Universidad de las Ciencias Informáticas

Facultad 2

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

Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux.

Autores:

Alberto González Esquivel

Fernando García Alemán

Tutores:

Ing. Lilian Saúco Altuna

Ing. Ubel Ángel Fonseca Cedeño

La Habana, 2017

Declaración de Auditoría

Declaramos que somos los únicos autores de este trabajo titulado:

Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux y autorizamos al centro de Telemática (TLM) de la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Fernando García Alemán

_____

Firma del autor

Alberto González Esquivel

_____

Firma del autor

Lilian Saúco Altuna

_____

Firma del tutor

Ubel Ángel Fonseca Cedeño

_____

Firma del tutor

Agradecimientos

Agradecemos a todos los que, de alguna manera, han influido en que sea posible el hecho de convertirnos en ingenieros en ciencias informáticas.

A nuestros padres gracias por el apoyo, la paciencia, la comprensión y la dedicación mostrada durante todo este tiempo.

A nuestros amigos gracias por haber estado a nuestro lado cuando más lo necesitábamos, cuando nos hizo falta una mano amiga para levantarnos ante los retos que la vida nos hizo pasar durante este tiempo en la universidad.

A nuestros profesores gracias por las enseñanzas que nos han podido aportar permitiéndonos el hecho de haber podido realizar esta tesis.

A nuestros tutores les estamos muy agradecidos por el apoyo y la guía que nos han brindado durante el transcurso de realización de este trabajo de diploma.

A todos MUCHAS GRACIAS.

Dedicatoria

Dedicamos especialmente esta tesis a nuestros padres por el apoyo y la ayuda durante este tiempo de carrera.

A nuestros compañeros de brigada que siempre nos ayudaron durante todos estos años a sobrepasar los retos que posee el estudiante universitario en el día a día.

A nuestros hermanos y a al resto de la familia por creer en nosotros y darnos los ánimos para alcanzar nuestras metas personales y profesionales.

Resumen

Este trabajo aborda la investigación, desarrollo e implementación de un módulo que para la detección de virus en archivos adjuntos de correo electrónico para el software Segurmática Antivirus 1.7 para Linux. Incluye un estado del arte donde se hace un análisis de la metodología de desarrollo, tecnologías, lenguajes de programación y herramientas propuestas para el desarrollo de la solución. La aplicación y sus funciones fueron desarrolladas siguiendo las estrategias propuestas por la metodología de software Proceso Unificado Ágil (AUP) en su variante para la Universidad de las Ciencias Informáticas (AUP-UCI), haciendo uso de los lenguajes C++ y Phyton. El sistema elaborado permite hacer un filtrado de los archivos adjuntos detectando, en caso de existir, posibles amenazas que puedan ser introducidas por medio de correos electrónicos, apoyando el mantenimiento de la seguridad de la información y los medios que se posean en una entidad.

Palabras clave: complemento, virus, archivos adjuntos, seguridad.

Índice

Introducción:

Las Tecnologías de la Información y la Comunicación (TIC), desde su surgimiento han formado parte de la vida diaria; influyendo en diferentes esferas de la sociedad como la educación, la salud, la comunicación y los negocios. Su creciente evolución ha permitido a la humanidad progresar rápidamente en la ciencia y la técnica desplegando las armas más poderosas para el hombre: el conocimiento y la información (Brito, 2012).

La información generalmente es procesada, intercambiada y conservada en redes de datos, equipos informáticos y soportes de almacenamiento, que son parte de lo que se conoce como sistemas informáticos. (Brito, 2012) Los sistemas informáticos están sometidos a potenciales amenazas, originados tanto desde dentro de la propia organización, como desde fuera.

Dentro de las posibles amenazas se encuentran los virus informáticos, que no son más que programas que infectan computadoras sin el conocimiento o permiso de sus operadores. Están catalogados también como mecanismos parásitos, puesto que atacan a los archivos o sectores de arranque y se replican para continuar su esparcimiento provocando no solo la pérdida de información, sino también tiempo en la reinstalación de los sistemas operativos, entre otros daños incluso más graves. (Dulce María Canes , 2012)

La seguridad informática es el conjunto de métodos y herramientas destinados a proteger los bienes informáticos de una institución, y tiene como objetivo mantener la confidencialidad, integridad y disponibilidad de la información. (Ruiz, 2016) Las amenazas a un sistema informático pueden ser muy diversas pero se clasifican de manera general en ataques de:

- Intercepción: acceso a la información por personas no autorizadas; es un ataque de tipo pasivo contra la confidencialidad de la información, ejemplo de esto es el robo de contraseñas o la copia ilícita de programas
- Modificación: acceso no autorizado a la información en el que se produce una modificación de la misma; es un ataque activo contra la integridad de los datos, ejemplo de esto es la desfiguración de un sitio web
- Interrupción: deja de funcionar total o parcialmente un sistema informático; es un ataque activo contra la disponibilidad de la información, ejemplo de esto es el bloqueo de los servicios de servidores web o de correo electrónico.

- Fabricación: Se crean objetos falsificados en un sistema de cómputo, el intruso puede insertar operaciones no esenciales de un sistema de comunicación de red; este es un ataque contra la autenticidad, ejemplo de esto es el hecho de añadir registros a un archivo o a una base de datos existente.

Para prevenir estos ataques se emplean mecanismos de defensa que pueden ser:

- Preventivos: aumentan la seguridad del sistema, previniendo la ocurrencia de violaciones a la seguridad
- Detección: detectan la ocurrencia de una violación a la seguridad en el momento en que se produce la misma
- Recuperación: retornar el sistema a su normal funcionamiento después de una violación

Entre los ejemplos de mecanismos de defensa, que pueden entrar dentro de las categorías anteriormente mencionadas, podemos nombrar los muros de seguridad o firewall, los antivirus, los sistemas detectores de intrusos, las copias de respaldo o backups entre otros.

Por ello, para resguardar los recursos informáticos de las instituciones, se hace necesario velar por el correcto cumplimiento de las políticas de seguridad en cada organismo ya que, por un lado, muestra el posicionamiento de la organización con relación a la seguridad, y por otro lado sirve de base para desarrollar los procedimientos concretos de seguridad. (Benitez, 2013)

Cuba no ha estado exenta de este desarrollo vertiginoso ni de las amenazas asociadas al mismo y ha sabido incrementar el uso de las TIC en todos los sectores del país, teniendo en cuenta las pautas que rigen su seguridad, y garantizando un respaldo legal que responda a las condiciones y necesidades del proceso de informatización del país. Cada entidad que haga uso de las tecnologías de la información para el desempeño de sus actividades está en la obligación de tener instalado y mantener actualizado un sistema de antivirus encargado de mantener protegidos los recursos de la entidad minimizando los riesgos a que están sometidos diariamente. En nuestra nación el cumplimiento de la seguridad informática en las organizaciones está regido por la Resolución 127 del 2007 que integra el Reglamento de Seguridad para las Tecnologías de la Información, cuyo objetivo esta basado en el establecimiento de los requerimientos que rigen la seguridad de las tecnologías de la información y garantizar su respaldo legal, respondiendo a las condiciones y necesidades del proceso de informatización del país.

La Empresa de Consultoría y Seguridad Informática SEGURMÁTICA es una empresa estatal cubana perteneciente al Ministerio de Informática y Comunicaciones. Su misión estratégica es brindar los servicios de Seguridad Informática que sean demandados por entidades y particulares radicados en Cuba, sustituyendo importaciones y garantizando la seguridad del país. (Bidot, 2015) Esta empresa conjuntamente con Telemática (TLM) centro productivo de la Facultad 2 de la Universidad de las Ciencias Informáticas (UCI), encargado de desarrollar sistemas relacionados con las telecomunicaciones y la seguridad informática desarrollan el software Segurmática Antivirus 1.7 para Linux: programa que entre sus funcionalidades debe ser capaz de detectar programas malignos que se ejecutan en Linux procedentes de archivos adjuntos en correos electrónicos.

Con frecuencia a través del correo electrónico se reciben archivos adjuntos infectados con algún tipo de virus, ocupando un porcentaje que, si bien es relativamente pequeño aproximadamente de un siete por ciento, no deja de ser relevante ante las posibles consecuencias a la que estaríamos expuestos; por eso se le considera como uno de los medios de propagación de códigos maliciosos a tener en cuenta. Su acción comienza al abrir un archivo infectado, el código que conforma al virus se guarda en memoria y se añade a los programas que se ejecuten ya que toma el control del sistema operativo. (Quiroga, 2016) Dentro de los virus más habituales se encuentran:

- Troyanos: es el malware más popular diseñado para controlar de forma remota un ordenador; el virus más temido por empresas y usuarios, conocido como caballo de Troya, es un tipo de software que tiene como objetivo infiltrarse, dañar una computadora o el sistema de información sin el consentimiento de su propietario
- Gusanos o Worm: son programas que realizan copias de sí mismos, alojándolas en diferentes ubicaciones del ordenador; el objetivo de este malware suele ser colapsar los ordenadores y las redes informáticas, impidiendo así el trabajo a los usuarios
- Sobre escritura: se caracterizan por destruir la información de los ficheros a los que infecta dejándolos inservibles
- Mutantes o Polimórficos: se ocultan en un archivo y son cargados en la memoria cuando el archivo es ejecutado y en lugar de hacer una copia exacta de éste cuando infecta otro archivo, éste modificará la copia cada vez que sea ejecutado

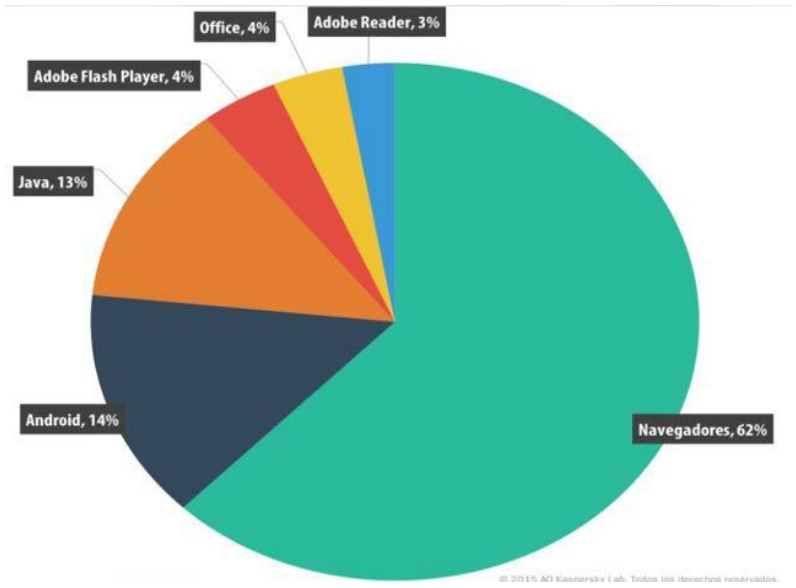

Figura 1: porcentaje de transmisión de virus. (kasperski, 2015)

La imagen anterior muestra el porcentaje de transmisión de virus a nivel global. Se evidencia como entre los compendios de datos guardados en documentos de Office y de Adobe Reader, representando en una cantidad considerable los elementos transmitidos mediante adjuntos de correo electrónico, ocupan aproximadamente un siete por ciento del porcentaje de transmisión mundial.

Dependiendo de la programación del virus es el daño que causa en el equipo infectado, por lo que contar con un software antivirus que posea un módulo capaz de detectar y tomar medidas contra estos programas malignos procedentes de archivos adjuntos en correos electrónicos es una necesidad.

La aplicación principal (Segurmática Antivirus 1.7 para Linux) la cual se encuentra en su fase de desarrollo, proporciona servicios que los plugins podrían utilizar, incluyendo un método para que se registren a sí mismos y un protocolo para el intercambio de datos. Los plugins dependen de los servicios prestados por la aplicación de acogida y no suelen funcionar de forma aislada. Por el contrario, la aplicación principal funciona independientemente de ellos, lo que le permite a los usuarios finales añadir y actualizar los complementos de forma dinámica sin necesidad de hacer cambios a la aplicación principal.

Procedente de lo anterior expuesto se define como **problema científico:** ¿Cómo permitirle al usuario final identificar virus provenientes de archivos adjuntos de correo electrónico?

Derivado del problema antes definido se obtiene el **objeto de estudio:** Proceso de gestión de archivos adjuntos en correo electrónico.

Para el cumplimiento del problema planteado anteriormente se establece como **objetivo general:** Implementar un módulo que se integre con Segurmática Antivirus 1.7 para Linux, que permita al usuario final realizar una búsqueda en su buzón de correo electrónico e informar sobre posibles amenazas.

Se especifica como **campo de acción:** los procesos de implementación, diseño e integración de un módulo para antivirus.

Para darle cumplimiento al objetivo planteado se responderán las siguientes **preguntas científicas**:

1. ¿Qué aspectos deben tenerse en cuenta para realizar un análisis y diseño de un módulo que se integre con Segurmática Antivirus 1.7 para Linux?
2. ¿Cómo implementar a partir del análisis y diseño realizado, el Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux?
3. ¿Qué resultados se obtendrán al validar el Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux?

Para ello se trazaron los siguientes **objetivos específicos:**

1. Caracterizar los fundamentos teóricos relacionados con los sistemas de integración a aplicaciones.
2. Definir las tecnologías, las herramientas y la metodología de desarrollo para la implementación del Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux.
3. Diseñar el Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux.
4. Implementar el Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux.

5. Validar el Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux.

Para darle cumplimiento a los objetivos específicos se proponen las siguientes **tareas de investigación:**

1. Realización de un estudio sobre los sistemas de integración a aplicaciones para la investigación y comprensión de los mismos.
2. Selección de las tecnologías, herramientas y estándares que se necesitan para implementar el Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux.
3. Selección de la metodología de desarrollo de software que guie el desarrollo del Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux.
4. Elaboración de los artefactos requeridos por la metodología de desarrollo seleccionada.
5. Implementación del Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux.
6. Validar las funcionalidades del Módulo de protección de correo electrónico para verificar su correcto funcionamiento.

En el desarrollo del trabajo de diploma se utilizaron los siguientes métodos de investigación científica:

Métodos Teóricos:

- **Histórico-lógico:** permite conocer los elementos que han caracterizado a los módulos para antivirus, así como sus antecedentes y tendencias actuales.
- **Analítico-sintético:** facilita el estudio de las fuentes bibliográficas utilizadas en la investigación con el objetivo de elaborar el marco teórico, identificar los elementos relacionados con la solución propuesta y brindar un análisis de los procesos relacionados al campo de acción.
- **Modelación:** brinda una abstracción de la solución, ya que representa mediante modelos y diagramas los diferentes elementos que la componen.

Métodos Empíricos:

- **Entrevista:** permite conocer el criterio respecto al problema presentado por parte de personas que están directamente vinculadas y/o afectadas por sus consecuencias e identificar posibles funcionalidades a incorporar en la solución.

El presente documento se encuentra conformado por 3 capítulos como se expresa a continuación:

Capítulo 1: Estudio de sistemas de integración a aplicaciones, herramientas, tecnologías y metodología para su desarrollo.

Se definen las herramientas y tecnologías, así como metodología de desarrollo de software que se utilizará, constituyendo la base teórica de la investigación.

Capítulo 2: Caracterización del Módulo de protección de correo electrónico para el software Segurmática Antivirus para Linux.

Este capítulo abarca el desarrollo del flujo actual de los procesos, y se describe la propuesta de solución para resolver el problema planteado. Además de, especificar los requisitos funcionales, no funcionales y los elementos fundamentales del diseño y de la arquitectura.

Capítulo 3: El capítulo contiene los elementos que forman parte de las pruebas realizadas a la solución, con el objetivo de validar sus funcionalidades así como los resultados obtenidos en la evaluación del objetivo de la investigación.

Capítulo 1: Fundamentación teórica

En este capítulo se hace referencia a los principales conceptos concernientes con el trabajo de diploma. Además se estudian los sistemas relacionados con la temática para comprender mejor el objetivo. Se selecciona la metodología de desarrollo, las diferentes herramientas y tecnologías a utilizar.

1.1 Conceptos asociados

A continuación se enunciaran los conceptos que se consideran claves con el fin de lograr una mejor comprensión del presente trabajo:

-TIC: Son tecnologías de la información y de comunicaciones, constan de equipos de programas informáticos y medios de comunicación para reunir, almacenar, procesar, transmitir y presentar información en cualquier formato es decir voz, datos, textos e imágenes. Las TIC son aquellas tecnologías que permiten transmitir, procesar y difundir información de manera instantánea. (slideshare, 2015)

-Comunicación cliente servidor: Es una arquitectura de paso de mensajes, que puede operar variables compartidas mediante un archivo público de acceso remoto; es una solución muy socorrida pues su propiedad de distribución de trabajo aprovecha la versatilidad de que un servidor puede también ser cliente de otros servidores. El acceso a los terceros recursos se hace mediante el intermedio del servidor, quien se convierte en cliente de otros. (Perezrul, 2015)

-Empaquetador (wrapper): Una función de envoltura es una subrutina en una biblioteca de software o un programa de computadora cuyo propósito principal es llamar a una segunda subrutina o una llamada de sistema con poca o ninguna computación adicional. (Callao, 2013) Las funciones del envoltorio son un medio de delegación y pueden utilizarse para una serie de propósitos:

- Comodidad de programación: Las funciones de Wrapper son útiles en el desarrollo de aplicaciones que utilizan funciones de biblioteca de terceros. Un envoltorio se puede escribir para cada una de las funciones de terceros y se utiliza en la aplicación nativa.

En caso de que las funciones de terceros cambien o se actualicen, sólo los modificadores de las aplicaciones nativas deben modificarse en lugar de cambiar todas las instancias de funciones de terceros en la aplicación nativa.

- Adaptación de interfaces de clase u objeto: Las funciones de Wrapper se pueden utilizar para adaptar una clase u objeto existente para tener una interfaz diferente. Esto es especialmente útil cuando se utiliza código de bibliotecas existentes.

-Correo electrónico: Es un servicio de red que permite a los usuarios enviar y recibir mensajes (también denominados mensajes electrónicos o cartas digitales) mediante sistemas de comunicación electrónica. (Kingsley, 2012)

-Archivos adjuntos: Un archivo adjunto, archivo anexo, adjunto de correo, es un archivo que se envía junto a un mensaje correo electrónico. (Kingsley, 2012)

-Virus: Es un programa que infecta las computadoras sin el conocimiento o permiso de sus operadores. Está catalogado también como un mecanismo parásito, puesto que ataca a los archivos o sectores de arranque y se replica para continuar su esparcimiento de modo que provoca no solamente la pérdida de información, sino también la de tiempo en la reinstalación de los sistemas operativos, entre otros daños incluso más graves. Uno de los líderes mundiales de la seguridad informática el conocido ruso Eugene Kaspersky, asevera: “Combatir contra los virus es una carrera contra el tiempo. Es un trabajo muy duro”. (Fauces, 2014)

-Antivirus: Un antivirus es un programa de seguridad que se instala en la computadora o dispositivo móvil para protegerlo de infecciones por software malintencionado (malware). (Cohen, 2015)

-Plug-in (plugin o complemento): “Un complemento es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Son paquetes modulares de archivos que se utilizan para llevar a cabo una acción especializada. Constituye un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.” (Membrides, 2010)

Los plugins permiten:

- Que los desarrolladores externos colaboren con la aplicación principal extendiendo sus funciones
- Reducir el tamaño de la aplicación
- Separar el código fuente de la aplicación a causa de la incompatibilidad de las licencias de software

Como funcionan:

Un programa asigna una única carpeta (generalmente en la misma ruta que la del programa) donde se buscaran los nuevos plugins creados por terceros, el programa principal escanea la carpeta al iniciarse y mediante procedimientos llamados por defecto y requeridos en los complementos obtiene los datos necesarios para (por ejemplo) visualizarlos en una interface gráfica u obtener la lista de procedimientos alternativos que exporta el plugin (mediante cadenas). (Membrides, 2010)

1.2 Protocolos de acceso a correo

Protocolo para Transferencia Simple de Correo (SMTP):

Protocolo de red utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA, teléfonos móviles, impresoras, etc.). Su funcionamiento es en línea, de manera que opera en los servicios de correo electrónico. Sin embargo, este protocolo posee algunas limitaciones en cuanto a la recepción de mensajes en el servidor de destino (cola de mensajes recibidos). Como alternativa a esta limitación se asocia normalmente con otros protocolos, como el POP o IMAP, otorgando a SMTP la tarea específica de enviar correo, y recibirlos empleando los otros protocolos antes mencionados.

El protocolo SMTP funciona con comandos de textos enviados al servidor SMTP (al puerto 25 de manera predeterminada). A cada comando enviado por el cliente (validado por la cadena de caracteres ASCII CR/LF, que equivale a presionar la tecla Enter) le sigue una respuesta del servidor SMTP compuesta por un número y un mensaje descriptivo. (Beltran, 2017)

Capa de Puertos Seguros (SSL):

Protocolo diseñado para permitir que las aplicaciones puedan transmitir información de ida y de manera segura hacia atrás. Las aplicaciones que utilizan este protocolo saben cómo dar y

recibir claves de cifrado con otras aplicaciones, así como la manera de cifrar y descifrar los datos enviados entre los dos.

Algunas aplicaciones que están configuradas para ejecutarse con SSL incluyen navegadores web como Firefox y los programas de correo como Outlook, Mozilla Thunderbird, Mail.app de Apple, y SFTP (Secure File Transfer Protocol), estos programas son capaces de recibir de forma automática conexiones SSL. Para establecer una conexión segura SSL, se debe tener una clave de cifrado que le asigna una autoridad de certificación en la forma de un Certificado. Una vez que haya una única clave de su cuenta, se puede establecer una conexión segura utilizando el protocolo SSL. (John Merrill, 2017)

Un certificado SSL sirve para brindar seguridad al visitante de su página web, una manera de decirles a sus clientes que el sitio es auténtico, real y confiable para ingresar datos personales. Este protocolo de seguridad hace que sus datos viajen de manera íntegra y segura; es decir, la transmisión de los datos entre un servidor y usuario web, y en retroalimentación es totalmente cifrada o encriptada. Para que los datos viajen cifrados se emplean matemáticos y un sistema de claves que sólo son identificados entre la persona que navega y el servidor. Al tener un certificado SSL confiable se puede asegurar que nadie puede leer el contenido de nuestros datos. Todo esto lleva a entender que la tecnología que brinda un certificado SSL es la transmisión segura de información a través de internet y así confirmar que los datos no serán accedidos por personas no deseadas. (Grupo Advantage, 2016)

Protocolo de Acceso a Mensajes de Internet (IMAP):

Protocolo de la capa de aplicación que permite el acceso a mensajes almacenados en un servidor de internet. Mediante IMAP se puede tener acceso al correo electrónico desde cualquier equipo que tenga una conexión a internet. Constantemente se sincroniza el programa de correo electrónico con el servidor, mostrando los mensajes que están presentes en la carpeta en cuestión. Todas las acciones realizadas en los mensajes (leer, mover, eliminar…) se realizan directamente en el servidor. (Cantón, 2013) Dentro de sus características se encuentran:

- Gran número de transacciones simultáneas
- No es necesario descargar los correos completos (se puede visualizar solo el encabezado)

- Permite la búsqueda de mensajes por medio de palabras claves
- Buena integración con el Protocolo Ligero de Acceso a Directorios (LDAP)

Protocolo de oficina de correos (POP):

Protocolo de red que se utiliza en clientes locales de correo electrónico para obtener los mensajes de correo electrónico almacenados en un servidor remoto. Está configurado para descargar todos los mensajes del servidor de correo electrónico al ordenador desde el que se conecta. Esto significa que todas las acciones realizadas en los mensajes (leer, mover, borrar…) se realizarán en el propio ordenador. Al descargarse, por defecto, se eliminan los mensajes del servidor y, por ello, el usuario no podrá volver a ver los mensajes desde cualquier lugar que no sea el equipo en el que los mensajes han sido descargados. (Cantón, 2013)

Conclusiones de los protocolos de acceso a correo:

Se define como protocolo de acceso a correos a emplear IMAP en su versión más actualizada IMAP4rev1. Mediante este protocolo se puede tener acceso al correo electrónico desde cualquier equipo que tenga una conexión a internet y es poseedor de varias ventajas sobre POP, como por ejemplo: es posible especificar carpetas del lado servidor, permite visualizar los mensajes de manera remota y no descargando los mensajes como lo hace POP, proporciona tiempos de respuestas más rápidos para usuarios que tienen una gran cantidad de mensajes ya que permanecen conectados el tiempo que su interfaz permanezca activa y descargan los mensajes bajo demanda.

1.3 Metodología de desarrollo de software

En la actualidad no está definida una metodología universal que se aplique a todos los proyectos, su selección depende de las características específicas que tenga cada uno de estos.

Se podrían clasificar en dos grandes grupos:

Metodologías Pesadas: orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas y notaciones a utilizar.

Metodologías ligeras/ágiles: orientadas a la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente

en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando.

AUP

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas (test driven development-TDD en inglés)
- Modelado ágil
- Gestión de Cambios ágil
- Refactorización de Base de Datos para mejorar la productividad

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva. (Sánchez, 2015)

Fases AUP:

Inicio: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.

Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.

Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.

Transición: El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

Variación de AUP para la UCI.

La creación de software es una tarea sumamente compleja y no existe una metodología que integre todas las posibles situaciones y características de un equipo de desarrollo demandando

así que el proceso sea adaptable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.

De las cuatro fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes tres fases de AUP en una sola, nombrada Ejecución y se agrega la fase de Cierre. (Sánchez, 2015)

Con la adaptación de AUP que se propone para la actividad productiva de la UCI se logra estandarizar el proceso de desarrollo de software, cumpliendo además con las buenas prácticas que se definen en el Modelo CMMI-DEV v1.3 que sirve como guía para aumentar la calidad del software que se produce. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. Se redujo a una la cantidad de metodologías que se usaban y de más de veinte roles en total que se definían se redujeron a once.

Metodología escogida

Para la realización del proyecto se definió la metodología AUP-UCI debido al enfoque ágil que posee, dirigida a aumentar la eficiencia de las personas involucradas en los proyectos productivos de la universidad. AUP-UCI garantiza que el trabajo sea exitoso, satisface las necesidades cambiantes del cliente así como, la relación estrecha que establece con el mismo por lo que se adapta fácilmente a las circunstancias y, es ideal para equipos de trabajo pequeños. Además se apoya en las buenas prácticas que propone el Modelo CMMI-DEV v1.3 como guía para lograr un producto con la suficiente calidad.

1.4 Lenguajes de programación

Según la definición teórica, como lenguaje se entiende a un sistema de comunicación que posee una determinada estructura, contenido y uso. La programación es en el vocabulario propio de la informática, el procedimiento de escritura del código fuente de un software. De esta manera, puede decirse que la programación le indica al programa informático qué acción tiene que llevar a cabo y cuál es el modo de concretarla. Con estas nociones en claro, podemos afirmar que un lenguaje de programación es aquella estructura que, con una cierta base sintáctica y semántica imparte distintas instrucciones a un programa de computadora. (Porto, y otros, 2012)

C++ v3.5.1

Es un lenguaje imperativo de propósito general, orientado a objetos basado en C, manteniendo considerables potencialidades para la programación a bajo nivel, añadiéndosele elementos que le permiten también un estilo con alto nivel de abstracción y algunas utilidades adicionales de librerías. (FreeFind, 2012)

Principales características:

- Tiene un conjunto completo de instrucciones de control
- Permite la agrupación de instrucciones
- Incluye el concepto de puntero (variable que contiene la dirección de otra variable)
- Los argumentos de las funciones se transfieren por su valor
- E/S no forma parte del lenguaje, sino que se proporciona a través de una biblioteca de funciones
- Permite la separación de un programa en módulos que admiten compilación independiente

Python v2.7

Es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y en menor medida programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Es administrado por la Python Software Foundation. (Pyton.org, 2014) El motivo para la utilización de este lenguaje es la creación de un script encargado de realizar la conexión con el servidor de correo, apoyándose en la librería Imaplib proporcionando ventajas sobre C++, pues en este lenguaje no existe una librería no privativa que realice estas funciones.

1.5 Herramientas utilizadas

Las herramientas informáticas son programas, aplicaciones o simplemente instrucciones usadas para efectuar otras tareas de modo más sencillo. En este caso los lenguajes de programación C++ y Python que son los usados para la realización del proyecto, condicionan la selección de algunas herramientas para el desarrollo de la solución. A continuación se muestra

una breve reseña de las herramientas que fueron usadas en el desarrollo del módulo y que a su vez son, la mayoría, las establecidas por el centro TLM.

1.5.1 Entorno de desarrollo integrado (IDE)

Un entorno de desarrollo integrado, es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). (AC, 2013)

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java, C#, Delphi, Visual Basic, entre otros; en algunos lenguajes puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. (AC, 2013)

Un IDE debe tener las siguientes características:

- Multiplataforma
- Soporte para diversos lenguajes de programación
- Integración con Sistemas de Control de Versiones
- Reconocimiento de Sintaxis
- Extensiones y Componentes para el IDE
- Integración con Framework populares
- Depurador
- Importar y Exportar proyectos
- Múltiples idiomas
- Manual de Usuarios y Ayuda

Ventajas de los IDEs:

- La curva de aprendizaje es muy baja
- Es más ágil y óptimo para los usuarios que no son expertos en manejo de consola
- Formateo de código
- Funciones para renombrar variables, funciones

- Warnings y errores de sintaxis en pantalla de algo que no va a funcionar al interpretar o compilar
- Poder crear proyectos para poder visualizar los archivos de manera gráfica
- Herramientas de refactoring como por ejemplo seria extraer una porción de código a un método nuevo

1.5.2 IDEs seleccionados:

Qt Creator v3.5.1

Principales características:

- Posee un avanzado editor de código C++
- Además soporta los lenguajes: C#/.NET Lenguajes (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby
- Posee también una GUI integrada y diseñador de formularios
- Herramienta para proyectos y administración
- Ayuda sensible al contexto integrado
- Depurador visual
- Resaltado y auto-completado de código
- Soporte para refactorización de código

PyCharm v3.4

Entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con VCS / DVCS y soporte para el desarrollo web con Django, entre otras bondades. En la versión 3.4 se ha mejorado la funcionalidad, el desempeño y se han añadido nuevas características cuyo objetivo es hacer las jornadas de desarrollo aún más productivas. (Arrs, 2014) Entre los aspectos principales se pueden destacar:

- Gestión de los intérpretes de Python con una nueva interfaz de usuario

- Nuevo soporte refinado para intérpretes remotos
- Soporte para Django 1.7
- Múltiples selecciones
- Soporte completo de depuración en la consola interactiva Python

1.5.3 Herramienta de modelado

Visual Paradigm v8.0

Es una herramienta profesional que usa un Lenguaje Unificado de Modelado (UML) para el diseño e implementación de sistemas de software complejos, soportando su ciclo de vida de desarrollo: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. (Nazar, 2014) En el desarrollo de este sistema se hace uso de esta herramienta para el diseño de los prototipos de interfaces así como para la realización de diferentes diagramas que representan el ciclo de vida del proyecto.

1.5.4 Herramienta de gestión de bases de datos

SQLite

Sistema de gestión de bases de datos relacional. A diferencia de los sistemas de gestión de bases de datos cliente-servidor, su motor no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza esta funcionalidad a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. (SQLite.org, 2015)

1.5.5 Librería

Imaplib

Imaplib implementa un cliente para comunicarse con los servidores de la versión 4 del Protocolo de acceso a mensajes de Internet (IMAP). El protocolo IMAP define un conjunto de comandos enviados al servidor y las respuestas devueltas al cliente. La mayoría de los comandos están

disponibles como métodos del objeto IMAP4 que se utiliza para comunicarse con el servidor. Este módulo define tres clases, IMAP4, IMAP4_SSL e IMAP4_stream, que encapsulan una conexión a un servidor IMAP4 e implementan un gran subconjunto del protocolo de cliente IMAP4rev1 como se define en RFC 2060. Es compatible con servidores IMAP4 (RFC 1730). (Python Software Foundation, 2017) Esta librería se usara, dentro del script creado, para el establecimiento de la conexión con el servidor de correo.

1.6 Conclusiones del capítulo

Con el desarrollo del Capítulo 1 se concluye que:

- La solución para dar cumplimiento al problema planteado es un módulo integrado a Segurmática Antivirus que permita notificar la presencia de virus en archivos adjuntos de correo electrónico.
- Se determina emplear AUP-UCI como metodología de desarrollo, utilizando C++ y Python como lenguajes de programación, para la gestión de bases se usa SQLite y para la conexión cliente-servidor se utilizará la librería Imaplib.

-

Capítulo 2: Propuesta de solución

2.1 Introducción

En este capítulo se formalizará la propuesta de solución al problema existente con el apoyo de la metodología seleccionada, presentando un análisis y definiendo sus características. Además se describe el modelo de dominio, las historias de usuario (HU) definidas por el equipo de desarrollo, la gestión de requerimientos y se desarrollan las disciplinas seleccionadas que pertenecen a la metodología.

2.2 Propuesta de solución

La propuesta de solución está dirigida a establecer una conexión con el servidor IMAP de correo electrónico (imap.uci.cu), realizar la búsqueda en la casilla de correo (IMBOX), descargar los archivos adjuntos en una carpeta predefinida y luego del escaneo, por parte del antivirus (Segurmática antivirus 1.7 para Linux), notificar sobre la existencia o no de virus en dichos archivos descargados.

2.2.1 Conexión con el servidor IMAP de correo electrónico

Se desarrollará un módulo que implemente una interfaz que contenga los campos necesarios dígase usuario, contraseña, servidor y puerto, para establecer la conexión con el servidor (imap.uci.cu), así como su subdominio (imap.estudiantes.uci.cu); usará para una conexión segura el protocolo Secure Sockets Layer (SSL; en español “capa de puertos seguros”).

2.2.2 Detección y descarga de archivos adjuntos

El módulo contará con un script de Python, apoyado en la librería Imaplib.py para la conexión con el servidor, que permitirá el escaneo de la casilla de correo electrónico, detectando los correos que posean adjuntos y descargando los mismos en una carpeta predefinida para su posterior análisis por parte del antivirus.

2.2.3 Notificación

El módulo emitirá una notificación al finalizar el escaneo de los archivos descargados en la carpeta predefinida con la descripción de los archivos infectados y los tipos de virus

encontrados. En caso de no encontrar virus emitirá una notificación informando la no presencia de códigos malignos en los adjuntos analizados.

2.3 Fase de Ejecución de AUP-UCI

En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se aplica la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software. (Sánchez, 2015)

2.4 Modelado de negocio

El Modelado de negocio es definido como un proceso de representación de uno o más aspectos de una empresa u organización:

- Su propósito
- Su estructura
- Su funcionalidad
- Su dinámica
- Su lógica de negocios
- Sus componentes

Con el objetivo de garantizar que el producto, en este caso el software cumpla con su propósito. (Pressman, 2010)

2.4.1 Modelo de dominio

El Modelo de dominio es “la representación visual de los conceptos u objetos más importantes de un negocio, sus características y relaciones entre dichos conceptos. Es el mecanismo fundamental para comprender el dominio del problema y para establecer conceptos comunes. Es un diccionario visual del dominio del problema. (Pressman, 2010)

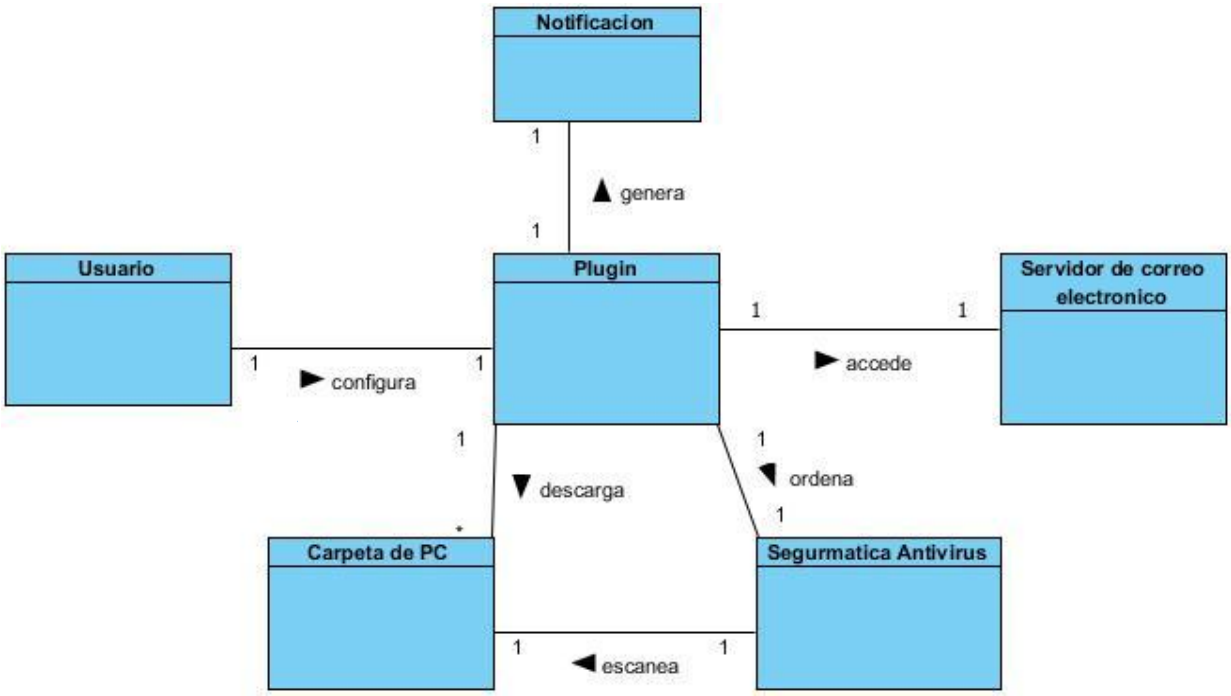

Figura 2: Diagrama de clases del Modelo de Dominio.

2.5 Disciplina Requisitos

El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos Casos de Uso del Sistema (CU), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP), agrupados en cuatro escenarios condicionados por el Modelado de negocio. (Sánchez, 2015)

2.5.1 Técnicas de capturas de requisitos

“La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cumplir dicho sistema’’. (Koch, 2012) Este proceso incluye un conjunto de tareas que conducen a comprender, que es lo que el cliente quiere y como interactuaran los usuarios finales con la propuesta de solución.

Para la realización del levantamiento de requisitos y determinar las funcionalidades necesarias para la elaboración del módulo se utilizó la técnica de Entrevistas, resultando “una técnica muy

aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural". (Koch, 2012)

2.5.2 Requisitos funcionales

"Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, sin tomar en consideración ningún tipo de restricción física, de manera que se mantienen invariables sin importar con que propiedades o cualidades se relacionen". (Rumbaugh, 2012) Para la determinación de los requisitos funcionales se aluden algunas características como el nombre, la descripción, la complejidad, su prioridad, los campos que incluye, tipo de datos y sus reglas o restricciones, además de las observaciones necesarias. La prioridad es determinada según su importancia para la realización del escaneo y la complejidad según el nivel que demanda su implementación. A continuación se listan los requisitos:

- Requisito funcional 1: Guardar configuración
- Requisito funcional 2: Probar conexión
- Requisito funcional 3: Buscar virus
- Requisito funcional 4: Notificar resultados

2.5.3 Requisitos no funcionales

"Los requisitos no funcionales son aquellas propiedades que debe tener la solución y que no son una funcionalidad", (Rumbaugh, 2012) se agruparon según su clasificación en:

Interfaz:

- Apariencia o Interfaz externa (interfaz comprensible, amigable, profesional y fácil de usar)
- Cumple con las normas de diseño de la empresa Segurmática

Usabilidad:

- Fácil de usar para clientes con poca experiencia en el manejo de aplicaciones de escritorio

Hardware:

- Sistemas operativos: Distribuciones de Linux RedHat, CentOS, SuSE, Mandrake, Debian, Ubuntu, Nova de 32 y 64 bit
- En cuanto a RAM, CPU y HDD son los requerimientos mínimos de la instalación del propio sistema operativo. Por tanto, en un sistema en funcionamiento como los mencionados anteriormente se puede instalar.

Legales, de Derecho de Autor y otros:

- El plugin pertenece al proyecto Segurmática Antivirus 1.7 para Linux

2.5.4 Escenario seleccionado para la disciplina de requisitos: Escenario No4

Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido donde el cliente estará siempre acompañando al equipo de desarrollo. Para elaborar las historias de usuario (HU) el proyecto debe establecer conversaciones acerca de las necesidades de los clientes. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información. (Sánchez, 2015)

2.5.5 Historias de usuario

Las historias de usuario se utilizan para definir los requerimientos de un sistema de software y también para crear estimaciones para la planeación de las interacciones. Estas son escritas por los clientes en forma de las cosas que quieren que el sistema haga por ellos. Deben describir comportamientos externos del sistema, los cuales puedan ser entendidos por los involucrados. (K.I, 2010) A continuación se definen los siguientes parámetros:

- Número: Número de la historia de usuario incremental en el tiempo
- Nombre de la historia de usuario: Nombre de la historia de usuario especificado por el programador
- Iteración asignada: Número de la iteración
- Prioridad en negocio (Baja, Media, Alta):

  - Baja: Se le otorga a las HU que son de funcionalidades auxiliares y que son independientes del sistema.

- Media: Se le otorga a las HU que son de funcionalidades a tener en cuenta, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
- Alta: Se le otorga a las HU que son de funcionalidades fundamentales en el desarrollo del sistema.

- Riesgo en desarrollo (Bajo, Medio, Alto):

  - Bajo: Cuando en la implementación de las HU puedan existir errores, pero éstos son tratados fácilmente y no afectan el desarrollo del sistema.
  - Medio: Cuando en la implementación de las HU puedan existir errores y retrasen la entrega del producto.
  - Alto: Cuando en la implementación de las HU pueda existir algún error y afecte la disponibilidad del sistema.
- Puntos estimados: Tiempo estimado, en semanas, que se demorará el desarrollo de la HU
- Descripción: Breve descripción de la HU
- Observaciones: Señalamiento o advertencia del sistema
- Prototipo de interfaz: Prototipo de interfaz solo si aplica

Número: 1	Nombre de la HU: Guardar configuración
Programadores: Alberto González Esquivel y Fernando García Alemán	Usuario: Cliente
Prioridad: Alta	Iteración asignada: 1
Riesgo en desarrollo: Alto	Puntos estimados: 1
Descripción: Muestra los campos necesarios donde el cliente deberá introducir la configuración para conectarse al servidor. Usuario (Obligatorio. Campo tipo texto. Permite registrar un usuario en el servidor de correo) Contraseña (Obligatorio. Campo tipo password. Permite registrar una contraseña en el	

servidor de correo) Servidor (Obligatorio. Campo tipo texto. Permite introducir el servidor al que se va a conectar) Puerto (Obligatorio. Campo tipo texto. Permite introducir el puerto por el cual se va a establecer la conexión)
Observaciones: En caso de no modificarse el campo “Puerto” se tomará el 993 como predefinida. Se establece como servidor predefinido: imap.uci.cu Al finalizar emitirá una notificación informando el resultado de la operación.
Prototipos de interfaz: 

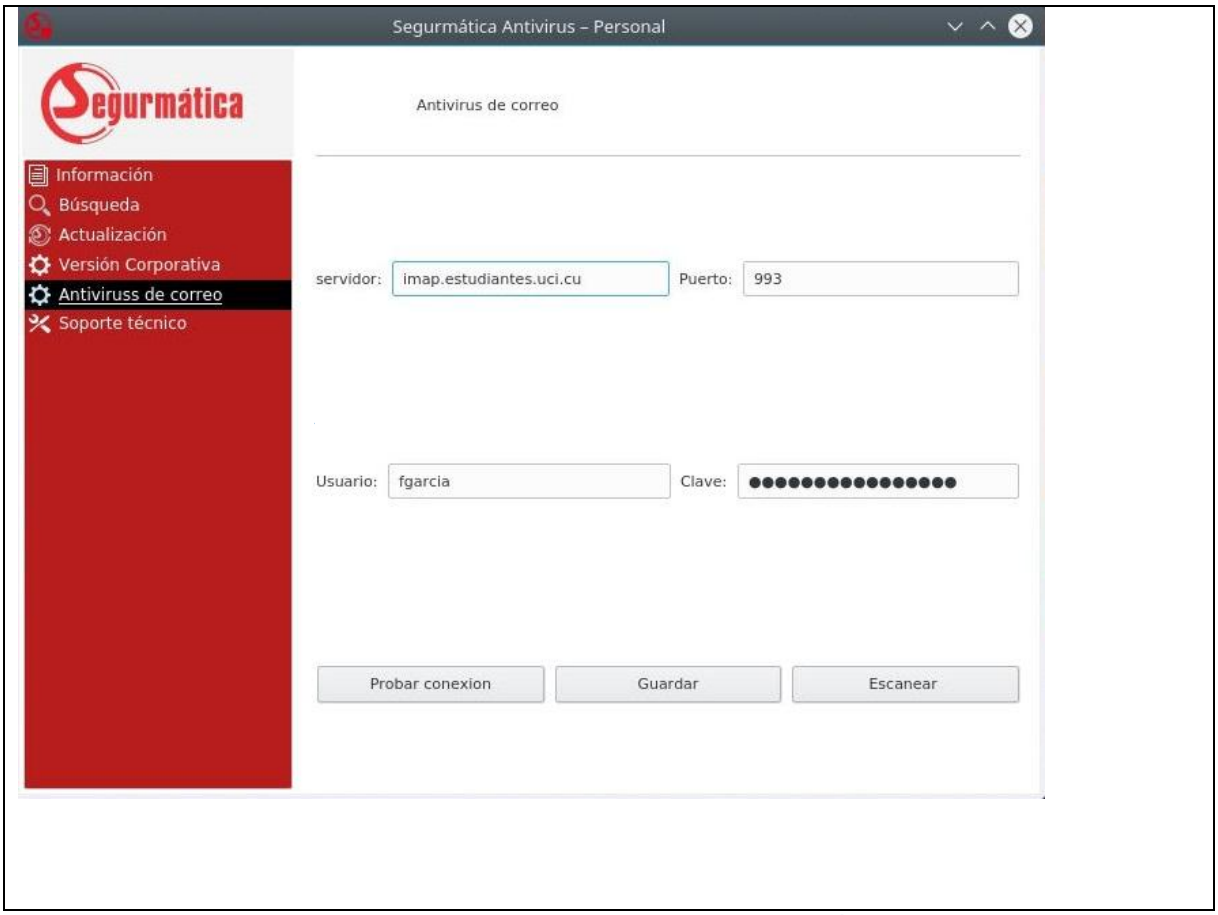

Tabla 1: HU1 Guardar configuración

Número: 2	Nombre de la HU: Probar conexión
Programadores: Alberto González Esquivel y Fernando García Alemán	Usuario: Cliente
Prioridad: Media	Iteración asignada: 1
Riesgo en desarrollo: Medio	Puntos estimados: 1
Descripción: Prueba la validez de los datos introducidos.	
Observaciones: Siempre emitirá una notificación con el resultado de la validación.	
Prototipo de interfaz:	

Tabla 2: Probar conexión

Número: 3	Nombre de la HU: Buscar virus
Programadores: Alberto González Esquivel y Fernando García Alemán	Usuario: Cliente
Prioridad: Alta	Iteración asignada: 1
Riesgo en desarrollo: Alto	Puntos estimados: 2
Descripción: 1. Descarga los archivos en una carpeta predefinida por el antivirus. 2. Realiza un análisis de todos los archivos descargados en la carpeta. 3. Elimina todos los archivos.	
Observaciones:	
Prototipo de interfaz:	

Tabla 3: HU3 Buscar virus

Número: 4	Nombre de la HU: Notificar resultados
Programadores: Alberto González Esquivel y Fernando García Alemán	Usuario: Cliente
Prioridad: Alta	Iteración asignada: 2
Riesgo en desarrollo: Medio	Puntos estimados: 2
Descripción: Notifica el resultado de la búsqueda.	
Observaciones: En caso de detectar uno o varios archivos infectados muestra una ventana con el nombre de dichos archivos.	
Prototipo de interfaz: 	

Tabla 4: HU4 Notificar resultados

2.6 Disciplina Análisis y diseño

En esta disciplina si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos. Además, en esta disciplina se modela el

sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. (Sánchez, 2015)

2.6.1 Arquitectura basada en componentes

Para la construcción de la solución se utilizará la arquitectura basada en componentes ya que es la más acorde para la creación de este módulo. Esta arquitectura se emplea en el sub-sistema de actualizaciones en el cliente y se rige por el estándar de un bajo acoplamiento en la implementación, lo que permite realizar modificaciones en las clases sin que esto implique modificar el resto. (Gil, 2015) El Cliente de Actualización se compone de varios plugins (considerados componentes) que interactúan entre sí para facilitar el funcionamiento de la aplicación. Estos plugins pueden ser modificados sin tener que modificar los restantes. Los plugins en el cliente-base se ubican en tres niveles principales:

- Plugins del core, que ocupan la mayor relevancia en la aplicación y se encargan de las tareas principales que son requeridas por varios plugins del segundo nivel.
- Plugins de aplicaciones, que son los plugins encargados de realizar tareas específicas dentro del funcionamiento, dependen de los plugins del core y son los que contienen a los plugins del tercer nivel.
- Sub-plugins, que son los plugins que utilizan los plugins de aplicaciones para delegar responsabilidades.

2.6.2 Patrones de diseño

Los patrones de diseño son un conjunto de prácticas de óptimo diseño que se utilizan para abordar problemas recurrentes en la programación orientada a objetos. Los patrones de diseño que se especifican a continuación son los aplicados en la solución propuesta. Aunque no son de utilización obligatoria, seguir estos patrones garantiza mayor robustez en una aplicación informática y que el mantenimiento o ampliación de la misma pueda realizarse más fácilmente (Kioskea, 2015).

Patrones Gang of Four (GoF)

Son propiamente orientados a objetos y clasificados según el propósito para el cual han sido definidos, permitiendo la construcción de un diseño elegante y robusto. Entre los patrones usados se encuentran los siguientes:

Patrones de comportamiento: solucionan problemas respecto a la interacción y responsabilidades entre clases y objetos, así como los algoritmos que encapsulan

- **Observador (Observer):** Define una dependencia entre objetos, de forma tal que cuando uno cambie su estado, el observador notifica este cambio y se actualizan todos los objetos que dependen de él. El objetivo principal es desacoplar la clase de los objetos clientes del objeto aumentando la modularidad del lenguaje.

El patrón Observador es el encargado de enviar los eventos desde la clase SearchSvc y tiene la capacidad de cambiar el estado del objeto a la hora de la búsqueda, por ejemplo del archivo que se está analizando.

Pandilla de los cuatro:

- **Fachada:** Provee una interfaz unificada sencilla que haga de intermediaria entre el módulo y Segurmática antivirus.

Patrones generales de software para asignar responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma estructurada. (Larman, 2005)

- **Experto:** Se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad. Contribuye a estructurar sistemas más robustos y de fácil mantenimiento. Este patrón se evidencia mediante las clases SearchSvc.cpp y Mail.cpp.
- **Creador:** El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. Se aplicará en todos los casos donde una clase tiene la responsabilidad de crear una nueva instancia de la otra. La correcta asignación permite que la aplicación pueda soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y reutilización. Este patrón está presente en la relación entra las clases mail.h y Mail.cpp.
- **Alta Cohesión:** Asigna una responsabilidad de forma tal que la cohesión siga siendo alta, asumiendo como principal objetivo que cada elemento debe de realizar una labor única dentro del módulo. Facilitando la claridad con que se comprende el diseño y una mayor capacidad de reutilización.

2.7 Diagrama de componentes

El diagrama de componentes muestra como el sistema está dividido en componentes y las dependencias entre ellos previendo una vista arquitectónica de alto nivel del sistema. Ayuda a los desarrolladores a visualizar el camino de la implementación permitiendo tomar decisiones.

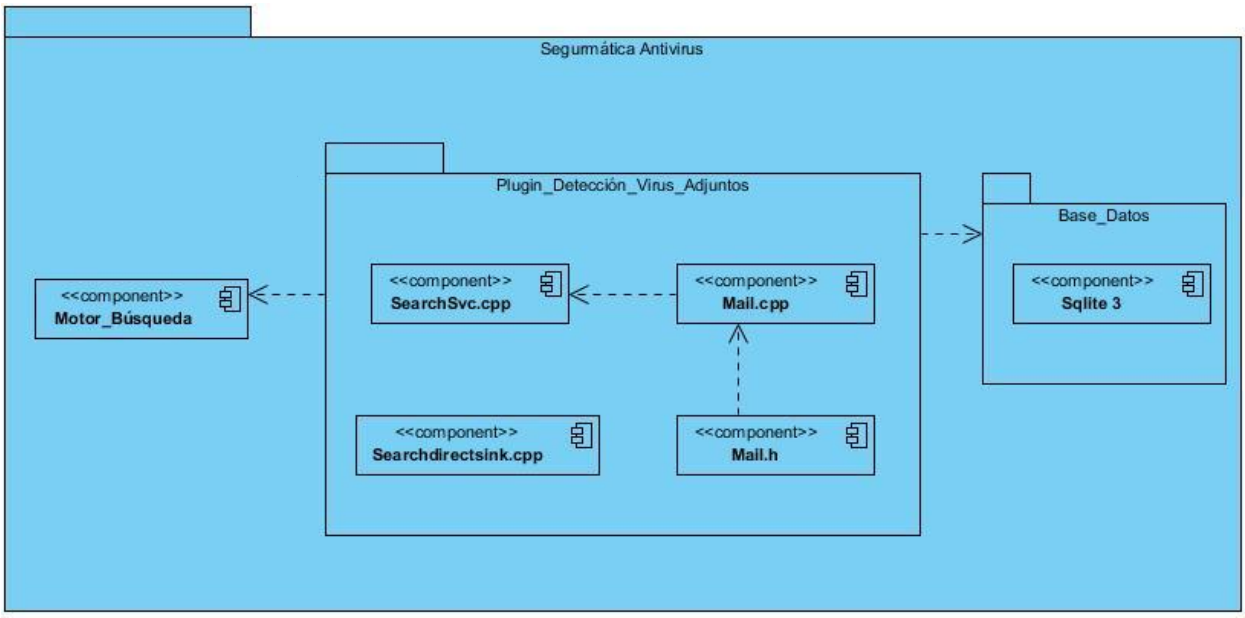

Figura 3: Diagrama de componentes.

2.8 Modelo de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema, es decir, como se distribuyen las funcionalidades entre los nodos de cómputo. A continuación se muestra el diagrama donde es desplegado el componente.

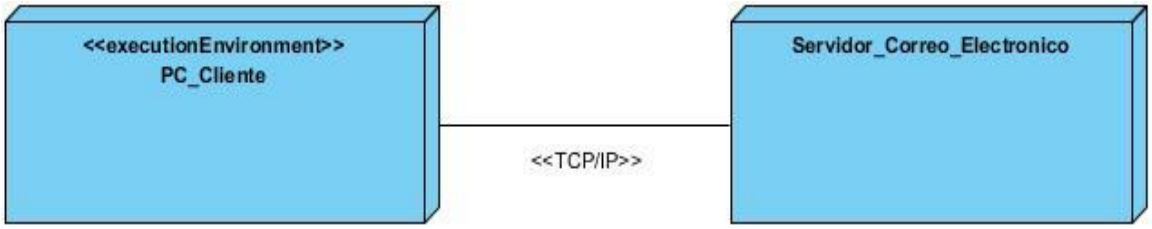

Figura 4: Modelo de despliegue.

2.9 Conclusiones del capítulo

En el presente capítulo se presentaron las características y comportamiento de los principales conceptos asociados a la propuesta de solución para lograr una mejor comprensión acerca de la misma. En el mismo se obtuvo el modelo de negocio, los requisitos funcionales y no funcionales, historias de usuario, arquitectura, patrones de diseño, diagramas de componentes y despliegue; esto permitió conocer con detalles las funcionalidades del sistema y sus comportamientos.

Capítulo 3: Pruebas al software.

En el presente capítulo se describe el proceso de verificación utilizado para evaluar la propuesta de solución. Se verifica la calidad del resultado de la implementación mediante los productos de trabajos generados durante las disciplinas de pruebas, exponiendo los resultados obtenidos.

3.1 Pruebas de Software.

Las aplicaciones informáticas no están exentas de errores por lo que a todas ellas es necesario aplicarles un conjunto de pruebas para garantizar que cuentan con la calidad requerida. Este es el factor fundamental para lograr entregar un software que cumpla o supere las expectativas del cliente. La meta de probar es encontrar errores, y una buena prueba es aquella que tiene una alta probabilidad de encontrar uno. Por tanto, un sistema basado en computadora o un producto debe diseñarse e implementarse teniendo en mente la “comprobabilidad”. Al mismo tiempo, las pruebas en sí mismas deben mostrar un conjunto de características que logren la meta de encontrar la mayor cantidad de errores con el mínimo esfuerzo. (Pressman, 2010)

Cualquier producto sometido a ingeniería (y la mayoría de otras cosas) pueden probarse dos formas:

- Al conocer la función específica que se asignó a un producto para su realización, pueden llevarse a cabo pruebas que demuestren que cada función es completamente operativa mientras al mismo tiempo se buscan errores en cada función.
- Al conocer el funcionamiento interno de un producto, pueden realizarse pruebas para garantizar que las operaciones internas se realizan de acuerdo con las especificaciones y que todos los componentes internos se revisaron de manera adecuada.

Uno de los enfoques de pruebas considera una visión externa y se llama prueba de caja negra. El otro requiere una visión interna y se denomina prueba de caja blanca. La prueba de caja negra se refiere a las pruebas que se llevan a cabo en la interfaz del software. Una prueba de caja negra examina algunos aspectos fundamentales de un sistema con poca preocupación por la estructura lógica interna del software. La prueba de caja blanca del software se basa en el examen cercano de los detalles de procedimiento. Las rutas lógicas a través del software y las colaboraciones entre componentes se ponen a prueba al revisar conjuntos específicos de condiciones y/o bucles. (Pressman, 2010)

3.2 Prueba de caja blanca.

La prueba de caja blanca en ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. (Pressman, 2010) Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que:

- Garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez
- Revisen todas las decisiones lógicas en sus lados verdadero y falso
- Ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas
- Revisen estructuras de datos internas para garantizar su validez

3.2.1 Prueba de ruta básica.

Para el análisis de la implementación, a partir de su código fuente, y el poder examinar los detalles procedimentales de la propuesta de solución se selecciona la técnica del camino básico, la cual determina la complejidad ciclomática de una porción de código. Este método permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. (Pressman, 2010)

```
bool Mail::testConexion()                    1
{

    ParamList params,result;                 2
    params.push<std::string>                   3
(nsearch::FNC_TEST_EMAIL_CONNECTION);        4
    bool success = sharedContext->             5
ipcCall("SearchSvc",params,result);            6
    if(success){                               7
        int ouput = result.pop<int>();         8
        if(ouput == 0){                        9
            QMessageBox  * msg = new QMessageBox;  10
            msg->setText("Conexion establecida");  11
            msg->exec();                           12

        }
        else{                        13
            QMessageBox  msg;
            msg.critical(0,"Error","No se puede conectar al
servidor. \n"
                                     "Verifique que los datos
son correctos");
        }
    }                14
    else             15
        std::cout << "Call SearchSvc Fail" << std::endl;
}                    16
```

Figura 5: Código de implementación de la HU Probar conexión.

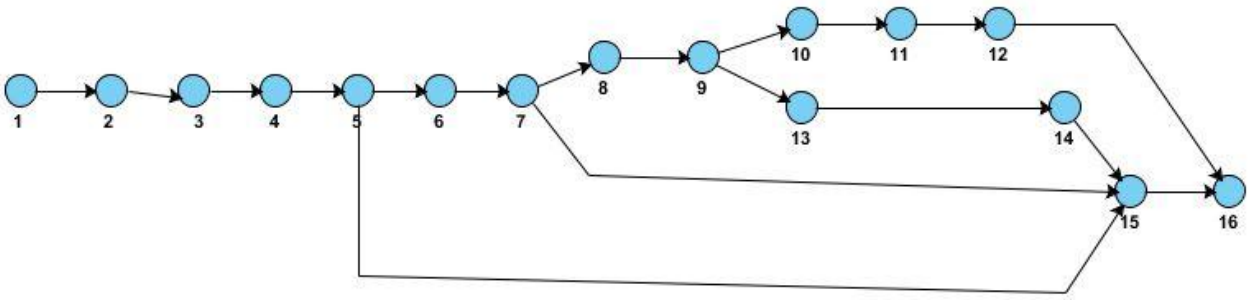

Figura 6: Representación de la prueba de ruta básica de Probar conexión.

- **Rutas básicas:**

  a) 1-2-3-4-5-15-16

b) 1-2-3-4-5-6-7-15-16

c) 1-2-3-4-5-6-7-8-9-13-14-15-16

d) 1-2-3-4-5-6-7-8-9-10-11-12-16

- **Complejidad ciclomática[V(G)]:**

  A: Número de aristas

  N: Número de nodos

  V (G) = A – N + 2

  A = 17 N = 16

  V (G) = 17 – 16 + 2

  V (G) = 3

- **Diseño de las pruebas por ruta:**

1. Ruta a: Al no haber éxito en la conexión con el servicio.

   Respuesta: Se obtiene el lanzamiento de una excepción y la finalización del algoritmo.

2. Ruta b: Se tiene éxito en la conexión con el servicio, pero no en la llamada en la que se pasan los dos parámetros (params,result).

   Respuesta: Se obtiene el lanzamiento de una excepción y la finalización del algoritmo.

3. Ruta c: Se tiene éxito en la conexión con el servicio, en la llamada en la que se pasan los dos parámetros (params,result), pero no en la respuesta de la pila result.

   Respuesta: Se genera una notificación informando que no se puede conectar con el servidor, luego se lanza una excepción y se finaliza el algoritmo.

4. Ruta d: Se tiene éxito en la conexión con el servicio, en la llamada en la que se pasan los dos parámetros (params,result) y en la respuesta de la pila result.

   Respuesta: Se notifica que la conexión está establecida mediante un mensaje en pantalla.

- **Resultado general de la prueba:**

  A partir de la aplicación del caso de prueba expuesto anteriormente se comprobó que el flujo de trabajo de las funcionalidades es correcto, ya que se demostró que cada sentencia es ejecutada al menos una vez, cumpliéndose así las condiciones de la prueba y el resultado esperado es satisfactorio.

3.3 Prueba de caja negra.

Las pruebas de caja negra también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. Las pruebas de caja negra no son una alternativa para las técnicas de caja blanca. En vez de ello, es un enfoque complementario que es probable que descubra una clase de errores diferente que los métodos de caja blanca. (Pressman, 2010)

Las pruebas de caja negra intentan encontrar errores en las categorías siguientes:

- Funciones incorrectas o faltantes
- Errores de interfaz
- Errores en las estructuras de datos o en el acceso a bases de datos externas
- Errores de comportamiento o rendimiento
- Errores de inicialización y terminación

En el módulo se emplea la técnica de partición de equivalente para la verificación de datos tanto válidos como inválidos valorando las respuestas del sistema. La partición equivalente está dirigida a la definición de casos de pruebas que descubran clases de errores, reduciendo así las clases de prueba que hay que desarrollar.

La tabla siguiente muestra los datos y sus entradas válidas e inválidas de la funcionalidad Probar conexión.

Identificador	Entrada	Clases válidas (V)	Clases inválidas (I)
Usuario	Nombre de usuario	usuario=caracteres alfabéticos	usuario=caracteres alfanuméricos y especiales usuario = [null]
Clave	Contraseña	clave=caracteres alfanuméricos y especiales	clave = [null]
Servidor	Confirmar	servidor=imap.<dominio>.uci.c	servidor=[null]

	dominio	u	servidor!= imap.<dominio>.uci.cu
Puerto	Confirmar puerto	puerto= 993	puerto= [null]

Tabla 5: Entradas y clases validas/invalidas del Probar conexión

Id	Escenario	Usuario	Clave	Servidor	Puerto	Respuesta del sistema	Resultado de la prueba
1	El usuario introduce los datos correctamente	V	V	V	V	El sistema debe verificar la veracidad del usuario y la clave, además de verificar el dominio del servidor y el puerto.	El sistema muestra una notificación informando el establecimiento de la conexión.
2	El usuario introduce datos erróneos o deja campos vacíos.	I	V	V	V	El sistema no debe permitir el establecimiento de la conexión.	El sistema muestra una notificación informando la existencia de datos erróneos o campos vacíos.
		V	I	V	V		
		V	V	I	V		
		V	V	V	I		
		I	V	I	V		
		V	I	V	I		
		I	V	V	I		
		V	I	I	V		
		I	I	V	V		
		V	V	I	I		
		I	I	I	V		

		V	I	I	I		
		I	V	I	I		
		I	I	V	I		
		I	I	I	I		

Tabla 6: Planificación por escenarios de la prueba de caja negra

Id	Usuario	Clave	Servidor	Puerto	Respuesta del sistema	Resultado de la prueba
1	aesquivel	qwerty	imap.estudiantes.uci.cu	993	El sistema muestra una notificación informando el establecimiento de la conexión.	Satisfactoria
2	Aesqui223	Qwerty789	imap.estudiantes.uci.cu	993	El sistema muestra una notificación informando la existencia de datos erróneos. 1. Usuario incorrecto 2. Clave incorrecta 3. Servidor no corresponde 4. Puerto incorrecto	Satisfactoria
	aesquivel	qwerty	imap.uci.cu	35		
	aesquivel	qwerty123	imap.estudiantes.uci.cu	993		
	aesquivel	qwerty	imap.uci.cu	993		
	aesquivel	qwerty	imap.estudiantes.uci.cu	34		

Tabla7: Ejemplo de ejecución de la prueba de caja negra

3.3.1 Resultados de la prueba

Luego de la realización de la prueba de caja negra se concluye que:

1. Tiene correspondencia la respuesta de la interfaz con los datos de entrada y el valor esperado
2. Los mensajes de error se muestran correctamente como ventanas emergentes en la página desde donde se genera
3. No existen errores en la interfaz, ya sean errores ortográficos o de funcionalidad de los botones

Por tanto se deriva que no fueron detectadas no conformidades.

3.4 Conclusiones del capítulo

En este capítulo se analizaron las principales técnicas y métodos de pruebas de software, tanto de caja blanca como de caja negra. Se determinó aplicar el método del camino básico, correspondiente al método de caja blanca llegando a las siguientes conclusiones:

- La solución desarrollada cumple con las funcionalidades especificadas en el capítulo 2.
- Las pruebas realizadas permitieron valorar los resultados obtenidos evidenciando así la calidad del módulo desarrollado.

-

Conclusiones Generales

Como resultado de la investigación se desarrolló un módulo para la detección de virus en archivos adjuntos de correo electrónico, dando con esto cumplimiento a los objetivos específicos y tareas trazadas. Se puede concluir con el desarrollo del mismo:

- El análisis de los principales conceptos relacionados con las herramientas encargadas del diseño e implementación del módulo, proporcionó los elementos teóricos necesarios para guiar el proceso de desarrollo. Además el estudio realizado de las metodologías, herramientas y tecnologías sentó las bases para el desarrollo de la solución.
- Los artefactos generados en el flujo de análisis y diseño permitieron implementar de forma satisfactoria la solución.
- La implementación del módulo para la detección de virus en archivos adjuntos de correo electrónico permitió dar cumplimiento a los requisitos funcionales identificados con anterioridad, satisfaciendo las necesidades del cliente.
- Las pruebas realizadas permitieron evaluar los resultados obtenidos.

Recomendaciones

- Incorporar al módulo un algoritmo que permita notificar al encontrar virus en un adjunto el identificador del correo recibido, el asunto así como la primera oración del contenido del mensaje.

Referencias

AC, Fernando Garcia. 2013. fergarciaac. [En línea] febrero de 2013. [Citado el: 22 de enero de 2017.] https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

Begue, Daniela Gonzalez y Potete Muñoz, Hector Vladimir. 2015. *Plataforma de Seguridad en las Tecnologías de la Información v2.0. Módulo de gestión de auditoría web.* La Habana : s.n., 2015.

Benitez, Moisés. 2013. *Políticas de seguridad informáticas.* 2013.

Bidot, Jose. 2015. Segurmatica consultoria y seguridad informática. [En línea] 2015. [Citado el: 6 de diciembre de 2016.] http://www.segurmatica.cu/laboratorio/lab2.jsp..

Brito, Karenny Acuña. 2012. Metodologías tradicionales y metodologías ágiles. [En línea] Biblioteca virtual de Derecho, Economía y Ciencias Sociales, 2012. [Citado el: 27 de octubre de 2017.] http://www.eumed.net/libros-gratis/2009c/584/Metodoligías.

Cantón, David. 2013. POP vs IMAP: Diferencias. Ventajas e inconvenientes . [En línea] 26 de enero de 2013. [Citado el: 7 de diciembre de 2016.] https://david.cantón.com/blog/informatica/pop-vs-imap-diferencias-ventajas-e-inconvenientes/1204/.

Cohen, Fred. 2015. Computer Viruses-Theory and Experiments. [En línea] 2015. [Citado el: 8 de diciembre de 2016.] http://web.eecs.umich.edu/~aprakash/eecs588/handouts/cohen-viruses.html.

Díaz, Dayné Calle. 2015. Seguridad Informática. [En línea] Infomedinstituciones, 2015. [Citado el: 6 de diciembre de 2016.] http://instituciones.sld.cu/dnspminsap/seguridad-informatica/.

Dulce María Canes . 2012. Acerca de los virus informáticos: una amenaza persistente. [En línea] 2012. [Citado el: 6 de diciembre de 2016.] http://www.redalyc.org/articulo.oa?id=368445227018. .

Fauces, Dulce Maria Canes. 2014. Acerca de los virus informáticos: una amenaza persistente. [En línea] 2014. [Citado el: 26 de octubre de 2016.] http://www.redalyc.org/articulo.oa?id=368445227018.

Gil, José A. San. 2015. Arquitectura Basada en Componentes. [En línea] 12 de abril de 2015. [Citado el: 12 de febrero de 2017.] http://es.scribd.com/doc.

Hernández, Vlamar Santos. 2004. *La industria de software.Estudio a nivel global y América Latina.* La Habana : s.n., 2004. 1.

K.I, Tong. 2010. *In Essential Skills for Agile Development.* 2010.

kasperski, Yevgueni. 2015. Kaspersky Lab. [En línea] 2015. [Citado el: 6 de diciembre de 2016.] https://www.kaspersky.com/.

Kingsley, Patrick. 2012. Father of the email attachment. [En línea] 26 de marzo de 2012. [Citado el: 7 de diciembre de 2016.] https://www.theguardian.com/technology/2012/mar/26/ather-of-the-email-attachment.

Kioskea. 2015. Patrones de diseño. [En línea] 18 de marzo de 2015. [Citado el: 13 de febrero de 2017.] http://es.kioskea.net/contents/224-patrones-de-diseno..

Koch, María José Escalona y Nora. 2012. *Ingeniería de Requisitos en Aplicaciones para la Web-Un estudio corporativo.* Sevilla : s.n., 2012.

Membrides, Antonio. 2010. *Manual de GeneSIG.* Habana : s.n., 2010.

Nazar, Luis Alejandro Fabian. 2014. slideshare. [En línea] 13 de abril de 2014. [Citado el: 22 de enero de 2016.] https://es.slideshare.net/fabiannazar1/uml-tutorialvisualparadigm.

Perezrul, Adrian Herrero. 2015. Comunicación: comunicación con cliente-servidor. [En línea] 2015. [Citado el: 7 de diciembre de 2016.] https://sites.google.com/site/mrtripus/home/sistemas-operativos-2/2-1-comunicacion-comunicacion-con-cliente-servidor-comunicacion-con-llamada-a-procedimiento-remoto-comunicacion-en-grupo-tolerancia-a-fallos. .

Porto, Julián Péres y Merino, María. 2012. definicion. [En línea] 2012. [Citado el: 22 de enero de 2016.] http://definicion.de/lenguaje-de-programacion/.

Pressman, Roger S. 2010. *Ingeniería de software, Un enfoque práctico.* México : McGraw Hill, 2010.

Quiroga, Fabricio. 2016. Virus y Antivirus. [En línea] 2016. [Citado el: 6 de diciembre de 2016.] http://www.pandasecurity.com/spain/homeusers/security-info/classic-malware/..

Ruiz, Carlos. 2016. Gestión de Riesgo em la Seguridad Informática. [En línea] 2016. [Citado el: 2 de diciembre de 2016.] https://protejete.wordpress.com/gdr_principal/definicion_si/.

Rumbaugh, James, Jacobson, Ivar y Grady Booch. 2012. *El Lenguaje Unificado de Modelado.* s.l. : I Addison Wesley Iberoamericana, 2012.

Sánchez, Tamara Rodríguez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

slideshare. 2015. es.slideshare. [En línea] 2015. [Citado el: 5 de noviembre de 2016.] https://es.slideshare.net/guidopg/qu-son-las-tics-13067328.

Yograterol. 2014. PyCharm. *PyCharm.* [En línea] 16 de 9 de 2014. [Citado el: 10 de 4 de 2016.] http://www.cristalab.com/tutoriales/pycharm-el-mejor-ide-para-tus-proyectos-en-python-c1140841/. 1.

—. 2014. PyCharm: El mejor IDE para tus proyectos en Python. *PyCharm: El mejor IDE para tus proyectos en Python.* [En línea] 16 de Septiembre de 2014. [Citado el: 10 de Abril de 2016.] http://www.cristalab.com/tutoriales/pycharm-el-mejor-ide-para-tus-proyectos-en-python-c114084l/. 2.

Bibliografía

AC, Fernando Garcia. 2013. fergarciaac. [En línea] febrero de 2013. [Citado el: 22 de enero de 2017.] https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

Arrs, Andre. 2014. Hipertextual. [En línea] 10 de junio de 2014. [Citado el: 6 de diciembre de 2016.] https://hipertextual.com/archivo/2014/06/pycharm-ide-python/.

Begue, Daniela Gonzalez y Potete Muñoz, Hector Vladimir. 2015. *Plataforma de Seguridad en las Tecnologías de la Información v2.0. Módulo de gestión de auditoría web.* La Habana : s.n., 2015.

Beltran, Carlos. 2017. CCM. [En línea] 2017. [Citado el: 2 de febrero de 2017.] http://es.ccm.net/contents/279-protocolos-de-mensajeria-smtp-pop3-e-imap4.

Benitez, Moisés. 2013. *Políticas de seguridad informáticas.* 2013.

Bidot, Jose. 2015. Segurmatica consultoria y seguridad informática. [En línea] 2015. [Citado el: 6 de diciembre de 2016.] http://www.segurmatica.cu/laboratorio/lab2.jsp..

Brito, Karenny Acuña. 2012. Metodologías tradicionales y metodologías ágiles. [En línea] Biblioteca virtual de Derecho, Economía y Ciencias Sociales, 2012. [Citado el: 27 de octubre de 2017.] http://www.eumed.net/libros-gratis/2009c/584/Metodoligías.

Callao, Raysha P Vera. 2013. Programacion. [En línea] 23 de diciembre de 2013. [Citado el: 5 de diciembre de 2016.] http://ayudaparaprogramacion.blogspot.com/2013/12/wrappers.html.

Cantón, David. 2013. POP vs IMAP: Diferencias. Ventajas e inconvenientes . [En línea] 26 de enero de 2013. [Citado el: 7 de diciembre de 2016.] https://david.cantón.com/blog/informatica/pop-vs-imap-diferencias-ventajas-e-inconvenientes/1204/.

Cohen, Fred. 2015. Computer Viruses-Theory and Experiments. [En línea] 2015. [Citado el: 8 de diciembre de 2016.] http://web.eecs.umich.edu/~aprakash/eecs588/handouts/cohen-viruses.html.

Díaz, Dayné Calle. 2015. Seguridad Informática. [En línea] Infomedinstituciones, 2015. [Citado el: 6 de diciembre de 2016.] http://instituciones.sld.cu/dnspminsap/seguridad-informatica/.

Dulce María Canes . 2012. Acerca de los virus informáticos: una amenaza persistente. [En línea] 2012. [Citado el: 6 de diciembre de 2016.] http://www.redalyc.org/articulo.oa?id=368445227018. .

Fauces, Dulce Maria Canes. 2014. Acerca de los virus informáticos: una amenaza persistente. [En línea] 2014. [Citado el: 26 de octubre de 2016.] http://www.redalyc.org/articulo.oa?id=368445227018.

FreeFind. 2012. C++ con clase. [En línea] 2012. [Citado el: 8 de diciembre de 2016.] http://c.conclase.net/.

Gil, José A. San. 2015. Arquitectura Basada en Componentes. [En línea] 12 de abril de 2015. [Citado el: 12 de febrero de 2017.] http://es.scribd.com/doc.

Grupo Advantage. 2016. Certsuperior. [En línea] 2016. [Citado el: 3 de febrero de 2017.] https://www.certsuperior.com/QueesunCertificadoSSL.aspx.

Hernández, Vlamar Santos. 2004. *La industria de software.Estudio a nivel global y América Latina.* La Habana : s.n., 2004. 1.

John Merrill. 2017. digeicert. [En línea] 2017. [Citado el: 3 de febrero de 2017.] https://www.digicert.com/es/ssl.htm.

K.I, Tong. 2010. *In Essential Skills for Agile Development.* 2010.

kasperski, Yevgueni. 2015. Kaspersky Lab. [En línea] 2015. [Citado el: 6 de diciembre de 2016.] https://www.kaspersky.com/.

Kingsley, Patrick. 2012. Father of the email attachment. [En línea] 26 de marzo de 2012. [Citado el: 7 de diciembre de 2016.] https://www.theguardian.com/technology/2012/mar/26/ather-of-the-email-attachment.

Kioskea. 2015. Patrones de diseño. [En línea] 18 de marzo de 2015. [Citado el: 13 de febrero de 2017.] http://es.kioskea.net/contents/224-patrones-de-diseno..

Koch, María José Escalona y Nora. 2012. *Ingeniería de Requisitos en Aplicaciones para la Web-Un estudio corporativo.* Sevilla : s.n., 2012.

Larman, Craig. 2005. Introducción al análisis orientado a objetos. [aut. libro] Hall Prentice. *UML y Patrones.* México : s.n., 2005.

Membrides, Antonio. 2010. *Manual de GeneSIG.* Habana : s.n., 2010.

Nazar, Luis Alejandro Fabian. 2014. slideshare. [En línea] 13 de abril de 2014. [Citado el: 22 de enero de 2016.] https://es.slideshare.net/fabiannazar1/uml-tutorialvisualparadigm.

Perezrul, Adrian Herrero. 2015. Comunicación: comunicación con cliente-servidor. [En línea] 2015. [Citado el: 7 de diciembre de 2016.] https://sites.google.com/site/mrtripus/home/sistemas-operativos-2/2-1-comunicacion-

comunicacion-con-cliente-servidor-comunicacion-con-llamada-a-procedimiento-remoto-comunicacion-en-grupo-tolerancia-a-fallos. .

Porto, Julián Péres y Merino, María. 2012. definicion. [En línea] 2012. [Citado el: 22 de enero de 2016.] http://definicion.de/lenguaje-de-programacion/.

Pressman, Roger S. 2010. *Ingeniería de software, Un enfoque práctico.* México : McGraw Hill, 2010.

Python Software Foundation. 2017. IMAP4 protocol client. [En línea] 27 de marzo de 2017. [Citado el: 10 de mayo de 2017.] https://docs.python.org/2/library/imaplib.html.

Pyton.org. 2014. Historias y Licencias de python. [En línea] Docs.python.org, enero de 2014. [Citado el: 10 de diciembre de 2016.] http://docs.python.org/license.html.

Quiroga, Fabricio. 2016. Virus y Antivirus. [En línea] 2016. [Citado el: 6 de diciembre de 2016.] http://www.pandasecurity.com/spain/homeusers/security-info/classic-malware/..

Ruiz, Carlos. 2016. Gestión de Riesgo em la Seguridad Informática. [En línea] 2016. [Citado el: 2 de diciembre de 2016.] https://protejete.wordpress.com/gdr_principal/definicion_si/.

Rumbaugh, James, Jacobson, Ivar y Grady Booch. 2012. *El Lenguaje Unificado de Modelado.* s.l. : I Addison Wesley Iberoamericana, 2012.

Sánchez, Tamara Rodríguez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

slideshare. 2015. es.slideshare. [En línea] 2015. [Citado el: 5 de noviembre de 2016.] https://es.slideshare.net/guidopg/qu-son-las-tics-13067328.

SQLite.org. 2015. SQLite Copyright. [En línea] SQLite.org, diciembre de 2015. [Citado el: 22 de febrero de 2017.] http://www.sqlite.org/copyright.html.

Yograterol. 2014. PyCharm. *PyCharm.* [En línea] 16 de 9 de 2014. [Citado el: 10 de 4 de 2016.] http://www.cristalab.com/tutoriales/pycharm-el-mejor-ide-para-tus-proyectos-en-python-c1140841/. 1.

—. 2014. PyCharm: El mejor IDE para tus proyectos en Python. *PyCharm: El mejor IDE para tus proyectos en Python.* [En línea] 16 de Septiembre de 2014. [Citado el: 10 de Abril de 2016.] http://www.cristalab.com/tutoriales/pycharm-el-mejor-ide-para-tus-proyectos-en-python-c114084l/. 2.

Anexos

Entrevista al Lic. José Bidot Peláez

Cargo: director general de Segurmática hasta 2014

- ¿Qué considera de la existencia e incremento de los virus informáticos a nivel nacional?
- Según su criterio, ¿se explotan las TIC, en función de instruir a la población, sobre el manejo de códigos malignos en archivos adjuntos?
- ¿Cuáles son los principales factores que inciden en el estado actual del desarrollo de los nuevos virus informáticos?
- ¿Conoce algunas iniciativas que se desarrollen con respecto a esta situación? ¿Alguna de ellas hace uso de las TIC?
- ¿Qué conocimientos considera necesario dominar para lograr, en un nivel básico, estar protegido contra toda posible amenaza informática?