Universidad de las Ciencias Informáticas

Facultad 2

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

Automatizar las supervisiones de seguridad informática a través del Gestor de Recursos de Hardware y Software versión 2.0.

Autor(es):

Damir Reyes Torres

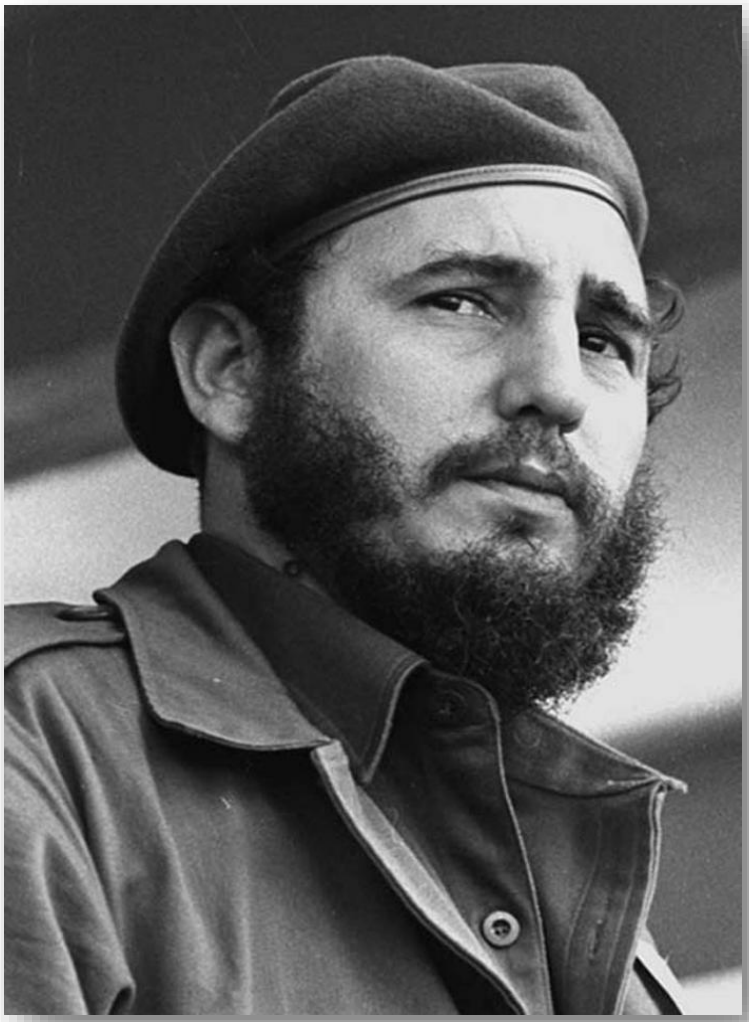
Yanira Caridad Baró González

Tutor(es):

Ing. Dania Carmenate Cantero

Ing. José Antonio Rodríguez Cascaret

La Habana, 2017



"El futuro de nuestra patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento, porque precisamente es lo que más estamos sembrando: lo que más estamos sembrando son oportunidades a la inteligencia (...)"

Datos del contacto

Declaramos ser los autores del presente trabajo de diploma y autorizamos a la Facultad 2 de la Universidad de las Ciencias Informáticas hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los _ días del mes de junio de 2017.

Autor

Yanira Caridad Baró González

Firma del Autor

Autor

Damir Reyes Torres

Firma del Autor

Tutor

Dania Carmenate Cantero

Firma del Tutor

Tutor

José Antonio Rodríguez Cascaret

Firma del Tutor

Agradecimientos

Yanira

Agradezco en primer lugar a Mukita y Tático por ser los padres más maravillosos del mundo. Por darme todo su amor, comprensión, por guiar mis pasos. Gracias a ustedes por engendrar en mí sentimientos positivos y ayudarme a ser mejor persona cada día. Gracias por los regaños, el amor y gracias por el esfuerzo que han hecho durante toda su vida para construir, formar y mantener la bella familia que tenemos.

Agradezco a José Miguel por ser tan buen amigo. Gracias por el cariño, el apoyo, la comprensión, el tiempo dedicado. Gracias te doy mi Ototе por ser mi hermano, mi otra mitad en esta vida. Te quiero hoy y siempre.

Agradezco a mis abuelas y abuelos por su cariño y sus regaños. En especial te agradezco a ti Mima por ser la abuela más maravillosa del mundo, yo sé que aunque te separaste de mi cuando tenía solo 11 años tu me ves y me cuidas desde el lugar donde estás. No me acostumbro todavía a estar sin ti.

Gracias a mis hermanas por su tiempo y cariño. En especial te agradezco Yanara por ser peleona, regañarme más que mi mamá y por todo el cariño brindado; eso me ha ayudado a ser mejor persona cada día. Te llevo en el corazón y te extraño más de lo que imaginas.

Agradezco a mis tías, tíos, primas y primos por… todo. Es realmente increíble el cariño que nos une. Gracias por su apoyo, comprensión y dedicación hacia mi persona. Gracias Adel y

Agradecimientos

Ariaxni por ser los mejores primos del mundo y ayudarme en todo lo que pueden. Tía Mercedes no sé ni que decirte eres la mejor, mi segunda madre. Tíos Alberto y Lorenzo los amo gracias por el cariño y el tiempo dedicado.

Agradezco a mi madrina Reina por todo lo que hace por mí y a mi padrino que aunque hace mucho tiempo no está físicamente su cariño brindado en tan poco tiempo también me han ayudado.

Gracias a ti Ana Laura que a pesar de los 7 años de diferencia en la edad tu apoyo y tiempo son especiales para mí. Gracias por ser esa hermana pequeña que la vida no me pudo dar. Porque sí eso eres para mí, mi hermana pequeña que cuido con tanto amor para que nadie me lastime.

Agradezco a mis maravillosos vecinos por su apoyo. En especial a Rebeca y a Yadiana, mis amigas del barrio, de la vida. Gracias por sus consejos y por escucharme siempre. Gracias a todas las amistades que tengo, por su apoyo y cariño. Gracias Fraddy por ser como eres.

Gracias a mis compañeras de apartamento: Daynela, Sandra, Yenisey, Yenlis y Yuliet por todos estos años que hemos compartido juntas. Gracias por el apoyo, las risas, los momentos vividos; pero en especial gracias por su cariño.

Gracias a mis compañeros de aula por el apoyo, los repasos, las risas. En especial agradecerte Marien por ser mi amiga en la universidad. Por escucharme, apoyarme y ayudarme en todo lo que necesito. Gracias Luis Angel por ser una persona tan maravillosa y un compañero espectacular. Yinet aunque no estés

aquí en la escuela gracias por ser esa amiga loca que toda persona debe tener. Te quiero mucho.

Agradezco a todas las amistades que he conocido en la universidad por su tiempo, dedicación y cariño. Pino, Yasiel, Ana Fe, Cuso, Naila, Xiomara, Yadián gracias por los momentos de alegría, el apoyo, las comidas, las fiestas, las conversaciones constructivas. Gracias Héctor Mario por el apoyo y la comprensión de estos últimos meses.

Gracias a todos los profesores que han formado parte de mi formación como profesional. Gracias Sailyn por el apoyo brindado en esta etapa. Gracias a todas las personas que brindaron su ayuda en el trabajo realizado.

Gracias a Dios y a la vida por permitirme estar aquí este día.

... y por último y no menos importante agradecer a mi compañero de tesis por los chistes, los momentos compartidos y su apoyo brindado. Gracias Damir.

Damir

Agradezco principalmente a mi madre por su apoyo incondicional ante cualquier adversidad, a mi hermana por alentarme día tras día, a mi sobrina por hacerme feliz con sus ocurrencias, a Yuli por motivarme a luchar hasta el final, a mis tíos, primos, a todos mis amigos, a mi compañera de tesis. A todos gracias pues sin ellos no hubiese podido ser posible la culminación de este trabajo y por sobre todo gracias a Dios

Dedicatoria

Dedico el trabajo realizado a mis padres y mi hermana Yanara. A Ana Laura y José Miguel por permitirme ser su hermana. A Alberto y Mercedes. A mis sobrinas Yelennis y Sarahí. A mima y tío Raudel que yo sé que desde el cielo me cuidan el sueño y velan por mi felicidad.

Yanira

Dedico este trabajo a mi madre, familia, amigos y a Dios.

Damir

Resumen

Resumen

En la Universidad de las Ciencias Informáticas se encuentra el centro de Telemática perteneciente a la Facultad 2. Dicho centro es el encargado del desarrollo y mantenimiento del Gestor de Recursos de Hardware y Software (GRHS). Este sistema cuenta con un módulo que verifica las políticas de seguridad informática. La comunicación del servidor hacia el cliente es necesario establecerla para que, desde un servidor, se puedan enviar mensajes al cliente para que verifique las políticas de seguridad en la red de computadoras donde esté instalado el cliente del GRHS. Para el desarrollo de la solución se actualizó el subplugin antivirus, que presentaba un problema en el momento de obtener las propiedades relacionadas con la actualización. Se estableció la comunicación del servidor hacia el cliente, empleando AMQP como protocolo de mensajería AMQP y RabbitMQ como sistema de mensajería. Con el desarrollo de la solución el asesor de seguridad puede desde un servidor verificar las políticas de seguridad de una estación de trabajo cuando lo necesite. Para el desarrollo de la solución se siguieron los pasos que propone la metodología AUP_UCI. Se seleccionaron las herramientas y tecnologías necesarias para el desarrollo de la misma.

Palabras claves: cliente, comunicación, GRHS, políticas de seguridad informática**,** servidor.

Índice

Índice de contenido

Introducción 1
Capítulo 1 Fundamentación Teórica 6
1.1 Introducción 6
1.2 Conceptos 6
1.3 Protocolo de mensajería AMQP 7
1.4 Sistemas de mensajería: 7
1.4.1 Conclusiones de los sistemas de mensajería estudiados: 8
1.5 Metodología 8
1.5.1 Metodología seleccionada: AUP_UCI 9
1.5.2 Descripción de la metodología AUP_UCI 9
1.6Tecnología y herramientas 10
1.6.1 Lenguajes de programación. 10
1.6.2 Framework 11
1.6.3 Herramienta CASE 12
1.6.4 Sistema Gestor de Bases de Datos (SGBD) 12
1.6.5 Sistema distribuido 12
1.6.6 Editor de código 12
1.7 Conclusiones Parciales del Capítulo 13
Capítulo 2 Propuesta y diseño de la solución. 14
2.1 Introducción 14
2.2 Propuesta de solución 14
2.3 Modelo de dominio 15
2.3.1 Descripción de los elementos del modelo de dominio. 16
2.4 Requisitos 16
2.4.1 Requisitos funcionales: 16
2.4.2 Requisitos no funcionales: 16
2.5 Diagrama de caso de uso del sistema 17
2.5.1 Descripción de los casos de uso del sistema. 17
2.6 Análisis y diseño 23
2.6.1 Diagrama de colaboración 23
2.6.2 Diagrama de clases con estereotipos web 24
2.7.2 Arquitectura cliente-servidor. 25
2.7.3 Patrón arquitectónico 26
2.7. Modelo de datos 27
2.9 Patrones de diseño 29
2.9.1 Patrones GRASP 29

Índice

Capítulo 3 Implementación y prueba 31
3.2 Diagrama de componentes 31
3.3 Modelo de despliegue 32
3.3Aplicación y resultados de las pruebas 33
3.3.1 Pruebas internas 33
3.3.2 Pruebas de aceptación 34
Conclusiones Generales 39
Referenciasbibliográficas 41
Bibliografía 45

Índice

Índice de figuras

Figura 1: Funcionamiento de la comunicación del servidor con el cliente. 15
Figura 2: Modelo de dominio 15
Figura 3: Diagrama de Caso de uso del sistema 17
Figura 4: Diagrama de colaboración del Caso de uso Verificar políticas. 24
Figura 5: Diagrama de colaboración del Caso de uso Cargar plugin 24
Figura 6: Diagrama de clases de diseño con estereotipos web del caso de uso Verificar políticas. 25
Figura 7: Funcionamiento de la arquitectura cliente servidor 26
Figura 8: Patrón Modelo-Plantilla-Vista 27
Figura 9: Modelo físico de la base de datos 28
Figura 10: Diagrama de componentes 32
Figura 11: Diagrama de despliegue. 33
Figura 12: Muestra de los casos de pruebas unitarias aplicados. 34
Figura 13: Resultado de los casos de prueba aplicados 34
Figura14: Comportamiento de las clases satisfactorias y no satisfactorias 37

Índice

Índice de tablas

Tabla 1: Caso de uso Verificar políticas 17
Tabla 2: Caso de uso: Cargar plugin 21
Tabla 3: Uso del patrón Experto 29
Tabla 4: Caso de Prueba Comprobar políticas de seguridad seleccionando la localización. 36
Tabla 5: Caso de prueba: Comprobar políticas de seguridad seleccionando estación de trabajo. 36
Tabla 6: Obtener antivirus instalado. 37
Tabla 7. Caso de prueba: Verificar actualización de antivirus en la máquina. 47

Introducción

Introducción

El avance de la ciencia y la técnica ha propiciado el desarrollo de una sociedad donde tienen un protagonismo cada vez mayor las Tecnologías de la Información y las Comunicaciones (TIC), trayendo consigo un auge en el ámbito de la informática y las redes de computadoras. El empleo de estas redes y los beneficios que aportan para las instituciones y entidades ha propiciado que su uso se extienda hacia todos los sectores de la sociedad.

Una red de computadoras está constituida por un conjunto de dispositivos que permiten el intercambio de información, servicios y recursos entre ellos. Tener el control de la información referente al hardware y software de los dispositivos que operan y componen la red, es una labor necesaria para los propietarios o administradores. Con el objetivo de satisfacer esta necesidad surgen sistemas con la capacidad de brindar información de hardware y software reflejada a modo de inventario (1).

La Universidad de las Ciencias Informáticas (UCI) es una de las entidades productoras de software en el país. Cuenta con gran cantidad de dispositivos conectados a la red, haciendo que sea extensa y compleja. Para inventariar estos dispositivos, la Dirección de Seguridad Informática de la universidad hace uso del Gestor de Recursos de Hardware y Software (GRHS). Este gestor, controla los medios informáticos determinando cambios en ellos y alertando en caso de que un cambio sea no autorizado. El GRHS está compuesto por los subsistemas: Gadmin, Gserver (ambos en el servidor) y Gclient (en los clientes). El procesamiento de la información suministrada por el cliente se realiza de manera asíncrona en el servidor a través del protocolo AMQP (Advanced Message Queuing Protocol).

El centro Telemática (TLM), desarrolla sistemas relacionados con las telecomunicaciones y la seguridad informática, es el encargado del desarrollo y mantenimiento del GRHS. En el centro se desarrolla el módulo de Seguridad Informática para este gestor. Dicho módulo se encarga de comprobar las políticas de seguridad en la red de computadoras de cada centro donde se encuentre instalada la aplicación cliente del GRHS. El asesor de seguridad es la persona que se debe encargar de conocer en que estación de trabajo se están cumpliendo o no con las políticas de seguridad.

El módulo Seguridad informática de GRHS está compuesto por un plugin para gclient y un módulo para la aplicación gadmin del sistema GRHS. El plugin de seguridad informática por su parte está integrado por subplugins, los cuales son los encargados

de obtener de las máquinas donde está instalado gclient, aquellas propiedades vinculadas a las políticas de seguridad: nombre del dominio, identificador, instalación de antivirus y firewall, grupos de administradores, carpetas compartidas, sistema operativo, entre otras propiedades. Luego el asesor de seguridad se encarga de visualizar si la máquina cumple con el uso de las políticas. La aplicación gclient envía la información al servidor (gserver), donde se encuentran almacenadas en una base de datos, la información enviada por los clientes de la red de computadoras con que cuenta el centro TLM. Dicha información, que se encuentra en el servidor, es utilizada por el módulo de seguridad informática en la aplicación gadmin. En dicha aplicación el asesor de seguridad puede visualizar todas las estaciones de trabajo del centro con sus propiedades (2).

En el módulo Seguridad Informática del GRHS la comunicación del servidor hacia el cliente se realiza por http, por razones de seguridad no se pueden realizar peticiones desde el servidor hacia el cliente ya que se podía suplantar la identidad de un usuario además de que se puede apagar o reiniciar cualquier máquina, lo que traía como consecuencia la pérdida de información o la interrupción de algún servicio. Debido a esto el sistema verifica las políticas de seguridad cuando el cliente es instalado o reiniciado; por lo que el asesor, desde un servidor, no puede verificar las políticas de seguridad en el momento que sea necesario.

Además en el módulo, la verificación de las políticas presenta como deficiencia que la actualización del antivirus no se muestre de la forma correcta. Si el antivirus se actualiza en el transcurso del día en que fueron verificadas las políticas de seguridad, el sistema lo muestra como que se encuentra desactualizado. Esto trae como consecuencia que el asesor informe que no se cumple con esa política de seguridad en dicha estación de trabajo.

A partir de la **problemática** descrita anteriormente se define como **problema a resolver**: ¿Cómo establecer la comunicación del servidor hacia el cliente para iniciar la verificación de las políticas de seguridad informática en el sistema GRHS?

Se define como **objeto de estudio**: El proceso de comunicación entre el servidor y el cliente.

Para dar solución al problema a resolver **el objetivo general es**: Establecer la comunicación desde el servidor hacia el cliente para iniciar la verificación de las políticas de seguridad informática en el sistema GRHS.

Introducción

El **campo de acción** en que se va a desarrollar la investigación es: El proceso de comunicación del servidor hacia el cliente en el sistema GRHS.

Para guiar la investigación se establecen las siguientes **preguntas científicas**:

¿Qué sistema de mensajería se utilizará para el envío de mensajes del servidor al cliente?

¿Qué herramientas y metodología serán utilizadas para el diseño y desarrollo de la solución?

¿Qué resultados se alcanzarán con la aplicación de pruebas a la solución?

Para dar solución al objetivo general se han propuesto las siguientes **tareas de investigación:**

- Estudio de los sistemas de mensajería para seleccionar cual será el empleado en el paso de mensajes del servidor al cliente.
- Estudio y selección de las herramientas y metodología utilizadas para el diseño y posterior desarrollo de la solución.
- Análisis de los elementos correspondientes al diseño de software para guiar la implementación de la solución.
- Revisión de las pruebas que propone la metodología a seleccionar para verificar el correcto funcionamiento de la aplicación.

Para la realización de este trabajo se aplican los métodos científicos de la investigación, **teóricos y empíricos**. Entre los métodos teóricos se utiliza específicamente el **Analítico-Sintético** y la **Modelación**. El primero permite buscar la esencia de los fenómenos, así como los rasgos que lo caracterizan y los distinguen, por lo que facilita el análisis de las teorías y documentos. En el trabajo de diploma, este método permite extraer y sintetizar los elementos más relacionados y de mayor importancia con el objeto de estudio. El segundo crea abstracciones, con el objetivo de explicar la realidad a través de los modelos. En el trabajo de diploma se puede apreciar el uso de este método cuando se crean los diferentes diagramas. Se emplea el método empírico **Entrevista** ya que es una conversación planificada entre el investigador y el entrevistado para obtener información. Su uso constituye un medio para el conocimiento cualitativo de los fenómenos. Para la realización de este trabajo de diploma es necesario establecer conversaciones planificadas con el fin de obtener

información y opiniones del cliente para así obtener un software que cumpla con los requerimientos del mismo.

El trabajo de diploma está dividido en tres capítulos estructurados de la siguiente forma:

Capítulo 1 Fundamentación Teórica: En el capítulo se hace referencia a los principales conceptos relacionados con el trabajo diploma. Además, se estudian los sistemas relacionados con el tema para comprender mejor el objetivo. Se selecciona la metodología de desarrollo y las diferentes herramientas y tecnologías a usar.

Capítulo 2 Propuesta y diseño de la solución: En el presente capítulo se exponen las características con las que cuenta la solución que se va a desarrollar. Se presentan los requisitos funcionales y no funcionales de la solución. Se describen las disciplinas de Requisitos y Análisis y diseño, propias de la metodología seleccionada. Se define la arquitectura a utilizar en la solución.

Capítulo 3 Implementación y prueba: A partir del modelado de la propuesta en el capítulo anterior se representa el diagrama de componente y despliegue de la solución desarrollada. Se verifica la calidad del resultado de la implementación, mediante las pruebas realizadas al producto obtenido.

Capítulo 1 Fundamentación Teórica

1.1 Introducción

En el presente capítulo se hace referencia a los principales conceptos relacionados con el trabajo diploma. Se describe el protocolo de mensajería seleccionado y se estudian sistemas de mensajería para seleccionar cuál será el empleado para dar solución al problema a resolver. En el capítulo se describe la metodología de desarrollo de software seleccionada. Se definen las herramientas y tecnologías para la implementación de la propuesta de solución.

1.2 Conceptos

Un **sistema informático** es el conjunto de partes interrelacionadas, hardware, software y de recursos humanos que permite almacenar y procesar información. El hardware incluye computadoras o cualquier tipo de dispositivo electrónico inteligente, que consisten en procesadores, memoria, sistemas de almacenamiento externo, entre otros. El software incluye al sistema operativo y aplicaciones, siendo especialmente importante los sistemas de gestión de bases de datos. Por último, el soporte humano incluye al personal técnico que crean y mantienen el sistema (analistas, programadores, operarios, entre otros) y a los usuarios que lo utilizan (3).

La 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 (4).

Los **sistemas de mensajería** son una categoría de software para la comunicación de forma asincrónica, desacoplada, confiable, escalable y segura entre aplicaciones distribuida o sistemas. Permiten la comunicación entre aplicaciones, usando las interfaces de aplicaciones programadas que ofrece cada proveedor (5).

La **comunicación asíncrona** es la conexión que se establece entre el cliente y el servidor que permite la transferencia donde el cliente puede realizar varias peticiones al servidor sin necesidad de esperar por la respuesta del primero (6).

Un **plugin** es un componente que se carga dinámicamente en un programa informático para añadir funcionalidades adicionales o una nueva característica al software. Es ejecutado mediante el software principal, con el que interactúa a través de una interfaz. La aplicación principal proporciona servicios que el plugin puede utilizar, pero el plugin depende de los servicios prestados por la aplicación de acogida.

La aplicación principal funciona independientemente de ellos, lo que permite a los usuarios finales añadir y actualizar los plugin de forma estática o dinámica sin necesidad de hacer cambios a la misma (7).

1.3 Protocolo de mensajería AMQP

El módulo seguridad informática del GRHS tiene establecida la comunicación del cliente al servidor por el protocolo AMQP. Con el objetivo de mantener el mismo protocolo de comunicación del cliente al servidor y del servidor al cliente se selecciona este protocolo para establecer la comunicación en el módulo Seguridad Informática del GRHS. Es un protocolo estándar abierto orientado a mensajes, cuyo objetivo principal es la interoperabilidad de diferentes servicios. Soporta modelos de comunicación punto a punto y publicación/subscripción (8). Define el formato de los datos que van a ser enviados, de manera que cualquier aplicación pueda crear y leer mensajes en este formato de datos independientemente del lenguaje usado. En esta tecnología los mensajes se almacenan en colas, las cuales garantizan que estos se entreguen en el mismo orden en el que han llegado siguiendo el mecanismo FIFO (Fist In, FirstOut).

1.4 Sistemas de mensajería:

Apache ActiveMQ

Es un mediador de mensajes entre aplicaciones emisoras y receptoras. Es de código abierto. Se distribuye bajo Licencia Apache. Implementa la especificación de Java Message Service(JMS). Ofrece características como clustering[1], múltiples almacenes para mensajes, así como la capacidad de emplear cualquier administrador de base de datos como proveedor de persistencia JMS. El objetivo principal de ActiveMQ es suministrar estándares basados en la mensajería orientada a la integración de aplicaciones a través de múltiples lenguajes y plataformas (9).

RabbitMQ

Es un intermediario de mensajes de código abierto. Los editores envían mensajes a los intercambios y los consumidores recuperan los mensajes de las colas. Ofrece una serie de escenarios de implementación distribuida (y requiere que todos los nodos puedan resolver nombres de host). Este brinda a las aplicaciones una plataforma común para mandar y recibir mensajes, y a los mensajes un entorno seguro para

[1]técnica que permite combinar múltiples sistemas para que trabajen en paralelo y se comporten como un recurso informático unificado para: servir a un grupo de tareas, proporcionar tolerancia a fallos y tener disponibilidad continua.

persistir hasta que estos sean recibidos, utiliza AMQP como protocolo de comunicación (10).

Apache Kafka

Es un sistema de mensajería de publicación-suscripción rápido, escalable y duradero. Kafka proporciona un almacén de mensajes duradero, similar en cierto modo a una base de datos, que se ejecuta en un clúster de servidores, que almacena los flujos de registros en categorías llamadas temas. Retiene todos los mensajes por un período de tiempo determinado, y los consumidores son responsables de rastrear su ubicación en cada registro. Kafka puede soportar un gran número de consumidores y retener grandes cantidades de datos con muy poca sobrecarga (11).

1.4.1 Conclusiones de los sistemas de mensajería estudiados:

Luego de realizar un estudio a los sistemas descritos anteriormente se puede concluir que no se selecciona Apache Kafka ya que utiliza un protocolo propio basado en TCP y Apache Zookeeper, y para desarrollar la solución se emplea el protocolo AMQP. Por su parte ActiveMQ presenta problemas si tiene que tratar mensajes perdidos. El sistema que se selecciona es el RabbitMQ, ya que rastrea el mal comportamiento en las colas, evitando que el sistema de encolamiento falle. Soporta plugins para extender su comportamiento y permite un entorno seguro a los mensajes hasta que sean recibidos. Además RabbitMQ implementa el protocolo de comunicación asíncrono AMQP que es el protocolo seleccionado.

1.5 Metodología

Una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. La metodología indica cómo hay que obtener los distintos productos parciales y finales (12). Teniendo en cuenta su filosofía de desarrollo se pueden clasificar en Metodologías Tradicionales, conocidas también como metodologías pesadas y Metodologías Ágiles (13).

Las metodologías tradicionales o pesadas exigen una abundante y exhaustiva documentación, así como una especificación precisa de requisitos y del modelado. Se centran en una detallada planificación, desde la fase inicial del proyecto. Se ajustan a proyectos de largo plazo de duración. Entre este tipo de metodología se encuentra RUP (Rational Unified Procces), MSF (Microsoft Solution Framework), Win-Win Spiral Model, Iconix.

En las metodologías ágiles es más importante construir un buen equipo que construir el entorno. Se centra fundamentalmente en el desarrollo de un buen software y no en la documentación del mismo. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Estas metodologías tienen la habilidad de responder a los cambios que puedan surgir a lo largo del proyecto.Entre los ejemplos de tipos de metodologías ágiles se encuentra XP (Extreme Programming), Scrum, Crystal Clear, AUP(Agile UnifiedProcess), ASD (Adaptive Software Development), AUP_UCI.

1.5.1 Metodología seleccionada: AUP_UCI

La metodología a escoger para el desarrollo del presente trabajo es AUP_UCI ya que integra en una, todas las metodologías usadas en la UCI(XP, OPEN UP, RUP, BPM, DAC, KIMBALL, SXP, SCRUM, NOVA OPEN UP). Además, presenta un enfoque ágil y está adaptada al ciclo de vida definido para la actividad productiva de la UCI. Garantiza que el trabajo sea exitoso, satisface las necesidades cambiantes del cliente, así como la relación estrecha que se establece con él. AUP_UCI es la metodología empleada en el GRHS y como el objetivo es desarrollar e integrar un módulo a este gestor, se debe seguir la misma línea de trabajo.
El desarrollo de la solución se centra en la fase de Ejecución de la metodología. Además, se generarán los productos de trabajo definidos para las disciplinas Requisitos, Análisis y diseño, Implementación, Pruebas internas y de Aceptación. En el epígrafe siguiente se describe la metodología seleccionada.

1.5.2 Descripción de la metodología AUP_UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable; 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. Con la adaptación de AUP para la actividad productiva de la UCI se logra estandarizar el proceso de desarrollo de software. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos (14).

Fases de Variación de AUP para la UCI

La metodología Variación de AUP para la UCI está formada por tres fases (Inicio, Ejecución y Cierre) para el ciclo de vida de los proyectos de la universidad. Las características de las fases de la metodología de la universidad son:

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el software es transferido al ambiente de los usuarios finales o entregado al cliente junto con la documentación.
- **Cierre:** En el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Disciplinas de Variación de AUP para la UCI

Para el ciclo de vida de los proyectos de la UCI se decidió contar con 8 disciplinas: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas interna, Pruebas de liberación, Pruebas de Aceptación y Despliegue.

1.6Tecnología y herramientas

1.6.1 Lenguajes de programación.

Python 2.7

Es un lenguaje de programación de alto nivel, diseñado para ser fácil de leer y simple de implementar. Es código abierto (de libre uso). Es usado para desarrollar aplicaciones web y contenido web dinámico(15). Es el lenguaje de programación que utiliza el sistema GRHS.

HTML5

Es el lenguaje de marcado que hace que el proceso de codificación sea más fácil. Muchas de las características de HTML5 permiten a los usuarios ejecutar contenidos complejos en plataformas de baja potencia (16). Se utilizará este lenguaje para el diseño de las interfaces en la solución, aprovechando las características del soporte para CSS3.

CSS3

Ofrece una gran variedad de opciones muy importantes para las necesidades del diseño web actual. Desde opciones de sombreado y redondeado, hasta funciones avanzadas de movimiento y transformación (17). Se utilizará CSS3 para obtener interfaces con un diseño más agradable.

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente del lado del cliente, permitiendo crear efectos atractivos y dinámicos en las páginas web. La ventaja de JavaScript es que al estar alojado en el ordenador del usuario los efectos son muy rápidos y dinámicos. Al ser un lenguaje de programación permite toda la potencia de la programación como uso de variables, condicionales, bucles (18). Este lenguaje se utilizará para el manejo de los datos del lado del cliente.

1.6.2 Framework

Django 1.8

Es un framework de código abierto escrito en Python, que nos permitirá construir aplicaciones web rápidamente y con menos programación. Está orientado a cumplir con el paradigma Modelo-Vista-Controlador (MVC). Posee su propia librería de mapeo Objeto-Relacional, contiene bibliotecas HTTP y sistema de plantillas(19). Es el marco de trabajo que utiliza GRHS.

JQuery 1.9

JQuery es una biblioteca JavaScript que facilita la tarea de programar páginas web. Reduce al mínimo nuestro código de muchas líneas en JavaScript a unas cuantas, que hacen exactamente lo mismo, pero con JQuery (20). Este framework se utilizará para manejar todas las operaciones de las acciones del cliente en la aplicación gadmin del sistema GRHS.

Twitter Bootstrap 3.0

Twitter Bootstrap es una colección de herramientas HTML y plantillas CSS, que se utilizan para el desarrollo y diseño de un sitio web o aplicaciones web (21). Se utilizará para la creación de las interfaces proporcionando un diseño sólido basado en estándares como CSS3 y HTML5 que también serán usados en la solución.

Backbone 1.1

Este pequeño framework que permite construir aplicaciones usando JavaScript siguiendo el patrón MVC (modelo-vista-controlador). El objetivo de Backbone ha sido desde sus inicios probar y definir un conjunto de estructuras de datos (Models y Collections) junto al manejo de la interfaz por medio de Vistas y URLs que fueran útiles cuando se construyan aplicaciones JavaScript (22).

Xilema Base Web

Es un marco de trabajo desarrollado en el Centro de TLM que está constituido por Django como framework base, librerías de JavaScript como son JQuery y Backbone además cuenta con las pautas de diseño de la Universidad.

1.6.3 Herramienta CASE

Visual Paradigm 8.0

Es una herramienta CASE que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un software. Permite coherencia entre diagramas, controla que el modelado con UML sea correcto (23). Esta herramienta es utilizada para el diseño de los artefactos generados por la metodología seleccionada.

1.6.4 Sistema Gestor de Bases de Datos (SGBD)

PostgreSQL 9.3

Es un SGBD. Sus características técnicas lo hacen un gestor de base de datos potente y robusto. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes, cadenas de bits y se le incluye entre los gestores objeto-relacionales. Es ideal para tecnologías Web y es fácil de administrar (24).

1.6.5 Sistema distribuido

Celery

Es una cola de tareas asíncrona Open Source, basada en el envío distribuido de mensajes, que nos permite enviar tareas a una cola, para que se ejecuten cuando sea posible. Las unidades de ejecución, llamadas tareas, se ejecutan de manera concurrente en uno o más nodos de trabajo. Estas tareas pueden ejecutarse tanto asíncrona como síncronamente, siendo el sistema en general altamente configurable (10). Celery es utilizado para distribuir las tareas relacionadas con la comunicación del cliente y el servidor.

1.6.6 Editor de código

SublimeText 3

Es uno de los editores de texto líderes del mercado, la extensibilidad de su sistema de paquetes le hace muy flexible. Cada desarrollador puede seleccionar los paquetes que más le interesan para cada proyecto. Su sistema de resaltado de sintaxis soporta un gran número de lenguajes como CSS, HTML, JavaScript y Python (25).

1.7 Conclusiones Parciales del Capítulo

En el presente capítulo se analizaron los principales conceptos relacionados con la investigación. Se realizó un estudio de los sistemas de mensajería para seleccionar al RabbitMQ como el sistema a emplear en la solución.Como metodología a emplear se escogió la metodología AUP_UCI. Se realizó un estudio de las posibles herramientas y tecnologías a emplear, escogiendo para el desarrollo de la solución; como lenguajes de programación: Python, HTML5, CSS 3 y JavaScript, como entorno de desarrollo Sublime Text 3 y como SGBD se seleccionó PostgreSQL. Como framework de desarrollo para los lenguajes seleccionados: Django, JQuery, Twitter Bootstrap, Backbone y Xilema Base Web fueron los seleccionados.

Capítulo 2

Capítulo 2 Propuesta y diseño de la solución.

2.1 Introducción

En el presente capítulo se exponen las características con las que cuenta la aplicación que se va a desarrollar. Se presentan los requisitos funcionales y no funcionales de la solución. Se describen las disciplinas Requisitos y Análisis y diseño, propias de la metodología seleccionada.

2.2 Propuesta de solución

La propuesta de solución está encaminada a establecer la comunicación del servidor hacia el cliente en el módulo Seguridad Informática para que se puedan verificar las políticas de seguridad informática en el cliente a partir de una petición del servidor. Esta comunicación se realizaba por http, y por razones de seguridad no se podían realizar peticiones desde el servidor hacia el cliente ya que se podía suplantar la identidad de un usuario. Además se podía apagar o reiniciar cualquier máquina tryend como consecuencia pérdida de información, interrupción de servicio o una falla técnica en el hardware. Para establecer la comunicación de forma segura se a va emplear el servidor de mensajería RabbitMQ que implementa el protocolo AMQP y utiliza colas de mensajes como intermediario entre el cliente y el servidor.

La aplicación Gserver tiene almacenada en una base de datos la información enviada por los clientes de la red de computadoras, el servidor le solicitará a la aplicación cliente la verificación de las políticas de seguridad de una determinada subred, una determinada máquina o una localización. Gserver va a producir mensajes para que estos lleguen a su destinatario. Estos mensajes son guardados en una cola que tiene Gclient en el servidor de mensajería RabbitMQ. El cliente es el encargado de desencolar los mensajes y procesarlos. Posteriosmeente Gadmin mostrará la información obtenida desde Gclient a través de Gserver. Si la aplicación falla mientras se está procesando alguna petición esta estrategia permite al mensaje persistir en la cola hasta que sea procesado por completo. En la figura 1 se muestra la propuesta de solución anteriormente descrita.

Además, el subplugin antivirus no muestra correctamente la información referente a su actualización. Si se actualizó el mismo día en que son verificadas las políticas de seguridad no muestra que se encuentra actualizado. Para ello se propone realizar un cambio en el resultado de la diferencia entre la fecha actual y la fecha de verificación de las políticas.

Figura 1: Funcionamiento de la comunicación del servidor con el cliente.

2.3 Modelo de dominio

Un modelo de dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos del software. Muestra (a los modeladores) clases conceptuales significativas en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos (26). En la Figura 2 se muestra el modelo de dominio realizado.

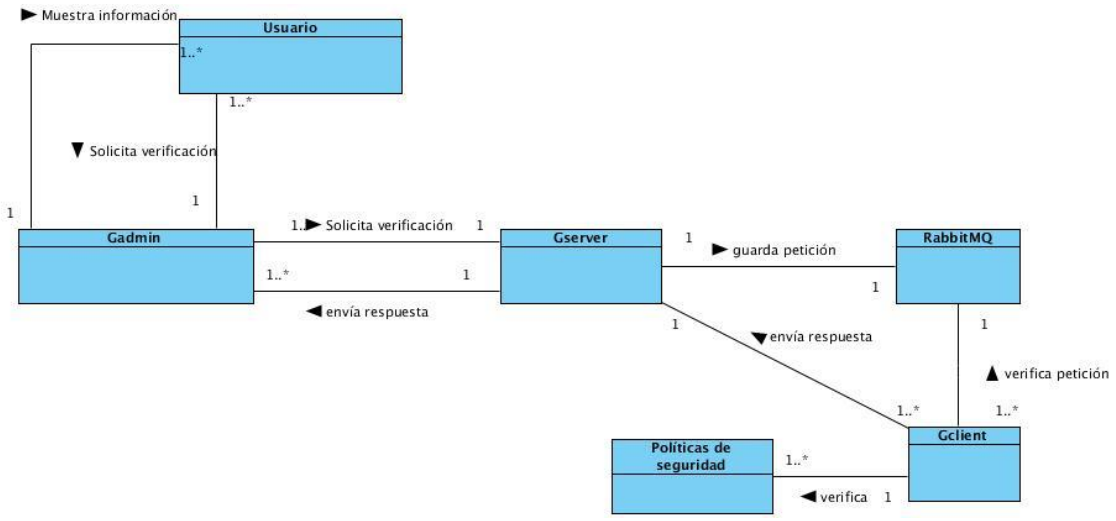


Figura 2: Modelo de dominio

2.3.1 Descripción de los elementos del modelo de dominio.

Usuario: persona que interactúa con el sistema.

Gadmin: consola de administración que permite visualizar toda la información a través de las interfaces.

Gserver: sistema del lado del servidor, encargado de recibir toda la información de los inventarios realizados; así como los cambios en estos y las incidencias; además de almacenar en base de datos.

Gclient: aplicación instalada en las estaciones de trabajo que se encarga de capturar la información referente a las políticas de seguridad.

RabbitMQ: se encarga de enviar y recibir mensajes permitiéndoles un entorno seguro para persistir hasta que estos sean recibidos.

Políticas de Seguridad: las políticas de seguridad informática que se verifican en cada estación de trabajo.

2.4 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.

2.4.1 Requisitos funcionales:

RF1: Comprobar políticas de seguridad seleccionando la localización.
RF2: Comprobar políticas de seguridad seleccionando estación de trabajo.
RF3: Obtener propiedades del antivirus.
RF3.1: Verificar actualización de antivirus en la máquina.

2.4.2 Requisitos no funcionales:

RNF1: Usuabilidad
Para interactuar con el sistema es necesario que el usuario posea conocimientos básicos de informática.
RNF2: Hardware
Características de hardware del servidor
2 GB de RAM
2.3 GHZ el procesador
10 GB de disco duro

Características de hardware del cliente

256 MB de RAM

2.0 GHZ el procesador

2 GB de espacio en disco duro

RNF3: Software

Características de software en el servidor

Servidor web: Nginx 1.2, uWsgi

Python 2.7

Características del software en el cliente

Sistema operativo: Windows 7, Windows 8, Windows 10, Ubuntu 11.04, Ubuntu 12.04, Ubuntu 14.04, Ubuntu 15.04, Ubuntu 15.10, Ubuntu 16.04.

Python 2.7.

2.5 Diagrama de caso de uso del sistema

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario. Determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema puede ejecutar (27). Los actores que intervienen en el sistema se corresponden con los usuarios que permite la aplicación. En este caso pueden acceder a la aplicación los asesores de seguridad que serían los usuarios del sistema. En la figura 3 se muestra el diagrama de caso de uso del sistema.

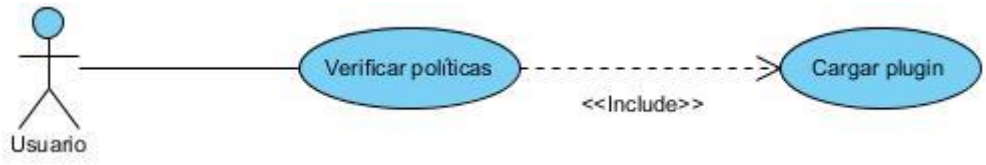


Figura 3: Diagrama de Caso de uso del sistema

2.5.1 Descripción de los casos de uso del sistema.

Para la descripción de los casos de usos se emplean tablas. En las celdas se presentan elementos del caso de uso como: su nombre, la prioridad y complejidad, sus condiciones, así como sus flujos de eventos básico y alterno, y los actores que intervienen.

Tabla 1: Caso de uso Verificar políticas

Objetivo	Comprobar las políticas de seguridad a cada una de las

	estaciones de trabajo.
Actor	Usuario
Resumen	El caso de uso inicia cuando el usuario selecciona en el menú, el módulo Seguridad. Luego selecciona la opción Comprobar Políticas de Seguridad. El sistema muestra una nueva interfaz donde el usuario debe seleccionar la localización a la que desee comprobar las políticas de seguridad. El sistema verifica las políticas de seguridad para la localización seleccionada y muestra la información obtenida.
Complejidad	Alta
Prioridad	Alta
Precondiciones	Seleccionar el módulo Seguridad.
Poscondiciones	Se muestra la información de las políticas de seguridad verificadas.
Flujo de eventos	
Flujo básico: “Seleccionar localización”	

	Actor	**Sistema**
1	Selecciona el módulo Seguridad.	Accede a la interfaz del módulo Seguridad.
2	Selecciona la opción Comprobar Políticas de Seguridad.	Selecciona la localización. Selecciona las políticas de seguridad. (Ver Sección 1: Seleccionar políticas de seguridad.)

		Selecciona la opción Aceptar.
3	Selecciona la localización. Selecciona las políticas de seguridad. Selecciona la opción Aceptar.	Carga el plugin del lado del cliente. Recopila la información referente a las políticas de seguridad. Almacena la información en su base de datos. Muestra la información. Finaliza el caso de uso.
Flujos alternos		
Evento 2: Si selecciona al menos una estación de trabajo.		
	Actor	**Sistema**
1	Selecciona la(s) estación(es) de trabajo que desea verificar en ese momento. Selecciona la opción Comprobar Políticas de Seguridad.	Muestra una nueva interfaz con las políticas (Ver Sección 1: Seleccionar políticas de seguridad.)
2	Selecciona la opción Aceptar.	Carga el plugin del lado del cliente. Recopila la información referente a las políticas de seguridad. Almacena la información en su base de datos. Muestra la información. Finaliza el caso de uso.

Sección 1: "Seleccionar políticas de seguridad".		
	Actor	**Sistema**
1		Muestra todas las políticas de seguridad.
2	Selecciona la opción Aceptar	Verifica todas las políticas de seguridad.
Flujo alterno		
Evento 2 Si selecciona una o más políticas de seguridad.		
	Actor	**Sistema**
1	Selecciona una o más políticas de seguridad.	Verifica las políticas de seguridad seleccionadas.
Relaciones	**CU incluido**	Cargar plugin (Ver Caso de uso Cargar plugin).
	CU extendido	No aplica
Prototipo de interfaz:		

Capítulo 2

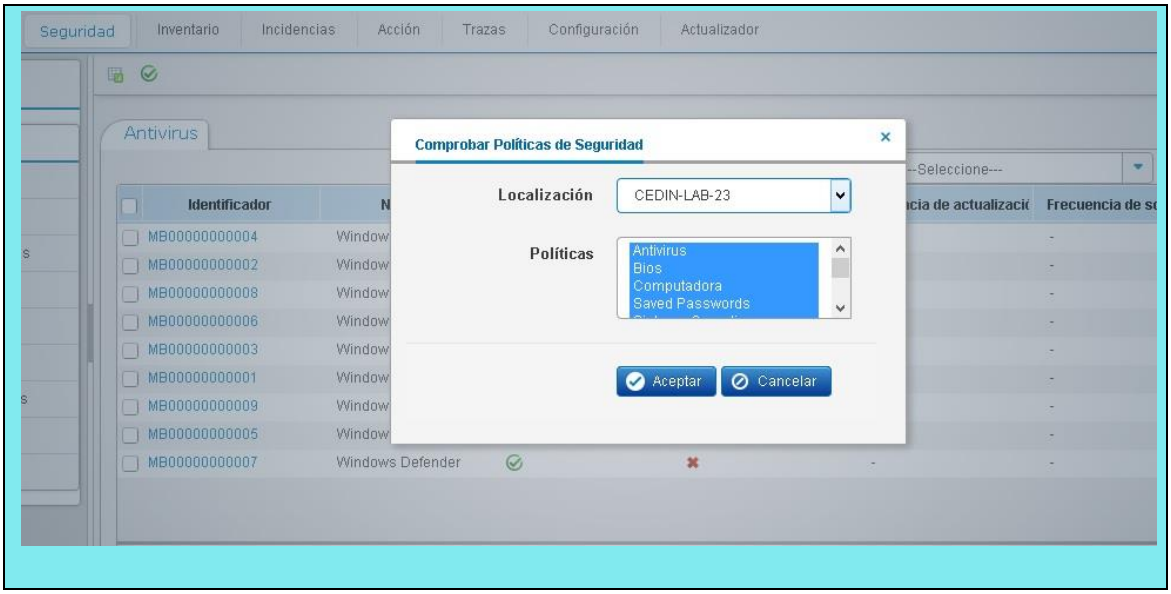


Tabla 2: Caso de uso: Cargar plugin

Objetivo	Cargar el plugin que obtiene la información referente a las políticas de seguridad de cada una de las estaciones de trabajo.
Actor	Usuario
Resumen	El caso de uso inicia cuando después del usuario seleccionar la localización, el sistema carga el plugin para obtener la información referente a las políticas de seguridad de cada estación de trabajo.
Complejidad	Alta
Prioridad	Alta
Precondiciones	Seleccionar la opción Comprobar Políticas de Seguridad.
Poscondiciones	Se obtiene la información de las políticas de seguridad
Flujo de eventos	

Flujo básico: “Seleccionar localización”		
	Actor	**Sistema**
1	Selecciona la localización. Selecciona la opción Aceptar.	Carga el plugin del lado del cliente. Recopila la información referente a las políticas de seguridad. Almacena la información en su base de datos. Muestra la información. Finaliza el caso de uso.
Flujos alternos		
Evento 1: No selecciona la opción Comprobar políticas de seguridad, accede directamente a seleccionar al menos una estación de trabajo.		
	Actor	**Sistema**
1	Selecciona la(s) estación(es) de trabajo que desea verificar en ese momento. Selecciona la opción Comprobar Políticas de Seguridad.	Muestra una nueva interfaz con las políticas de seguridad que se van a comprobar. (Ver Sección 1: Seleccionar políticas de seguridad.)
2	Selecciona la opción Aceptar.	Carga el plugin del lado del cliente. Recopila la información referente a las políticas de seguridad. Almacena la información en su base de datos. Muestra la información. Finaliza el caso de uso.
Sección 1: “Seleccionar políticas de seguridad”.		
1		Muestra todas las políticas de seguridad.

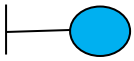
2	Selecciona la opción Aceptar.	Verifica todas las políticas de seguridad.
Flujo alterno		
Evento 2 Si selecciona una o más políticas de seguridad.		
	Actor	**Sistema**
1	Selecciona las políticas de seguridad que desea verificar.	Verifica las políticas de seguridad seleccionadas.
Relaciones	**CU incluido**	No
	CU extendido	No
Prototipo de interfaz: No aplica.		

2.6 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.

2.6.1 Diagrama de colaboración

El diagrama de colaboración es un tipo de diagrama de interacción cuyo objetivo es describir el comportamiento dinámico del sistema de información mostrando como interactúan los objetos entre sí, es decir, con qué otros objetos tiene vínculo o intercambia mensajes un determinado objeto (28).



Interfaz: Toda la funcionalidad especificada en las descripciones de los casos de uso que depende directamente de los aspectos externos del sistema se ubica en los objetos de interfaces. Se utilizan para modelar la interacción entre el sistema y sus actores.

Entidad: son usadas para modelar la información que tiene permanencia en el tiempo y es persistente. Además, modelan la información y el comportamiento asociado de algún concepto como una persona, evento u objeto del mundo real.

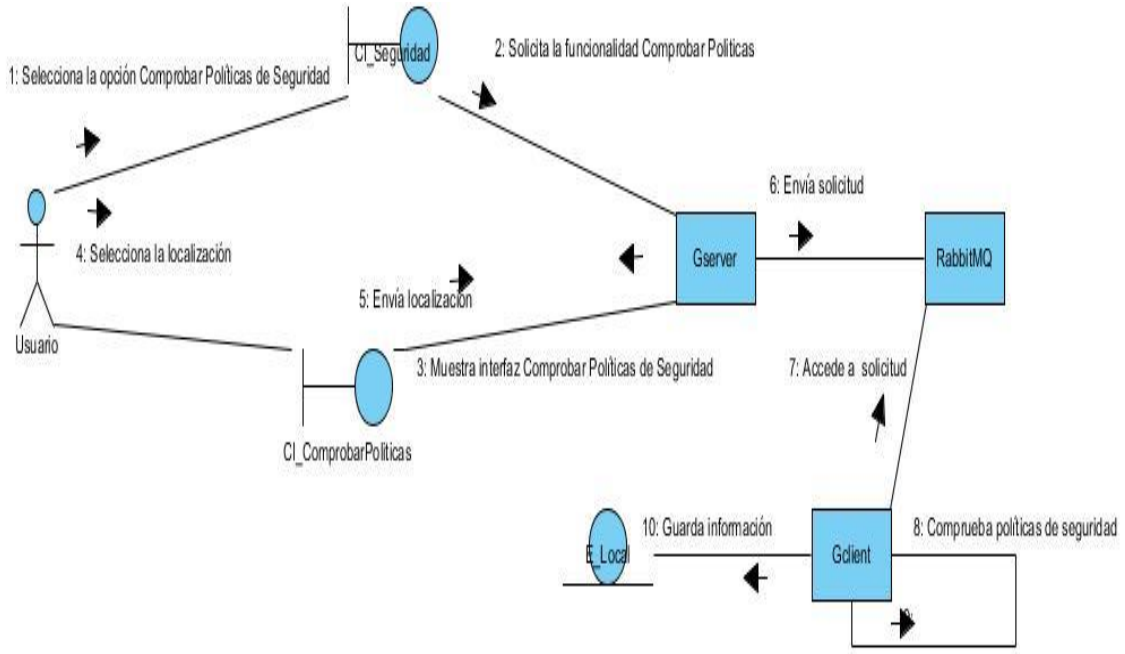


Figura 4: Diagrama de colaboración del Caso de uso Verificar políticas.

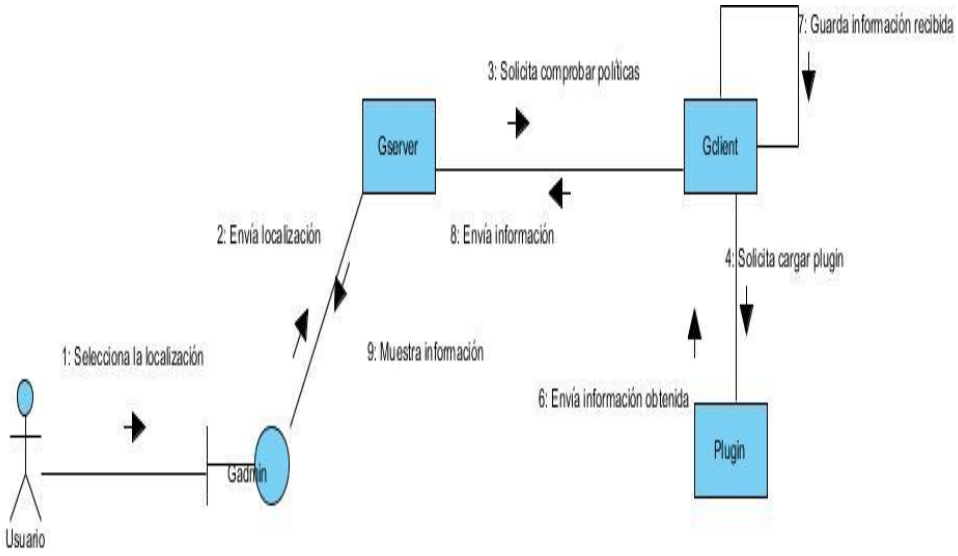


Figura 5: Diagrama de colaboración del Caso de uso Cargar plugin

2.6.2 Diagrama de clases con estereotipos web

Para el desarrollo de la vista lógica del sistema, a partir del modelado del análisis se presentan los diagramas de clases de diseño con estereotipos web. La extensión UML

para web, establece como elementos significativos, tres clases estereotipadas de la siguiente manera (27):

Página servidora (Server page): encargada de la representación de la página que contiene el código que es ejecutado en el servidor (27).

Página cliente (Client Page): mezcla datos, presentación y lógica, interpretados por el navegador (27).

Formulario (Form): elementos de entrada que son parte de una página cliente. Sus atributos son los elementos de entrada del formulario (27).

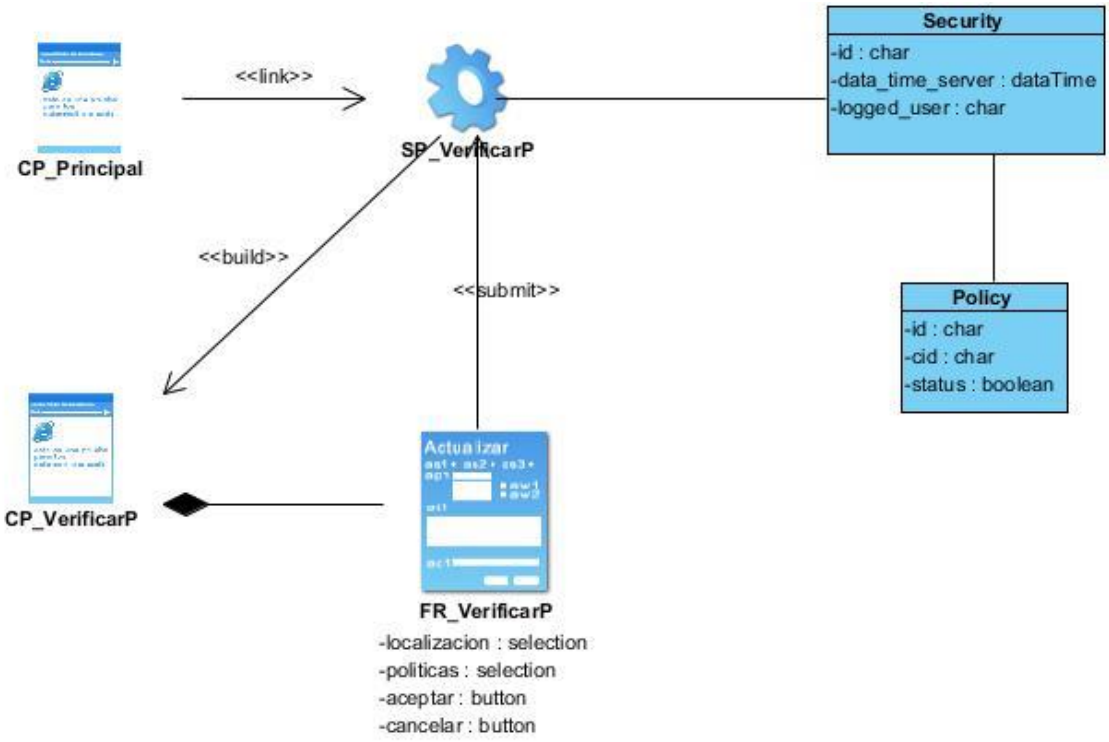


Figura 6: Diagrama de clases de diseño con estereotipos web del caso de uso Verificar políticas.

2.7.2 Arquitectura cliente-servidor.

La arquitectura de software es la organización fundamental de un sistema referenciado a sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución (29).

La arquitectura cliente-servidor es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor, al proceso que

responde a las solicitudes. Los principales componentes del esquema cliente/servidor son los clientes, los servidores y la infraestructura de comunicaciones. El cliente envía mensajes directamente al servidor de aplicación el cual debe administrar y responder todas las solicitudes. El servidor, en dependencia del tipo de solicitud, accede y se conecta con la base de datos (30).

En la solución se evidencia esta arquitectura con Gserver y Gclient. Gserver inicia el diálogo solicitando la comprobación de las políticas de seguridad a Gclient. Gclient realiza la comprobación de estas políticas a la estación de trabajo en que se encuentra instalado y envía la información a Gserver. Este último muestra la información a través de la interfaz Gadmin. La figura 7 muestra el funcionamiento de esta arquitectura en la solución.

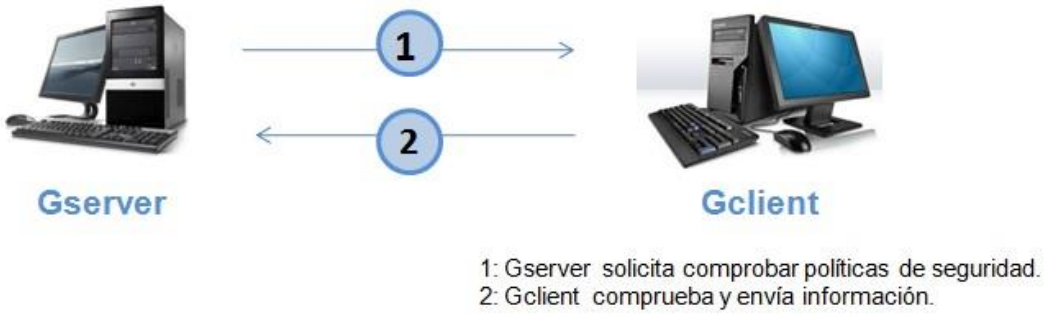


Figura 7: Funcionamiento de la arquitectura cliente servidor

2.7.3 Patrón arquitectónico

Los patrones arquitectónicos son patrones de software que ofrecen soluciones a problemas de arquitectura de software, adaptabilidad a requerimientos cambiantes, rendimiento, modularidad y acoplamiento. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes de un sistema. La solución que plantean es la creación de patrones de llamadas entre objetos y el empaquetado de funcionalidades (31).

El Patrón Arquitectónico que se utiliza en la realización de las clases del diseño para el sistema es el Model-Template-View (Modelo Plantilla Vista). Se escoge este patrón ya que Django es uno de los frameworks usados en el desarrollo de la solución y este es un framework MTV. En comparación con el patrón MVC, el Modelo en Django sigue siendo Modelo; a la Vista en Django se le denomina Plantilla y en el caso del Controlador se le denomina Vista (32).

Modelo: la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.

Plantilla (Template): la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento.

Vista: la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada, es como un puente entre el modelo y las plantillas.

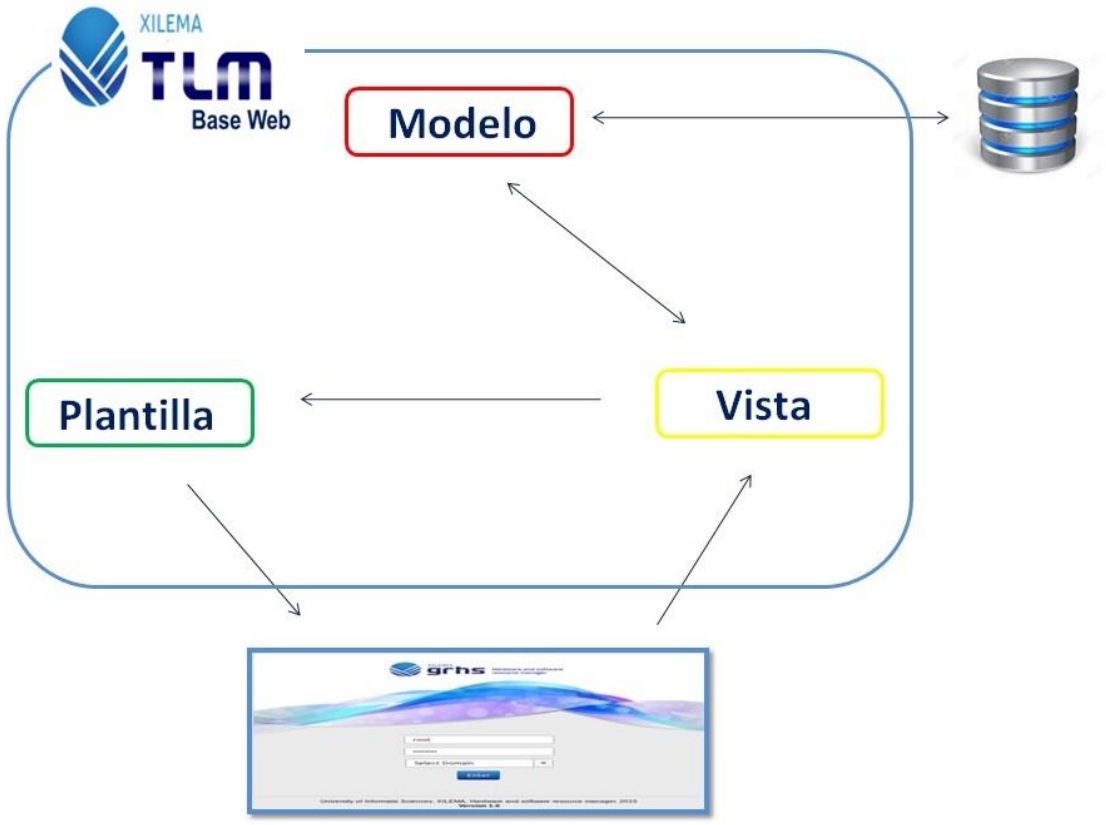


Figura 8: Patrón Modelo-Plantilla-Vista

Este patrón comienza su funcionamiento cuando el navegador envía una solicitud a la vista. La vista interactúa con el modelo para obtener los datos. Este interactúa con la base de datos para obtener los datos solicitados. La vista llama a la plantilla y esta envía la respuesta de la solicitud al navegador.

2.7. Modelo de datos

El modelo de datos es una colección de herramientas conceptuales que se emplean para especificar datos, las relaciones entre ellos, su semántica asociada y las

restricciones de integridad. Entre sus clasificaciones está el modelo de datos lógico que se usa para describir datos a nivel conceptual y externo caracterizándose por tener una amplia capacidad expresiva, son muy flexibles y permiten especificar ciertas restricciones de integridad (33). La Figura 9 muestra el modelo de datos que contiene las tablas de la base de datos del sistema, así como sus atributos y relaciones.

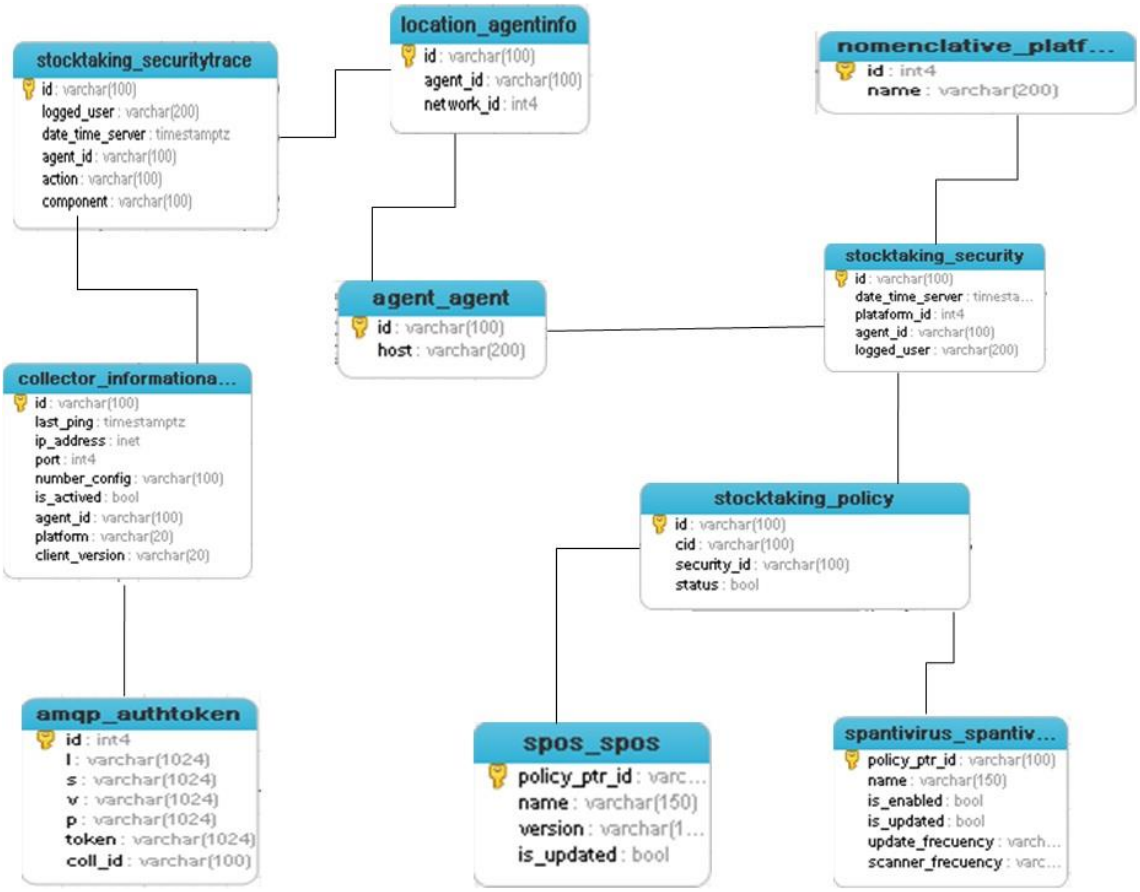


Figura 9: Modelo físico de la base de datos

stocktaking_security: tabla donde se almacenan los datos del servidor: fecha, hora, plataforma, agente y usuario logueado.

amqp_authtoken: tabla donde se almacena la información relacionada con los tokens del cliente.

nomenclative_plataform: tabla donde se almacenan las plataformas de los clientes.

agent: tabla donde se almacena la información de los agentes (host de las computadoras clientes).

stocktaking_securitytrace: tabla donde se almacena la información de las trazas.

collector_informationagent: tabla donde se almacena y reúne la información del cliente.

location_agentinfo: tabla donde se almacenan las ubicaciones de las computadoras clientes por locales físicos.

spantivirus: tabla donde se almacenan las propiedades de los clientes en cuanto a antivirus.

spos: tabla donde se almacenan las propiedades de los clientes en cuanto a sistema operativo.

2.9 Patrones de diseño

Los patrones de diseño describen las clases y objetos que se comunican entre sí de manera que puedan resolver un problema general de diseño en un contexto particular. (34).

2.9.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (26). GRASP son las siglas en inglés para denominar a los Patrones Generales de Asignación de Responsabilidades. Estos patrones de diseño son utilizados para la asignación de responsabilidades a una determinada clase. El grupo GRASP está conformado por 5 patrones principales y 4 adicionales aplicables al diseño orientado a objetos. A continuación, se refleja la utilización de dos de los patrones principales en la solución:

Experto: Consiste en asignar la responsabilidad al experto en información, es decir a la clase que cuenta con la información necesaria para cumplir la responsabilidad. De esta forma se conserva el encapsulamiento de la información, puesto que los objetos ejecutan las tareas que le corresponden de acuerdo a la información que poseen, lo que da lugar a sistemas más robustos y fáciles de mantener (26). En la presente investigación siguiendo el patrón Experto se le asigna responsabilidad determinada a la clase AMQPManager. De esta forma se mantiene el encapsulamiento de la información, puesto que los objetos utilizan su propia información para llevar a cabo las tareas.

Clase	Responsabilidad
AMQP Manager	Se encarga de administrar el flujo de comunicación entre el cliente y el servidor.

Tabla 3: Uso del patrón Experto

Alta cohesión: En el diseño orientado a objetos, la cohesión es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento (clase o subsistema). Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que colaboran entre sí y con otros objetos para simplificar su trabajo. Una clase muy cohesiva puede destinarse a un propósito muy específico. Una clase tiene responsabilidades moderadas y colabora con las otras para llevar a cabo las tareas (29). En la presente investigación siguiendo el patrón descrito se le asigna la responsabilidad determinada a la clase Task, esta clase almacena el contenido de una tarea. Envía la tarea para que otras clases la ejecuten y obtiene el resultado.

2.10 Conclusiones Parciales del Capítulo

Al culminar el presente capítulo se puede concluir que el modelo de dominio da una visión clara de las relaciones entre los conceptos asociados al desarrollo de la solución. Como parte de las características del sistema se identificaron una serie de requisitos funcionales y no funcionales. Se destaca el uso de los patrones de diseño Alta cohesión y Experto, que garantizan la reutilización de las funcionalidades y el uso de buenas prácticas en la codificación.

Capítulo 3

Capítulo 3 Implementación y prueba

3.1 Introducción

A partir del modelado de la propuesta en el capítulo anterior se representa el diagrama de componente y despliegue de la solución desarrollada. Se verifica la calidad del resultado de la implementación, mediante las pruebas realizadas al producto obtenido.

3.2 Diagrama de componentes

El diagrama de componentes muestra las organizaciones y dependencias lógicas entre los componentes del software, sean estos de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo (31).

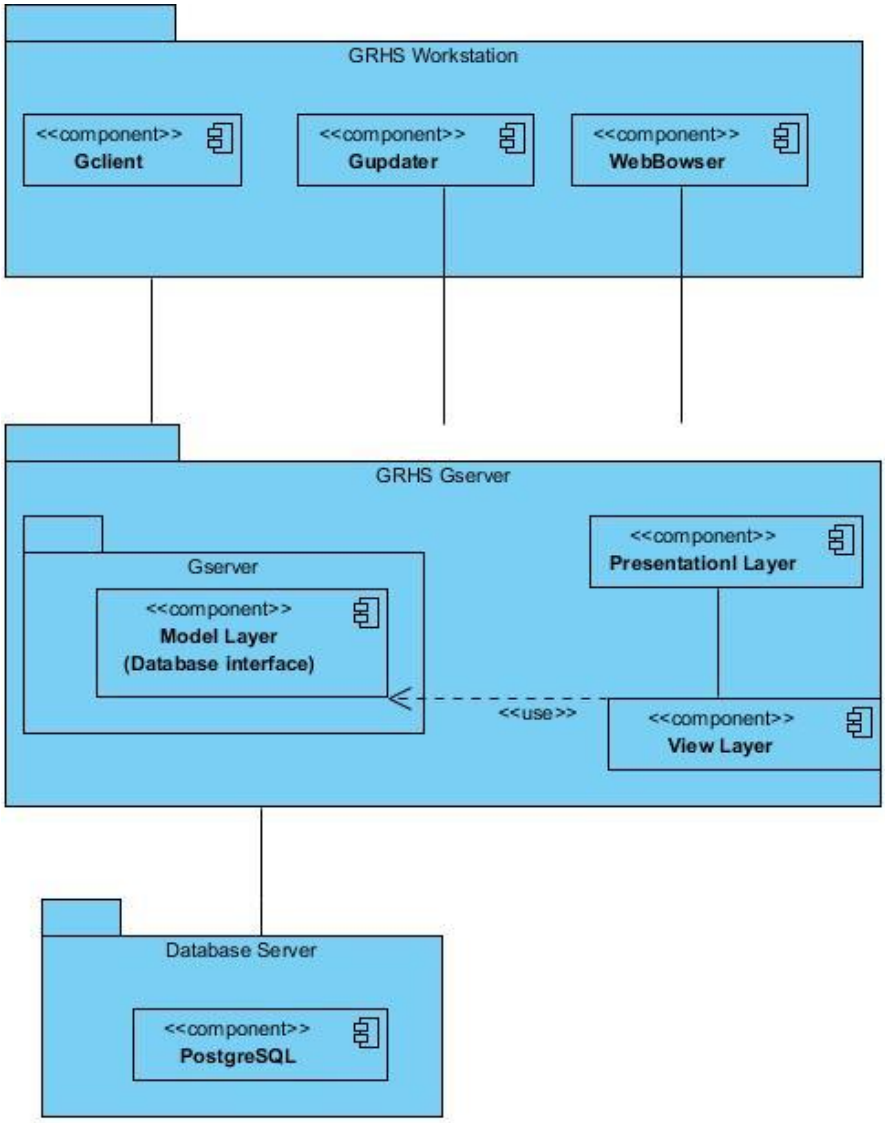


Figura 10: Diagrama de componentes

3.3 Modelo de despliegue

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Es un grafo de nodos unidos por conexiones de comunicación. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria (27).

Para la estructuración física del sistema se utilizó un nodo Gclient que representa el subsistema en los clientes que se encarga de capturar la información referente a las

políticas de seguridad. Se conecta mediante el protocolo AMQP al nodo Gserver, en el cual se encontrará el sistema del lado del servidor, encargado de recibir toda la información de los clientes y guardar en su base de datos. Esta base de datos está representada por el nodo Servidor de BD y se conecta al Gserver mediante el protocolo TCP/IP. En la figura 11 se muestra el diagrama de despliegue anteriormente descrito.

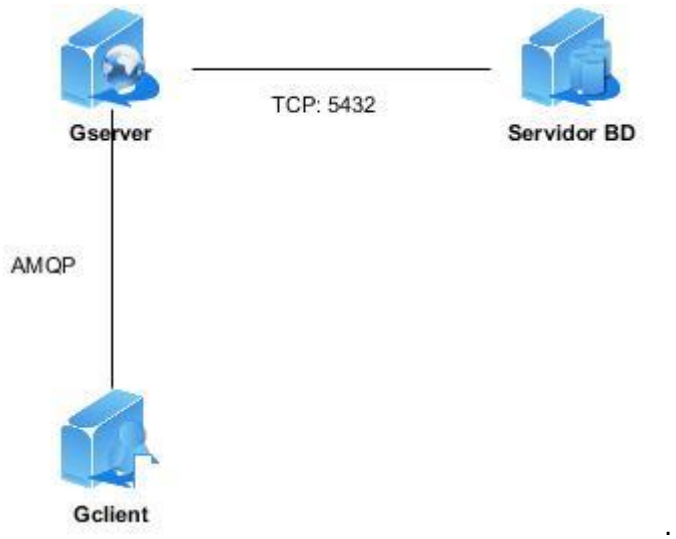


Figura 11: Diagrama de despliegue.

3.3Aplicación y resultados de las pruebas

3.3.1 Pruebas internas

Las pruebas internas verifican el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible, componentes de pruebas ejecutables para automatizar las pruebas.

Prueba de unidad o unitarias

Se concentran en probar cada componente individualmente para asegurar que funcione de manera apropiada como unidad. Emplean técnicas de prueba que recorren caminos específicos en la estructura de control de los componentes (pruebas estructurales) (31). Las pruebas unitarias fueron realizadas al culminar la implementación de una funcionalidad, para lo cual se utilizó la librería de Python (unittesting), lo cual permitió comprobar el correcto funcionamiento de la solución.

```
class GclientTestCase(unittest.TestCase):

    # Kaspersky not installed
    def testAntivirusPath(self):
        self.assertIsNone(get_path_to_kaspersky())

if __name__ == "__main__":
    unittest.main()
```

Figura 12: Muestra de los casos de pruebas unitarias aplicados.

```
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK

B:\Tesis\new\Thesis\code\grhs\gclient\plugins\security\plugins\windows\spantivirus>python test.py
.
----------------------------------------------------------------------
```

Figura 13: Resultado de los casos de prueba aplicados

3.3.2 Pruebas de aceptación

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Las pruebas de aceptación correspondiente a cada una de las funcionalidades de la solución serán representadas mediante tablas divididas por las siguientes secciones:

- **Clases válidas:** describe los pasos seguidos durante el desarrollo de la prueba, se tiene en cuenta las entradas válidas y opciones seleccionadas por el usuario con el objetivo de verificar si se obtiene el resultado esperado.
- **Clases inválidas:** describe los pasos seguidos durante el desarrollo de la prueba, se tiene en cuenta las entradas inválidas y opciones seleccionadas por el usuario con el objetivo de verificar si se obtiene el resultado esperado y cómo responde el sistema.
- **Resultado esperado:** breve descripción del resultado que se espera ya sea para clases válidas o inválidas.
- **Resultado de la prueba:** breve descripción del resultado que se obtiene.
- **Observaciones**: algún señalamiento o advertencia que sea necesario hacerle a la sección que se está probando.

Capítulo 3

Para llevar a cabo la detección de las no conformidades presentes en la aplicación desarrollada, se realizaron 3 iteraciones de pruebas. A continuación, se muestran tres casos de prueba correspondiente a tres funcionalidades del sistema.

Clases válidas	Clases inválidas	Resultado es perado	Resultado de la prueba	Observacione s
El usuario en la interfaz del módulo Seguridad selecciona la opción Comprobar Políticas de Seguridad.		El sistema muestre la interfaz Comprobar políticas de seguridadcon la localización y las políticas de seguridad que se desean comprobar.	Satisfactorio	
El usuario en la interfaz Comprobar políticas de seguridad selecciona la localización deseada y presiona el botón Aceptar.		El sistema verifica las políticas de seguridad para la localización seleccionada y muestra la información actualizada para esa localización.	Satisfactorio	
	El usuario en la interfaz Comprobar políticas de seguridad no selecciona ninguna localización.	El sistema en la interfaz Comprobar políticas de seguridad desabilita el botón aceptar.	Satisfactorio	
El usuario en la interfaz Comprobar políticas de seguridad selecciona las políticas de seguridad que desea verificar y selecciona la opción Aceptar.		El sistema verifica las políticas de seguridad seleccionadas y muestra la información actualizada de las políticas de seguridad seleccionadas.	Satisfactorio	
El usuario en la interfaz Comprobar políticas de		El sistema regresa a la interfaz del módulo	Satisfactorio	

seguridad selecciona la opción cancelar.		Seguridad.		

Tabla 4: Caso de Prueba Comprobar políticas de seguridad seleccionando la localización.

Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba	Observaciones
El usuario en la interfaz del módulo Seguridad selecciona una o varias estaciones de trabajo.		El sistema muestre lasestaciones de trabajo seleccionadas marcadas con un cuadradito gris.	Satisfactorio	
El usuario selecciona la opción Comprobar políticas de seguridad.		El sistema muestre la interfaz Comprobar políticas de seguridad con las políticas de seguridad que se desean comprobar.	Satisfactorio	
El usuario en la interfaz Comprobar políticas de seguridad selecciona las políticas de seguridad que desea verificar y selecciona la opción Aceptar.		El sistema verifica las políticas de seguridad seleccionadas y muestra la información actualizada de las políticas de seguridad seleccionadas.	Satisfactorio	
El usuario en la interfaz Comprobar políticas de seguridad selecciona la opción cancelar.		El sistema regresa a la interfaz del módulo Seguridad		

Tabla 5: Caso de prueba: Comprobar políticas de seguridad seleccionando estación de trabajo.

Clase válidas	Clases Inválidas	Resultado esperado	Resultado de la prueba	Observaciones
El usuario		El sistema	Satisfactorio	

selecciona la opción de Antivirus en el menú izquierdo del módulo seguridad.		muestra una tabla dividida por secciones (nombre, habilitado, actualizado, frecuencia de actualización, frecuencia de escaneo). En la columna del nombre se muestra el nombre del antivirus instalado.		

Tabla 6: Obtener antivirus instalado.

Al terminar las pruebas de aceptación se obtuvo un total de 4 resultados no satisfactorios en la primera iteración, todos fueron resueltos. En la segunda iteración se encontraron 2 resultados no satisfactorios que fueron resueltos y en la tercera iteración, no se encontró resultados no satisfactorios. En la figura 14 se muestran los resultados correspondientes a estas pruebas.

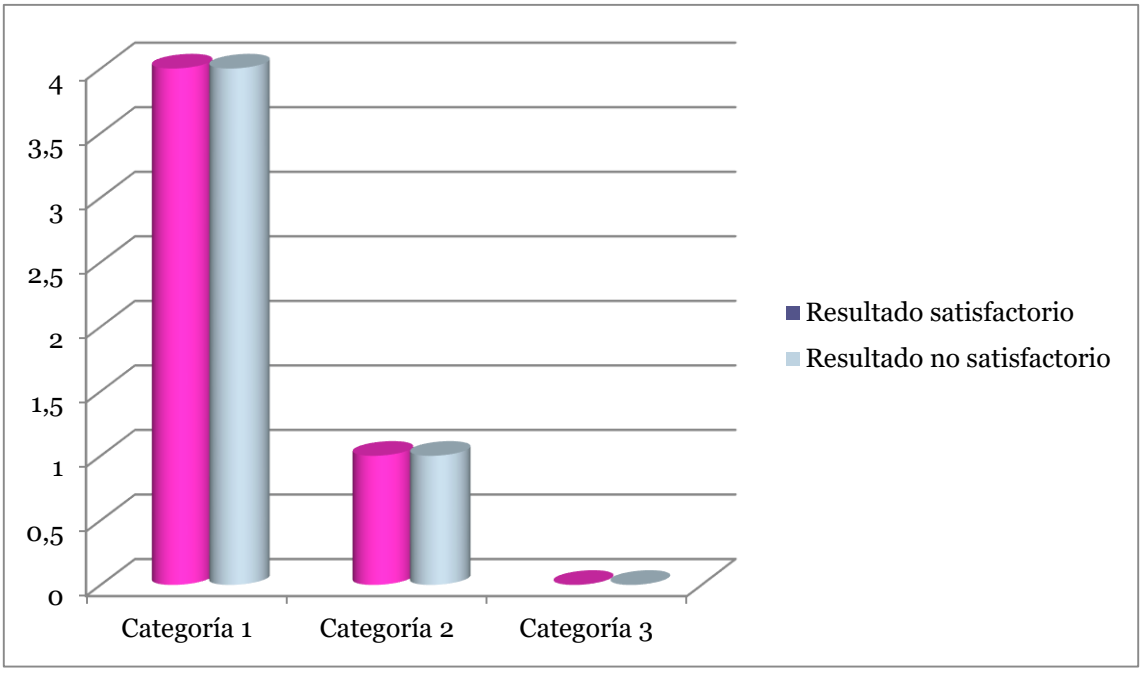


Figura14: Comportamiento de las clases satisfactorias y no satisfactorias

3.4 Conclusiones del capítulo

En este capítulo se obtuvieron los diagramas de componente y despliegue correspondientes a la aplicación. Se realizaron pruebas unitarias, permitiendo comprobar el correcto funcionamiento de procedimientos internos del software. Con

las pruebas de aceptación se validó que las funciones a nivel de interfaces de usuario fueran operativas.

Conclusiones Generales

Una vez finalizada la investigación para el desarrollo de la solución propuesta, se arriba a las siguientes conclusiones:

- La correcta asimilación de las herramientas, tecnologías y metodología definidas, permitió la obtención de la solución con las características esperadas según los requisitos definidos.
- La confección de los artefactos pertenecientes a las disciplinas de trabajo definidas según la metodología seleccionada, dieron como resultado la documentación necesaria para el desarrollo exitoso de la solución.
- Se realizaron pruebas internas y de aceptación para comprobar la correcta implementación de cada una de las funcionalidades y así evidenciar la calidad del sistema desarrollado.

Por todo lo anteriormente expuesto, se concluye que la puesta en práctica de las tareas propuestas contribuyó a establecer la comunicación desde el servidor hacia el cliente para iniciar la verificación de las políticas de seguridad informática en el sistema GRHS

Recomendaciones

Recomendaciones

Una vez concluida la investigación y desarrollada la propuesta de solución los autores recomiendan emplear el protocolo de comunicación asíncrono AMQP y el servidor de mensajería RabbitMQ en los restantes módulos del GRHS.

Referencias Bibliográficas

Referenciasbibliográficas

1. HÉRNANDEZ CAMEJO, Yoel y MARTÍNEZ SANCHEZ, Arianna. Instalación automática de Gclient para el sistema GRHS. Trabajo de diploma. La Habana: Universidad de las Ciencias Informáticas, 2015.

2. PÉREZ MONTESINOS, Dagoberto Félix y CARMENATE CANTERO, Dania. Comprobación de políticas de seguridad informática a través del sistema Gestor de Recursos de Hardware y Software (GRHS).La Habana: Universidad de las Ciencias Informáticas, 2015.

3. MARIO. 1. Concepto de sistema informático. In: PCPI 12/13 INFORMATICA MARIO [online]. [Accessed 10 noviembre 2016]. Available from: https://sites.google.com/site/pcpi1213informaticamario/home/modulos/2-mantenimiento/1-mantenimiento-de-sistemas-informáticos/1-concepto-de-sistema-informático.

4. ADRIAN HERRERO PEREZRUL, [sin fecha]. 2.1 Comunicación: comunicación con cliente – servidor, comunicación con llamada a procedimiento remoto, comunicación en grupo, tolerancia a fallos. mrTripas [en línea]. [Consulta: 24 noviembre 2016]. Disponible en: 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.

5. ROCA FUMERO, Luis Armando y HIRIBARNE GUEDES, Roberto. Plataforma de Integración de Aplicaciones y Servicios de redes. Trabajo de diploma. Santa Clara: Universidad Central Marta Abreu, 2012.

6. GARCÍA SÁNCHEZ, Henar y ANTÓN RODRÍGUEZ, Miriam.Framework de gestión de consumo de servicios. Trabajo fin de grado. Valladolid: Universidad de Valladolid, 2016

7. http://definicion.de/plugin/

8. POVEDANO MOLINA, Javier. Una aproximación publicación/subscripció centrada en datos para la provisión de servicios multimedia. Tesis Doctoral. Granada :Universidad de Granada, 2014.

9. About: Apache ActiveMQ. In: [online]. [Accessed 14 enero 2017]. Available from: http://es.dbpedia.org/page/Apache_ActiveMQ.

Referencias Bibliográficas

10. PIÑEIRO CRUZ, Jorge Bárbaro y PONCE PÉREZ, Javier Ricardo. Desarrollo de mecanismos de seguridad para el sistema Gestor de Recursos de Hardware y Software (GRHS). Trabajo de diploma. La Habana: Universidad de las Ciencias Informáticas, 2016.

11. PÉREZ ESTESO, Mario y DUEÑAS LÓPEZ, Juan Carlos. Análisis de arquitecturas deprocesado de Streaming BigData. Trabajo fin de Máster. Universidad Politécnica de Madrid, 2015

12. REYES GONZÁLEZ, Yunia, GONZÁLEZ BRAVO, Lisbet Maria y RUIZ CONSTANTEN, Yadira. ESTUDIO COMPARATIVO SOBRE LAS PRINCIPALES METODOLOGÍAS PESADAS Y ORIENTADAS A OBJETO EN EL DESARROLLO DE SOFTWARE. 2 diciembre 2008. S.l.: s.n.

13. BRITO ACUNA, Karenny. Metodologías tradicionales y metodologías ágiles. eumed.net [online]. 2009. [Accessed 7 December 2016]. Available from: http://www.eumed.net/librosgratis/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm

14. RODRÍGUEZ SÁNCHEZ, Tamara. Metodología de desarrollo para la Actividad productiva de la UCI. 6 marzo 2015. S.l.: s.n. ahora es 14

15. ALEGSA, Leandro. Definición de Python (lenguaje de programación). In: Alegsa.com.ar [online]. 7 julio 2014. [Accessed 7 diciembre 2016]. Available from: http://www.alegsa.com.ar/Dic/python.php.

16. Algunas Características de HTML5 que usted debe saber. [online]. 13 October 2013. [Accessed 18 January 2017]. Available from: http://latindesigner.net/algunas-caracteristicas-de-html5-que-usted-debe-saber/

17. KHALID. CSS3: Características, Funcionalidades y Recursos. In: Exp3rto [online]. 25 septiembre 2014. [Accessed 14 Canuary 2017]. Available from: http://www.exp3rto.com/introduccion-a-css3-nuevas-caracteristicas-capacidades-y-recursos/.

18. GONZÁLEZ, Enrique. ¿Qué es y para qué sirve JavaScript? Embeber JavaScript en HTML. Ejercicio ejemplo básico (CU00731B). In: Aprender a programar [online]. [Accessed 14 enero 2017]. Available from: http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=590:ique-es-y-para-que-sirve-javascript-embeber-javascript-en-html-ejercicio-

ejemplo-basico-cu00731b&catid=69:tutorial-basico-programador-web-html-desde-cero&Itemid=192.

19. GARCIA, Oscar. [Django] – Te presentamos el Framework web para perfeccionistas | El Club del Programador. In: [online]. 14 diciembre 2011. [Accessed 7 diciembre 2016]. Available from: http://www.elclubdelprogramador.com/2011/12/14/django-te-presentamos-el-framework-web-para-perfeccionistas/.

20. JQuery, qué es, orígen y características | El Viento 365. Elviento 365 [online]. 7 March 2016. [Accessed 18 January 2017]. Availablefrom: http://elviento365.com/ct/internet/3786/

21. BELIAL9826. 6 Caracteristicas Que Hacen Tan Popular A Bootstrap. JaGonzalez [online]. 22 July 2014. [Accessed 14 December 2016]. Available from: http://jagonzalez.org/6-caracteristicas-que-hacen-tan-popular-a-bootstrap/

22. RODRÍGUEZ, Txema. Backbone.js, el framework para construir aplicaciones usando Javascript siguiendo el patrón MVC alcanza la versión 1.0. In: GenbetaDev [online]. 23 marzo 2013. [Accessed 14 January 2017]. Available from: http://www.genbetadev.com/desarrollo-web/backbone-js-el-framework-para-construir-aplicaciones-usando-javascript-siguiendo-el-patron-mvc-alcanza-la-version-1-0.

23. Guión Visual Paradigmfor UML. S.l. 2013.

24. ESCUELA POLITECNICA NACIONAL. PostgreSQL: Características, limitaciones y ventajas. PostgreSQL [online]. 17 November 2012. [Accessed 7 December 2016]. Available from: http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html

25. ÁLVAREZ, Cecilio. 10 Packages de SublimeText para Desarrolladores Web. In: GenbetaDev [online]. 29 febrero 2016. [Accessed 14 January2017]. Available from: http://www.genbetadev.com/desarrollo-web/10-packages-de-sublimetext-para-desarrolladores-web.

26. CRAIG Larman. UML y Patrones. 2da Edición. S.l.: Pearson Educación, 2004. ISBN 84-205-3438-2.

27. DEL TORO FONSECA, Ana Fe, PEÑA DIEGUEZ, Yordan. Módulo de administración, configuración y seguridad del Sistema de Gestión para la

Ingeniería Clínica y Electromedicina v2.0. Trabajo de diploma. La Habana: Universidad de las Ciencias Informáticas. Junio de 2014.

28. CILLERO, Manuel. manuel.cillero.es Mi circunstancia digital [Accessed 11 April 2017]. Available from: https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-interaccion/diagrama-de-colaboracion/

29. CÁCERES TELLO, Jesús. Patrones de diseño: ejemplo de aplicación en los GenerativeLearningObject. Universidad de Alacalá. 2008

30. RODRÍGUEZ VILLAZÓN, Dania. Tendencias y Tecnologías Web actuales a considerar. TINO: Revista informático-técnologica de la familia. 18 Julio de 2014. ISSN 1995-9419

31. DÍAZ ROMEU, Ivaniet. SISTEMA INFORMÁTICO DE GESTIÓN DE INDICADORES PARA LA FORMACIÓN DE ESTUDIANTES POTENCIALMENTE TALENTOSOS EN LA UCI. Trabajo de diploma. La Habana: Universidad de las Ciencias Informáticas. Junio de 2015.

32. El patrón de diseño MTV .In: Libros Web. 2017. [Accessed 4 March 2017]. Available from: http://librosweb.es/libro/django1.0/capitulo5/el_patrondedisenomtv.html

33. CARRILLO, SILVIA ACID, et al. Introducción a las Bases de Datos: El modelo relacional. Editorial Paraninfo, 2005. ISBN 978-84-9732-396-3

34. COSMIN, PuiuIonut. Patrones de Diseño. MoleQla: revista de Ciencias de la Universidad Pablo de Olavide, 2016, no 23, p. 36.

Bibliografía

Bibliografía

1. HERNÁNDEZ LEÓN, Rolando Alfredo y COELLO GONZÁLEZ, Sayda. El proceso de investigación científica. S.l.: Editorial Universitaria del Ministerio de Educación Superior, 2011. ISBN 978-959-16-1307-3.
2. GUARDO GARCÍA, María Elena. Los componentes del Diseño Téorico de la Investigación Científica. Una reflexiónpraxiológica. In: . 2009, Vol. 14.
3. BRITO ACUÑA, Karenny. Metodologías tradicionales y metodologías ágiles. In: BIBLIOTECA VIRTUAL de Derecho, Economía y Ciencias Sociales [online]. [Accessed 7 noviembre 2016]. Available from: http://www.eumed.net/libros-gratis/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm.
4. REYES GONZÁLEZ, Yunia, GONZÁLEZ BRAVO, Lisbet Maria y RUIZ CONSTANTEN, Yadira. ESTUDIO COMPARATIVO SOBRE LAS PRINCIPALES METODOLOGÍAS PESADAS Y ORIENTADAS A OBJETO EN EL DESARROLLO DE SOFTWARE. 2 diciembre 2008. S.l.: s.n.
5. WIESEL JONATHAN. Como utilizar colas de mensajes con RabbitMQ - Parte I. *CODEHERO* [online]. 4 March 2014. [Accessed 21 January 2017]. Available from: http://codehero.co/como-utilizar-colas-de-mensajes-con-rabbitmq-parte/
6. Roger, Pressman. Ingeniería de software. Un enfoque práctico. Ingeniería de software. s.l. : 7ma, 2007.
7. CRAIG Larman. UML y Patrones. 2da Edición. S.l.: Pearson Educación, 2004. ISBN 84-205-3438-2.
8. RODRÍGUEZ SÁNCHEZ, Tamara. Metodología de desarrollo para la Actividad productiva de la UCI. 6 marzo 2015. S.l.: s.n.
9. DÍAZ ROMEU, Ivaniet. SISTEMA INFORMÁTICO DE GESTIÓN DE INDICADORES PARA LA FORMACIÓN DE ESTUDIANTES POTENCIALMENTE TALENTOSOS EN LA UCI. Trabajo de diploma. La Habana: Universidad de las Ciencias Informáticas. Junio de 2015.
10. RODRÍGUEZ VILLAZÓN, Dania. Tendencias y Tecnologías Web actuales a considerar. TINO: Revista informático-técnologica de la familia. 18 Julio de 2014. ISSN 1995-9419
11. DEL TORO FONSECA, Ana Fe, PEÑA DIEGUEZ, Yordan. Módulo de administración, configuración y seguridad del Sistema de Gestión para la

Bibliografía

Ingeniería Clínica y Electromedicina v2.0. Trabajo de diploma. La Habana: Universidad de las Ciencias Informáticas. Junio de 2014.

12. Publicación/Suscripción, 2014. [online]. [Accessed 9 June 2017]. Available from:https://www.ibm.com/support/knowledgecenter/es/SSMKHH_9.0.0/com.ibm.etools.mft.doc/aq01120_.htm?view=embed
13. HÉRNANDEZ CAMEJO, Yoel y MARTÍNEZ SANCHEZ, Arianna. Instalación automática de Gclient para el sistema GRHS. Trabajo de diploma. La Habana: Universidad de las Ciencias Informáticas, 2015.
14. ROCA FUMERO, Luis Armando y HIRIBARNE GUEDES, Roberto. Plataforma de Integración de Aplicaciones y Servicios de redes. Trabajo de diploma. Santa Clara: Universidad Central Marta Abreu, 2012.

Clase válidas	Clases Inválidas	Resultado esperado	Resultado de la prueba	Observaciones
El usuario selecciona la opción de Antivirus en el menú izquierdo del módulo seguridad.		El sistema muestra una tabla dividida por secciones (nombre, habilitado, actualizado, frecuencia de actualización, frecuencia de escaneo). En la columna de actualizado se muestra si el antivirus se encuentra actualizado o desactualizado.	Satisfactorio	

Tabla 7. Caso de prueba: Verificar actualización de antivirus en la máquina.