrones GoF. [online]. 19 March 2014. [Accessed 19 March 2014]. Available from: <http://geektheplanet.net/5462/patrones-gof.xhtml>

30. Revisiones de código y estándares de codificación. [online]. [Accessed 9 April 2014]. Available from: [http://msdn.microsoft.com/es-es/library/aa291591\(v=vs.71\).aspx](http://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx)
  
31. Las pruebas de software forman el único instrumento adecuado para determinar la calidad de software. [online]. [Accessed 9 April 2014]. Available from: <http://pruebasdesoftware.com/laspruebasdesoftware.htm>

## Glosario de términos

**DHCP:** Protocolo de configuración de *host* dinámico. Permite que un equipo conectado a una red pueda obtener su configuración de red en forma dinámica.

**Directivas:** son variables almacenadas en el archivo de texto de configuración para alterar y controlar el funcionamiento de Apache en tiempo de ejecución según sus valores y después de haber reiniciado el proceso servidor.

**Framework:** es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Un *framework* puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

**Historias de Usuario (HU):** secuencias de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de las secuencias.

**IDE:** un Entorno de Desarrollo Integrado, también IDE por sus siglas en inglés de *Integrated Development Environment*, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

**Licencia GPL:** la Licencia Pública General de GNU pretende garantizarle la libertad de compartir y modificar software libre, para asegurar que el software es libre para todos sus usuarios. Esta Licencia Pública General se aplica a la mayor parte del software de la *Free Software Foundation* y a cualquier otro programa si sus autores se comprometen a utilizarla.

**Malware:** se llama *malware* al software que se encarga de hacer que cualquier programa instalado en el ordenador no funcione correctamente.

**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.

**Sistemas operativos GNU/Linux:** el sistema operativo Linux junto con las utilidades del proyecto GNU constituyen lo que se conoce como GNU/Linux.

**Spam:** se llama *spam*, correo basura o mensaje basura a los mensajes no solicitados, no deseados o de remitente no conocido (correo anónimo), habitualmente de tipo publicitario, generalmente enviados en grandes cantidades que perjudican de alguna o varias maneras al receptor.

**SSH:** *Secure SHell* es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red.

**Unix:** Sistema operativo multitarea y multiusuario, lo cual significa que puede ejecutar varios programas simultáneamente, así como gestionar a varios usuarios de forma concurrente.

## Anexos



Figura 6: Interfaz de instalación del antivirus

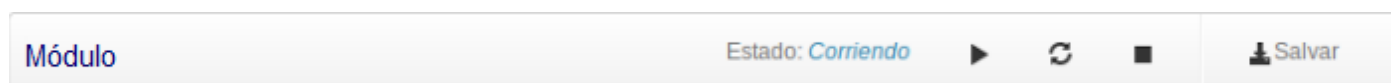


Figura 7: Interfaz gestionar estado del antivirus

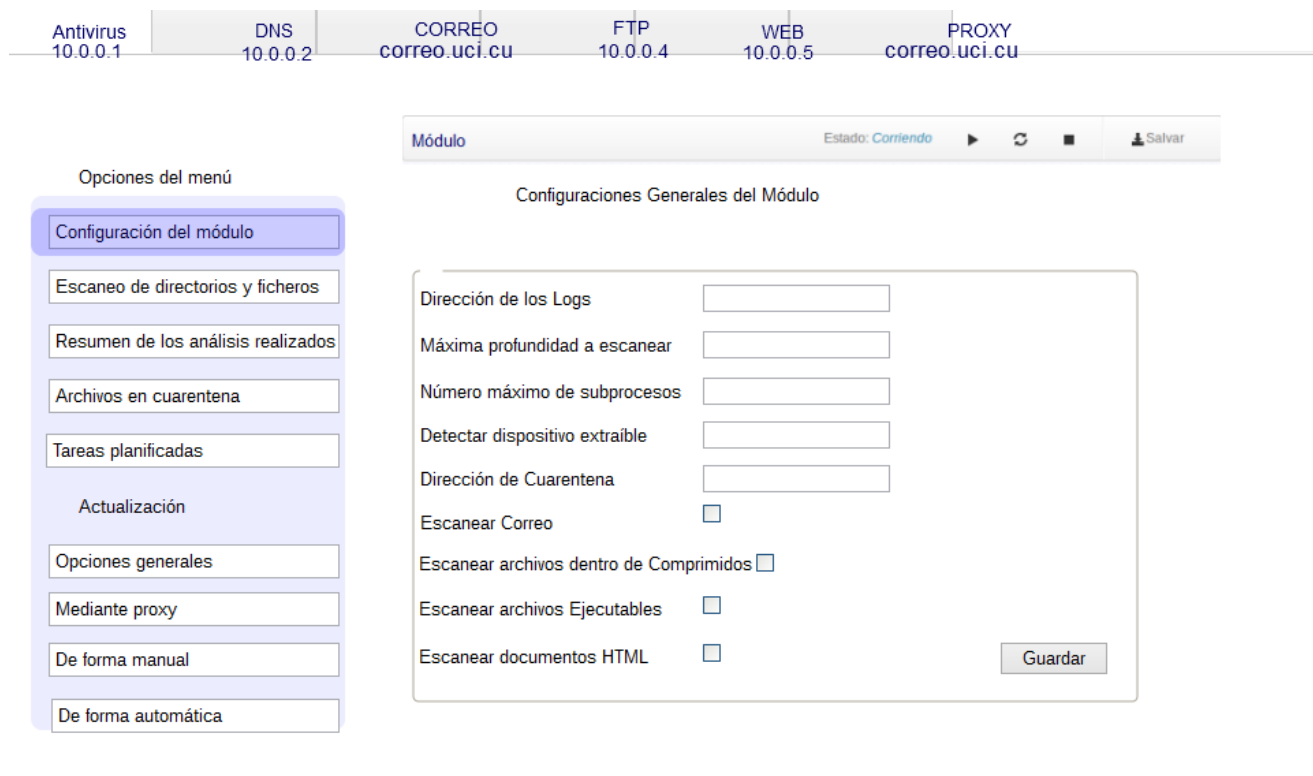


Figura 8: Interfaz de la configuración general del antivirus

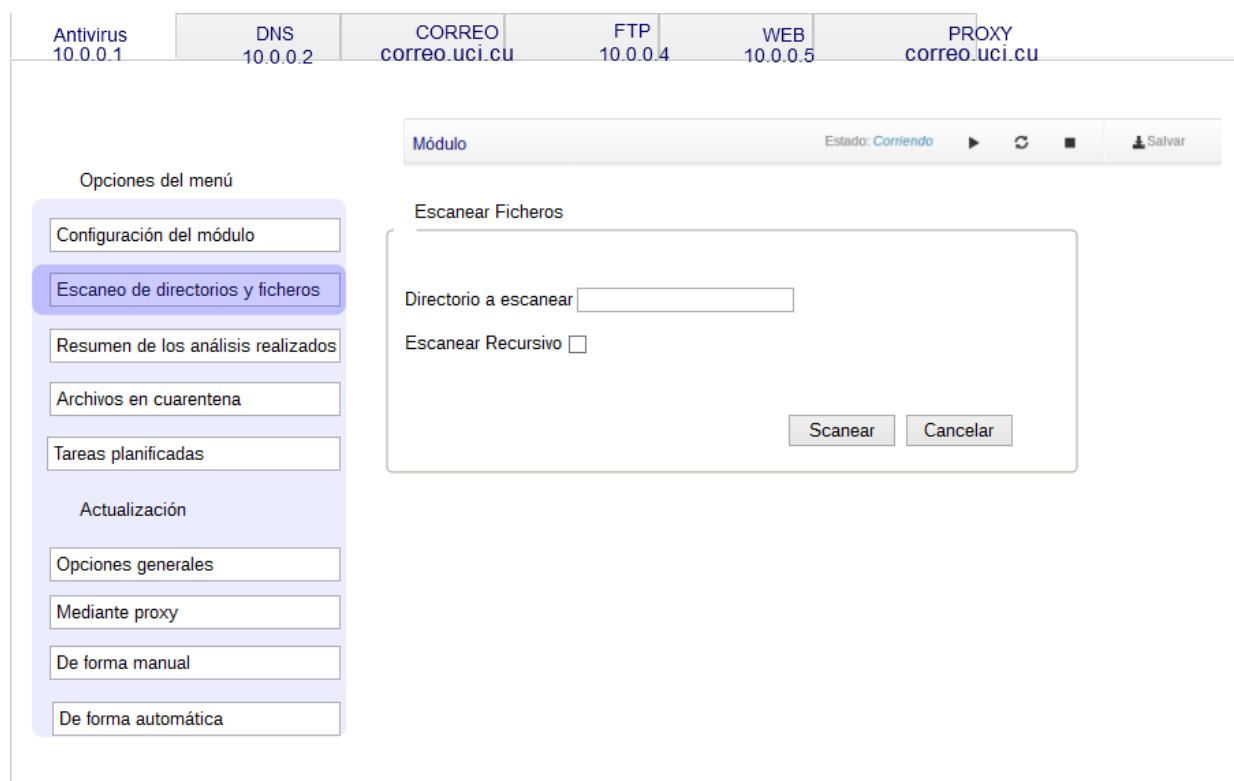


Figura 9: Interfaz del escaneo del antivirus

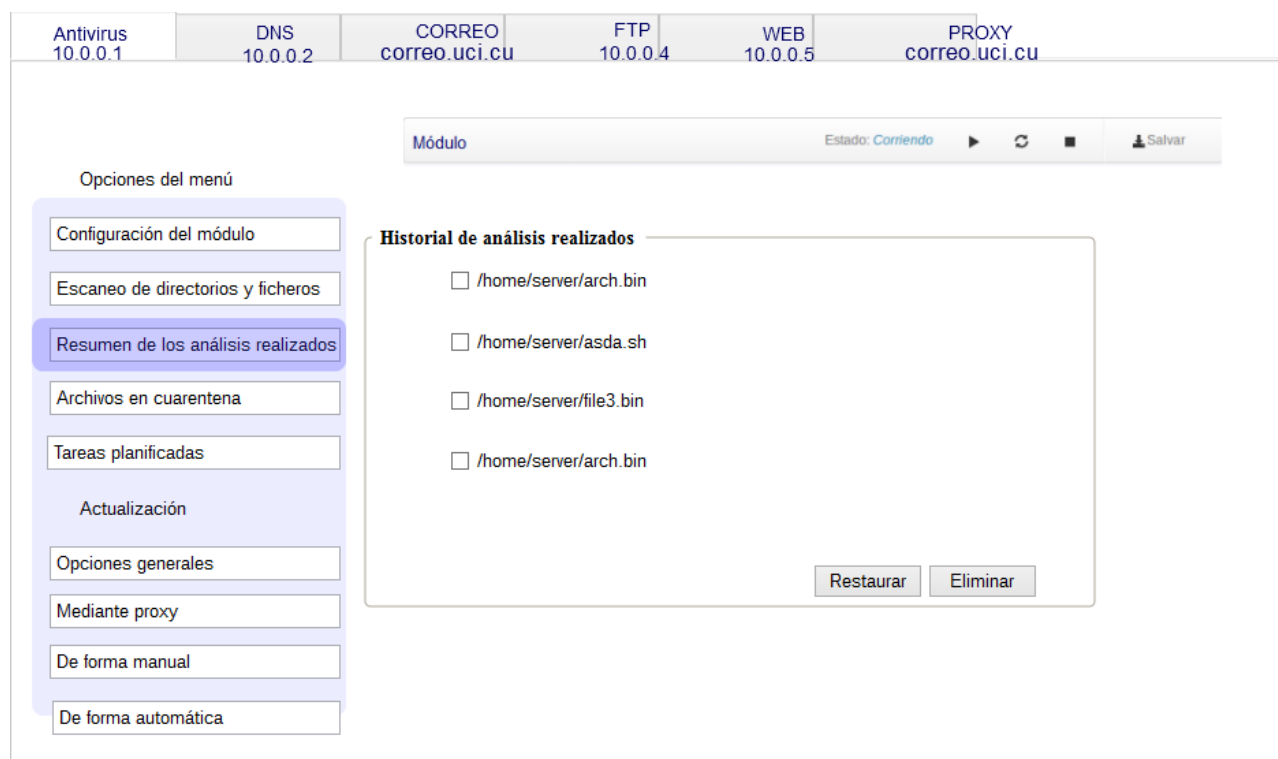


Figura 10: Interfaz del resumen de los análisis realizados



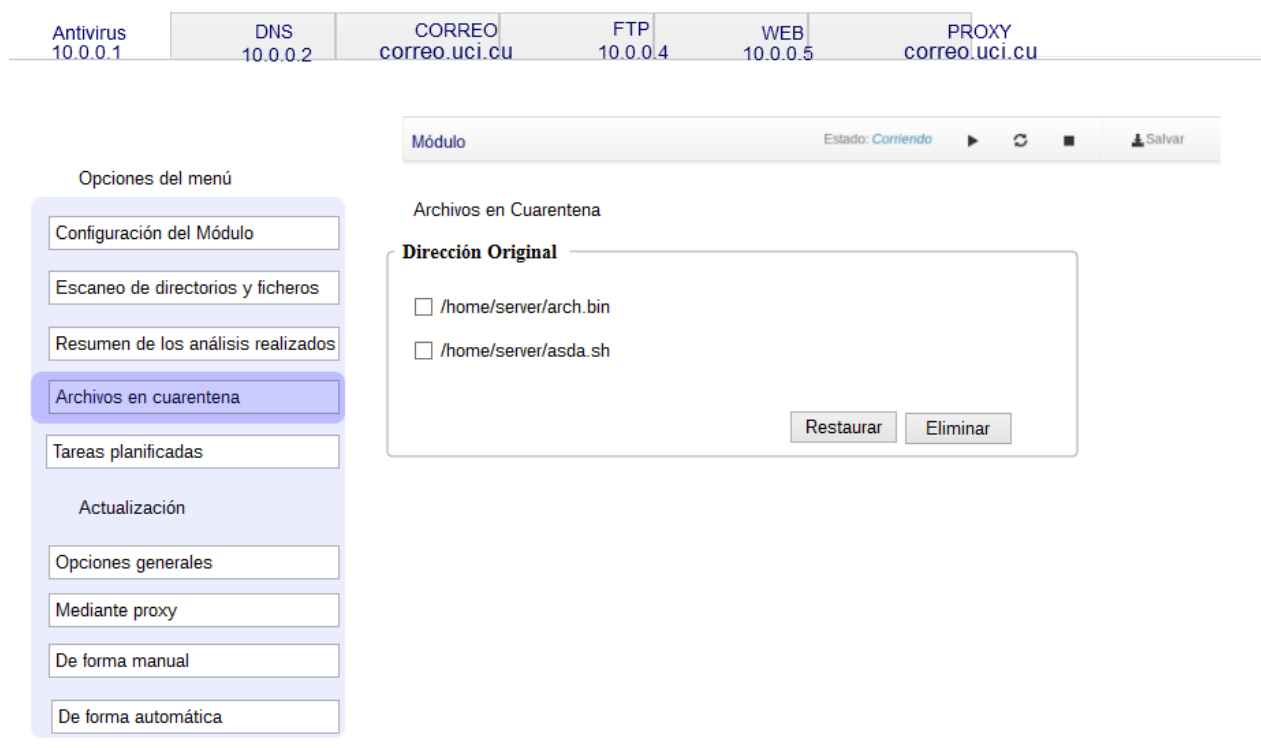


Figura 11: Interfaz de los ficheros en cuarentena

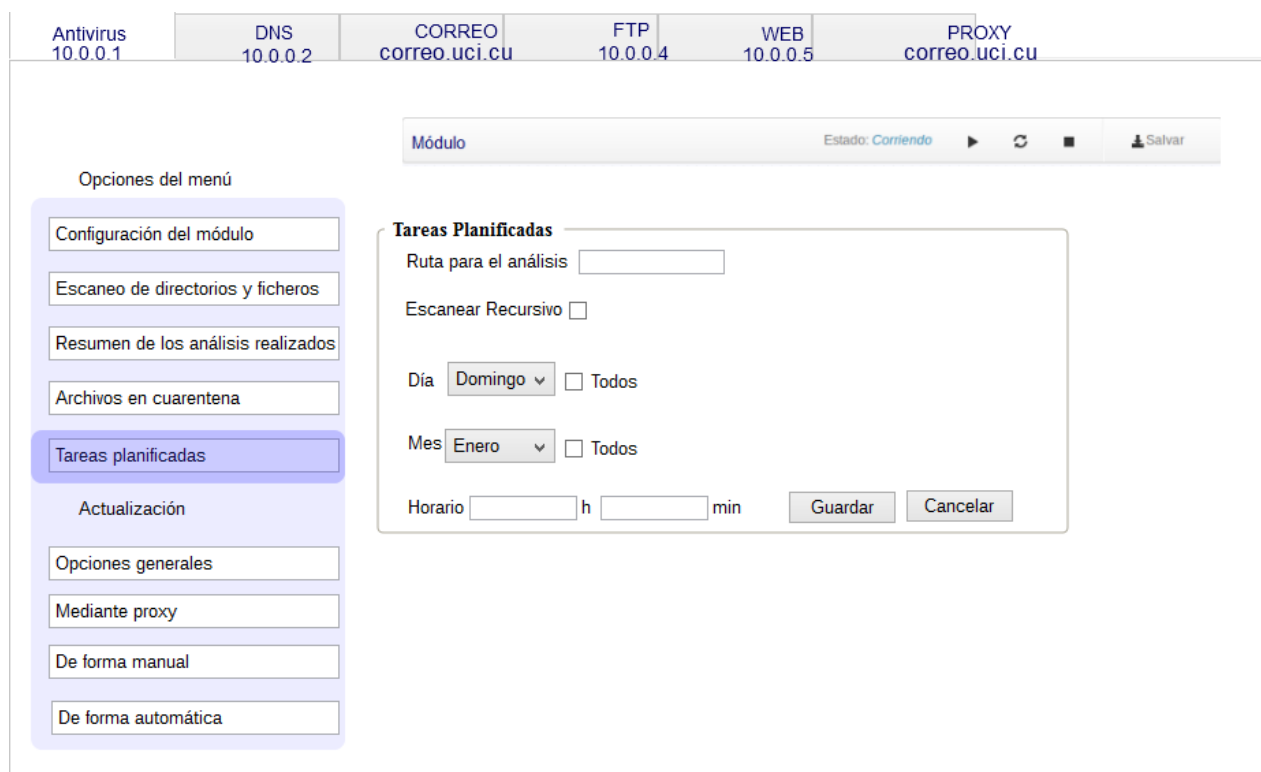


Figura 12: Interfaz de las tareas planificadas

Antivirus 10.0.0.1	DNS 10.0.0.2	CORREO correo.uci.cu	FTP 10.0.0.4	WEB 10.0.0.5	PROXY correo.uci.cu
-----------------------	-----------------	-------------------------	-----------------	-----------------	------------------------

Módulo Estado: Corriendo ▶ ↻ ◼ ⬇ Salvar

Opciones del menú

- Configuración del módulo
- Escaneo de directorios y ficheros
- Resumen de los análisis realizados
- Archivos en cuarentena
- Tareas planificadas
- Actualización
- Opciones generales
- Mediante proxy
- De forma manual
- De forma automática

Configuraciones Generales de las Actualizaciones

**Opciones generales**

Directorio de las bases de Actualizaciones

Dirección de los logs de Actualizaciones

Logs detallados

Origen de las Actualizaciones

Figura 13: Interfaz de las opciones generales para la actualización

Antivirus 10.0.0.1	DNS 10.0.0.2	CORREO correo.uci.cu	FTP 10.0.0.4	WEB 10.0.0.5	PROXY correo.uci.cu
-----------------------	-----------------	-------------------------	-----------------	-----------------	------------------------

Módulo Estado: Corriendo ▶ ↻ ◼ ⬇ Salvar

Opciones del menú

- Configuración del módulo
- Escaneo de directorios y ficheros
- Resumen de los análisis realizados
- Archivos en cuarentena
- Tareas planificadas
- Actualización
- Opciones generales
- Mediante proxy
- De forma manual
- De forma automática

Configuración de la conexión a través de proxy

**Opciones del proxy**

Servidor Proxy

Puerto

Usuario

Contraseña

Figura 14: Interfaz de la actualización mediante proxy

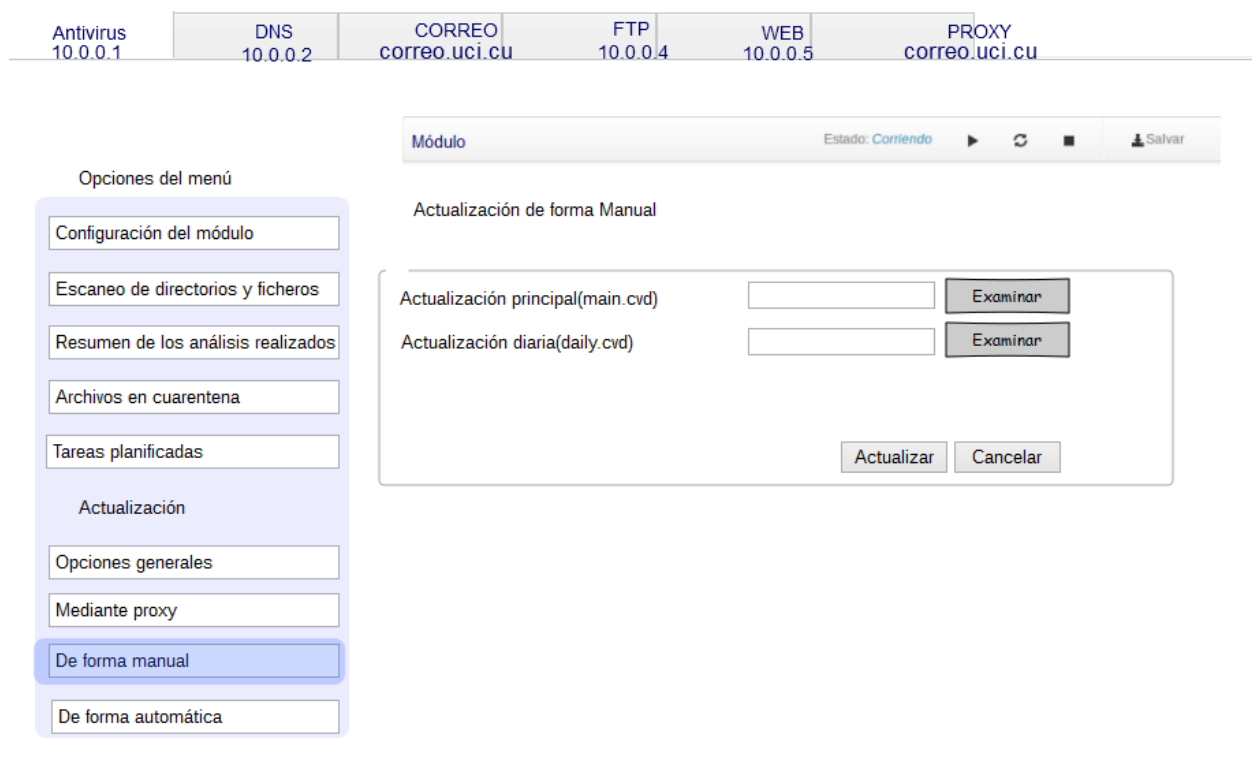


Figura 15: Interfaz de la actualización manual

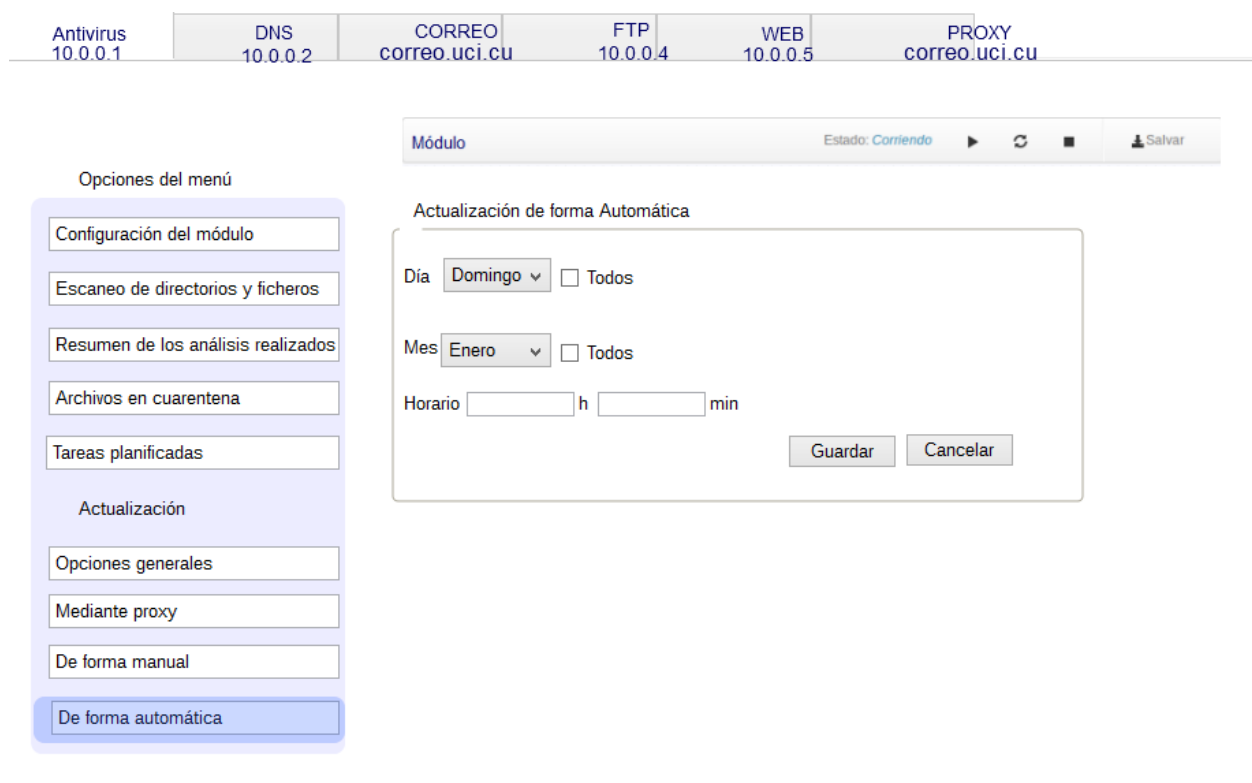


Figura 16: Interfaz de la actualización automática