UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

Título: Comprobación de políticas de seguridad informática a través del sistema Gestor de Recursos de Hardware y Software (GRHS).

Autor(es):

Dania Carmenate Cantero

Dagoberto Félix Pérez Montesinos

Tutor(es):

Ing. Yor Alex Remond Recio

MSc. Ramón Alexander Anglada Martínez

"La Habana, Junio del 2015"

"El aspecto fundamental en el cual la juventud debe señalar el camino es precisamente en el aspecto de ser vanguardia en cada uno de los trabajos que le compete"

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores de este trabajo y autorizamos a la Facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____	_____
Dania Carmenate Cantero	Dagoberto Pérez Montesinos
Firma Autor	Firma Autor
_____	_____
Ing. Yor Alex Remond Recio	Msc. Ramón Alexander Anglada Martínez
Firma Tutor	Firma Tutor

DATOS DE CONTACTO

Autor:

Dania Carmenate Cantero

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: dcarmenate@estudiantes.uci.cu

Autor:

Dagoberto Félix Pérez Montesinos

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: dfmontesinos@estudiantes.uci.cu

Tutor:

Ing. Yor Alex Remond Recio

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: reymond@uci.cu

Tutor:

Msc. Ramón Alexander Anglada Martínez

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: raanglada@uci.cu

AGRADECIMIENTOS

Agradezco en primer lugar a mi familia. A mi mamá por su cariño y confianza, por ser mi guía; a mi papá por ser como es, por estar en cada momento que necesito su ayuda; a ellos le debo este sueño que hoy se ha hecho realidad.

A mi hermana que tanto quiero. A mi prima Diana, por su ayuda durante estos 5 años y a mi prima Karen, a ambas las adoro.

A mis abuelos Emilia y Luis, por su apoyo, sacrificio y amor incondicional.

A mis tíos Loly y Luisi, por quererme tal y como soy.

A mi novio y compañero de tesis Dagoberto, por su amor durante estos 7 años.

A mi otra familia, Noemí, Dagoberto, Mercy, a ellos, gracias por su cariño.

A mis tutores Yor Alex y Ramón Alexander, por su ayuda incondicional para la realización de este trabajo.

A mis amigas Lázara y Elaine, por estar a mi lado durante estos 5 años, por aguantarme, por su ayuda, consejos, regaños y ser mis fieles confidentes, las quiero.

A mis amigos Rayco y Juan Luis, por ser los mejores, por esas largas noches de explicaciones para que yo saliera bien en todo, por poder contar con ellos en cualquier momento.

A mis compañeros de grupo por estos bellos 5 años junto a ellos. A Julio, Mario, Elizabeth, Saylin, Kirenia, Osmanys, a todos mis amigos, por formar parte de este logro.

A los profesores que de una forma u otra se relacionaron conmigo y me ayudaron con sus conocimientos a lograr mis sueños. A toda la Universidad y Facultad 2 por forjarme y educarme como una verdadera profesional.

Dania

Agradezco en primer lugar a mi familia. A mis padres por ser la fuerza que me impulsó cada día con el "tú si puedes".

A mi hermana que aunque no está aquí hoy sirvió como mi guía profesional. A mis tías Mercy, Mirian y Lourdes que de cierto modo también contribuyeron a realizar este sueño, en especial mi tía Mercy.

A mis abuelas Idalia y Cira, que también estuvieron ahí en los momentos difíciles.

A mi querida novia Dania, por haber sido como mi madre en esta universidad, brindándome su amor y momentos de estudios que de verdad sin ellos no hubiera sido posible graduarme este día .

A mi otra familia, Estherlily, Ángel, Deborah , Diana , Ernesto, Emilia a todos muchas gracias.

A mis tutores Yor Alex y Ramón Alexander, por su ayuda incondicional para la realización de este trabajo.

A mi amigo Rayco que también estuvo en cada momento difícil de esta carrera, a mis demás amigos Mario, Elizabeth, Osmany, Gilberto, Michel, y a todos mis colegas del vicio.

Dagoberto

DEDICATORIA

Dedico este trabajo a la persona más especial de mi vida, mi mamá, que aunque hoy no se encuentre disfrutando este momento con nosotros, siempre ha estado pendiente de todo, sé que se siente muy orgullosa de mí. A mi papá, hermana, abuelos, tíos y primas. A mis tutores por su ayuda incondicional y a mis amistades.

Danía

Dedico este trabajo a mi familia, tutores y amigos.

Dagoberto

RESUMEN

Los controles de seguridad informática son un punto esencial para todas aquellas empresas donde se haga uso de las Tecnologías de la Información y la Comunicación. El control automático mediante un software, sin la intervención humana, garantiza ahorro en cuanto a tiempo, recurso y personal.

El presente trabajo de diploma tiene como objetivo desarrollar un módulo al sistema GRHS que permita comprobar las políticas de seguridad informática del centro Telemática de manera automatizada. Surge ante la necesidad de mantener el control de las políticas de seguridad informática del centro TLM.

A partir de la realización del módulo de seguridad se obtuvo como resultado la comprobación de las políticas de seguridad en la red de computadoras del centro TLM, verificando las políticas relacionadas a: antivirus, BIOS, sistema operativo, protección de datos, contraseñas guardadas en Firefox y Thunderbird, bloqueo automático de la pantalla, carpetas compartidas, dominio de la máquina, cortafuegos, grupos de administradores y cuentas de usuarios en las máquinas.

Palabras Clave:

Controles, GRHS, políticas de seguridad informática, red de computadoras

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La creciente evolución tecnológica ha hecho que las Tecnologías de la Información y la Comunicación (TIC) faciliten la vida cotidiana y profesional de las personas. Las TIC están presentes en gran parte de las actividades humanas: en el ocio, en la educación, en la comunicación, en la forma de relacionarnos con los demás y en el mundo de los negocios. Variadas empresas y organizaciones son completamente dependientes de estas tecnologías para llevar a cabo sus actividades (1).

Esta revolución ha sido propiciada por la aparición de la tecnología digital. La tecnología digital, unida al desarrollo de ordenadores cada vez más potentes, han permitido a la humanidad progresar rápidamente en la ciencia y la técnica desplegando las armas más poderosa: el conocimiento y la información (2).

La información en ocasiones es almacenada, procesada y transmitida en formato digital, encontrándose constantemente expuesta a múltiples amenazas: robos, duplicación, uso indebido, eliminación, entre otros; provocando la pérdida de la disponibilidad, confidencialidad o integridad de la información; lo cual, generalmente implica graves consecuencias para las empresas y en muchas ocasiones daños irreparables.

Muchos de los riesgos que afectan a la información se deben a la actividad humana, pero en realidad es a la ausencia o mal uso de un control de la seguridad informática donde se encuentran estas almacenadas. El control de la seguridad informática se puede realizar de forma manual o automatizada, siendo la forma automatizada la más recomendable para instituciones y empresas donde se haga uso de las TIC, debido a que es un proceso menos complejo y más efectivo en un entorno de constantes amenazas de seguridad, y teniendo en cuenta la gran cantidad de medidas a implementar.

La gestión automatizada de un control de seguridad informática implica que la operación, monitorización y revisión tanto del software como del hardware de una estación de trabajo se realice de forma automática, mediante sistemas informáticos; sin que se produzca la intervención humana en la realización de estas acciones (34).

En Cuba el control de la seguridad informática en todas las entidades se ha convertido en un papel importante, los avances alcanzados en los últimos años en la informatización de la sociedad

con el incremento de tecnologías de la información en todos los sectores y en particular de las redes informáticas y sus servicios asociados, y el impulso orientado por la dirección del país al desarrollo acelerado de programas que multipliquen dichos logros, han requerido la adopción de medidas que garanticen un adecuado nivel de seguridad para su protección y ordenamiento (3).

Una de las instituciones educacionales presentes en el país que impulsa la gestión automatizada de las políticas de seguridad informática es la Universidad de las Ciencias Informáticas (UCI). Como centro de excelencia en la formación de profesionales competentes en la rama de la Informática y en el desarrollo de software, cuenta con una estructura especializada para la actividad productiva, dentro de la cual se encuentra el Centro de Telemática (TLM), encargado del desarrollo de sistemas y servicios informáticos en las ramas de las Telecomunicaciones y la Seguridad Informática.

El centro TLM tiene definidas políticas de seguridad informática con el objetivo de garantizar la protección de la información contenida en sus máquinas y el conocimiento generado por sus especialistas. Dichas políticas deben estar establecidas en cada una de las estaciones de trabajo del centro y son verificadas frecuentemente por el asesor de tecnología o por un grupo de personas a los cuales se asigna esta tarea. La verificación del cumplimiento de estas políticas se realiza de manera manual, computadora por computadora, lo cual implica costo en cuanto al tiempo que se emplea en comprobar el cumplimiento de las políticas en todas las máquinas del centro. Al realizar de manera manual esta comprobación, aumenta la probabilidad de cometer errores y obviar la comprobación de una determinada política que pueda ser objeto luego de una brecha de seguridad.

El centro además cuenta con un sistema informático denominado Gestor de Recursos de Hardware y Software (GRHS), sistema encargado de obtener las propiedades de las piezas y programas instalados en una red de computadoras, con el fin de contabilizar los cambios realizados y tomar acciones automáticas en caso de cambios no autorizados (35). El sistema a pesar de obtener la información relacionada al software de una estación de trabajo no contribuye a verificar cada una de las políticas de seguridad informática establecidas, pues no brinda información detallada de las mismas.

Teniendo en cuenta lo expuesto anteriormente se plantea el siguiente **problema a resolver**: ¿Cómo contribuir a la comprobación de las políticas de seguridad informática, en las computadoras del centro TLM?

Se define como **objeto de estudio** el proceso de control de políticas de seguridad informática.

Para dar solución al problema a resolver se define como **objetivo general**: desarrollar un módulo al sistema GRHS que permita comprobar las políticas de seguridad informática del centro TLM de manera automatizada.

El **campo de acción** se centra en la red de computadoras del centro TLM donde se encuentra desplegado el sistema GRHS.

Para dar cumplimiento al objetivo propuesto se organizan las siguientes **tareas de investigación**:

1. Realización del estudio de sistemas informáticos basados en el control de políticas de seguridad informática para analizar sus características.
2. Análisis del sistema GRHS para identificar los mecanismos de obtención del software instalado en una red de computadoras.
3. Estudio de las políticas de seguridad informática del centro TLM para el correcto desarrollo de las nuevas funcionalidades.
4. Selección de la metodología y herramientas para el desarrollo del módulo al sistema GRHS que permita comprobar las políticas de seguridad informática del centro TLM de manera automatizada.
5. Implementación de un plugin en la aplicación gclient del sistema GRHS para la comprobación de las políticas de seguridad informática en cada una de las estaciones de trabajo del centro TLM.
6. Diseño e implementación de un módulo para GRHS que permita desde gadmin controlar las políticas de seguridad informática en las estaciones de trabajo del centro TLM.
7. Realización de pruebas a cada una de las funcionalidades del sistema para detectar posibles errores.

Para el desarrollo de esta investigación y el logro de su objetivo, se utilizarán los siguientes **métodos de investigación científica**:

- **Métodos Teóricos**

Analítico sintético: Este método permitirá analizar las teorías y los documentos referentes al objetivo de la investigación, facilitando de esta forma la extracción de los elementos más importantes relacionados con el objeto de estudio.

Modelación: Se empleará para la representación de las funcionalidades a desarrollar del módulo de seguridad.

- **Métodos Empíricos:**

Entrevista: Este método se utilizará para el desarrollo del sistema, detectando las necesidades del cliente a través del intercambio con él. Se realizan una serie de entrevistas con el cliente para identificar los principales problemas existentes y así poder satisfacer sus necesidades y obtener un software con la calidad requerida.

El trabajo de diploma se divide en 3 capítulos, los cuales estarán estructurados de la siguiente forma:

Capítulo1. **“Fundamentación teórica”**: se describen los principales conceptos a tratar: seguridad informática, sistema informático, políticas de seguridad informática y plugin. Se realizará la investigación y el estudio del estado del arte, analizando los sistemas similares que permitan llevar a cabo el proceso automatizado de la verificación de las políticas de seguridad informática. Se definirá la metodología de desarrollo, herramientas y tecnologías a utilizar para el desarrollo del módulo.

Capítulo 2. **“Propuesta de solución”**: se describe la propuesta de solución, se realiza una descripción de las historias de usuario que se proponen para dar solución al objetivo general derivado de la situación problemática. Se realiza la estimación de esfuerzo por cada historia de usuario y el plan de duración de las iteraciones a llevar a cabo.

Capítulo 3. **“Implementación y prueba”**: se especificará la arquitectura del sistema y con ello los patrones arquitectónicos empleados. Se adapta el diseño al entorno de implementación y se describen las tareas de ingeniería. Se explicarán las pruebas realizadas al software para comprobar que el producto de software funciona según lo diseñado y que las funcionalidades se han implementado de forma adecuada.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se explican los conceptos relacionados con el desarrollo del trabajo de diploma. Se realiza un estudio acerca de los estándares y regulaciones sobre seguridad informática y los sistemas que permiten la automatización de controles de políticas de seguridad informática. Se especifica la metodología de desarrollo escogida, así como las tecnologías y lenguajes utilizados para la implementación del sistema.

1.2 Conceptos

1.2.1 Seguridad Informática

La seguridad informática es el conjunto de métodos y herramientas destinados a proteger los bienes o activos informáticos de una institución.
La información es el activo más preciado y la seguridad en la información tiene el objetivo de garantizar:

- Confidencialidad: la información o los activos informáticos son accedidos solo por las personas autorizadas para hacerlo.
- Integridad: los activos o la información solo pueden ser modificados por las personas autorizadas y de la forma autorizada.
- Disponibilidad: los activos informáticos son accedidos por las personas autorizadas en el momento requerido (4).

1.2.2 Sistema Informático

Un sistema informático es el conjunto de partes interrelacionadas, hardware, software y de recurso humano que permite almacenar y procesar información. Un sistema informático puede formar parte de un sistema de información; en este último la información, el uso y acceso a la misma, no necesariamente se encuentre informatizada (5).

1.2.3 Políticas de Seguridad Informática

Las políticas de seguridad informática tienen por objeto establecer las medidas de índole técnica y de organización, necesarias para garantizar la seguridad de las tecnologías de información

(equipos de cómputo, sistemas de información, redes (voz y datos)) y personas que interactúan haciendo uso de los servicios asociados a ellos y se aplican a todos los usuarios de cómputo de las empresas (6).

1.2.4 Plugin

Pequeño programa que añade alguna función a otro programa, habitualmente de mayor tamaño. Son muy utilizados en los navegadores web como Mozilla Firefox para ampliar sus funcionalidades (7).

1.3 Principales estándares y regulaciones sobre seguridad informática

En el contexto de la seguridad informática existen estándares que constituyen normas certificables, marcos de trabajo que representan una recopilación de mejores prácticas y regulaciones que son de obligatorio cumplimiento en determinadas naciones.

- **ISO/IEC 27001**

La Organización Internacional para la Estandarización (ISO) y la Comisión Electrotécnica Internacional (IEC) han elaborado un grupo de estándares que constituyen una referencia a nivel mundial en la temática de seguridad de la información. La serie ISO/IEC 27000 está destinada completamente a la seguridad informática, donde los estándares ISO/IEC 27001 y 27002 abordan el tema de manera general, y el resto tratan temas específicos que complementan a los anteriores.

El proceso de gestión de la seguridad informática se encuentra descrito en el estándar ISO/IEC 27001, el cual constituye una norma certificable a nivel internacional. Esta norma ofrece un modelo para el establecimiento, implementación, operación, monitorización, revisión, mantenimiento y mejora de un sistema de gestión de la seguridad de la información (SGSI). La norma certificable ISO/IEC 27001, en su Anexo A plantea 133 controles de seguridad divididos en 11 dominios y 39 objetivos de control. Estos controles son explicados detalladamente en la guía de buenas prácticas ISO/IEC 27002, donde pueden apreciarse un grupo de controles técnicos, otros más relacionados con los recursos humanos y un grupo de controles de tipo organizativo (8).

- **NIST SP 800-53**

La publicación especial (SP) 800-53 del Instituto Nacional de Estándares y Tecnologías de EEUU (NIST) forma parte de la serie 800 dedicada específicamente a temas relacionados con la seguridad de la información en agencias federales de dicho país. En particular la NIST SP 800-53 ofrece un catálogo de controles de seguridad y privacidad de los sistemas de información federales, y un proceso para la selección de estos para proteger las operaciones de la organización (misión, funciones, imagen y reputación), activos, individuos, otras organizaciones y la nación de un conjunto diverso de amenazas, incluyendo ataques hostiles cibernéticos, desastres naturales, fallas estructurales, y los errores humanos (tanto intencionales como no intencionales).

A pesar de que el estándar NIST SP 800-53 constituye una regulación solo en EEUU, esta publicación es muy utilizada y referenciada a nivel internacional por el prestigio que normalmente poseen los estándares del NIST. El estándar contiene una recomendación de controles de seguridad informática que pueden servir en general para diferentes organizaciones y toda la documentación pueden ser obtenida sin costo alguno (8).

- **Resolución 127/2007 del MIC**

El Reglamento de Seguridad para las Tecnologías de la Información del Ministerio de la Informática y las Comunicaciones de Cuba, recogido en la resolución 127/2007 de dicho ministerio constituye el documento rector en materia de seguridad informática del país. Tiene por objeto establecer los requerimientos que rigen la seguridad de las tecnologías de la información y garantizar un respaldo legal que responda a las condiciones y necesidades del proceso de informatización de la nación (8).

- **ISF-SoGP. Recomendación de buenas prácticas de seguridad.**

Este estándar aborda el tema de la seguridad de la información desde la perspectiva del negocio, enfocándose en la implementación de los controles necesarios para mitigar los riesgos asociados con los sistemas de información críticos.

El SoGP posee gran aceptación entre los miembros del Fórum para la Seguridad de la Información (ISF por sus siglas en inglés), pero a nivel internacional su aplicación es limitada, debido a que predominan otros estándares con mayor aceptación como ISO/IEC 27001 (8).

- **ISM3. Modelo de madurez para la gestión de la seguridad informática.**

Este modelo pretende extender los principios de calidad establecidos en ISO 9001 a un Sistema de Gestión de la Seguridad de la Información. En lugar de estar orientado a controles, se enfoca en los procesos de seguridad informática que pueden ser comunes a todas las organizaciones.

En ISM3 los procesos relacionados con la seguridad informática son descritos detalladamente, estableciendo objetivos y métricas que permitan establecer un sistema de calidad. El enfoque práctico y de medición, así como la orientación hacia los objetivos de negocio de la organización, es lo que diferencia este modelo del resto de los estándares relacionados con la seguridad de la información (8).

A pesar de que ISM3 plantea un enfoque novedoso, su publicación es relativamente reciente y tiene competidores muy fuertes como ISO/IEC 27001, COBIT e ITIL, por lo que en general todavía no posee un amplio reconocimiento a nivel internacional.

- **COBIT, ITIL**

Los Objetivos de Control para las Tecnologías de la Información (COBIT por sus siglas en inglés), constituyen un conjunto de mejores prácticas desarrollados por la Asociación para la Auditoría y Control de Sistemas (ISACA). COBIT plantea un marco de trabajo para el gobierno de las Tecnologías de Información (TI) que permite a los ejecutivos alinear las tecnologías de la información con los objetivos del negocio.

Por otra parte, la Biblioteca de Infraestructura de Tecnologías de Información (ITIL por sus siglas en inglés) es un conjunto de buenas prácticas para la gestión de servicios de TI, desarrollado por el gobierno británico. ITIL ofrece descripciones detalladas de un extenso conjunto de procedimientos de gestión ideados para ayudar a las organizaciones a lograr calidad y eficiencia en las operaciones de TI, contribuyendo de esta manera a una mayor efectividad del negocio (8).

A pesar de que ITIL y COBIT poseen un amplio reconocimiento, no son dedicados por entero a la seguridad de la información, sino que este tema lo abordan por parte de la gestión de TI.

1.4 Trabajos sobre la automatización en seguridad informática.

En el campo de la gestión de la seguridad informática pocas investigaciones y estudios abordan la temática de la automatización con una visión integral, analizando el amplio espectro de controles

de seguridad recomendados por estándares internacionales. El trabajo que resulta un poco más abarcador en este sentido es SCAP (Protocolo de Automatización de Contenido de Seguridad, por sus siglas en inglés), desarrollado por el Instituto Nacional de Estándares y Tecnologías de Estados Unidos (NIST).

- **SCAP. Protocolo de Automatización de Contenido de Seguridad.**

SCAP fue creado para proveer una forma estandarizada de gestionar la seguridad de los sistemas informáticos. Permite automatizar, en cierto grado, el chequeo y gestión de vulnerabilidades, evaluar posibles impactos de vulnerabilidades, mediciones de seguridad, y evaluación de políticas a adoptar. En definitiva, provee una infraestructura poderosa para la elaboración de análisis e informes sobre vulnerabilidades en los sistemas de información.

Los componentes definidos por SCAP consisten en:

- Common Vulnerabilities and Exposures (CVE): Nomenclatura y diccionario de vulnerabilidades de seguridad de los sistemas.
- Common Configuration Enumeration (CCE): Nomenclatura y diccionario de configuraciones de seguridad de los sistemas.
- Common Platform Enumeration (CPE): Nomenclatura y diccionario para hardware, sistemas operativos y aplicaciones.
- Common Vulnerability Scoring System (CVSS): Especificación para aplicar criterios de medida a la severidad relativa de las vulnerabilidades de los sistemas.
- Extensible Configuration Checklist Description Format (XCCDF): Lenguaje para la especificación de listas de chequeo de configuraciones de seguridad y para el reporte de los resultados obtenidos en las verificaciones.
- Open Vulnerability and Assessment Language (OVAL): Lenguaje para la especificación de procedimientos de pruebas de seguridad de bajo nivel empleados por las listas de chequeo.
- Open Checklist Interactive Language (OCIL) Versión 2.0: Lenguaje para representar chequeos que recolectan información de personas y almacenes de datos
- Asset Identification (AI): Formato para identificar unívocamente a los activos mediante identificadores conocidos.

- Asset Reporting Format (ARF): Formato para el intercambio de información y generación de reportes sobre los activos.
- Common Configuration Scoring System (CCSS): Especificación para aplicar criterios de medida relativos a la configuración de los sistemas.
- Trust Model for Security Automation Data (TMSAD): Modelo común de confianza que puede ser aplicado a las especificaciones de un dominio de automatización de seguridad (8).

1.5 Sistemas que permiten la automatización de controles de políticas de seguridad informática.

Desde el punto de vista práctico existen varios sistemas, aplicaciones y herramientas que permiten la automatización de un grupo determinado de controles de seguridad informática. A continuación se describen algunas características de estos sistemas.

- **OCS Inventory NG**

Es una herramienta que permite realizar de forma automatizada el inventario de todos los equipos conectados a la red de datos. Registra tanto las características de hardware como de software de las máquinas (9).

- **OpenVAS**

Es un sistema de detección de vulnerabilidades, permite escanear de forma programada todos los dispositivos conectados a la red para detectar la presencia de vulnerabilidades en los sistemas operativos y las aplicaciones (10).

- **Rsyslog**

Es un sistema de gestión de trazas, permiten la recolección y almacenamiento de trazas de seguridad de forma centralizada. Rsyslog es un eficiente y rápido sistema de procesamiento de registros de sistema. Ofrece un diseño modular de alto desempeño y niveles de seguridad apropiados (11).

- **Snort**

Es un sistema de detección de intrusiones, que permite detectar en tiempo real los ataques que se producen a través de la red, además de alertar y responder ante cualquier anomalía (12).

- **Script au**

En la Universidad de las Ciencias Informáticas es empleado para el control de las políticas de seguridad el script llamado au. Este script es un programa basado en un lenguaje de programación interpretado, que ejecuta las instrucciones para la comprobación de las políticas de seguridad informática en una estación de trabajo. Registra en un archivo la información referente a la máquina auditada en cuanto a: auditor, área, local, usuario o propietario de la estación de trabajo, uso de TrueCrypt, comprobación del dominio, empleo del firewall y activación del antivirus.

- **GRHS**

En la facultad 2 se cuenta con el sistema GRHS, como el gestor de recursos de hardware y software, este posee tres aplicaciones: gserver, gadmin y gclient. La aplicación gclient es la encargada de realizar los inventarios, detectar incidencias y ejecutar acciones ante estas, realiza un inventario de los componentes internos de la computadora y de los dispositivos conectados a la misma en los sistemas operativos GNU/Linux y Windows. La aplicación gserver es el centralizador de inventarios que recibe la información enviada por los clientes, envía órdenes a las computadoras inventariadas y notifica las alertas a los interesados cuando ocurren incidencias. La aplicación gadmin, mediante este módulo web se describe lo que constituye una incidencia, los niveles de incidencia, así como cada qué tiempo debe realizarse un inventario de hardware o de software, también se gestionan los usuarios, los períodos de cambios y los reportes (35).

1.6 Conclusiones de los principales estándares y regulaciones sobre seguridad informática, trabajos y sistemas que automatizan controles de políticas de seguridad informática.

La revisión de los estándares y regulaciones contribuyó a un mejor entendimiento de las políticas de seguridad del centro TLM y aportó de manera teórica a la implementación de las políticas ya definidas en dicho centro.

El estudio del trabajo sobre la automatización de políticas de seguridad: SCAP, permitió determinar que su alcance aún es limitado, solamente posee controles relacionados con la detección de vulnerabilidades (CVE, CVSS), el chequeo de las configuraciones de seguridad y del cumplimiento de regulaciones (XCCDF, OVAL, CCE, CCSS), así como el inventario de activos (CPE, AI, ARF).

Como puede apreciarse existen varios sistemas que automatizan controles de seguridad informática (OCS Inventory NG, OpenVAS, Rsyslog, Snort, GRHS). Sin embargo, la mayoría de ellos poseen deficiencias y no abarcan completamente el objetivo de la investigación.

El script au aunque registra gran parte de la información recogida en las políticas de seguridad, continúa siendo un trabajo engorroso para los asesores de tecnología, debido a que el script debe ejecutarse de forma manual en cada una de las estaciones de trabajo, el asesor debe brindar cierta información sobre la máquina y luego comprobar cada una de las políticas de seguridad en el fichero donde au guarda la auditoría.

Luego de realizado el análisis de las herramientas el que más facilidades brinda para dar solución al problema antes planteado es el sistema GRHS, debido a que el mismo es un sistema multiplataforma, permite la integración de plugin y actualmente dicho sistema obtiene propiedades que se encuentran en las políticas de seguridad por lo que se podrán reutilizar esas funcionalidades para el desarrollo del módulo de seguridad.

1.7 Metodología de desarrollo

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software (13). Las metodologías se pueden clasificar en ágiles y tradicionales.

Las metodologías tradicionales centradas específicamente en el control del proceso, han demostrado ser efectivas y necesarias en un gran número de proyectos, sobre todo aquellos proyectos de gran tamaño (respecto a tiempo y recursos) (14). Dentro de las metodologías tradicionales podemos encontrar a OPEN, METRICA y Proceso Unificado Relacional (RUP según sus siglas en inglés)

Para las metodologías ágiles los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados en el desarrollo del software. Es más relevante crear un software que funcione, que escribir documentación exhaustiva. La colaboración con el cliente debe prevalecer sobre la negociación de contratos. La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan. Algunos ejemplos de metodologías de este tipo son: SCRUM y Programación Extrema (XP según sus siglas en inglés).

1.7.1 Metodología de desarrollo seleccionada. XP

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes y simplicidad en las soluciones implementadas. XP se define como adecuada para proyectos con requisitos imprecisos y muy cambiantes y donde existe un alto riesgo técnico (14).

Se selecciona la metodología XP para el desarrollo del presente trabajo debido a que el mismo se trata de un proyecto pequeño y de corto tiempo, donde no se necesita la generación de tantos artefactos y los requisitos pueden cambiar con el tiempo a medida que avanza el trabajo; el equipo de desarrollo está compuesto por dos programadores, en el cual el grado de interacción entre los miembros es alto pues existe una buena comunicación y entendimiento entre ambos; además el cliente pertenece al equipo de desarrollo, lo que posibilita un mayor intercambio de opiniones logrando un producto que satisfaga las necesidades del cliente en el menor tiempo posible y con la calidad requerida.

1.8 Herramientas y Tecnologías

Los lenguajes y las herramientas empleadas en el desarrollo del módulo fueron definidos por el centro TLM. Constituyendo Python el lenguaje principal del centro. A continuación se describen las herramientas y lenguajes utilizados.

1.8.1 Lenguaje de programación

Un lenguaje de programación permite crear un grupo de instrucciones que luego se convertirán en un programa. Teniendo en cuenta que el centro TLM define las tecnologías con que se desarrollarán sus productos, para el desarrollo del módulo será utilizado Python 2.7 como lenguaje del lado del servidor y HTML5, CSS3 y JavaScript como lenguajes del lado del cliente. A continuación una breve descripción de estos lenguajes.

1.8.1.1 HTML5

El HTML5 es la última versión del Lenguaje de Marcado de Hipertexto (código en que se programan los sitios web), y cambia los paradigmas de desarrollo y diseño web que se tenían al

introducir herramientas notables como etiquetas que permiten la publicación de archivos de audio y video; cambios en los llenados de formularios; y una web semántica mucho mejor aprovechada (15). Se utilizará el lenguaje HTML5 para el diseño de las interfaces del módulo, aprovechando las características del soporte para CSS3 y el manejo mejorado de formularios en el navegador.

1.8.1.2 CSS3

Las hojas de estilo en cascada (Cascading Style Sheets o CSS) son las que ofrecen la posibilidad de definir las reglas y estilos de representación en diferentes dispositivos, ya sean pantallas de equipos de escritorio, portátiles, móviles, impresoras u otros dispositivos capaces de mostrar contenidos web. Permiten definir de manera eficiente la representación de nuestras páginas y es uno de los conocimientos fundamentales que todo diseñador web debe manejar a la perfección para realizar su trabajo (16). Su uso permitirá obtener un mayor control de la presentación de la aplicación al poder tener todo el código CSS3 reunido en uno, lo que facilitará su modificación.

1.8.1.3 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. Los navegadores interpretan el código JavaScript integrado en las páginas web (17). Se utilizará este lenguaje para el manejo de los datos del lado del cliente.

1.8.1.4 Python 2.7

Python es un lenguaje de programación que soporta múltiples paradigmas, incluyendo programación orientada a objetos, programación imperativa y funcional. El código Python define objetos incorporados como listas enlazadas, tuplas, tablas hash y enteros de longitud arbitraria (18). Es el lenguaje de programación que utiliza el sistema GRHS.

1.8.2 Framework de desarrollo

Un framework de desarrollo es un ambiente de trabajo que contiene librerías de códigos y módulos que pueden ser reutilizados para el rápido desarrollo de aplicaciones. El centro TLM definió para el desarrollo del módulo la utilización de Django 1.4, jQuery 1.9, Twitter Bootstrap 3.0 y Backbone 1.1.

1.8.2.1 Django 1.4

Django es un framework web de código abierto escrito en Python que permite construir aplicaciones web de forma rápida y con menos código. Usa una modificación de la arquitectura Modelo-Vista-Controlador (MVC), llamada MTV (Model – Template – View), que sería Modelo-Plantilla-Vista. (19). Está liberado bajo la licencia BSD[1] y se centra en automatizar todo lo posible. Es el marco de trabajo que utiliza GRHS.

1.8.2.2 jQuery 1.9

jQuery es una biblioteca gratuita de Javascript, cuyo objetivo principal es simplificar las tareas de creación de páginas web responsives, acordes a lo estipulado en la web 2.0, la cual funciona en todos los navegadores actuales (20). Se utilizará el framework de desarrollo jQuery en su versión 1.9 ya que se utiliza para manejar todas las operaciones de las acciones del cliente en la aplicación gadmin del sistema GRHS.

1.8.2.3 Twitter Bootstrap 3.0

Bootstrap es un framework que simplifica el proceso de creación de diseños web combinando CSS y JavaScript. Es un framework potente con numerosos componentes web que ahorrará mucho esfuerzo y tiempo (21). En el desarrollo de la aplicación se empleará Boostrap 3.0 para la creación de las interfaces, ofreciendo una mejor integración con librerías como JQuery y proporcionando un diseño sólido basado en estándares como CSS3 y HTML5 que también serán usados en la solución.

1.8.2.4 Backbone 1.1

Backbone es un pequeño framework que permite construir aplicaciones usando Javascript, basado en el paradigma de diseño de aplicaciones Modelo Vista Controlador. Su objetivo consiste en probar y definir un conjunto de estructuras de datos junto al manejo de la interfaz por medio de vistas que son útiles cuando se construyen aplicaciones Javascript (22). Es utilizado en el módulo en la construcción de las tablas.

[1] **BSD:** Es una licencia de software libre que permite el uso del código fuente en software no libre.

1.8.3 Herramienta CASE: Visual Paradigm 5.0

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación (23). Esta herramienta es utilizada para el modelado de los diagramas de clases persistentes y el modelo físico de la base de datos, además para la representación de las capas del patrón arquitectónico Modelo – Vista – Plantilla.

1.8.4 Editor de código: Sublime Text 3

Para la implementación del software se empleará el editor de código Sublime Text. Es un editor multiplataforma y su sistema de resaltado de sintaxis soporta un gran número de lenguajes como CSS, HTML, JavaScript y Python. También tiene como características una mejor edición de HTML, incluyendo etiquetas de cierre automático y autocompletamiento de las mismas (24).

1.8.5 Sistema Gestor de Bases de Datos (SGBD)

Un SGBD es un conjunto de programas que permiten la creación de base de datos y proporciona herramientas para añadir, borrar, modificar y eliminar datos de la base de datos, además de mantener la integridad, confidencialidad y seguridad de los datos.

1.8.5.1 PostgreSQL 9.3

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente.

PostgreSQL tiene las siguientes características principales:

- Arquitectura cliente-servidor con un amplio rango de drivers y clientes.
- Diseño de alta concurrencia donde los lectores y escritores no se bloquean.
- Altamente configurable y extensible para muchos tipos de aplicación.
- Excelente escalabilidad y rendimiento con características de ajustes extensas.
- Optimizador de consultas sofisticado, adecuado para inteligencia de negocios.

- Soporta totalmente el acceso y procedimientos de base de datos en Java, Python, Perl, PHP, entre otros.
- Altamente confiable con características extensivas para durabilidad y alta disponibilidad.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas aunque no entre objetos, ya que no existen (26).

1.9 Conclusiones

En este capítulo se analizaron los principales conceptos a emplear para el desarrollo de la investigación. Se realizó un estudio de los principales estándares y regulaciones relacionados con la seguridad informática para un mejor entendimiento de las políticas de seguridad del centro TLM. Se investigaron los sistemas que permiten la automatización de controles de políticas de seguridad informática, arribando a la conclusión de utilizar el sistema GRHS, porque aporta funcionalidades que pueden ser reutilizadas. Se hizo un análisis detallado de la metodología, herramientas y lenguajes a utilizar para el correcto desarrollo de la solución, definiendo como metodología guía a la metodología ágil XP, como entorno de desarrollo Sublime Text 3, como herramienta de modelado Visual Paradigm en su versión 5.0 y como SGBD se seleccionó PostgreSQL en su versión 9.3, como lenguaje de programación del lado del servidor Python 2.7; y del lado del cliente HTML5, CSS 3 y JavaScript, como framework de desarrollo para los lenguajes seleccionados: Django 1.4, JQuery 1.9, Twitter Bootstrap 3.0 y Backbone 1.1.

CAPÍTULO 2 PROPUESTA DE SOLUCIÓN

2.1 Introducción

En este capítulo se describen las características del sistema a desarrollar, se realiza una descripción de las historias de usuario que se proponen para dar cumplimiento al objetivo general originado por la situación problemática. Se describen las características funcionales con que debe contar la aplicación y las propiedades del producto. Se realiza el plan de duración de las iteraciones a llevar a cabo.

2.2 Propuesta de solución

La solución propuesta está diseñada para permitir el control de las políticas de seguridad informática en las estaciones de trabajo del centro TLM de manera automatizada. El presente sistema permitirá a los asesores de tecnologías o jefes de centro llevar un control de todas aquellas máquinas inventariadas que cumplen o no con las políticas de seguridad establecidas por la universidad para los centros de producción, departamentos y laboratorios de docencia.

La solución está compuesta por un plugin para gclient y un módulo para la aplicación gadmin del sistema GRHS. El plugin de seguridad informática por su parte estará integrado por subplugins, los cuales serán los encargados de obtener de las máquinas donde esté instalado gclient, aquellas propiedades vinculadas a las políticas de seguridad, dígase entre estas: nombre del dominio, identificador, instalación de antivirus y firewall, uso de contraseña BIOS, grupos de administradores, carpetas compartidas, sistema operativo, entre otras propiedades; para su posterior verificación por el asesor de seguridad si la máquina cumple con el uso de las políticas. Las propiedades obtenidas de la máquina serán almacenadas en una base de datos local, además contará con un módulo visual en la interfaz de la aplicación gclient, donde se mostrarán los datos recogidos. En la ilustración #1 se muestra el funcionamiento del plugin.

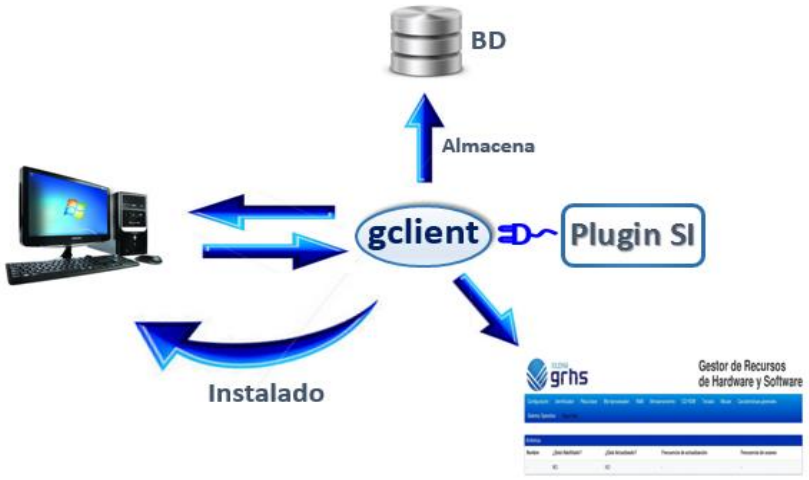


Ilustración 1: Funcionamiento del plugin de seguridad informática.

La aplicación gclient enviará la información al servidor (gserver), donde se encontrarán 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 será utilizada por el módulo de seguridad informática en la aplicación gadmin, en la cual el asesor de seguridad podrá visualizar todas las estaciones de trabajo del centro con sus propiedades, además podrá realizar búsquedas por diferentes criterios y exportar esta información a formato Excel. En la siguiente ilustración se muestra la propuesta de solución para el módulo de seguridad informática.

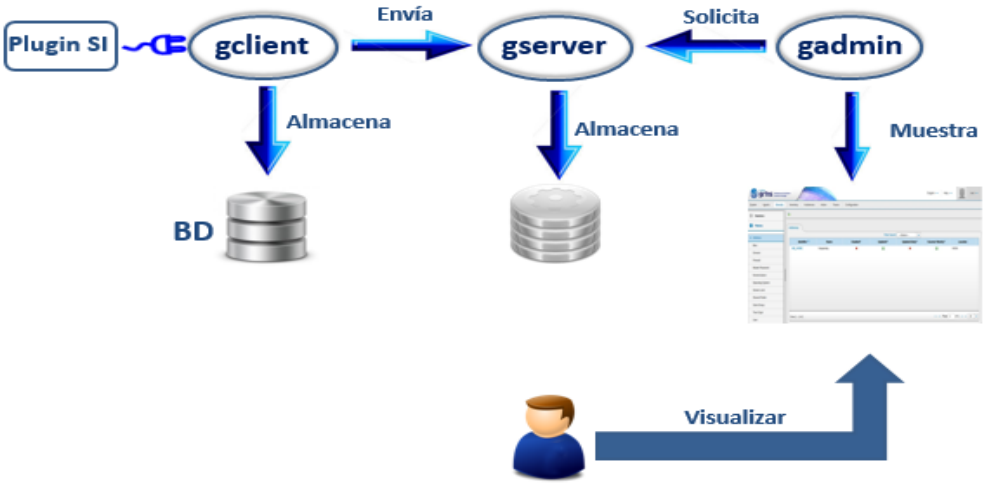


Ilustración 2: Propuesta de solución para el módulo de seguridad informática.

2.3 Funcionalidades del sistema

A continuación se detallan las funcionalidades del sistema. El sistema debe ser capaz de:

- Obtener propiedades del antivirus
  - ✓ Obtener antivirus instalado en la máquina para GNU/Linux.
  - ✓ Verificar activación de antivirus en la máquina para GNU/Linux.
  - ✓ Verificar actualización de antivirus en la máquina para GNU/Linux.
  - ✓ Verificar frecuencia de actualización de antivirus para GNU/Linux.
  - ✓ Verificar frecuencia de escaneo de antivirus para GNU/Linux.
  - ✓ Obtener antivirus instalado en la máquina para Windows.
  - ✓ Verificar activación de antivirus en la máquina para Windows.
  - ✓ Verificar actualización de antivirus en la máquina para Windows.
  - ✓ Verificar frecuencia de actualización de antivirus para Windows.
  - ✓ Verificar frecuencia de escaneo de antivirus para Windows.
- Obtener propiedades del BIOS
  - ✓ Verificar uso de contraseña BIOS para GNU/Linux.
  - ✓ Verificar secuencia de booteo para GNU/Linux.
  - ✓ Verificar uso de contraseña BIOS para Windows.
  - ✓ Verificar secuencia de booteo para Windows.
- Obtener propiedades del sistema operativo
  - ✓ Verificar actualizaciones de seguridad pendientes para GNU/Linux.
  - ✓ Verificar actualizaciones de seguridad pendientes para Windows.
- Obtener propiedades del bloqueo de pantalla
  - ✓ Verificar bloqueo automático de la pantalla para GNU/Linux.
  - ✓ Obtener tiempo de espera para bloqueo automático de la pantalla para GNU/Linux.
  - ✓ Obtener usuario logueado en la máquina para GNU/Linux.
  - ✓ Obtener entorno de escritorio instalado para GNU/Linux.
  - ✓ Verificar bloqueo automático de la pantalla para Windows.
  - ✓ Obtener tiempo de espera para bloqueo automático de la pantalla para Windows.
  - ✓ Obtener usuario logueado en la máquina para Windows.
- Obtener propiedades de carpetas compartidas
  - ✓ Obtener carpetas compartidas en la máquina para GNU/Linux.

- ✓ Verificar estado de las carpetas compartidas para GNU/Linux.
- ✓ Obtener ruta de la carpeta compartida para GNU/Linux.
- ✓ Obtener carpetas compartidas en la máquina para Windows.
- ✓ Verificar estado de las carpetas compartidas para Windows.
- ✓ Obtener ruta de la carpeta compartida para Windows.

➢ Obtener propiedades de la computadora
- ✓ Verificar activación del cortafuego en la máquina para GNU/Linux.
- ✓ Verificar activación del cortafuego en la máquina para Windows.

➢ Obtener propiedades del grupo de administradores
- ✓ Obtener grupo de dominio de administradores para GNU/Linux.
- ✓ Verificar activación del grupo de administradores para GNU/Linux.
- ✓ Obtener grupo de dominio de administradores para Windows.
- ✓ Verificar activación del grupo de administradores para Windows.

➢ Obtener propiedades de las contraseñas almacenadas
- ✓ Verificar contraseñas guardada en el Firefox para GNU/Linux.
- ✓ Verificar contraseña guardada en Thunderbird para GNU/Linux.
- ✓ Verificar contraseñas guardada en el Firefox para Windows.
- ✓ Verificar contraseña guardada en Thunderbird para Windows.

➢ Obtener propiedades del software para la protección de datos
- ✓ Obtener programa instalado para la protección de datos para GNU/Linux.
- ✓ Verificar activación de programa para la protección de datos para GNU/Linux.
- ✓ Obtener programa instalado para la protección de datos para Windows.
- ✓ Verificar activación de programa para la protección de datos para Windows.

➢ Obtener propiedades de los usuarios
- ✓ Identificar el grupo al que pertenece el usuario en GNU/Linux.
- ✓ Identificar el grupo al que pertenece el usuario en Windows.

✓ Mostrar propiedades del antivirus.

✓ Mostrar propiedades del BIOS.

✓ Mostrar propiedades del Sistema Operativo.

✓ Mostrar propiedades del bloqueo de pantalla.

✓ Mostrar propiedades de las carpetas compartidas.

✓ Mostrar propiedades de la computadora.

- ✓ Mostrar propiedades de las contraseñas.
- ✓ Mostrar propiedades del grupo de administradores.
- ✓ Mostrar propiedades del programa para la protección de datos.
- ✓ Mostrar propiedades de las cuentas de usuario.
- ✓ Listar máquinas con las propiedades del antivirus.
- ✓ Listar máquinas con las propiedades del BIOS.
- ✓ Listar máquinas con las propiedades del sistema operativo.
- ✓ Listar máquinas con las propiedades del bloqueo de pantalla.
- ✓ Listar máquinas con las propiedades de las carpetas compartidas.
- ✓ Listar máquinas con las propiedades de la computadora.
- ✓ Listar máquinas con las propiedades de las contraseñas almacenadas.
- ✓ Listar máquinas con las propiedades del grupo de administradores.
- ✓ Listar máquinas con las propiedades del software para la protección de datos.
- ✓ Listar máquinas con las propiedades de los usuarios.
- ✓ Filtrar búsquedas por las propiedades del antivirus.
- ✓ Filtrar búsquedas por las propiedades del BIOS.
- ✓ Filtrar búsquedas por las propiedades del sistema operativo.
- ✓ Filtrar búsquedas por las propiedades del bloqueo de pantalla.
- ✓ Filtrar búsquedas por las propiedades de las carpetas compartidas.
- ✓ Filtrar búsquedas por las propiedades de la computadora.
- ✓ Filtrar búsquedas por las propiedades de las contraseñas almacenadas.
- ✓ Filtrar búsquedas por las propiedades del grupo de administradores.
- ✓ Filtrar búsquedas por las propiedades del software para la protección de datos.
- ✓ Filtrar búsquedas por las propiedades de los usuarios.
- ✓ Exportar reporte estadístico.
- ✓ Internacionalización del módulo.

2.4 Propiedades del producto

Para un correcto funcionamiento del sistema se deben tener en cuenta los siguientes requisitos no funcionales:

2.4.1 Usabilidad

El usuario que interactúe con el módulo de seguridad del sistema GRHS, deberá tener conocimientos básicos de informática y notará que es de fácil manejo y comprensión.

2.4.2 Hardware

Características del hardware del servidor para 100 clientes:

- 4 GB de RAM.
- 20 GB de disco duro.
- 2 núcleos, cada uno de 3.0 GHz.

Características del hardware del cliente:

- 256 MB de RAM.
- 1 procesador de 2.0 GHz.
- 2 GB de espacio en el disco duro.

2.4.3 Software

Características del software en el servidor:

- Sistema operativo: Ubuntu Server 12.04, Debian 7, Centos 6.x, Centos 7.
- Servidor web: Nginx 1.2, uWsgi
- Python 2.7.

Características del software en el cliente:

- Sistema operativo: Windows 7, Windows 8, Debian 6.0, Debian 7.0, Ubuntu 11.04, Ubuntu 12.04, Ubuntu 14.04, Nova 2013.
- Python 2.7.

2.4.4 Interfaz

La solución propuesta poseerá una interfaz que cumpla con las pautas de diseño de Xilema Base – Web.

2.5 Fase de Exploración

Es la primera fase de la metodología XP, en esta, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la entrega del producto, las cuales les posibilitan a los programadores estimar la duración del proyecto. Durante esta fase el equipo de desarrollo se familiariza con las herramientas, lenguajes y aplicaciones necesarias para el desarrollo del sistema.

2.5.1 Historias de Usuario

Las historias de usuario (HU) son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (27).

Las HU son representadas mediante tablas divididas por las diferentes secciones:

- 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.
- Usuario: personas involucradas en el desarrollo de las HU.
- 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 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.

A continuación se describen 3 de las historias de usuario, el resto se podrán encontrar en el Anexo 1:

Tabla 1: Historia de Usuario # 1: Obtener antivirus instalado en la máquina.

Historia de Usuario	
Número: 1	**Usuario:** Usuario
Nombre de historia: Obtener antivirus instalado en la máquina para Linux.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1/5	**Iteración asignada:** 1
Programadores responsables: Dania Carmenate, Dagoberto F. Pérez	
Descripción: Permite obtener el antivirus instalado en la estación de trabajo.	
Observaciones:	
Prototipo de Interfaz: No aplica.	

Tabla 2: Historia de Usuario # 56: Listar máquinas con las propiedades del antivirus.

Historia de Usuario	
Número: 56	**Usuario:** Usuario

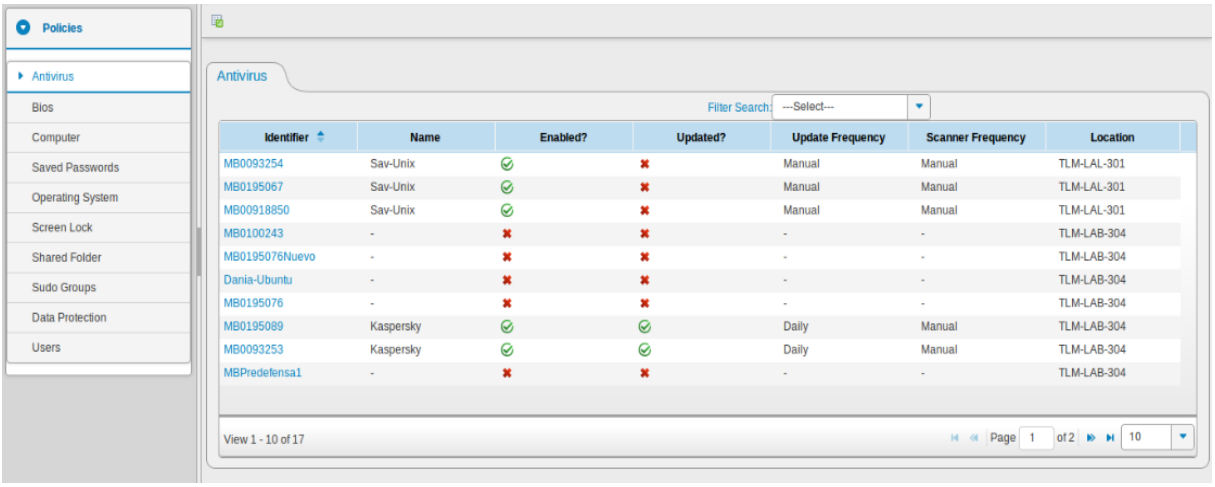
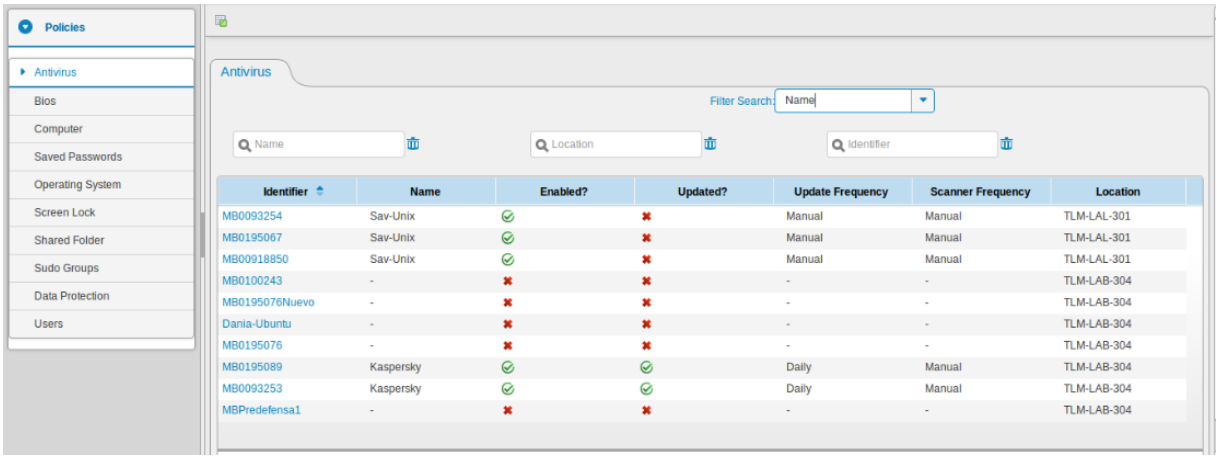
Nombre de historia: Listar máquinas con las propiedades del antivirus.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2/5	**Iteración asignada:** 3
Programadores responsables: Dania Carmenate, Dagoberto F. Pérez	
Descripción: Se listan todas las estaciones de trabajo con sus respectivas propiedades del antivirus (nombre, estado (activo o no activo), si se encuentra actualizado, frecuencia de actualización y frecuencia de escaneo).	
Observaciones:	
Prototipo de Interfaz: 	

Tabla 3: Historia de Usuario # 66: Filtrar búsquedas por las propiedades del antivirus.

Historia de Usuario	
Número: 66	**Usuario:** Usuario
Nombre de historia: Filtrar búsquedas por las propiedades del antivirus.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta

Puntos estimados: 1/5	**Iteración asignada:** 3
Programadores responsables: Dania Carmenate, Dagoberto F. Pérez	
Descripción: Se listan aquellas estaciones de trabajo que posean el criterio de búsqueda seleccionado por el usuario de acuerdo a las propiedades del antivirus (identificador, nombre, estado, actualización, frecuencia de actualización, frecuencia de escaneo, localización).	
Observaciones:	
Prototipo de Interfaz: 	

2.6 Fase de Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la entrega y se determina un cronograma en conjunto con el cliente (27). Las estimaciones de esfuerzo asociado a la implementación de las HU la establecen los programadores utilizando como medida el *punto*. Un punto, equivale a una semana ideal de programación, planificada para 5 días, donde los miembros del equipo de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción.

2.6.1 Estimación de esfuerzo por historias de usuario

A continuación se muestra la estimación de esfuerzo por cada historia de usuario propuesta para el desarrollo de la solución:

Tabla 4: Estimación de esfuerzo por HU.

Número	Historia de Usuario	Puntos estimados
1	Obtener antivirus instalado en la máquina para GNU/Linux.	1/5
2	Verificar activación de antivirus en la máquina para GNU/Linux.	1/5
3	Verificar actualización de antivirus en la máquina para GNU/Linux.	1/5
4	Verificar frecuencia de actualización de antivirus para GNU/Linux.	1/5
5	Verificar frecuencia de escaneo de antivirus para GNU/Linux.	1/5
6	Verificar uso de contraseña BIOS para GNU/Linux.	1/5
7	Verificar secuencia de booteo para GNU/Linux.	1/5
8	Verificar actualizaciones de seguridad pendientes para GNU/Linux.	1/5
9	Verificar bloqueo automático de la pantalla para GNU/Linux.	1/5
10	Obtener tiempo de espera para bloqueo automático de la pantalla en GNU/Linux.	1/5
11	Obtener usuario logueado en la máquina para GNU/Linux.	1/5
12	Obtener entorno de escritorio instalado para GNU/Linux.	1/5
13	Obtener carpetas compartidas en la máquina para GNU/Linux.	1/5
14	Verificar estado de las carpetas compartidas en la máquina para GNU/Linux.	1/5
15	Obtener ruta de la carpeta compartida para GNU/Linux.	1/5

16	Verificar activación de cortafuego en la máquina para GNU/Linux.	1/5
17	Obtener grupo de dominio de administradores para GNU/Linux.	1/5
18	Verificar activación del grupo de administradores para GNU/Linux.	1/5
19	Verificar contraseñas guardada en el Firefox para GNU/Linux.	2/5
20	Verificar contraseña guardada en Thunderbird para GNU/Linux.	2/5
21	Obtener programa instalado para la protección de datos en GNU/Linux.	1/5
22	Verificar activación de programa para la protección de datos para GNU/Linux.	1/5
23	Identificar el grupo al que pertenece el usuario en GNU/Linux.	1/5
24	Mostrar propiedades del antivirus.	1/5
25	Mostrar propiedades del BIOS.	1/5
26	Mostrar propiedades del Sistema Operativo.	1/5
27	Mostrar propiedades del bloqueo de pantalla.	1/5
28	Mostrar propiedades de las carpetas compartidas.	1/5
29	Mostrar propiedades de la computadora.	1/5
30	Mostrar propiedades de las contraseñas.	1/5
31	Mostrar propiedades del grupo de administradores.	1/5
32	Mostrar propiedades del programa para la protección de datos.	1/5

33	Mostrar propiedades de las cuentas de usuario.	1/5
34	Obtener antivirus instalado en la máquina para Windows.	1/5
35	Verificar activación de antivirus en la máquina para Windows.	1/5
36	Verificar actualización de antivirus en la máquina para Windows.	1/5
37	Verificar frecuencia de actualización de antivirus en la máquina para Windows.	1/5
38	Verificar frecuencia de escaneo de antivirus para Windows.	1/5
39	Verificar uso de contraseña BIOS para Windows.	1/5
40	Verificar secuencia de booteo para Windows.	1/5
41	Verificar actualizaciones de seguridad pendientes para Windows.	1/5
42	Verificar bloqueo automático de la pantalla para Windows.	1/5
43	Obtener tiempo de espera para bloqueo automático de la pantalla en Windows.	1/5
44	Obtener usuario logueado en la máquina para Windows.	1/5
45	Obtener carpetas compartidas en la máquina para Windows.	1/5
46	Verificar estado de las carpetas compartidas en la máquina para Windows.	1/5
47	Obtener ruta de la carpeta compartida para Windows.	1/5
48	Verificar activación de cortafuego en la máquina para Windows.	1/5
49	Obtener grupo de dominio de administradores para Windows.	1/5

50	Verificar activación del grupo de administradores para Windows.	1/5
51	Verificar contraseñas guardada en el Firefox para Windows.	1/5
52	Verificar contraseña guardada en Thunderbird para Windows.	1/5
53	Obtener programa instalado para la protección de datos en Windows.	1/5
54	Verificar activación de programa para la protección de datos para Windows.	1/5
55	Identificar el grupo al que pertenece el usuario en Windows.	1/5
56	Listar máquinas con las propiedades del antivirus.	2/5
57	Listar máquinas con las propiedades del BIOS.	2/5
58	Listar máquinas con las propiedades del sistema operativo.	2/5
59	Listar máquinas con las propiedades del bloqueo de pantalla.	2/5
60	Listar máquinas con las propiedades de las carpetas compartidas.	2/5
61	Listar máquinas con las propiedades de la computadora.	2/5
62	Listar máquinas con las propiedades de las contraseñas almacenadas.	2/5
63	Listar máquinas con las propiedades del grupo de administradores.	2/5
64	Listar máquinas con las propiedades del software para la protección de datos.	2/5
65	Listar máquinas con las propiedades de los usuarios.	2/5
66	Filtrar búsquedas por las propiedades del antivirus.	1/5

67	Filtrar búsquedas por las propiedades del BIOS.	1/5
68	Filtrar búsquedas por las propiedades del sistema operativo.	1/5
69	Filtrar búsquedas por las propiedades de las contraseñas almacenadas.	1/5
70	Filtrar búsquedas por las propiedades del bloqueo de pantalla.	1/5
71	Filtrar búsquedas por las propiedades del grupo de administradores.	1/5
72	Filtrar búsquedas por las propiedades del software para la protección de datos.	1/5
73	Filtrar búsquedas por las propiedades de los usuarios.	1/5
74	Exportar reporte estadístico.	1/5
75	Internacionalización del módulo.	1/5

2.6.2 Plan de iteraciones

Luego de ser identificada, descrita y estimada cada una de las HU se procede a planificar la fase de implementación, la cual se realizará en tres iteraciones, las cuales se describen a continuación.

2.6.2.1 Iteración 1

En la iteración 1 se llevará a cabo la implementación de la HU número 1 a la 33, perteneciente al desarrollo del plugin del gclient para el sistema operativo GNU/Linux. Al terminar la iteración esto representará un 44% de la solución.

2.6.2.2 Iteración 2

En la iteración 2 se llevará a cabo la implementación de la HU número 33 a la 55, perteneciente al desarrollo del plugin del gclient para el sistema operativo Windows. Al terminar la iteración esto representará un 73.3% de la solución.

2.6.2.3 Iteración 3

En la iteración 3 se llevará a cabo la implementación de la HU número 56 a la 75, perteneciente al desarrollo del módulo de seguridad informática de la aplicación gadmin. Al terminar la iteración esto representará un 100% de la solución.

2.6.3 Plan de duración de las iteraciones

Se encarga de mostrar las HU en el orden en que se implementarán en cada una de las iteraciones, así como la duración estimada para cada iteración. En la tabla # 5 se muestra el plan de duración de las iteraciones.

Tabla 5: Plan de duración de las iteraciones.

Iteración	Orden de la HU a implementar	Duración total
1	Obtener antivirus instalado en la máquina para GNU/Linux. Verificar activación de antivirus en la máquina para GNU/Linux. Verificar actualización de antivirus en la máquina para GNU/Linux. Verificar frecuencia de actualización de antivirus para GNU/Linux. Verificar frecuencia de escaneo de antivirus para GNU/Linux. Verificar uso de contraseña BIOS para GNU/Linux. Verificar secuencia de booteo para GNU/Linux. Verificar actualizaciones de seguridad pendientes para GNU/Linux. Verificar bloqueo automático de la pantalla para GNU/Linux. Obtener tiempo de espera para bloqueo automático de la pantalla para GNU/Linux. Obtener usuario logueado en la máquina para GNU/Linux. Obtener entorno de escritorio instalado para GNU/Linux.	**8 semanas**

	Obtener carpetas compartidas en la máquina para GNU/Linux. Verificar estado de las carpetas compartidas para GNU/Linux. Obtener ruta de la carpeta compartida para GNU/Linux. Verificar activación del cortafuego en la máquina para GNU/Linux. Obtener grupo de dominio de administradores para GNU/Linux. Verificar activación del grupo de administradores para GNU/Linux. Verificar contraseñas guardada en el Firefox para GNU/Linux. Verificar contraseña guardada en Thunderbird para GNU/Linux. Obtener programa instalado para la protección de datos para GNU/Linux. Verificar activación de programa para la protección de datos para GNU/Linux. Identificar el grupo al que pertenece el usuario en GNU/Linux. Mostrar propiedades del antivirus. Mostrar propiedades del BIOS. Mostrar propiedades del Sistema Operativo. Mostrar propiedades del bloqueo de pantalla. Mostrar propiedades de las carpetas compartidas. Mostrar propiedades de la computadora. Mostrar propiedades de las contraseñas. Mostrar propiedades del grupo de administradores. Mostrar propiedades del programa para la protección de datos. Mostrar propiedades de las cuentas de usuario.	

2	Obtener antivirus instalado en la máquina para Windows. Verificar activación de antivirus en la máquina para Windows. Verificar actualización de antivirus en la máquina para Windows. Verificar frecuencia de actualización de antivirus para Windows. Verificar frecuencia de escaneo de antivirus para Windows. Verificar uso de contraseña BIOS para Windows. Verificar secuencia de booteo para Windows. Verificar actualizaciones de seguridad pendientes para Windows. Verificar bloqueo automático de la pantalla para Windows. Obtener tiempo de espera para bloqueo automático de la pantalla para Windows. Obtener usuario logueado en la máquina para Windows. Obtener carpetas compartidas en la máquina para Windows. Verificar estado de las carpetas compartidas para Windows. Obtener ruta de la carpeta compartida para Windows. Verificar activación del cortafuego en la máquina para Windows. Obtener grupo de dominio de administradores para Windows. Verificar activación del grupo de administradores para Windows. Verificar contraseñas guardada en el Firefox para Windows. Verificar contraseña guardada en Thunderbird para Windows. Obtener programa instalado para la protección de datos para Windows. Verificar activación de programa para la protección de datos para	**5 semanas**

	Windows. Identificar el grupo al que pertenece el usuario en Windows.	
3	Listar máquinas con las propiedades del antivirus. Listar máquinas con las propiedades del BIOS. Listar máquinas con las propiedades del sistema operativo. Listar máquinas con las propiedades del bloqueo de pantalla. Listar máquinas con las propiedades de las carpetas compartidas. Listar máquinas con las propiedades de la computadora. Listar máquinas con las propiedades de las contraseñas almacenadas. Listar máquinas con las propiedades del grupo de administradores. Listar máquinas con las propiedades del software para la protección de datos. Listar máquinas con las propiedades de los usuarios. Filtrar búsquedas por las propiedades del antivirus. Filtrar búsquedas por las propiedades del BIOS. Filtrar búsquedas por las propiedades del sistema operativo. Filtrar búsquedas por las propiedades del bloqueo de pantalla. Filtrar búsquedas por las propiedades de las carpetas compartidas. Filtrar búsquedas por las propiedades de la computadora. Filtrar búsquedas por las propiedades de las contraseñas almacenadas. Filtrar búsquedas por las propiedades del grupo de administradores.	**6 semanas**

	Filtrar búsquedas por las propiedades del software para la protección de datos. Filtrar búsquedas por las propiedades de los usuarios. Exportar reporte estadístico. Internacionalización del módulo.	

2.6.4 Plan de entregas

El plan de entrega define la fecha fin de cada iteración, los productos obtenidos divididos en las tres iteraciones. En la tabla # 6 se muestra el plan de entrega.

Tabla 6: Plan de entrega.

Final de la Iteración 1 (24 de febrero del 2015)	Final de la Iteración 2 (2 de abril del 2015)	Final de la Iteración 3 (8 de mayo del 2015)
Plugin al Gclient para Linux	**Plugin al Gclient para Windows**	**Módulo de Seguridad Informática al GRHS**

2.7 Conclusiones

En este capítulo se describió la propuesta de solución a desarrollar. Se definieron las funcionalidades a implementar y los requisitos que debe cumplir el producto, así como la descripción de las HU divididas por tres iteraciones y la estimación del esfuerzo dedicado a la realización de cada una de ellas en el orden en que se les dará cumplimiento según las necesidades del cliente. Se obtuvo el plan de entregas y se realizó la planificación lo cual posibilitó delimitar el ciclo de desarrollo de la solución.

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

En este capítulo se describen las fases de implementación y prueba, propias de la metodología XP. Se define la arquitectura a utilizar en la solución. Se realiza un análisis de los patrones de diseño que se utilizará para su correcta implementación. Se definen las tarjetas Clase-Responsabilidad-Colaborador (CRC) para identificar y organizar las clases orientadas a objetos. Se exponen las clases persistentes y el modelo físico de la base de datos. Se describen las tareas de ingeniería generada para cada historia de usuario. Por último, se realizan las pruebas al producto para comprobar su correcto funcionamiento y si las funcionalidades se han implementado de forma adecuada.

3.2 Arquitectura

Una arquitectura de software define la estructura del sistema, la cual adquiere gran importancia debido a que las representaciones de arquitecturas de software permiten la comunicación entre todas las partes interesadas en el desarrollo de un sistema de cómputo; constituye un modelo comprensible de cómo está estructurado el sistema y cómo trabajan juntos sus componentes (28).

3.2.1 Arquitectura Cliente-Servidor

En esta arquitectura la computadora de cada uno de los usuarios, llamada cliente, produce una demanda de información a cualquiera de las computadoras que proporcionan información, conocidas como servidores estos últimos responden a la demanda del cliente que la produjo. Mediante esta arquitectura el usuario (cliente) puede acceder a la información sin tener en cuenta su ubicación física y donde pueda estar alojada la misma. A continuación se muestra en la ilustración # 3 la arquitectura cliente-servidor.

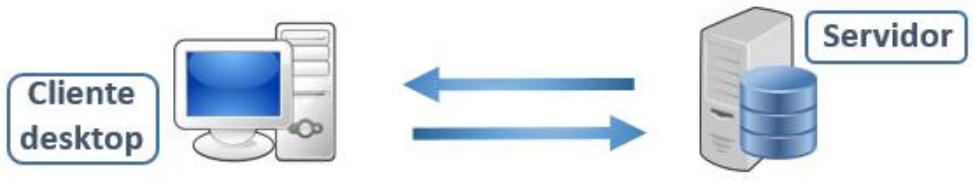


Ilustración 3: Arquitectura Cliente-Servidor

Para el desarrollo de la solución se definió la arquitectura cliente-servidor. En este caso se cuenta con el cliente desktop (gclient) y el servidor (gserver). El cliente luego de realizar el inventario de la estación de trabajo en que se encuentra instalado, le solicita al servidor el último inventario realizado a dicha computadora; el servidor responde a la solicitud del cliente, el cliente compara ambos inventarios y envía nuevamente al servidor la información actualizada de la máquina actualizada.

3.3 Patrón arquitectónico

Los patrones arquitectónicos son los que definen la estructura de un software, los cuales a su vez se componen de subsistemas con sus responsabilidades. Además poseen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño (29). Para el desarrollo del módulo el patrón a seguir es el Modelo –Plantilla –Vista (MTV).

3.3.1 Modelo –Plantilla –Vista

Uno de los framework a utilizar para el desarrollo de la solución es Django, el cual usa el Modelo –Plantilla –Vista como una modificación del patrón Modelo –Vista –Controlador (MVC), empleado en la implantación de la solución. En Django el modelo sigue siendo modelo, la vista pasa hacer la plantilla y el controlador la vista.

- 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 tienen, y las relaciones entre los datos.
- Plantilla: se refiere a la capa de presentación. Esta capa contiene las decisiones relacionadas con la presentación: cómo debería mostrarse algo en una página web u otro tipo de documento.
- Vista: se refiere a la capa de lógica del negocio. Esta capa contiene el acceso al modelo y delega en las plantillas apropiadas. (29).

A continuación en la ilustración 4 se muestra el funcionamiento del patrón MTV:

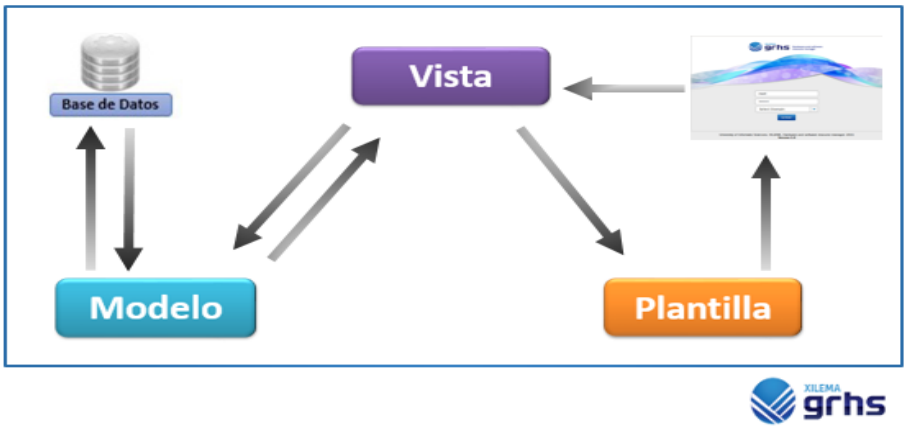


Ilustración 4: Patrón Modelo –Plantilla –Vista

Este patrón comienza su funcionamiento cuando el navegador envía una solicitud a la vista. La vista interactúa con la modelo para obtener los datos. La vista llama a la plantilla y esta envía la respuesta de la solicitud al navegador. En la siguiente ilustración se muestra el funcionamiento del patrón MTV:

3.4 Representación de las Capas de la Arquitectura

A continuación se describen las clases situadas en cada una de las capas de la arquitectura del patrón MTV. Esta arquitectura se encuentra estructurada por 3 capas: Vista, Plantilla y Modelo.

3.4.1 Capa Vista

La capa vista tiene como propósito determinar qué datos serán visualizados, es la encargada de la validación de datos a través de formularios. La ilustración 5 muestra una representación de las clases de la capa vista.

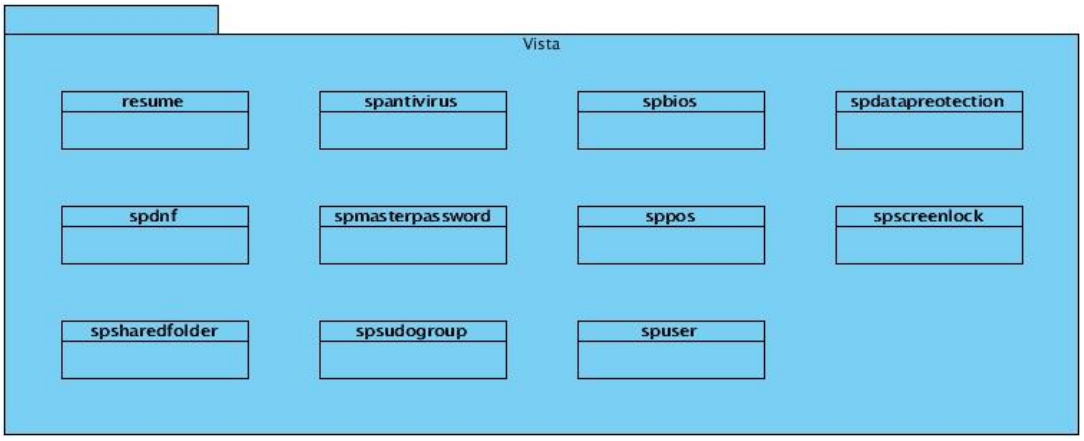


Ilustración 5: Capa Vista

resume: en este archivo se encuentran los métodos necesarios para el llenado de las tablas de cada una de las políticas de seguridad.

spantivirus: en este archivo se encuentran los métodos necesarios para el llenado de las tablas y filtrados por las propiedades del antivirus.

spbios: en este archivo se encuentran los métodos necesarios para el llenado de las tablas y filtrados por las propiedades del BIOS.

spdataprotection: en este archivo se encuentran los métodos necesarios para el llenado de las tablas y filtrados por las propiedades del programa para la protección de datos.

spdnf: en este archivo se encuentran los métodos necesarios para el llenado de las tablas y filtrados por las propiedades de la computadora.

spmasterpassword: en este archivo se encuentran los métodos necesarios para el llenado de las tablas y filtrados por las propiedades del almacenamiento de contraseñas.

spos: en este archivo se encuentran los métodos necesarios para el llenado de las tablas y filtrados por las propiedades del sistema operativo.

spscreenlock: en este archivo se encuentran los métodos necesarios para el llenado de las tablas y filtrados por las propiedades del bloqueo de pantalla.

spsharedfolder: en este archivo se encuentran los métodos necesarios para el llenado de las tablas y filtrados por las propiedades de las carpetas compartidas.

spsudogroup: en este archivo se encuentran los métodos necesarios para el llenado de las tablas y filtrados por las propiedades de los grupos de administradores.

spuser: en este archivo se encuentran los métodos necesarios para el llenado de las tablas y filtrados por las propiedades de los usuarios.

3.4.2 Capa Plantilla

La capa plantilla organiza los datos recibidos por la capa vista para su presentación en el navegador web. La ilustración 6 muestra una representación de las clases de la capa plantilla.

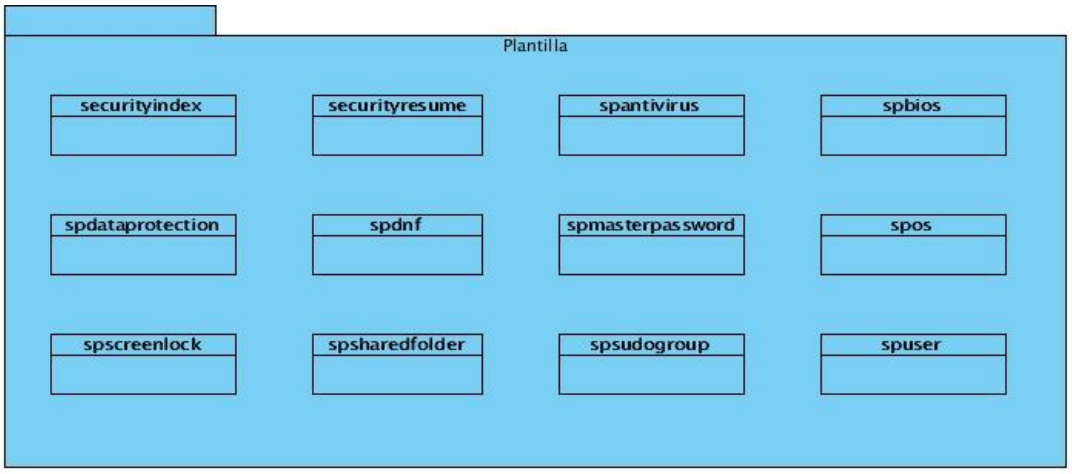


Ilustración 6: Capa Plantilla

3.4.3 Capa Modelo

La capa modelo permite indicar y controlar el comportamiento de los datos almacenados. La ilustración 7 muestra una representación de las clases de la capa modelo.

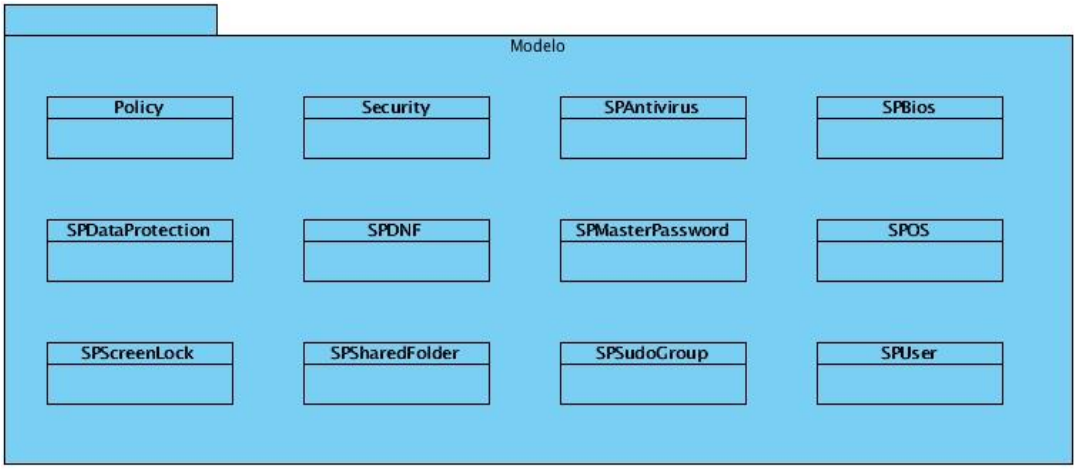


Ilustración 7: Capa Modelo

Security: clase responsable de brindar información del cliente.

Policy: clase que relaciona a la clase Security con las restantes clases.

SPUser: clase responsable de obtener de la base de datos detalles de las cuentas de usuarios creadas en las máquinas.

SPSudoGroup: clase responsable de obtener de la base de datos detalles de los grupos de administradores presentes en las máquinas**.**

SPMasterPassword: clase responsable de obtener de la base de datos detalles de las contraseña almacenadas en los programas Firefox y Thunderbird..

SPAntivirus: clase responsable de obtener de la base de datos detalles del antivirus instalado en las máquinas.

SPOS: clase responsable de obtener de la base de datos detalles del sistema operativo que presentan las máquinas.

SPSharedFolder: clase responsable de obtener de la base de datos detalles de las carpetas compartidas que presentan las máquinas.

SPScreenLock: clase responsable de obtener de la base de datos detalles del bloqueo automático de las máquinas.

SPBios: clase responsable de obtener de la base de datos detalles del BIOS de las máquinas.

SPDNF: clase responsable de obtener de la base de datos detalles de las máquinas (uso de firewall, identificador, dominio).

SPDataProtection: clase responsable de obtener de la base de datos detalles del programa instalado en las máquinas para la protección de datos.

3.5 Patrones de Diseño

Un patrón de diseño es un conjunto de reglas que describen cómo afrontar tareas y solucionar problemas que surgen durante el desarrollo del software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (30).

3.5.1 Patrón GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (31).

3.5.1.1 Experto en información

Problema: ¿De qué forma se puede saber qué responsabilidad delegar a cada objeto? Solución: Asignar una responsabilidad al experto en información; la clase que tiene la información necesaria para llevar a cabo la responsabilidad (31). Este patrón se verá reflejado en la clase Security del gclient.

3.5.1.2 Creador

Problema: ¿Quién debería ser responsable de crear una nueva instancia?

Solución: Crear una nueva instancia por la clase que: tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase y contiene o agrega la clase (31). Este patrón se verá reflejado en la clase Security del gclient al crear los restantes subplugins.

3.5.1.3 Controlador

Problema: ¿Quién gestiona un evento del sistema?

Solución: Asignar la responsabilidad de gestionar un mensaje de un evento del sistema a una clase que represente una de estas dos opciones: representa el sistema global, dispositivo o subsistema (controlador de fachada) y representa un escenario de caso de uso en el que tiene lugar el evento del sistema (controlador de caso de uso o de sesión) (31). Este patrón se verá reflejado en la clase Security del plugin en el gclient y en la clase Security del módulo en gadmin.

3.5.1.4 Alta cohesión

Problema: ¿Cómo mantener manejable la complejidad?

Solución: Asignar responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase (31). Este patrón se verá reflejado en las clases que reciben los datos del cliente: SPAntivirus, SPBios, SPOS del Gserver

3.5.1.5 Bajo acoplamiento

Problema: ¿Cómo dar soporte a las bajas dependencias y al incremento de la reutilización?

Solución: Diseñar con el objetivo de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases (31). Este patrón se verá reflejado en la clase Security del Gclient con los subplugins (spantivirus, spbios, spos, spdataprotection, spmasterpassword, spscreenlock, spdnf, spuser, spsudogroup).

3.6 Tarjeta Clase –Responsabilidad –Colaborador (CRC)

La metodología de desarrollo XP como parte de la fase de diseño propone el modelado de Clase-Responsabilidad-Colaborador (CRC), lo que constituye un modelo simple de organizar las clases más relevantes para las funcionalidades del sistema, con el objetivo de desarrollar una representación organizada de las clases. A continuación se muestra la tarjeta CRC de dos de las clases más importantes, el resto se encuentra en el Anexo 2.

Tabla 7: CRC: SPAntivirus – Gclient.

Clase: SPAntivirus	
Responsabilidad	**Colaborador**

Iniciar inventario. Verificar instalación de antivirus. Verificar si antivirus se encuentra habilitado. Verificar si antivirus se encuentra actualizado. Obtener frecuencia de escaneo. Obtener frecuencia de actualización.	Policy, Popen

Tabla 8: CRC: SPAntivirus - Gserver.

Clase: SPAntivirus	
Responsabilidad	**Colaborador**
Obtener el antivirus de la base de datos (BD). Obtener si el antivirus se encuentra habilitado de la BD. Obtener si el antivirus se encuentra actualizado de la BD. Obtener frecuencia de actualización de la BD. Obtener frecuencia de escaneo de la BD.	Policy, models

3.7 Diagrama de clases persistentes

El diagrama de clases persistentes muestra las clases que tienen durabilidad en el tiempo y la relación que tienen con las restantes clases. A continuación en la ilustración # 8 se muestra el diagrama de clases persistentes referente a la aplicación del lado del servidor, y en la ilustración # 9 el diagrama de la aplicación gclient.

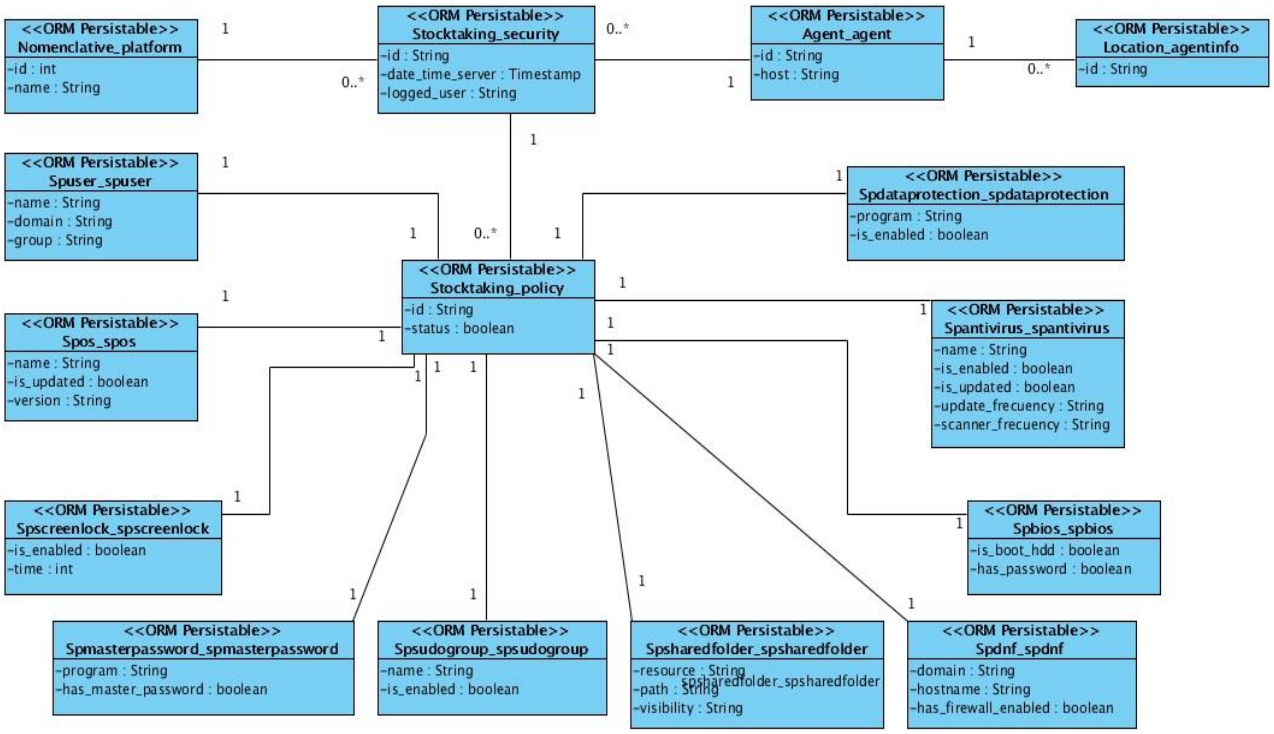


Ilustración 8: Diagrama de clases persistentes del lado del servidor

stocktaking_security: entidad principal, brinda la información del cliente (plataforma, agente, usuario y fecha).

nomenclative_plataform: entidad que brinda la plataforma por la cual se ha conectado el cliente.

stocktaking_policy: entidad abstracta, relaciona la clase stocktaking_security con las restantes clases.

agent: entidad que brinda el identificador del cliente.

location_agentinfo: entidad que brinda la localización de donde se ha conectado el cliente.

Policy: entidad intermediaria que recibe los datos del BIOS enviados por el cliente.

spbios: entidad que especifica las características del BIOS de una máquina.

spantivirus: entidad que especifica las características del antivirus de una máquina.

spsudogroup: entidad que especifica las características de los grupos de administradores de una máquina.

spsharedfolder: entidad que especifica las características de las carpetas compartidas de una máquina.

spscreenlock: entidad que especifica las características del bloqueo de pantalla de una máquina.

spos: entidad que especifica las características del sistema operativo de una máquina.

spmasterpassword: entidad que especifica las características del almacenamiento de contraseñas de una máquina.

spuser: entidad que especifica las características de los usuarios de una máquina.

dataprotection: entidad que especifica las características del programa para la protección de datos de una máquina.

spdnf: entidad que especifica las características en general de una máquina.

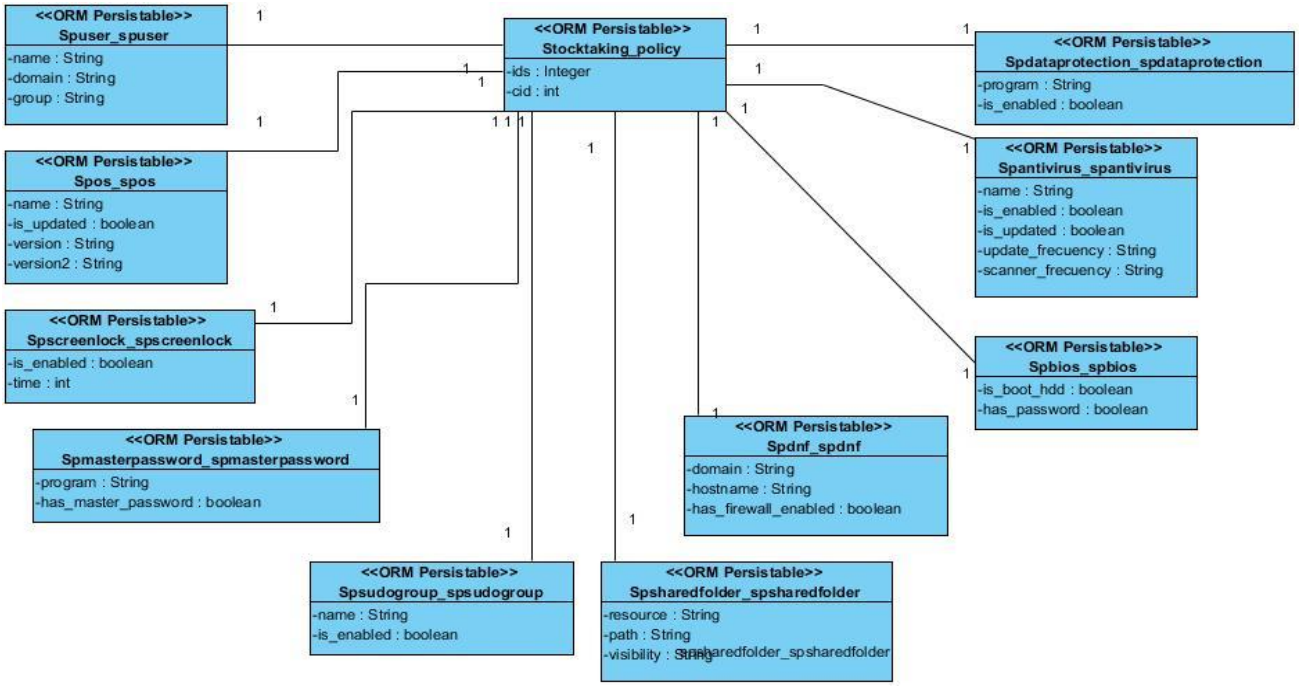


Ilustración 9: Diagrama de clases persistentes del lado del cliente

policy: entidad abstracta de la cual heredan las demás clases.

spuser: entidad encargada de obtener las propiedades de las cuentas de usuarios de la estación de trabajo que esté siendo inventariada.

spos: entidad encargada de obtener las propiedades del sistema operativo de la estación de trabajo que esté siendo inventariada.

spscreenlock: entidad encargada de obtener las propiedades del bloqueo de pantalla de la estación de trabajo que esté siendo inventariada.

spmasterpassword: entidad encargada de obtener las propiedades del almacenamiento de contraseñas en la estación de trabajo que esté siendo inventariada.

spsudogroup: entidad encargada de obtener las propiedades de los grupos de administradores de la estación de trabajo que esté siendo inventariada.

spsharedfolder: entidad encargada de obtener las propiedades de las carpetas compartidas en la estación de trabajo que esté siendo inventariada.

spdnf: entidad encargada de obtener las propiedades generales de la estación de trabajo que esté siendo inventariada.

spbios: entidad encargada de obtener las propiedades del BIOS de la estación de trabajo que esté siendo inventariada.

spantivirus: entidad encargada de obtener las propiedades del uso de antivirus en la estación de trabajo que esté siendo inventariada.

spdataprotection: entidad encargada de obtener las propiedades de los programa para la protección de datos instalados en la estación de trabajo que esté siendo inventariada.

3.8 Modelo físico de la Base de Datos

A continuación en la ilustración # 10 se muestra el modelo físico de la base de datos del servidor obtenido del diagrama de clases persistentes y en la ilustración # 11 se muestra el modelo físico de la base de datos del cliente, estos modelos muestran la relación que existe entre las entidades de las bases de datos.

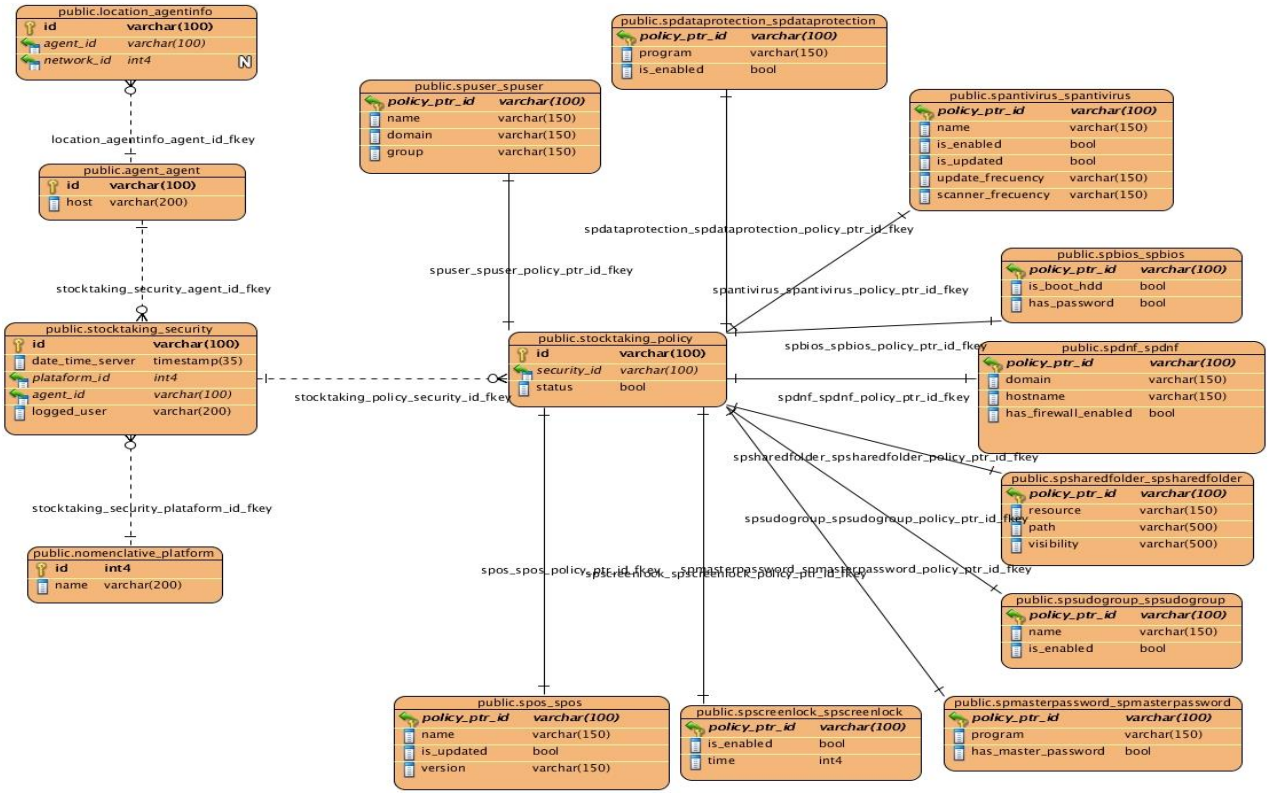


Ilustración 10: Modelo físico de la Base de Datos del servidor.

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

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).

location: tabla donde se almacena las ubicaciones de las computadoras clientes por locales físicos.

spuser: tabla donde se almacenan las propiedades de los clientes en cuanto a las cuentas de usuario creadas en las máquinas.

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

spscreenlock: tabla donde se almacenan las propiedades de los clientes en cuanto a el bloqueo automático de la pantalla.

spmasterpassword: tabla donde se almacenan las propiedades de los clientes en cuanto a las contraseñas guardadas en Firefox y Thunderbird.

spsudogroup: tabla donde se almacenan las propiedades de los clientes en cuanto a los grupos de administradores.

spsharedfolder: tabla donde se almacenan las propiedades de los clientes en cuanto a las carpetas compartidas.

spdnf: tabla donde se almacenan las propiedades de los clientes en cuanto a sus propiedades generales.

spbios: tabla donde se almacenan las propiedades de los clientes en cuanto al BIOS.

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

spdataprotection: tabla donde se almacenan las propiedades de los clientes en cuanto al uso de programa para la protección de datos.

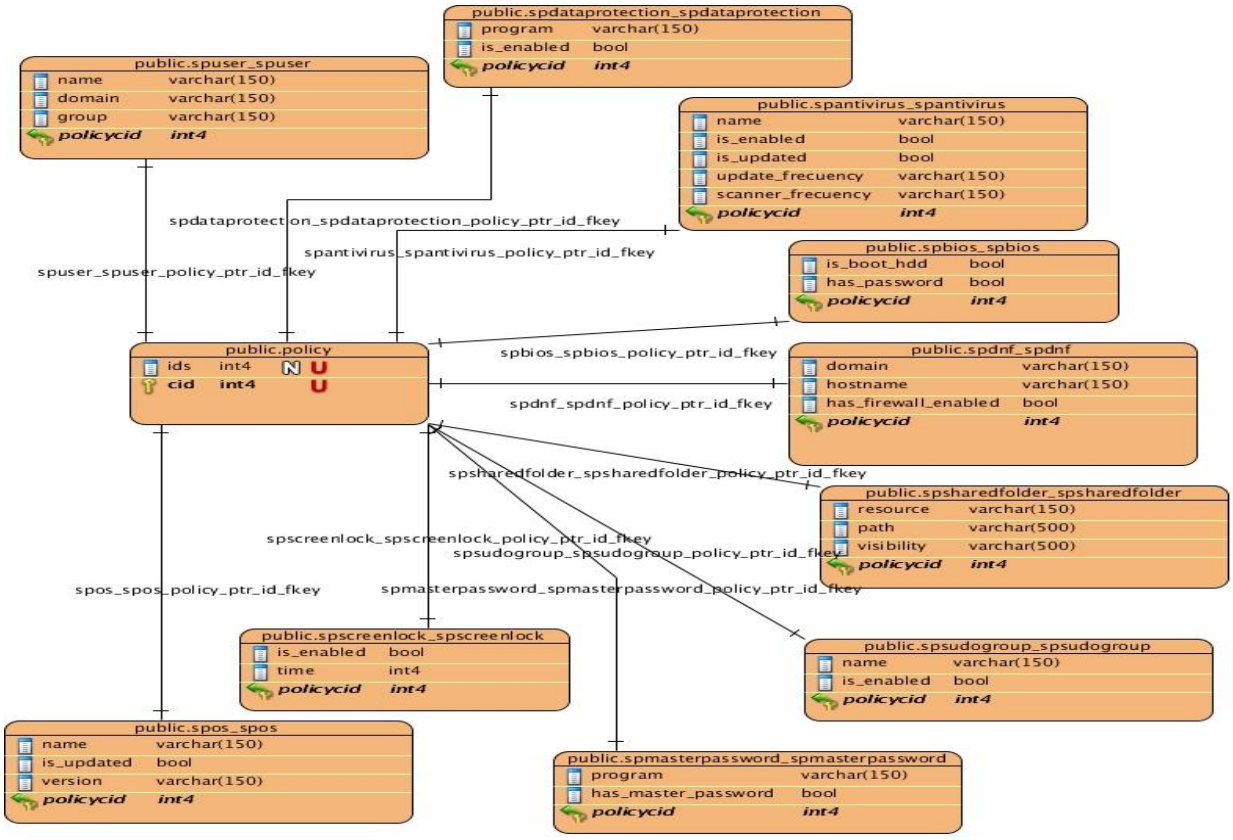


Ilustración 11: Modelo físico de la Base de Datos del cliente.

policy: tabla donde se almacenan los identificadores de las restantes tablas.

spuser: tabla donde se almacenan las propiedades de las cuentas de usuario de la máquina que está siendo inventariada.

spdataprotection: tabla donde se almacenan las propiedades en cuanto al uso de programa para la protección de datos de la máquina que está siendo inventariada.

spantivirus: tabla donde se almacenan las propiedades del uso de antivirus en la máquina que está siendo inventariada.

spbios: tabla donde se almacenan las propiedades del BIOS de la máquina que está siendo inventariada.

spdnf: tabla donde se almacenan las propiedades de la máquina que está siendo inventariada.

spsharedfolder: tabla donde se almacenan las propiedades de las carpetas compartidas en la máquina que está siendo inventariada.

spsudogroup: tabla donde se almacenan las propiedades de los grupos de administradores de la máquina que está siendo inventariada.

spmasterpassword: tabla donde se almacenan las propiedades en cuanto a las contraseñas guardadas en Firefox y Thunderbird de la máquina que está siendo inventariada.

spscreenlock: tabla donde se almacenan las propiedades en cuanto a el bloqueo automático de la pantalla de la máquina que está siendo inventariada.

spos: tabla donde se almacenan las propiedades del sistema operativo de la máquina que está siendo inventariada.

3.9 Tareas de Ingeniería

Las tareas de ingeniería son escritas por los desarrolladores, a partir de las historias de usuario creadas por el cliente, las cuales brindan una mayor información para la implementación. Las tareas de ingeniería se representan mediante tablas, a continuación se muestra una tabla donde se explican las secciones:

Tabla 9: Ejemplo de una Tarea de Ingeniería.

Tarea de Ingeniería	
Número tarea: (Número de la tarea, debe ser consecutivo)	**Número historia de usuario:** (Número de la historia de usuario a la que pertenece la tarea)
Nombre tarea: (Nombre que identifica la tarea)	
Tipo de tarea: (Las tareas se pueden clasificar en: desarrollo, mejora, corrección u otra)	**Puntos estimados:** (Tiempo que se le estimará al desarrollo de la tarea)
Fecha inicio: (Fecha en que inicia el desarrollo de la tarea)	**Fecha fin:** (Fecha en que culmina el desarrollo de la tarea)
Programador Responsable: (Nombre y apellidos del programador responsable)	
Descripción: (Breve descripción de la tarea)	

A continuación se muestran algunas de las tareas de ingeniería más importantes, las restantes se encuentran en el Anexo 3:

Tabla 10: Tarea de Ingeniería # 1: Obtener antivirus instalado en la máquina para Linux.

Tarea de Ingeniería	
Número tarea: 1	**Número historia de usuario:** 1
Nombre tarea: Obtener antivirus instalado en la máquina para Linux.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17-12-2014	**Fecha fin:** 18-12-2014
Programador Responsable: Dania Carmenate, Dagoberto F. Pérez	
Descripción: Al ser ejecutado el gclient, el subplugin spantivirus ejecuta el comando para verificar si el antivirus Kaspersky o el SavUnix se encuentran instalado en la máquina, se obtiene la información y se almacena en la base de datos local.	

Tabla 11: Tarea de Ingeniería # 57: Listar máquinas con las propiedades del antivirus.

Tarea de Ingeniería	
Número tarea: 57	**Número historia de usuario:** 56
Nombre tarea: Listar máquinas con las propiedades del antivirus.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 02-04-2015	**Fecha fin:** 04-04-2015
Programador Responsable: Dania Carmenate, Dagoberto F. Pérez	
Descripción: Al usuario seleccionar en la lista despegable que le brinda el sistema al lado izquierdo la opción Antivirus, el sistema mostrará todas las máquinas que envían sus propiedades a Gserver con las respectivas características que debe tener el antivirus, dígase nombre del antivirus, estado (activo o no activo), si se encuentra actualizado, frecuencia de actualización, frecuencia de escaneo e identificador y localización de cada una de las máquinas.	

Tabla 12: Tarea de Ingeniería # 67: Filtrar búsquedas por las propiedades del antivirus.

Tarea de Ingeniería	
Número tarea: 67	**Número historia de usuario:** 66
Nombre tarea: Filtrar búsquedas por las propiedades del antivirus.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 25-04-2015	**Fecha fin:** 27-04-2015
Programador Responsable: Dania Carmenate, Dagoberto F. Pérez	
Descripción: El usuario en la interfaz de Antivirus selecciona una de las opciones para realizar búsquedas (nombre del antivirus, estado (activo o no activo), si se encuentra actualizado, frecuencia de actualización, frecuencia de escaneo e identificador y localización), el sistema muestra de acuerdo a la selección realizada por el usuario aquellas máquinas que cumplan con la selección.	

3.10 Pruebas

Unos de los pilares de la metodología XP es el proceso de pruebas. XP propone comprobar constantemente cada una de las funcionalidades implementadas. Esto permite aumentar la calidad de los sistemas, reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. La metodología XP divide las pruebas en dos grupos: pruebas unitarias y pruebas de aceptación o funcionales.

3.10.1 Pruebas unitarias

Las pruebas unitarias son las encargadas de verificar el código y son diseñadas por los programadores. Cada uno de los desarrolladores tiene que ir probando constantemente lo que va obteniendo en el transcurso de la implementación de un sistema, para garantizar que las funcionalidades exigidas por el cliente estén siendo implementadas correctamente (32).

Las pruebas unitarias fueron realizadas al culminar la implementación de una funcionalidad, para lo cual se utilizó la librería de Python (unit testing), lo cual permitió comprobar el correcto funcionamiento del sistema.

3.10.1 Pruebas de aceptación

Las pruebas de aceptación también llamadas pruebas del cliente son especificadas por el cliente y son creadas en base a las historias de usuario. Estas son las encargadas de comprobar que las funcionalidades desarrolladas sean las esperadas por el cliente (33).

Las pruebas de aceptación correspondiente a cada una de las funcionalidades del Módulo de Seguridad Informática 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.

A continuación se muestra un caso de prueba correspondiente a una de las funcionalidades del sistema.

Tabla 13: Caso de Prueba Filtrar búsquedas por las propiedades del antivirus.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observacio-nes
El usuario en la interfaz de Sistema Operativo, selecciona la opción de buscar		El sistema muestra una tabla dividida por secciones (identificador, nombre, actualizado,	Satisfactorio.	

por identificador. El usuario introduce el identificador de la máquina y luego selecciona la opción de buscar.		localización) y en ella se lista la máquina que corresponde con el identificador introducido.		
	El usuario en la interfaz de Sistema Operativo, selecciona la opción de buscar por identificador. El usuario con el campo vacío del identificador selecciona la opción buscar.	El sistema muestra una tabla dividida por secciones (identificador, nombre, actualizado, localización) y en ella se listan todas aquellas máquinas que se encuentran en la base de datos.	Satisfactorio	
El usuario en la interfaz de Sistema Operativo, selecciona la opción de buscar por nombre del sistema operativo. El usuario introduce el nombre del sistema y luego selecciona la opción de buscar.		El sistema muestra una tabla dividida por secciones (identificador, nombre, actualizado, localización) y en ella se listan las máquinas que corresponde con el nombre del sistema operativo introducido.	Satisfactorio.	

	El usuario en la interfaz de Sistema Operativo, selecciona la opción de buscar por el nombre del sistema operativo. El usuario con el campo vacío del nombre selecciona la opción buscar.	El sistema muestra una tabla dividida por secciones (identificador, nombre, actualizado, localización) y en ella se listan todas aquellas máquinas que se encuentran en la base de datos.	Satisfactorio	
El usuario en la interfaz de Sistema Operativo, selecciona la opción de buscar por si se encuentra actualizado el sistema operativo. El usuario selecciona la opción (si o no) y luego selecciona la opción de buscar.		El sistema muestra una tabla dividida por secciones (identificador, nombre, actualizado, localización) y en ella se listan las máquinas de acuerdo a la selección realizada por el usuario.	Satisfactorio	
	El usuario en la interfaz de Sistema Operativo,	El sistema muestra una tabla dividida por secciones (identificador, nombre,	Satisfactorio	

	selecciona la opción de buscar por si se encuentra actualizado el sistema operativo. El usuario no realiza la selección y selecciona la opción de buscar.	actualizado, localización) y en ella se listan todas aquellas máquinas que se encuentran en la base de datos.		
El usuario en la interfaz de Sistema Operativo, selecciona la opción de buscar por localización. El usuario introduce la posible localización de la máquina y luego selecciona la opción de buscar.		El sistema muestra una tabla dividida por secciones (identificador, nombre, actualizado, localización) y en ella se listan aquellas máquinas que se encuentren en la localización introducida por el usuario.	Satisfactorio	
	El usuario en la interfaz de Sistema Operativo, selecciona la opción de buscar por localización. El usuario no	El sistema muestra una tabla dividida por secciones (identificador, nombre, actualizado, localización) y en ella se listan todas aquellas máquinas	Satisfactorio	

	introduce la posible localización de la máquina y luego selecciona la opción de buscar.	que se encuentran en la base de datos.		

Al terminar las pruebas de aceptación se obtuvo un total de 19 no conformidades en la primera iteración, de las cuales 17 fueron resueltas y 2 no procedían. En la segunda iteración se obtuvieron 13 no conformidades donde 11 fueron resueltas y 2 no procedían; y en la tercera iteración se encontraron 10 no conformidades y todas fueron resueltas. Durante las 3 iteraciones no quedó ninguna no conformidad por resolver. Los resultados correspondientes a estas pruebas se muestran en la ilustración # 12.

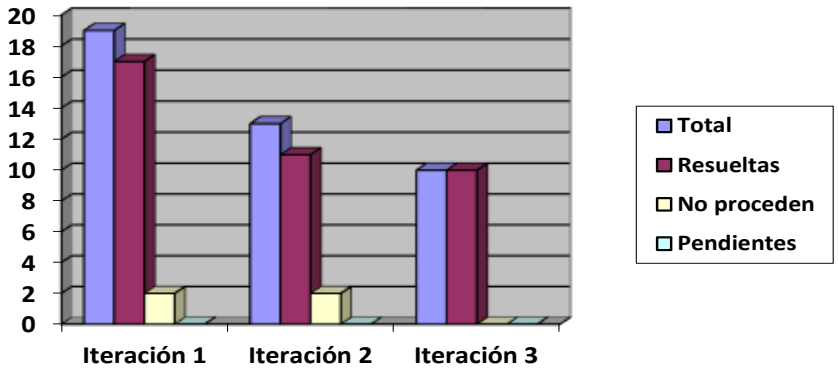


Ilustración 12: Resultados de las pruebas de aceptación.

3.11 Conclusiones

En este capítulo se estudió y se definió la arquitectura cliente servidor, para así diseñar y guiar el proceso de desarrollo del módulo de seguridad informática para la aplicación gadmin del sistema GRHS. Se realizó un análisis de diferentes patrones y posteriormente se seleccionaron los patrones de diseño que se utilizarán para el desarrollo del sistema. Se describieron las tareas de Ingeniería para cada historia de usuario, necesarias para la implementación del sistema. Además

se definieron las tarjetas CRC permitiendo identificar y organizar las clases orientadas a objetos. Se identificaron las clases que persisten en el tiempo para obtener a partir de las mismas el modelo físico de la base de datos. Se llevaron a cabo las pruebas unitarias, permitiendo comprobar el correcto funcionamiento de procedimientos internos del software. Con el uso de casos de prueba se validó que las funciones a nivel de interfaces de usuario fueran operativas.

CONCLUSIONES GENERALES

Una vez finalizada la investigación se arribaron a las siguientes conclusiones:

- ✓ Después de haber analizado las características de los sistemas que automatizan determinados controles de seguridad informática, se concluyó que el sistema que más facilidades brindaba para dar solución al problema planteado era el GRHS.
- ✓ Se realizaron pruebas de aceptación y unitarias con el objetivo de comprobar la correcta implementación de cada una de las funcionalidades y así evidenciar la calidad del sistema desarrollado.
- ✓ Con el desarrollo del plugin de seguridad para la aplicación gclient y el módulo al GRHS se logró obtener un producto que permita a los asesores controlar y monitorear el establecimiento de cada una de las políticas de seguridad en una red de computadoras.

Por todo lo anteriormente expuesto, se concluye que los objetivos propuestos para el presente trabajo se han cumplido satisfactoriamente, poniendo en práctica todas y cada una de las tareas propuestas para el desarrollo del módulo de seguridad informática.

RECOMENDACIONES

Teniendo en cuenta los resultados obtenidos en la realización del presente trabajo de diploma se recomienda:

- Permitir que las políticas se puedan agregar/quitar/modificar sin tener que volver a instalar el cliente a través de actualizaciones o plugin al sistema.
- Permitir al módulo de seguridad informática la determinación del cumplimiento de las políticas de seguridad para cada estación de trabajo.
- Realizar un estudio sobre el impacto, beneficios sociales y económicos que trae consigo el uso de esta solución basada en software libre.

REFERENCIAS

1. Emprendedores UNL. [En línea] Universidad Nacional del Litoral, 2010. [Citado el: 18 de Noviembre de 2014.] http://www.unl.edu.ar/emprendedores/?p=4776.
2. Servicios TIC. [En línea] [Citado el: 18 de Noviembre de 2014.] http://www.serviciostic.com/las-tic/definicion-de-tic.html.
3. Menéndez, Ramiro Váldes. *Resolución 127/2007.* La Habana, Cuba : s.n., 2007. ISSN 1682-7511.
4. Capitulo 1 Aspectos generales de Seguridad Informática. [En línea] [Citado el: 18 de Noviembre de 2014.] http://repositorio.utc.edu.ec/bitstream/27000/536/1/T-UTC-1052%281%29.pdf.
5. bligoo. [En línea] [Citado el: 21 de Noviembre de 2014.] http://bloginformatico.bligoo.com/sistema-informatico.
6. Benítez, Moisés. Gestión Integral. [En línea] [Citado el: 21 de Noviembre de 2014.] http://www.gestionintegral.com.co/wp-content/uploads/2013/05/Pol%C3%ADticas-de-Seguridad-Inform%C3%A1tica-2013-GI.pdf .
7. Comunicación online para todos los públicos. [En línea] [Citado el: 24 de Noviembre de 2014.] http://comunicacionparatodos.wordpress.com/category/glosario/.
8. Perurena, Ing. Raydel Montesino. *Modelo para la gestión automatizada e integrada de controles de seguridad informática.* La Habana, Cuba : s.n., 2012.
9. OpenVAS. [En línea] [Citado el: 24 de Noviembre de 2014.] http://www.openvas.org/.
10. OCS inventory ng. OCS inventory ng. [En línea] [Citado el: 24 de Noviembre de 2014.] http://www.ocsinventory-ng.org/en/.
11. sourceforge. [En línea] [Citado el: 25 de Noviembre de 2014.] http://sourceforge.net/projects/rsyslog/.
12. Snort. [En línea] [Citado el: 25 de Noviembre de 2014.] http://www.snort.org/.
13. Roger, Pressman. Ingeniería de software. Un enfoque práctico. Ingeniería de software. s.l. : 7ma, 2007.

14. InnovalProcess. [En línea] [Citado el: 30 de Noviembre de 2014.] http://www.innovalprocess.com/Temarios/Temario_IngenieriaSoftware.pdf.

15. Guioteca. [En línea] [Citado el: 5 de Diciembre de 2014.] http://www.guioteca.com/internet/%C2%BFque-es-html5-y-que-cambios-introduce/.

16. CSS3 HTML5. [En línea] [Citado el: 5 de Diciembre de 2014.] http://html5.dwebapps.com/que-es-css3/.

17. Aprende a programar. [En línea] [Citado el: 5 de Diciembre de 2014.] 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.

18. Etnassoft. [En línea] [Citado el: 7 de Diciembre de 2014.] http://www.etnassoft.com/biblioteca/algoritmos-y-programacion-con-lenguaje-python/Python.

19. Django. Django. [En línea] [Citado el: 8 de Diciembre de 2014.] http://django.es/.

20. Curos Mexired. [En línea] [Citado el: 8 de Diciembre de 2014.] http://www.mexired.com/blog/que-es-jquery/.

21. Mozilla hispano. Desarrollado con Twitter Bootstrap: Parte 1. [En línea] 8 de Dicimebre de 2014. [Citado el: 10 de Diciembre de 2013.] http://www.mozilla-hispano.org/desarrollando-con-twitter-bootstrap-parte-i/.

22. genbetadev. [En línea] [Citado el: 12 de Diciembre de 2014.] 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. Free Download Manager. Paradigma Visual para UML. [En línea] [Citado el: 12 de Diciembre de 2014.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

24. Genbeta. [En línea] [Citado el: 12 de Diciembre de 2014.] http://www.genbeta.com/herramientas/sublime-text-un-sofisticado-editor-de-codigo-multiplataforma .

25. Muy Linux. [En línea] [Citado el: 12 de Enero de 2015.] http://www.muylinux.com/2012/04/24/nginx-1-2-0-estable-disponible.

26. PostgreSQL. PostgresSQL-es. [En línea] [Citado el: 12 de Enero de 2015.] http://www.postgresql.org.es/sobre_postgresql.

27. Gestión de Estadísticas. [En línea] [Citado el: 20 de Enero de 2015.] https://xpgestionestadisticas.wordpress.com/pagina-principal/.

28. Cervantes, Dr. Humberto. Arquitectura de Software. Arquitectura. [En línea] Abril de 2010. [Citado el: 17 de Febrero de 2015.] http://sg.com.mx/content/view/922.

29. ISG3. [En línea] [Citado el: 25 de Febrero de 2015.] http://isg3.pbworks.com/w/page/7624479/Patrones%20Arquitect%C3%B3nicos.

30. LIBROSWEB. [En línea] [Citado el: 17 de Enero de 2015.] http://librosweb.es/libro/django_1_0/capitulo_5/el_patron_de_diseno_mtv.html.

31. Prácticas de Software. [En línea] [Citado el: 18 de Enero de 2015.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

32. Calidad y Software. [En línea] [Citado el: 20 de Abril de 2015.] http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.

33. globe TESTING. [En línea] [Citado el: 25 de Abril de 2015.] http://www.globetesting.com/glosario/pruebas-de-aceptacion/.

34. *Gestión automatizada e integrada de controles de seguridad informática.* Montesino Perurena, Raydel , Baluja García, Walter y Porvén Rubier, Joelsy . 1, La Habana, Cuba : s.n., 2013, Vol. 34. ISSN 1815-5928.

35. Wiki GRHS. [En línea] [Citado el: 20 de Noviembre de 2014.] http://10.128.50.236/grhs-doc/index.php/GRHS.

BIBLIOGRAFÍA

1. Emprendedores UNL. [En línea] Universidad Nacional del Litoral, 2010. [Citado el: 18 de Noviembre de 2014.] http://www.unl.edu.ar/emprendedores/?p=4776..
2. Servicios TIC. [En línea] [Citado el: 18 de Noviembre de 2014.] http://www.serviciostic.com/las-tic/definicion-de-tic.html.
3. Menéndez, Ramiro Váldes. *Resolución 127/2007.* La Habana, Cuba : s.n., 2007. ISSN 1682-7511.
4. Capitulo 1 Aspectos generales de Seguridad Informática. [En línea] [Citado el: 18 de Noviembre de 2014.] http://repositorio.utc.edu.ec/bitstream/27000/536/1/T-UTC-1052%281%29.pdf.
5. bligoo. [En línea] [Citado el: 21 de Noviembre de 2014.] http://bloginformatico.bligoo.com/sistema-informatico.
6. Benítez, Moisés. Gestión Integral. [En línea] [Citado el: 21 de Noviembre de 2014.] http://www.gestionintegral.com.co/wp-content/uploads/2013/05/Pol%C3%ADticas-de-Seguridad-Inform%C3%A1tica-2013-GI.pdf .
7. Comunicación online para todos los públicos. [En línea] [Citado el: 24 de Noviembre de 2014.] http://comunicacionparatodos.wordpress.com/category/glosario/.
8. Perurena, Ing. Raydel Montesino. *Modelo para la gestión automatizada e integrada de controles de seguridad informática.* La Habana, Cuba : s.n., 2012.
9. OpenVAS. [En línea] [Citado el: 24 de Noviembre de 2014.] http://www.openvas.org/.
10. OCS inventory ng. OCS inventory ng. [En línea] [Citado el: 24 de Noviembre de 2014.] http://www.ocsinventory-ng.org/en/.
11. sourceforge. [En línea] [Citado el: 25 de Noviembre de 2014.] http://sourceforge.net/projects/rsyslog/.
12. Snort. [En línea] [Citado el: 25 de Noviembre de 2014.] http://www.snort.org/.
13. Roger, Pressman. Ingeniería de software. Un enfoque práctico. Ingeniería de software. s.l. : 7ma, 2007.

14. Autores, Colectivo de. *Metodologías Ágiles en el Desarrollo de Software.* [ed.] Emilio A. Sánchez López Patricio Letelier Torres. Alicante, España : s.n., 12 de 11 de 2003.

15. InnovalProcess. [En línea] [Citado el: 30 de Noviembre de 2014.] http://www.innovalprocess.com/Temarios/Temario_IngenieriaSoftware.pdf.

16. Priolo, Sebastián. Programación Extrema. Programación Extrema. [En línea] [Citado el: 12 de Febrero de 2015.] http://www.fcad.uner.edu.ar/jai/6JAI/XP_6JAI.pdf.

17. Guioteca. [En línea] [Citado el: 5 de Diciembre de 2014.] http://www.guioteca.com/internet/%C2%BFque-es-html5-y-que-cambios-introduce/.

18. CSS3 HTML5. [En línea] [Citado el: 5 de Diciembre de 2014.] http://html5.dwebapps.com/que-es-css3/.

19. Aprende a programar. [En línea] [Citado el: 5 de Diciembre de 2014.] 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.

20. Etnassoft. [En línea] [Citado el: 7 de Diciembre de 2014.] http://www.etnassoft.com/biblioteca/algoritmos-y-programacion-con-lenguaje-python/Python.

21. Django. Django. [En línea] [Citado el: 8 de Diciembre de 2014.] http://django.es/.

22. Curos Mexired. [En línea] [Citado el: 8 de Diciembre de 2014.] http://www.mexired.com/blog/que-es-jquery/.

23. Mozilla hispano. Desarrollado con Twitter Bootstrap: Parte 1. [En línea] 8 de Dicimebre de 2014. [Citado el: 10 de Diciembre de 2013.] http://www.mozilla-hispano.org/desarrollando-con-twitter-bootstrap-parte-i/.

24. genbetadev. [En línea] [Citado el: 12 de Diciembre de 2014.] http://www.genbetadev.com/desarrollo-web/backbone-js-el-framework-para-construir-aplicaciones-usando-javascript-siguiendo-el-patron-mvc-alcanza-la-version-1-0.

25. Free Download Manager. Paradigma Visual para UML. [En línea] [Citado el: 12 de Diciembre de 2014.]

http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

26. Genbeta. [En línea] [Citado el: 12 de Diciembre de 2014.] http://www.genbeta.com/herramientas/sublime-text-un-sofisticado-editor-de-codigo-multiplataforma .
27. Muy Linux. [En línea] [Citado el: 12 de Enero de 2015.] http://www.muylinux.com/2012/04/24/nginx-1-2-0-estable-disponible.
28. PostgreSQL. PostgresSQL-es. [En línea] [Citado el: 12 de Enero de 2015.] http://www.postgresql.org.es/sobre_postgresql.
29. Gestión de Estadísticas. [En línea] [Citado el: 20 de Enero de 2015.] https://xpgestionestadisticas.wordpress.com/pagina-principal/.
30. Beas, José Manuel. José Manuel Beas. Ayudo a desarrollar mejor software. Historias de Usuario. [En línea] 23 de mayo de 2011. http://jmbeas.es/guias/historias-de-usuario/.
31. Cervantes, Dr. Humberto. Arquitectura de Software. Arquitectura. [En línea] Abril de 2010. [Citado el: 17 de Febrero de 2015.] http://sg.com.mx/content/view/922.
32. Mis primeros pasos por el mundo de la arquitectura del software. León, Jeimy. Octubre de 2009.
33. ISG3. [En línea] [Citado el: 25 de Febrero de 2015.] http://isg3.pbworks.com/w/page/7624479/Patrones%20Arquitect%C3%B3nicos.
34. LIBROSWEB. [En línea] [Citado el: 17 de Enero de 2015.] http://librosweb.es/libro/django_1_0/capitulo_5/el_patron_de_diseno_mtv.html.
35. Prácticas de Software. [En línea] [Citado el: 18 de Enero de 2015.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.
36. Calidad y Software. [En línea] [Citado el: 20 de Abril de 2015.] http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.
37. globe TESTING. [En línea] [Citado el: 25 de Abril de 2015.] http://www.globetesting.com/glosario/pruebas-de-aceptacion/.

38. autores, Colectivo de. *Guión Visual Paradigm for UML.* 2013.

39. Calabriza, Luis. *Metodología XP.* Uruguay : s.n., 2003.

40. Gutierrez, Demián. *Patrones de Diseno.* Venezuela : s.n., 2010.

41. J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres. *Pruebas del sistema en programación.* Sevilla : s.n.

42. Grau Abalo, Ricardo, Correa Valdés, Cecilia and Rojas Betancur, Mauricio. *METODOLOGÍA DE LA INVESTIGACIÓN.* [Documento] Ibagué : s.n.