Facultad 2

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

Módulo de inventario de servidores virtualizados para el cliente de XILEMA GRHS

Autores: Luis Angel Capestany Placias

Tutores: MSc. Mónica Peña Casanova

Ing. Leydis Rodríguez Zamora

Ing. Ramón Guzmán Alemañy

La Habana, junio de 2019

"Hay que despertar el interés de nuestra juventud para que investigue, para que conozca, para que se entrene, ya que esos conocimientos tienen valor en todos los órdenes"

Declaración de autoría

Declaro ser autor de la presente tesis que tiene por título: Módulo de inventario de servidores virtualizados para el cliente de XILEMA GRHS" y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los _____ días del mes de _____del año 2019.

Luis Angel
Capestany Placias

_____

Firma del Autor

MSc. Mónica Peña Casanova	Ing. Leydis Rodríguez Zamora	Ing. Ramón Guzmán Alemañy
_____	_____	_____
Firma del Tutor	Firma del Tutor	Firma del Tutor

Dedicatoria

Dedico esta tesis a mi familia, a mis padres para los cuales esta tesis representa un gran logro y orgullo, gracias por creer en mi hasta cuándo ni yo mismo lo hacía. Lamentablemente no expreso mis sentimientos hacia ustedes con frecuencia, pero tengan por seguro que si volviese a nacer mi mayor deseo sería que ustedes volviesen a ser mis padres. Los quiero.

A mi abuela Roselia para ti la vida, como mismo tú has sabido darla por mí a lo largo de estos años.

Agradecimientos

Agradecer de antes de todo a mi familia, mis padres que han sido mi ejemplo a seguir, mi hermana que ha sido la persona por la cual me he esforzado para ser mejor, a mi abuela por darme el amor más lindo e incondicional de este mundo. A mi abuelo por estar siempre presente cuidándome desde lo alto.

A mis tíos Olga Lidia y Miguel a mis primos Totica, Naje, y Daniel, ustedes son la mejor familia que alguien podría desear y agradezco a la vida haber tenido la oportunidad de compartir mi niñez cerca de ustedes.

A mis tutores Leydis y Ramón por saber guiarme, por tener paciencia conmigo y por aconsejarme durante todo este curso, sin ustedes este resultado no hubiese sido posible, muchas gracias de todo corazón.

Dicen que cada persona que pasa por tu vida deja su marca de alguna manera y todas ellas ayudan a construir a la persona que eres hoy en día. Gracias a la vida por poner en mi camino tantas buenas personas que me han enseñado tanto a lo largo de los años, ojalá pudiesen estar todos aquí, pero la vida sigue y no espera por nadie, lo entiendo.

Gracias a las primeras personas que conocí en esta universidad, mi gente del 88, Richard, Henry, Yerandy, el Yoe, Gerandi, Titi, Marquitos, ustedes son y serán siempre la familia a la que correr en búsqueda de refugio y soluciones.

Gracias a los juegos conocí maravillosas personas con las que compartí y discutí mucho tiempo, gracias a Yunior, José, Bryan, Raulito, Alain, Dago, Javier, Raydel, a toda mi gente del WoW gracias por tantas alegrías y tantos disgustos.

Si alguien siento que no esté hoy aquí conmigo eres tú Armando, gracias por los años que compartimos, aunque viviésemos la vida fajados y nos dijésemos barbaridades siempre salíamos después juntos de fiesta.

Muchas gracias a mis segundos padres Lidia y Roberto gracias por acogerme en su casa, gracias por tratarme como un hijo, ¡sí que fueron buenos tiempos aquellos que compartimos juntos!

Muchas gracias a mi angelita, tú sabes que ya no hablamos nunca, pero es porque yo soy así, para mi tú eres mi hermana y lo que importa es el sentimiento.

Mi negro Luky gracias mi hermano por los buenos momentos compartidos juntos, nunca voy a olvidar como empezamos a andar juntos de un momento a otro. Supongo que eso es lo que pasa cuando dos personas son muy parecidas.

Muchas gracias a mis padrinos Cuchun y Dennys, gracias por estar siempre hay para mí, gracias por ayudarme y velar por mí y estar dispuestos a correr conmigo ante cualquier situación, doy gracias a la vida y a todos los seres por haberme colocado en su camino.

Pedrito qué te puedo decir, no sé por qué tienes la habilidad de colmarme la paciencia.

A todas las personas que de una forma u otra se relacionan conmigo en la calle en una fiesta, en el comedor, muchas gracias por todo, gracias a todos ustedes yo soy la persona que soy hoy.

Resumen

El presente trabajo propone un módulo para realizar el inventario de Hardware y Software en máquinas virtuales que se encuentren conectadas a una red de ordenadores para el Gestor de Recursos de Hardware y Software (XILEMA GRHS). Este módulo se integró en el cliente de XILEMA GRHS (gClient) al ya existente módulo de inventario, agregándole los plugins para realizar el inventario a las máquinas virtuales tanto en Windows como en GNU/Linux. Permite la obtención de información referente a prestaciones de hardware asignadas a la máquina virtual, softwares instalados, sistema operativo presente en la máquina virtual, así como las políticas de seguridad presentes en la misma. Permite conocer información referente a dispositivos externos conectados a la estación de trabajo, así como el estado del antivirus y no solamente si posee un antivirus, también permite conocer si está actualizado o si la licencia ha caducado. El módulo está basado en una arquitectura cliente–servidor donde el cliente envía la información obtenida de la realización del inventario al servidor donde es almacenada para su posterior visualización por parte de los clientes de XILEMA GRHS. Se realizaron pruebas unitarias y pruebas de integración obteniendo resultados satisfactorios que garantizan la calidad del producto.

Palabras clave: hardware, inventario, máquina virtual, módulo, software

Abstract

The present work proposes a module to perform the Hardware and Software inventory in virtual machines that are connected to a computer network for the Hardware and Software Resource Manager (XILEMA GRHS). This module was integrated in the XILEMA GRHS client (gClient) to the existing inventory module, adding the plugins to inventory virtual machines in both Windows and GNU / Linux. It allows obtaining information regarding hardware features assigned to the virtual machine, installed softwares, operating system present in the virtual machine, as well as security policies present in it. It allows to know information regarding external devices connected to the workstation, as well as the status of the antivirus and not only if it has an antivirus, it also allows to know if it is updated or if the license has expired. The module is based on a client-server architecture where the client sends the information obtained from the inventory to the server where it is stored for later viewing by XILEMA GRHS customers. Unitary tests and integration tests were performed obtaining satisfactory results that guarantee the quality of the product.

Keywords: hardware, inventory, virtual machine, module, software

Índice

Índice de figuras

Índice de tablas

Introducción

Las tecnologías de la Información y las Comunicaciones se encuentran actualmente en constante desarrollo. Dada la necesidad de la sociedad de estar sincronizada con el avance tecnológico, tanto profesionales como estudiantes están motivados a buscar métodos y soluciones para informatizar sus tareas, en el menor tiempo y costo posible.

A pesar de vivir en un mundo donde existen gran cantidad de redes de computadoras, se pueden encontrar una variedad de recursos que no son controlados e incluso desconocidos por sus propietarios o administradores de red(1). Esta situación afecta los resultados de cualquier empresa, debido a que la falta de control en los recursos incide en la economía, así como propicia la pérdida o deterioro de los mismos a los cuales no se les brinda la debida atención y mantenimiento.

El desarrollo informático en la actualidad, ha traído consigo la aparición de herramientas que proporcionan información de hardware o software a través de las redes de computadoras. Estas herramientas son utilizadas para alertar a los administradores de red sobre las modificaciones existentes en sus áreas. Sin embargo, el problema se centra en elegir la más adecuada para cada situación, en cuanto a su licencia, funcionalidades, costo, rendimiento, dependencias, que se ajuste mejor a las necesidades del cliente y que permita un mejor control de los activos informáticos.

En la actualidad se cuenta con computadoras con características de hardware verdaderamente potentes capaces de realizar cálculos y tareas a velocidades muy rápidas. Esto, sin duda alguna, demuestra el desarrollo tecnológico alcanzado en la industria del hardware. En ocasiones se necesitan realizar diferentes tareas o brindar varios servicios desde diferentes sistemas operativos. Ante esta situación la primera respuesta sería comprar más servidores, pero, si se indaga un poco más se puede descubrir que existe la posibilidad de ejecutar varios sistemas operativos a la vez en el mismo servidor con las suficientes prestaciones de hardware para soportarlos.

La Universidad de las Ciencias Informáticas (UCI) es una de las organizaciones cubanas dedicadas al desarrollo de software(3). La misma cuenta con una red de centros productivos que apoyan la informatización de la sociedad cubana. En el centro de Telemática (TLM) se desarrolla el sistema Gestión de Recursos de Hardware y Software (XILEMA GRHS), el cual es un sistema que realiza el inventario de la información de los componentes de hardware y software en una red de computadoras. El mismo realiza el inventario del hardware y del software presentes en el equipo de cómputo. El sistema es capaz de detectar cambios no autorizados en el hardware, configurables según el tipo (hardware o software), componentes, estado (agregado o eliminado) y el nivel (alto, medio o bajo según la clasificación definida por la institución) según las necesidades de la institución y lanzar alertas de correo a los interesados. XILEMA GRHS permite conocer la localización del dispositivo inventariado según el lugar y la subred a la que pertenece. XILEMA GRHS ha sido desplegado en varias empresas, entre estas, en la Universidad de las Ciencias Informáticas con más de 3 500 computadoras reportándose. En esta institución XILEMA GRHS fue incluido entre las políticas de seguridad(4).

Como parte de la utilización de los clientes del sistema Gestor de Recursos de Hardware y Software (XILEMA GRHS) se han detectado algunas oportunidades de mejoras en el software como son, la posibilidad de poder inventariar servidores virtuales y poder determinar el sistema operativo que está ejecutándose, los softwares que está utilizando, así como los servicios que está brindando y las especificaciones de recursos físicos que tiene asignado dicho servidor. De esta forma tener un mejor control sobre los activos informáticos evitando el uso de softwares ilegales o con licencias ilegales. Teniendo en cuenta que hasta el momento no existe ningún tipo de monitoreo por parte de XILEMA GRHS sobre las máquinas virtuales, estas pudiesen no presentar las políticas de seguridad de la empresa para estaciones de trabajo o incluso las que son específicamente para máquinas virtuales, pueden ser fácilmente multiplicadas o incluso abandonadas sin haber sido usadas. La falta de control a los servidores virtualizados puede crear problemas de funcionamiento, ya que las demandas de procesador y espacio en disco pueden frenar

servicios significantes. La proliferación de servidores virtualizados puede crear dificultades de gestión, seguridad e inconsistencia en la red.

Teniendo en cuenta la situación problemática planteada se define como **problema a resolver**: ¿Cómo realizar el inventario de los recursos de hardware y software emulados en servidores virtuales? Teniéndose como **objeto de estudio** inventario de recursos de hardware y software. El **objetivo general** es desarrollar un módulo para el sistema XILEMA GRHS que permita inventariar los recursos de hardware y software emulados en servidores virtuales. El **campo de acción** se enmarca en el inventario de recursos de hardware y software mediante el cliente de GRHS.

Las **preguntas científicas** que guían y orientan el desarrollo del proceso investigativo son las siguientes:

- ¿Cuáles son los referentes teóricos a tener en cuenta para abordar la solución del problema planteado relacionado con el Gestor de recursos de Hardware y Software XILEMA GRHS?
- ¿Qué propuesta de solución se define para implantar en el módulo de inventario para máquinas virtuales, para el cliente de XILEMA GRHS, gClient?
- ¿Cómo desarrollar un módulo para el Gestor de Recursos de Hardware y Software XILEMA GRHS que permita el inventario de hardware y software en las máquinas virtuales que se encuentren activas en una red de ordenadores?
- ¿Cómo se valida el correcto funcionamiento del módulo de inventario para máquinas virtuales del cliente de XILEMA GRHS, gClient?

Para alcanzar el objetivo trazado y dar solución al problema planteado se elaboraron las siguientes **tareas de investigación**:

- Análisis sobre los sistemas similares que permiten el inventariado de servidores virtualizados en una red de computadoras.
- Análisis de los componentes sobre los que está desarrollado XILEMA GRHS.
- Análisis de las herramientas y tecnologías utilizadas para el diseño y posterior desarrollo del módulo.
- Diseño de una propuesta de solución que permita realizar el inventariado de máquinas virtuales en el sistema XILEMA GRHS.
- Implementación de un módulo para el sistema XILEMA GRHS que permita el inventariado de máquinas virtuales.
- Ejecución de pruebas para validar la solución propuesta.

Para el desarrollo del trabajo se tendrán en cuenta diferentes métodos los cuales son de suma importancia durante la investigación:

Métodos teóricos:

Análisis documental: para la revisión bibliográfica, la revisión de las fuentes básicas de información, el estudio de documentación relacionada con XILEMA GRHS, documentación acerca de los sistemas de inventario de hardware y software, entre otros.

Analítico-sintético: se utilizó para simplificar todas las citas, apuntes y datos tomados al respecto del tema tratado en la presente investigación. La revisión bibliográfica se realizó sobre un conjunto de libros, publicaciones y documentos en soporte electrónico, que se encuentran situados en páginas web, artículos científicos, entre otros.

Histórico-Lógico: se realizó un estudio del sistema XILEMA GRHS desde sus inicios hasta la actualidad, permitiendo obtener conocimiento del mismo referente a cómo ha evolucionado, cómo se comporta y las necesidades que tiene en estos momentos.

Métodos empíricos:

Entrevista: Se realizaron entrevistas al personal calificado en el tema de máquinas virtuales y obtención de inventariado de hardware y software, pertenecientes al proyecto XILEMA GRHS, entre los entrevistados: Ramón Guzmán Alemañy líder del proyecto y Fernando Ricardo Romero desarrollador del proyecto. Estas entrevistas permitieron obtener información necesaria, convirtiéndose en una técnica de recopilación.

La investigación consta de 3 capítulos:

Capítulo 1: Fundamentación teórica. En este capítulo se abordan los principales presupuestos teóricos asociados a la investigación, se realiza un estudio de varios sistemas de gestión de inventario utilizados en Cuba y el mundo, se realiza una caracterización del sistema XILEMA GRHS y sus principales deficiencias. Se describen la metodología, herramientas y lenguajes para implementar la propuesta de solución.

Capítulo 2: Análisis y diseño de la solución. En este capítulo se hace un levantamiento de los requisitos funcionales y no funcionales necesarios para el correcto funcionamiento del sistema. Se generan los artefactos según la metodología.

Capítulo 3: Implementación y pruebas. En este capítulo se muestra el modelo de implementación como resultado del diseño anteriormente desarrollado. Se describen las pruebas a realizar, con el objetivo de probar el correcto funcionamiento del módulo para el inventario de servidores virtualizados, así como los resultados de las mismas.

Capítulo I Fundamentación teórica

1.1 Introducción

En el siguiente capítulo se describe la fundamentación teórica del módulo a desarrollar y se realiza un análisis de los principales conceptos asociados al inventariado de máquinas virtuales. Como aspectos esenciales se profundiza en el estudio de los sistemas similares, así como la selección de la metodología y el lenguaje a emplear para dar solución al objetivo general planteado.

1.2 Principales conceptos asociados a la Investigación

Inventario de Hardware y Software

El inventariado de Hardware y Software se centra en el control de los recursos tangibles y no tangibles de cómputo presentes en las organizaciones. El inventario tiene como objetivo fiscalizar, identificar y categorizar los recursos tanto de hardware como de software que dispone y con los que trabaja la organización(5).

Máquinas Virtuales

Una máquina virtual es un software que simula un sistema de computación y puede ejecutar programas como si fuese una computadora real. Este software en un principio fue definido como "un duplicado eficiente y aislado de una máquina física". La acepción del término actualmente incluye a máquinas virtuales que no tienen ninguna equivalencia directa con ningún hardware real(2).

Virtualización de servidores

La virtualización de servidores es una tecnología basada en un software que posibilita la ejecución de varios sistemas operativos diferentes entre sí, como invitados dentro de un

único host[1] del servidor físico. Son las llamadas máquinas virtuales (VMs) que ejecutan en una imitación virtual del hardware del servidor. Es como si los recursos de un servidor físico, por ejemplo, fuesen divididos en diversos servidores virtuales que pueden ser usados con diferentes finalidades(6).

1.3 Sistemas para el inventario de hardware y software

Un sistema de gestión de inventario de hardware y software es una aplicación con soporte de datos, que registra información sobre los activos informáticos en una red de ordenadores(7). Debido al gran aumento del tamaño de las redes de computadoras y a la gran necesidad de mantener un correcto control sobre estos medios se han desarrollado un gran grupo de herramientas para poder llevar a cabo este objetivo. A continuación, se describen varios sistemas similares que se utilizan para el inventariado de hardware y software:

1.3.1 GLPI (Free Computer Equipment Manager por sus siglas en inglés)

GLPI es una herramienta de código abierto que ayuda a manejar y controlar los cambios en la infraestructura informática de manera sencilla, resolver problemas emergentes de manera eficiente y además hace posible el control fiable sobre el presupuesto y gastos que realiza la compañía en infraestructura tecnológica (IT)(8).

Permite el inventario de los activos informáticos, así como una vista en detalles de sus características de hardware, sus conexiones y ubicación en la red. Guarda un historial completo de todas las modificaciones realizadas. Permite la administración de los sistemas operativos brindando información como nombre, versión, edición, licencia. Reconocimiento

[1] El término host o anfitrión se usa en informática para referirse a las computadoras u otros dispositivos (tabletas, móviles, portátiles...) conectados a una red que proveen y utilizan servicios de ella.

de máquinas virtuales, así como información referente a la cantidad de recursos de hardware asignada a la máquina virtual, sistema operativo presente, softwares instalados.

No permite la posibilidad de realizar una exploración través de la red, hay que introducir manualmente todas las computadoras. Se tienen muchas opciones en la configuración, pero la documentación referente no es del todo asequible al usuario y cuesta un poco comprender muchas de sus funcionalidades(9).

1.3.2 Ivanti Discovery

Es una herramienta, que permite disponer de un inventario automático, hardware, software y comunicaciones, así como una gestión inteligente de sus activos TI. Ofrece información detallada de los equipos de su infraestructura informática(10).

Permite el reconocimiento y control de todo el hardware y el software presente en la red informática. Genera informes con toda información del inventario. Descubre y rastrea hardware y software virtuales, soporta VMWare y tecnologías de virtualización de Microsoft, ofrece información respectiva acerca de su sistema operativo, softwares que están instalados y servicios que brindan. Es multiplataforma. Produce un impacto mínimo sobre los recursos de la red(10).

1.3.3 Open Computers and Software Inventory Next Generation

Open Computer and Software Inventory Next Generation (OCS Inventory-NG) es un software libre que permite a los Administradores de TI (Tecnología de Información) gestionar el inventario de sus activos de TI. OCS Inventory-NG recopila información sobre el hardware y software de equipos que hay en la red que ejecutan el programa de cliente OCS ("agente OCS de inventario"). OCS Inventory-NG puede utilizarse para visualizar el inventario a través de una interfaz web. OCS Inventory-NG comprende la posibilidad de implementación de aplicaciones en los equipos de acuerdo a criterios de búsqueda(11).

OCS Inventory-NG permite descubrimiento de hardware y software virtual, genera reportes detallados de inventario de hardware y software tanto físico como virtual. Permite el escaneo de red por medio de IPDiscovery e instalar aplicaciones remotamente, aun así, muestra un bajo consumo de los recursos de red. Interfaz web para su administración.

No posibilita la visualización de versiones de las aplicaciones instaladas, ni la librería de software. No permite el control sobre sistemas IOS[2]. El servidor requiere una instalación previa de los servicios que necesita antes de instalar OCS Inventory-NG(12).

1.3.4 Pandora FMS

Pandora FMS es un software de código abierto que sirve para la monitorización de sistemas, aplicaciones y dispositivos en la red. Permite conocer el estado de cada elemento de un sistema pues dispone de un registro histórico de datos y eventos. No es un sistema como los descritos anteriormente pues su funcionalidad no se centra esencialmente en el inventariado, para Pandora FMS es algo opcional. Su inventario es flexible y dinámico, se puede comprobar remotamente(13).

Es multiplataforma, permite el monitoreo tanto de hardware como de software virtual obteniendo información como última fecha de encendido, cantidad de memoria disponible, cantidad de memoria utilizada, uso del CPU, porcentaje libre del espacio en disco asignado, sistema operativo activo en la máquina virtual, softwares instalados, versiones de las aplicaciones, permite la instalación de aplicaciones, borrar ficheros, así como levantar servicios de manera remota. Envía alertas y notificaciones ante cambios o amenazas que detecte en la infraestructura(13).

1.3.5 Gestor de Recursos de Hardware y Software (XILEMA GRHS)

[2] En inglés iPhone Operating System es un sistema operativo propietario de la empresa Apple Inc. utilizado en dispositivos como teléfonos, tabletas y otros dispositivos como televisiones o reproductores mp4, entre ellos los famosos iPhone, iPad y iPod.

XILEMA GRHS es un sistema desarrollado en la UCI por el centro Telemática el cual permite el monitoreo e inventariado de hardware y software de los activos informáticos conectados a la red. El sistema registra el hardware con que cuenta el equipo de cómputo: Memoria RAM[3], placa base, procesador, dispositivos de almacenamiento, así como periféricos conectados como: teclado, ratón, impresora, escáner, monitor. También guarda información acerca de los elementos del software como nombre de la computadora, dominio, localización, sistema operativo instalado, fabricante y versión del BIOS[4], detecta la existencia de antivirus, usuarios del sistema y controladores instalados. Es capaz de detectar cambios no autorizados en el hardware y lanzar alertas de correo a los interesados. Permite conocer la localización del dispositivo inventariado según el lugar y la subred a la que pertenece(4).

El sistema no dispone de la capacidad de detectar máquinas virtuales ni poder determinar la cantidad de servidores virtualizados que puede contener la misma computadora física, ni el software que está siendo ejecutado en estos servidores virtuales o los servicios que los mismo brindan.

1.3.6 Conclusiones del estudio de sistemas homólogos.

Tabla 1: Resumen del estudio de homólogos

Aplicaciones/Características	Inventario MV	Información detallada del Hardware	Exploración de la red	Información detallada de SW
GLPI	X	X		X

[3] Memoria de Acceso Aleatorio, memoria principal de la computadora, donde residen programas y datos, sobre la que se pueden efectuar operaciones de lectura y escritura.

[4] sistema básico de entrada y salida, es un software que localiza y reconoce todos los dispositivos necesarios para cargar el sistema operativo en la memoria RAM.

Ivanti Discovery	X	X	X	X
OSC Inventory-NG	X	X	X	
Pandora FMS	X	X	X	X
XILEMA GRHS		X	X	X

En la tabla anterior se mencionan los sistemas estudiados y se marca con una X las características que presentan estos sistemas. Se tuvieron en cuenta como características las antes expuestas porque son las funcionalidades que actualmente presentan los sistemas de inventario de hardware y software y algunas de ellas fueron especificadas por el cliente para el desarrollo del módulo.

El estudio realizado arrojó los siguientes resultados:

- Todos los sistemas realizan el inventario a máquinas virtuales excepto XILEMA GRHS.
- Todos los sistemas ofrecen información detallada del hardware.
- Solo GLPI no permite la exploración a través de la red.
- OSC Inventory-NG es el único sistema que no posibilita la información detallada del software.

Todas las herramientas descritas tienen como propósito fundamental realizar inventarios de hardware y software en una red de ordenadores, cada una con sus características específicas. Se puede destacar que las herramientas estudiadas son multiplataformas, sin embargo, no todas pueden realizar el inventario a máquinas virtuales.

De manera general, todas las herramientas estudiadas son semejantes en cuanto a algunos criterios y de ellas fueron tenidos en cuenta varios elementos positivos como, el bajo consumo de recursos de red, la obtención de información detallada de hardware y software,

y la posibilidad de permitir la exploración a través de la red. Teniendo en cuenta los elementos positivos se decide desarrollar un módulo que permita al sistema XILEMA GRHS la detección de hardware y software virtual e incorporarle estas características para así tener una mejor propuesta de solución.

1.4 Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Las técnicas indican cómo debe ser realizada una actividad técnica determinada identificada en la metodología. Combina el empleo de unos modelos o representaciones gráficas junto con el empleo de unos procedimientos detallados(14).

AUP-UCI El Proceso Unificado Ágil o Agile Unified Process (AUP) en inglés, en su variación para la UCI, define cuatro escenarios posibles, en los que la propuesta de solución puede enmarcarse. Permite su adaptación según las características de cada proyecto (equipo de desarrollo, recursos, etc.). Por lo que asegura que el proceso de desarrollo sea configurable y aumentar la calidad del software que se produce al aplicar las buenas prácticas en el proceso de desarrollo. La variación para la UCI de la metodología AUP es flexible y puede ajustarse en dependencia del proyecto y el software a implementar; definiendo tres fases: Inicio, Ejecución y Cierre. Durante el inicio se realizan actividades relacionadas con la planeación, definición del alcance y estimaciones de tiempo, costo y esfuerzo. En la fase de ejecución, se modela el negocio, se obtienen los requisitos, se elabora la arquitectura y diseño; se implementa y libera el producto. En el cierre se analizan los resultados del proyecto y su ejecución y se realizan las actividades formales de cierre del proyecto(15).

Durante la fase de ejecución en la disciplina Requisitos AUP-UCI, existen tres formas de encapsular los requisitos Historias de Usuarios (HU), Casos de uso del sistema (CU), Descripción de requisitos por procesos (DRP) y, a partir del Modelo Conceptual, se definen los escenarios posibles. El desarrollo de la solución que se propone, se enmarca en el Escenario número 2 de la metodología antes definida. Debido a que el proyecto no es

necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma se modularían exclusivamente los conceptos fundamentales del negocio. También se toma este escenario pues como política del centro de desarrollo TLM es el escenario con el que se trabaja y todos sus productos están enmarcados en el mismo.

1.5 Lenguaje de programación, herramientas y tecnologías informáticas

1.5.1 Herramienta CASE

La herramienta CASE (Computer Aided Software Engineering por sus siglas en inglés) permite Modelar los Procesos de Negocios de las organizaciones(16). Permite mejorar el diseño de los sistemas, elaborar los requerimientos de los usuarios y garantiza la eficiencia.

1.5.1.1 Visual Paradigm 8.0

Es una aplicación de modelado que emplea el lenguaje unificado de modelado (UML, por sus siglas en inglés, Unified Modeling Language), esta herramienta soporta el modelado de todos los diagramas UML, además genera documentación del sistema en varios formatos como PDF, HTML y Word, asimismo permite la generación de código a partir de algunos diagramas. La que puede ser utilizada en el proceso de modelado de aplicaciones informáticas que sigan la filosofía de software libre(16).

Mediante el empleo de esta aplicación es posible realizar los procesos de ingeniería tanto inversa como directa, ya que a partir de un modelo relacional es capaz de desplegar todas las clases asociadas a las tablas. Permite el control de versiones y es multiplataforma.

Por lo anteriormente descrito se decide utilizar Visual Paradigm como herramienta CASE para el modelado de la solución.

1.5.1.2 Lenguaje de Modelado BPMN

Business Process Modeling Notation (BPMN por sus siglas en inglés) es una notación gráfica que describe la lógica de los pasos de un proceso de negocio. Esta notación ha sido

especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades(17). BPMN proporciona un lenguaje común para que las partes involucradas puedan comunicar los procesos de forma clara, completa y eficiente. Esta notación se utiliza en el modelado del proceso de negocio del módulo a desarrollar.

1.5.2 Lenguaje de Programación

Un lenguaje de programación es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora(18). Para el desarrollo del módulo de inventario de servidores virtualizados serán utilizadas las mismas tecnologías que utiliza el sistema XILEMA GRHS, empleando como lenguaje del lado del servidor Python 2.7 y como lenguajes del lado del cliente HTML5 y JavaScript. A continuación, se muestra una breve descripción de cada uno de ellos.

1.5.2.1 Python 2.7

Python es un lenguaje de programación multipropósito de alto nivel. Su filosofía de diseño enfatiza la productividad del programador y la legibilidad del código. Tiene un núcleo sintáctico minimalista con unos pocos comandos básicos y simple semántica, pero además tiene una enorme y variada librería estándar, que incluye una Interfaz de Programación de Aplicaciones (API) para muchas de las funciones en el nivel del sistema operativo (SO)(19). Se decidió utilizar Python porque XILEMA GRHS está implementado sobre este lenguaje, además el módulo a implementar se va a integrar a este sistema y debe ser desarrollado bajo sus mismas tecnologías. En el centro de TLM se definió este lenguaje de programación para sus productos, lo que beneficia que cualquier persona pueda darle mantenimiento al sistema o programar nuevas funcionalidades cuando sea necesario.

1.5.2.2 HTML5

HTML5 es la última evolución de la norma que define el lenguaje de marcas de hipertexto (HTML por sus siglas en inglés). Se trata de una nueva versión del lenguaje HTML, con nuevos elementos, atributos y comportamientos. Diseñado para ser utilizado por todo tipo de desarrolladores. Describe la estructura y el contenido semántico de un documento web(20). Se utilizó el lenguaje HTML5 para el diseño de las interfaces del módulo, siendo este el utilizado en el sistema XILEMA GRHS actualmente.

1.5.2.3 JavaScript

JavaScript es un lenguaje ligero e interpretado, orientado a objetos, más conocido como el lenguaje de script para páginas web. Es un lenguaje script multiparadigma, basado en prototipos, dinámico, soporta estilos de programación funcional e imperativa(21). Este lenguaje es utilizado para realizar acciones o eventos de manera que permita visualizar el descubrimiento de cada subred.

1.5.3 Entorno de desarrollo integrado

Un IDE (Entorno de Desarrollo Integrado) es una aplicación de software que proporciona servicios integrales para los programadores informáticos para el desarrollo de software(22). Una IDE normalmente consiste en un editor de código fuente que permite construir herramientas de automatización.

1.5.3.1 PyCharm 2018

PyCharm es un IDE o entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código,

depuración gráfica, integración con VCS [5]/ DVCS [6]y soporte para el desarrollo web con Django, entre otras bondades. PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la Community que es gratuita y orientada a la educación y al desarrollo puro en Python y la Professional, que incluye más características como el soporte a desarrollo web(23). Los desarrolladores utilizan Pycharm para el desarrollo del sistema XILEMA GRHS.

1.5.4 Marco de trabajo

Un marco de trabajo se puede definir como un conjunto de componentes que estructuran un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web(24). Se definieron como marcos de trabajo a utilizar en el desarrollo del módulo de descubrimiento automático de una red de computadoras a Django 1.8 y Jquery 1.10. A continuación se describen cada uno de ellos.

1.5.4.1 Django 1.8

Django es un marco de trabajo web sobre Python que permite desarrollar rápidamente aplicaciones web, incluye el paradigma Modelo Vista Template. Se centra en automatizar todo lo posible(25). Es el marco de trabajo que se utilizó para la implementación del módulo por ser actualmente el que se emplea en el sistema XILEMA GRHS.

1.5.4.2 JQuery 1.10

jQuery es una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar

[5] En español, Sistema de control de versiones, gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.

[6] En español, Sistema de control de versiones distribuido, es un repositorio de código que puede sincronizarse con otros repositorios, pudiendo actuar tanto como cliente o como servidor de otros repositorios.

interacción con la técnica AJAX a páginas web. jQuery es software libre y de código abierto, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio(26).

1.5.5 Sistema Gestor de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad(27). Tiene como objetivo brindar una interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de tres lenguajes: lenguaje de definición de datos, lenguaje de manipulación de datos y lenguaje de consulta.

1.5.5.1 PostgreSQL 9.6

PostgreSQL es la base de datos relacional de código abierto más avanzada del mundo. Distribuida bajo licencia BSD (Berkeley Software Distribution por sus siglas en inglés). PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando(28). Es la que utiliza el sistema XILEMA GRHS en la actualidad.

1.5.5.2 pgAdmin III

PgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Código Abierto. Está diseñado para responder a las necesidades de los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente

para lanzar scripts[7] programados, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor puede hacerse mediante conexión TCP/IP y puede encriptarse mediante SSL[8] para mayor seguridad(29). La interfaz gráfica soporta todas las características de PostgreSQL y hace simple la administración.

1.6 Conclusiones parciales

Con la realización de este capítulo se fundamentaron las definiciones relacionadas con la investigación que ayudaron a un mejor entendimiento de la situación planteada. De los sistemas similares estudiados se determinó que no pueden ser integrados a la plataforma XILEMA GRHS porque sus funcionalidades no cumplen con las necesidades actuales del sistema. Debido a esto se decidió implementar un moduló que permita el inventariado de servidores virtualizados y que cuente con las características fundamentales de los sistemas estudiados. Se definieron las herramientas fundamentales para el desarrollo de dicho módulo y se seleccionó AUP-UCI como la metodología de desarrollo a utilizar. Se decidió utilizar Python 2.7 como lenguaje de programación, PyCharm 2018 como IDE de desarrollo y el marco de trabajo para el desarrollo web Django 1.8, por ser las tecnologías en las que se desarrolló el sistema XILEMA GRHS. Como sistema gestor de bases de datos se utilizó PostgreSQL 9.6 y la notación BPMN para el modelado del proceso de negocio del módulo a implementar.

[7] Archivo de órdenes, archivo de procesamiento por lotes o guion, es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.

[8] Secure Socket Layer en inglés, es un protocolo de seguridad que hace que sus datos viajen de manera íntegra y segura, es decir, la transmisión de los datos entre un servidor y usuario web, y en retroalimentación, es totalmente cifrada o encriptada.

Capítulo II Análisis y diseño de la solución

2.1 Introducción

En este capítulo se abordan los aspectos relacionados con el dominio de la investigación y la propuesta del trabajo del módulo de inventario de servidores virtualizados para el sistema de XILEMA GRHS. Se diseña el diagrama de modelo conceptual. Se identifican los requerimientos funcionales y no funcionales para el correcto funcionamiento. Se elaboran los diagramas de casos de uso del componente y los prototipos de interfaz. Se modela el diagrama de clases del diseño.

2.2 Propuesta de solución

En la propuesta de solución se decide implementar un módulo para el cliente de XILEMA GRHS gClient el cual se instalará en cada una de las máquinas virtuales y permitirá recopilar la información referente a la máquina virtual. Permitirá conocer que sistemas operativos están ejecutándose en estas MVs, que softwares tienen instalados. Identificará las características de hardware asignadas a la máquina virtual, así como todos los dispositivos y periféricos que estén conectadas a la misma. Se obtendrá información respecto al antivirus, cuál es el que se encuentra instalado en la máquina virtual, así como si está actualizado, si su licencia está activa y cuando expira. Se podrá obtener información referente al nombre de la máquina virtual, dominio al que pertenece, dirección IP, máscara de la subred, MAC[9].

[9] Media Access Control, es un identificador de 48 bits, que corresponde de forma única a una tarjeta o dispositivo de red. Se le conoce también como dirección física, y es única para cada dispositivo.

2.3 Modelo conceptual

El modelo conceptual es una representación que se realiza para entender el proyecto donde se está trabajando, cuando no se tiene una visión clara de los procesos que se realizan en el mismo. Se utiliza para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. Contiene conceptos que estarán asociados tanto a su definición natural como al papel que juegan desde el punto de vista informático.

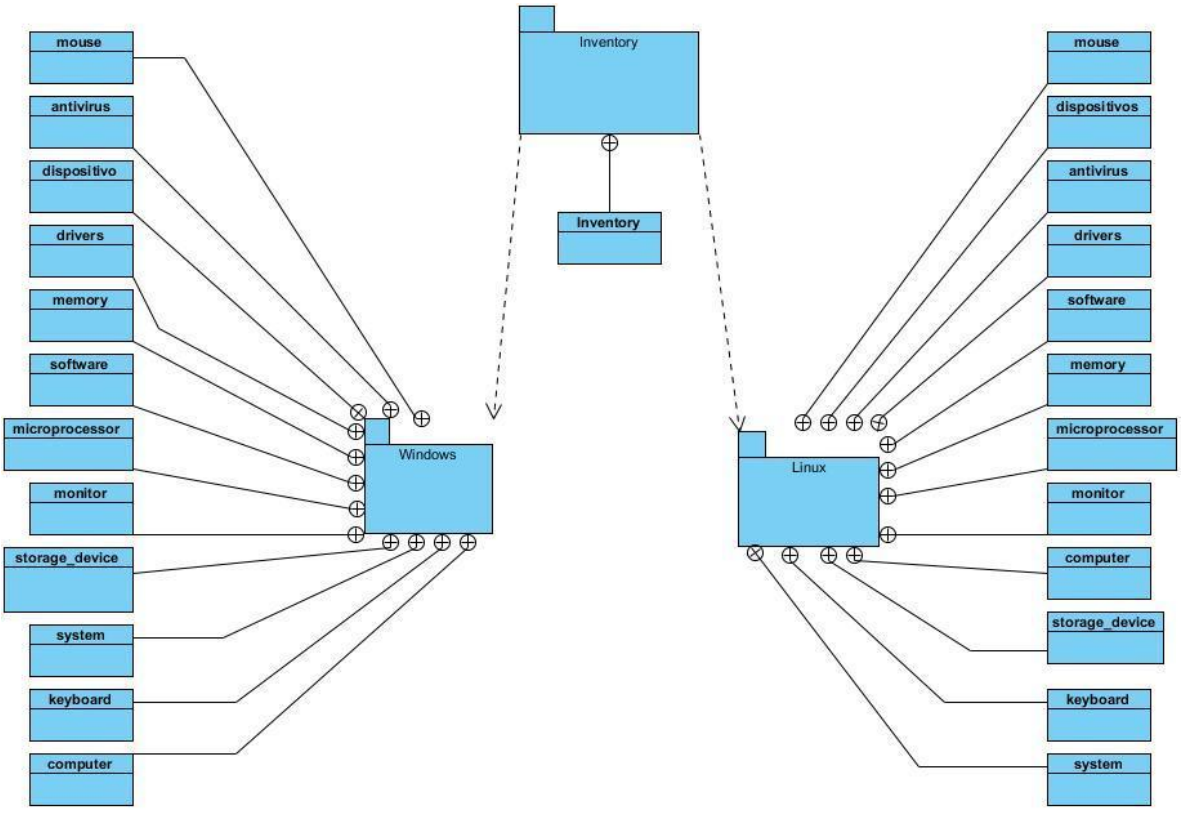

Figura 1 Modelo Conceptual

- Linux: contiene los plugins para la obtención de información referente al inventario para las distribuciones de GNU/Linux.

- Windows: contiene los plugins para la obtención de información referente al inventario para las versiones de Windows.
- **Mouse**: representa el módulo mediante el cual se obtiene la información referente al inventario del mouse conectado a la máquina virtual, tanto para Linux como para Windows.
- **Antivirus**: representa el módulo mediante el cual se obtiene la información referente al inventario del antivirus presente o no en la máquina virtual, tanto en Linux como en Windows.
- **Dispositivos:** representa el módulo mediante el cual se obtiene la información referente al inventario de los dispositivos y periféricos externos conectados a la máquina virtual, tanto para Linux como para Windows.
- **Controladores:** representa el módulo mediante el cual se obtiene la información referente al inventario de los controladores (drivers) presentes en la máquina virtual, ya sea para Windows como para Linux.
- **Software:** representa el módulo mediante el cual se obtiene la información referente al inventario de los softwares instalados en la máquina virtual, tanto para Linux como para Windows.
- **Memoria RAM:** representa el módulo mediante el cual se obtiene la información referente al inventario de la memoria RAM asignada a las máquinas virtuales, tanto para Linux como para Windows.
- **Microprocesador:** representa el módulo mediante el cual se obtiene la información referente al inventario del microprocesador asignado a las máquinas virtuales, ya sea para Linux como para Windows.
- **Monitor:** representa el módulo mediante el cual se obtiene la información referente al inventario del monitor conectado a la máquina virtual, ya sea para Linux como para Windows.
- **Sistema Operativo:** representa el módulo mediante el cual se obtiene la información referente al inventario del sistema operativo sobre el que se ejecuta la máquina virtual.

- **Almacenamiento**: representa el módulo mediante el cual se obtiene la información referente al inventario de Almacenamiento asignado a la máquina virtual, tanto para Linux como para Windows.
- **Teclado**: representa el módulo mediante el cual se obtiene la información referente al inventario del teclado conectado a la máquina virtual, ya sea para Linux como para Windows.
- **Computadora:** representa el módulo mediante el cual se obtiene la información referente al inventario del dominio, mascara de la subred, dirección IP[10], puerta de enlace, dirección MAC de las máquinas virtuales, tanto para Windows como para Linux.

2.3.1 Descripción del Modelo conceptual

El módulo inventario almacena los elementos para la obtención de la información referente al inventario de los componentes de la máquina virtual. Estos elementos están agrupados según el sistema operativo, pues la forma de obtener el inventario varía en dependencia del sistema operativo que se ejecute en la máquina virtual.

2.4 Especificación de los requisitos de software

Los requisitos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requisitos reflejan la necesidad de los clientes de un sistema que ayude a resolver algún determinado problema como el control de un dispositivo, hacer un pedido o encontrar información. El proceso de descubrir, analizar, documentar y verificar esos servicios y restricciones es denominado ingeniería de requisitos(30).

[10] Internet Protocol, es un número que identifica, de manera lógica y jerárquica, a una Interfaz en red de un dispositivo.

2.4.1 Requisitos funcionales

Los requisitos funcionales de un sistema son los que describen las capacidades o funciones que el componente debe cumplir, los servicios que de él se esperan, o los que proveerá(30). A continuación, las funcionalidades que el cliente planteo.

1. **RF1:** Obtener inventario de antivirus en máquinas virtuales en Windows y en Linux.
2. **RF2:** Obtener inventario de dispositivos conectados en máquinas virtuales en Windows y en Linux.
3. **RF3:** Obtener inventario de controladores (drivers) en máquinas virtuales en Windows y en Linux.
4. **RF4:** Obtener inventario de softwares en máquinas virtuales en Windows y en Linux.
5. **RF5:** Obtener inventario de memoria RAM en máquinas virtuales en Windows y en Linux.
6. **RF6:** Obtener inventario de microprocesador en máquinas virtuales en Windows y en Linux.
7. **RF7:** Obtener inventario de monitor en máquinas virtuales en Windows y en Linux.
8. **RF8:** Obtener inventario de sistema operativo en máquinas virtuales en Windows y en Linux.
9. **RF9:** Obtener inventario de dispositivos de almacenamiento en máquinas virtuales en Windows y en Linux.
10. **RF10:** Obtener inventario de teclado en máquinas virtuales en Windows y en Linux.
11. **RF11:** Obtener inventario de mouse en máquinas virtuales en Windows y en Linux.

12. **RF12:** Obtener inventario de computadora en máquinas virtuales en Windows y en Linux.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que definen las restricciones del sistema, son propiedades, cualidades que el sistema debe poseer. Representan las características del producto(30).

Confiabilidad

- RNF. El sistema debe ser capaz de recuperarse ante fallos.
- RNF. El sistema debe ser capaz de prevenir el acceso no autorizado a los datos.

Usabilidad

- RNF. El sistema debe ser fácil de entender para los usuarios.
- RNF. El sistema debe realizar todas las operaciones solicitadas por el usuario.

Hardware

- RNF. Definir la cantidad de memoria RAM, capacidad de disco duro a utilizar en los clientes, el servidor y el servidor de bases de datos.

Portabilidad

- RNF. El sistema debe ser adaptable a diferentes entornos especificados.
- RNF. El sistema debe poderse instalar en un entorno especificado.

Seguridad

- RNF. El sistema debe garantizar la integridad de la información.

- RNF: El sistema debe garantizar el acceso a datos solo de las personas autorizadas.

A continuación, se describe el requisito no funcional de confiabilidad, el resto de descripciones de requisitos se encuentran en el Anexo 1.

Tabla 2: Requisito no funcional de Confiabilidad

Atributo de Calidad	Confiabilidad
Sub-atributos/Sub-características	Tolerancia a fallos
Objetivo	Lograr que el sistema sea capaz de recuperarse ante fallos.
Origen	Interno al sistema
Artefacto	Servicios del sistema (cliente y servidor) / Canales de comunicación
Entorno	Operación normal
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Interrupción de comunicaciones de red en la PC	
	Tratar de conectarse cada cierto tiempo para terminar el proceso de comunicación / Continuar funcionando en modo normal
Medida de respuesta	
NA	

2.a Interrupción de comunicaciones de red en la PC servidor	
	Tratar de conectarse cada cierto tiempo para terminar el proceso de comunicación / Continuar funcionando en modo normal
Medida de respuesta	
NA	
3.a Inactividad del cliente	
	El servidor se encargará de notificar los clientes inactivos / Continuar funcionando en modo normal
Medida de respuesta	
NA	
4.a Ocurrencia de una excepción	
	Se notifica al usuario / Continuar funcionando en modo normal
Medida de respuesta	
NA	

2.5 Casos de uso del sistema

Los casos de uso son un tipo de requerimientos utilizados para especificar funcionalidad, especialmente en sistemas con un alto grado de interacción hombre/máquina(31). En esencia los casos de uso describen los intercambios entre el sistema que se está describiendo y las personas o sistemas externos que interactúan con el primero, por lo tanto, son útiles para describir funcionalidades a varios tipos de usuarios y con muchas interfaces.

A continuación, se describe el actor y el caso de uso del sistema.

Tabla 3: Descripción de autores

Actor	**Descripción del actor**
Sistema	Gestiona la obtención y almacenamiento de la información referente al inventario.

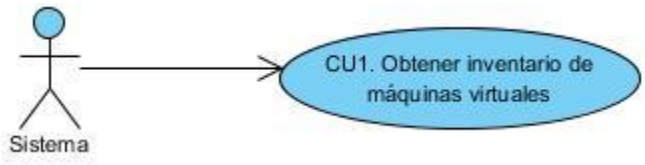

Figura 2 Diagrama de casos de uso del sistema.

CU1: Obtener inventario de máquinas virtuales.

2.5.1 Descripción de casos de uso del sistema

Para una mejor comprensión de las funcionalidades asociadas a cada caso de uso es necesario describirlos. Dicha descripción puede ser elaborada de forma breve o extendida. Una especificación de caso de uso proporciona detalles textuales de un caso de uso. A continuación, se describe detalladamente el caso de uso obtener inventario de antivirus en máquinas virtuales en Windows y en Linux, el resto de las secciones del caso de uso se encuentran en el Anexo 2.

CU 1. Obtener inventario de máquina virtual.

Tabla 4: Descripción CU 1. Obtener inventario de Máquina Virtual.

Objetivo	Obtener información referente al inventario de la Máquina Virtual
Actores	Sistema
Resumen	El caso de uso inicia cuando es instalado el gClient en la máquina virtual, entonces el Sistema comienza a obtener el inventario de dicha máquina y a enviarlo hacia base de datos.
Complejidad	Media
Prioridad	Alta
Precondiciones	El gClient debe estar instalado en la máquina virtual
Postcondiciones	Almacenar la información referente al inventario obtenida de la máquina virtual en cuestión.
Flujo de eventos	
Flujo básico: Obtener inventario de máquina virtual.	

	Actor	**Sistema**
•	El Sistema inicializa el servicio gClient.	
1.		Obtiene la información referente al inventario de la máquina virtual.
•	Envía información obtenida al gServer donde es almacenada en la base de datos.	
•		Termina el caso de uso.

Flujos alternos	
1. **No existe conexión con la base de datos.**	

	Actor	**Sistema**

13.		El Sistema obtiene la información del inventario y comprueba la conexión con la base de datos, hasta que exista conexión entre los dos.
Sección 1: Obtener inventario de antivirus en máquinas virtuales en Windows y en Linux.		
Flujo básico: Obtener inventario de antivirus en máquinas virtuales en Windows y en Linux.		
	Actor	Sistema
1.	El gClient inicializa el método antivirus_mv	
2.		Obtiene información referente al nombre del antivirus, fecha de activación, fecha de expiración, si está activo y si está actualizado.
3.	Envía información obtenida al gServer donde es almacenada en la base de datos.	
4.		Termina el caso de uso.
Flujos alternos		
1. No existe conexión con la base de datos.		
	Actor	**Sistema**
1.		El sistema obtiene la información del inventario y comprueba la conexión con la base de datos, hasta que exista conexión entre los dos.

2.6 Matriz de trazabilidad

La matriz de trazabilidad relaciona dos elementos esenciales para la buena ejecución de las labores de un proyecto: los requisitos establecidos para dicha ejecución y el valor que cada uno de ellos agrega al conjunto del proceso. Es una herramienta clave para la

ingeniería de los proyectos, así como para el seguimiento de los diversos elementos que los componen(32).

A continuación, se muestra la matriz de trazabilidad de los requisitos funcionales con los casos de uso (Requisitos_Casos), la cual establece la relación entre ellos.

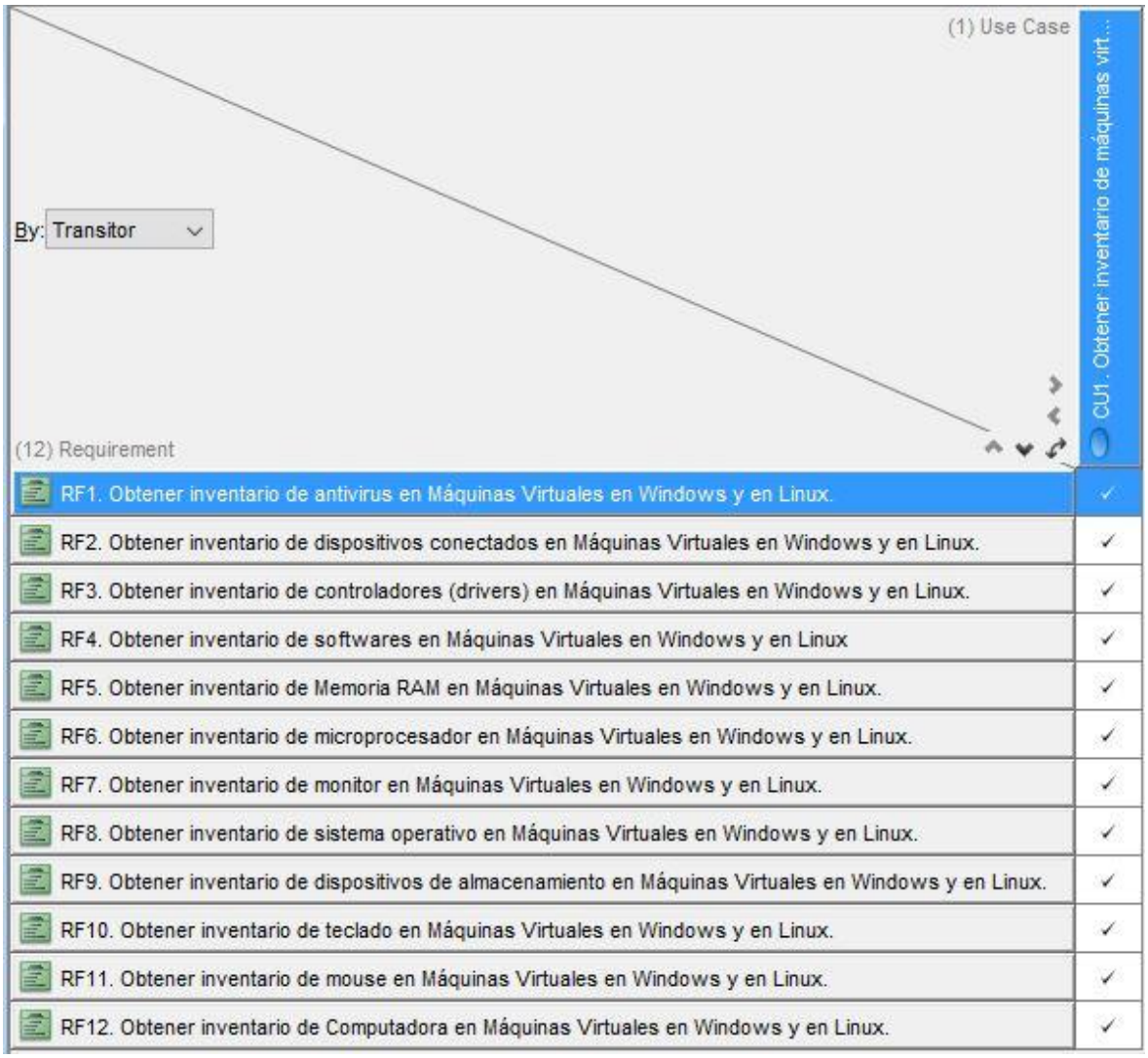

Figura 3 Matriz de Trazabilidad Requisitos - Casos.

2.7 Diagrama de clases del diseño

Los diagramas de clases contienen normalmente los siguientes elementos, clases, interfaces, colaboraciones, relaciones de dependencia, generalización y asociación. Los diagramas pueden también notas, restricciones, paquetes o subsistemas, los cuales se usan para agrupar los elementos de un modelo en partes más grandes(33).

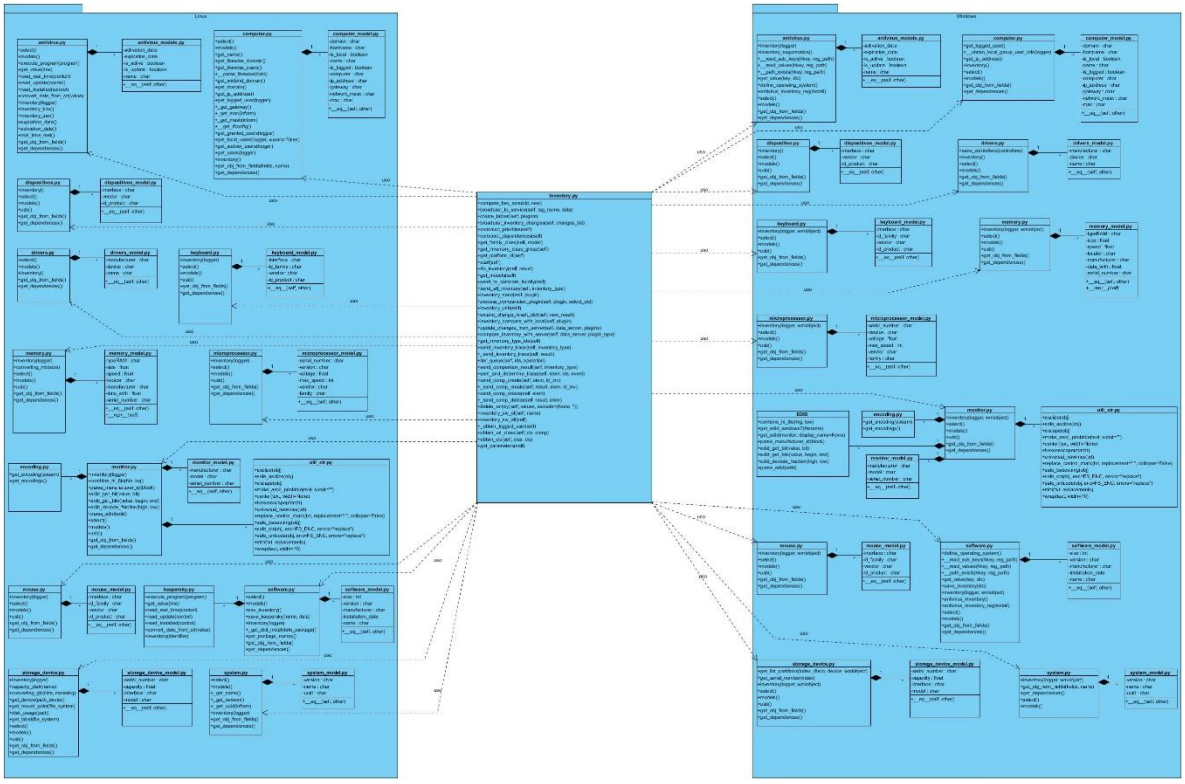

Figura 4 Diagrama de Clases del Diseño.

2.7.1 Descripción de clases

A continuación, se describe la clase Inventory, las demás clases se encuentran en el Anexo 3.

Clase Inventory

Tabla 5: Descripción de la Clase Inventory.py.

Número del módulo	1
Número de la clase	1
Clase	Inventory
Propósito	Clase modelo que contiene los métodos del inventario de máquinas virtuales.

Descripción	
Observaciones	NA

2.8 Diagrama de paquetes

El diagrama del paquete es un diagrama de estructura UML que muestra paquetes y dependencias entre los paquetes (18). Se diseña con el propósito de reflejar la organización que representan los paquetes con los que se trabaja en el desarrollo del sistema. A continuación, se presenta el diagrama de paquetes del sistema con un paquete principal Modulo Inventory el cual contiene anidado el paquete máquina virtual que a su vez contiene

los paquetes hardware y software y cada uno de estos contiene los paquetes Linux y Windows encargados de obtener las características de hardware y de software en dependencia del sistema operativo de la máquina virtual.

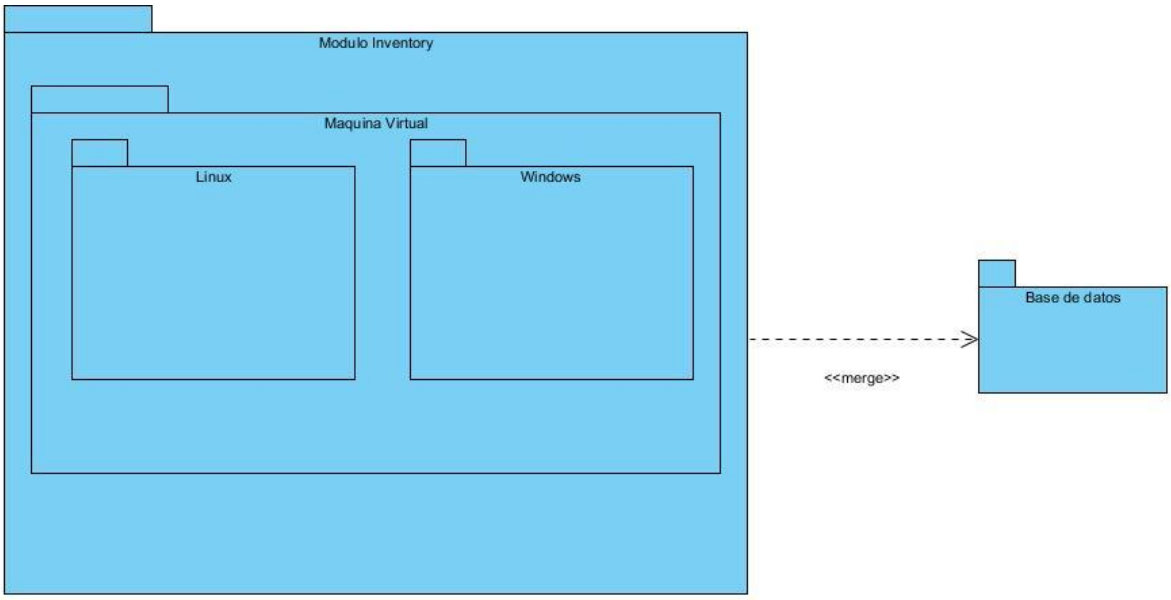

Figura 5 Diagrama de Paquetes.

2.9 Diagrama de despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos. Un nodo es un elemento de hardware o software(34).

A continuación, se muestra el modelo de despliegue propuesto para el componente en el cual los clientes se conectan por el protocolo HTTPS[11] al servidor de la aplicación el cual utilizará una conexión TCP-IP para la comunicación con el servidor de base de datos.

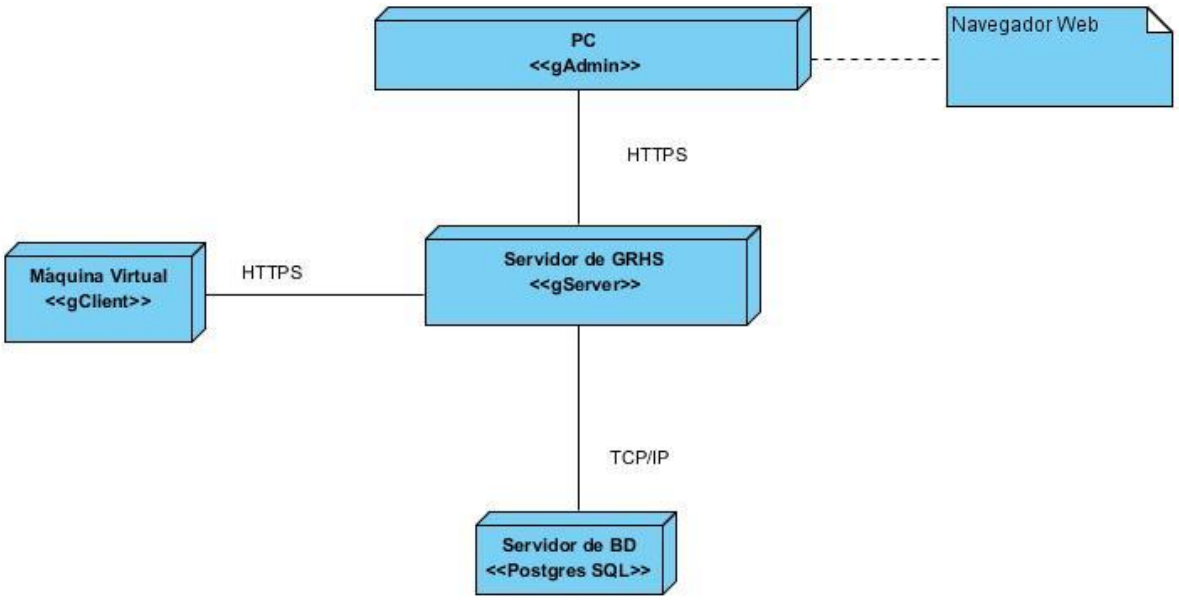

Figura 6 Diagrama de Despliegue.

2.10 Conclusiones del capítulo

Los artefactos obtenidos durante esta etapa del desarrollo de la solución propuesta dan cumplimiento a los requisitos funcionales y no funcionales definidos. Mediante la representación del modelo de dominio fueron identificadas las entidades relacionados con el sistema y las relaciones entre ellas. Los diagramas de clases y el diagrama de paquetes lograron que se obtuvieran de forma concreta y detallada las relaciones existentes entre las clases. El modelo de bases de datos ayudo a entender la relación entre las entidades en

[11] Protocolo seguro de transferencia de hipertexto en español, es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de Hipertexto, es decir, es la versión segura de HTTP.

base de datos. Todos estos artefactos son considerados la entrada principal para las siguientes actividades de implementación y prueba.

Capítulo III Implementación y prueba

3.1 Introducción

Luego de haber definido las herramientas informáticas a utilizar y las funcionalidades del componente, se debe proseguir a la implementación y a la realización de las pruebas al módulo. En este capítulo se definen importantes parámetros para la construcción de la implementación de la solución propuesta. Se explican, a continuación, la arquitectura empleada y el conjunto de patrones de diseños a utilizar para garantizar la calidad del producto. De igual forma se analiza la fase de implementación a partir de los resultados del diseño, describiendo el estado actual del componente. Para la fase de prueba se describen los tipos de pruebas a realizar y los resultados obtenidos de las mismas.

3.2 Arquitectura de software

La arquitectura del software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores. Es definida según la IEEE[12] Estándar 1471-2000 como: la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos, el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. [30]

La arquitectura del software alude a la estructura global del software y a las formas en que esta proporciona la integridad conceptual de un sistema. En su forma más simple, la arquitectura es la estructura jerárquica de los componentes del programa (módulos), la manera en que los componentes interactúan y la estructura de datos que van a utilizar los componentes. [31]

[12] Instituto de Ingeniería Eléctrica y Electrónica, es una asociación mundial de ingenieros dedicada a la normalización y el desarrollo en áreas técnicas

3.2.1 Patrones de arquitectura

La arquitectura de software es una descripción de los subsistemas y componentes de un sistema software, y de las relaciones entre ellos. Los subsistemas y componentes son especificados en diferentes vistas para mostrar las propiedades funcionales y no funcionales relevantes del sistema. La arquitectura de software es un artefacto; es el resultado de la actividad de diseño de software. (35)

Un patrón describe un problema que ocurre una y otra vez en el entorno y describe también el núcleo de la solución al problema, de forma que puede reutilizarse continuamente. Los patrones de arquitectura expresan los esquemas de organización estructural fundamental para sistemas software. Proveen de un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y guías para la organización de las relaciones entre ellos.(35)

3.2.2 Arquitectura cliente-servidor

El sistema XILEMA GRHS cuenta con una arquitectura cliente-servidor, como se muestra en la figura 8, el cual permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma. En la arquitectura cliente servidor, el cliente (gClient) envía un mensaje solicitando un determinado servicio al servidor (gServer), o lo que es lo mismo le hace una petición, y este envía uno o varios mensajes con la respuesta (provee el servicio).

Ventajas de la arquitectura Cliente-Servidor.

- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basada en el concepto de "servicio", que se establece entre clientes y servidores.

- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.

- Los clientes corresponden a procesos activos en cuanto a que son estos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.

- El ambiente es heterogéneo. La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre la misma. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.

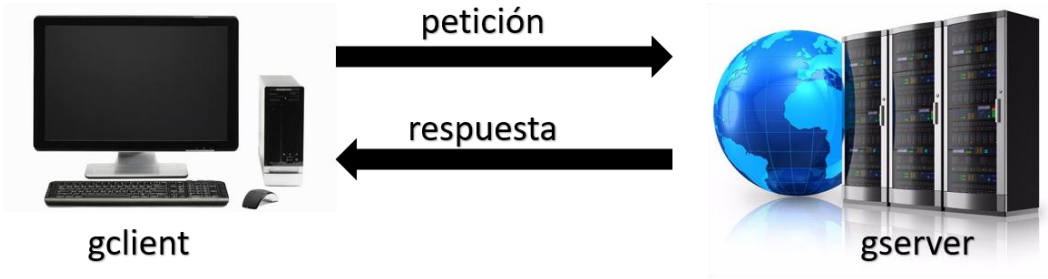

Figura 7 Patrón arquitectónico Cliente-Servidor

3.2.3 Patrón arquitectónico Modelo Plantilla Vista

Django se basa en el Modelo Plantilla Vista (MPV), que es una modificación del Modelo Vista Controlador (MVC). El controlador pasa a ser la vista y la vista pasa a denominarse plantilla. En Django, una vista describe los datos que se ofrecen al usuario, pero no necesariamente su aspecto. Una vista habitualmente delega los datos a una plantilla que describe la forma de presentarlos. El modelo es el encargado de las consultas a la base de datos. [33]

- Modelo (Model): Es la parte de acceso a la base de datos, es manejada por la capa de la base de datos Django.

- Plantilla (Template): Es la parte que contiene las decisiones relacionadas con la presentación, es el modo en el que algunas cosas son mostradas sobre una página web u otro tipo de documento y son los componentes que muestran la interfaz de usuario del sistema.

- Vista (View): Es la parte lógica del negocio que contiene la lógica que accede al modelo y la relaciona con la plantilla apropiada y es el componente que administra y controla la interacción del usuario.

3.3 Patrones GRASP

Los patrones General Responsibility Assignment Patterns (GRASP) describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones, lo que constituye una de las tareas fundamentales en el diseño de un software. También se les conoce como buenas prácticas de programación, y muchas veces son utilizados sin que el diseñador o programador tenga conciencia de ello(36). Los patrones GRASP que se utilizan son:

- Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos (una tarea muy común). La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. La ventaja de utilizar este patrón es eliminar la dependencia entre las clases logrando una mayor sostenibilidad y reutilización(36). Se evidencia en casi todas las clases como son monitor_mv, memory_mv, teclado_mv, antivirus_mv, programas_mv y el resto de clases que obtienen y guardan información de inventario en base de datos.

- Bajo acoplamiento: es un principio evaluativo que aplica un diseñador mientras evalúa todas las decisiones de diseño y soporta clases más independientes(36). Se evidencia en las clases utilizadas de manera que al realizar un cambio en una de las clases no se afecten las demás por esto la estructura de agrupar las clases

según el sistema operativo y dividirlas según el dispositivo o parte del inventario en específico sobre la cual interactúa.

- Alta cohesión: es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme(36). Dentro del sistema, este patrón fue utilizado en las acciones, las cuales contienen varias funcionalidades que están relacionadas.

- Experto: este patrón se utiliza en la asignación de responsabilidades a las clases que poseen la información necesaria para llevar a cabo una tarea específica(36). Se encuentra presente en las clases modelo Inventory.

3.4 Patrones GoF

Los patrones GOF presentan tres tipos de clasificación, las cuales son utilizadas en dependencia del propósito que se quiere alcanzar, estas son:

Patrones de Creación: Tratan la creación de instancias, y se abstraen a la forma en que los objetos son creados, de manera que permite tratar las clases a implementarse de forma genérica, y sin considerar las clases que serán desarrolladas ni la forma en que se hará (37).

- Patrones Estructurales: Tratan la relación entre clases, la combinación clases y la formación de estructuras más complejas(37).

- Patrones de Comportamiento: Tratan la interacción y la cooperación entre clases u objetos(37).

En el diseño del módulo fueron utilizados los patrones GOF explicados a continuación:

Patrón de Creación

- Singleton

Es un modelo que garantiza que solo exista una instancia y que todos puedan acceder a ella. Para esto en lugar de tener una variable global, la instancia se almacena en un atributo estático de la clase (37).

Las clases model implementan el patrón Singleton, las cuales construyen un modelo y solo hacen una instancia del mismo. Permite a la clase Inventory definir solo una instancia de cada objeto. Brinda un mayor control especialmente sobre el número de instancias. En el módulo se evidencia este patrón en cada componente de inventario.

Patrón de Comportamiento

- Observador (en inglés Observer)

Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y se actualicen automáticamente todos los objetos que dependen de él. Describe cómo establecer una consistencia entre objetos relacionados, sin hacer a las clases fuertemente acopladas ya que eso reduciría su reutilización (37).

El patrón Observador se emplea en el componente Inventory que es el encargado de realizar las acciones para la gestión del inventario.

3.5 Estándar de codificación

Como parte de la adopción de las buenas prácticas en la construcción de la solución planteada, se decide utilizar el estándar o estilo de programación de codificación del lenguaje Python: PEP 8 (en inglés, Python Enhancement Proposal). La utilización de este estándar asegura que el código exprese claramente el propósito del mismo y agilice el proceso de refactorización, que sea legible, entendible y que refleje un estilo armonioso(38).

El estilo de programación PEP 8 define esencialmente:

- Empleo de cuatro espacios para indentar.

- Limitación de los tamaños de línea a 79 caracteres como máximo, por ejemplo:

```
            model = elemento.get('ID_MODEL','')
            name = elemento.get('DEVNAME','')
            capacity = capacity_disk(name)
            device = StorageDevice.create(cid=str(uuid4()), serial_number=serial_number,
                                          capacity=capacity, interface=interface, model=model, name=name)
            list_storage_device.append(device)
```

Figura 8 PEP 8 Máximo de caracteres por línea

- Se debe separar las funciones de nivel superior y las clases con dos líneas en blanco, mientras que los métodos dentro de las clases se separan con una sola línea. Se permite la utilización de líneas en blanco dentro de las funciones para separar bloques que guardan cierta correlación lógica, por ejemplo:

```
PARAM = "-W -f='${Package}<->${Installed-Size}<->${Version}<->${Maintainer}\n'"


def select():
    """This method gets installed software instances in database"""
    data = [an for an in Antivirus.select().execute()]
    data.extend([p for p in Program.select().execute()])
    return data
```

Figura 9 PEP 8 Separación entre funciones de nivel superior y clases.

- Las sentencias de import deben estar separadas una en cada línea, por ejemplo:

```
import re
import os
import commands
from uuid import uuid4
from kaspersky import inventory as kaspersky_inventory
from software_models import Program
from software_models import Antivirus
```

Figura 10 PEP 8 Sentencias de import, separadas en líneas.

- Utilizar espacios alrededor de los operadores aritméticos.

- No utilizar espacios alrededor del signo igual cuando se encuentre en un listado de argumentos de una función.

- No se deben incluir comentarios obvios.

- No se deben comparar booleanos mediante “==”.

- Las clases deben nombrarse utilizando la convención “CapWords” (palabras que comienzan con mayúscula). Las clases para uso interno deben tener un guion bajo como prefijo, por ejemplo:

Figura 11 PEP 8 Nombramiento "CapWords" para las clases.

- Las funciones deben nombrarse utilizando la convención “lower_case” (todo en minúscula).

- Siempre se utiliza self, para el primer argumento de los métodos de instancia, excepto para los métodos definidos como estáticos.

3.6 Pruebas de software

Las pruebas son un conjunto de actividades que se planean con anticipado y se realizan de manera sistemática, por lo que se debe definir una plantilla para las pruebas de software que se realizarán, la cual se basa en un conjunto de pasos y procedimientos en que se puedan incluir técnicas y métodos específicos del diseño de casos de prueba.

Se definió una estrategia con niveles, tipos de pruebas, métodos y técnicas. Teniendo en cuenta en este análisis todos aquellos elementos que influyen en el desarrollo del sistema. La estrategia seleccionada permitió solucionar errores que presentaba el módulo y perfeccionar de forma satisfactoria la solución implementada.

Dentro de los niveles generales de pruebas se encuentran:

- Prueba de Desarrollador
- Prueba Independiente
- Prueba de Unidad
- Prueba de Integración
- Prueba de Sistema
- Prueba de Aceptación

Para elegir el nivel de prueba a utilizar se tuvo en cuenta las características de los mismos. Dentro de estos niveles de pruebas la estrategia elaborada hace énfasis en el nivel de Prueba de Desarrollador, debido a que estas pruebas son llevadas a cabo por el equipo de desarrollo y en el nivel de Prueba de Integración, llevada a cabo en la combinación de la propuesta solución con el cliente de XILEMA GRHS.

Tabla 6: Estrategia de pruebas

Niveles de Pruebas	Tipos de Pruebas	Métodos
Integración	Funcionalidad Función	Caja Negra
Desarrollador	Funcionalidad Función	Caja Blanca

3.6.1 Niveles de Pruebas

Pruebas de Integración: es la fase de la prueba de software en la cual se combinan los diferentes módulos y son probados como un grupo. Con estas pruebas se asegura que los componentes del modelo de implementación funcionen de manera correcta, cuando se combinan para ejecutar todos los CU que conforman las funcionalidades con las cuales debe contar el sistema.

Pruebas del Desarrollador: son las pruebas realizadas durante el desarrollo de la aplicación para verificar el funcionamiento correcto de lo implementado, y son llevadas a cabo por los desarrolladores.(39)

3.6.2 Tipos de Pruebas

Funcionalidad Función: Pruebas fijando su atención en la validación de las funciones, métodos, servicios, CU.

Pruebas de Caja Blanca

Se denomina cajas blancas al tipo de pruebas de software que es realizada sobre las funciones internas de un módulo determinado, es decir son las que tratan el código directamente no verifican si este trabaja o no correctamente, solo se encarga de revisar estructuras determinadas. Estas se llevan a cabo en primer lugar, sobre un módulo concreto.(39)

Pruebas PyUnit

PyUnit es una versión en lenguaje Python de JUnit, de Kent Beck y Erich Gamma. JUnit es, a su vez, una versión de Java del marco de prueba Smalltalk de Kent. El mismo utiliza el módulo unittest para la implementación de las pruebas. Unittest admite la automatización de pruebas, el uso compartido de la configuración y el código de apagado para pruebas, la agregación de pruebas en colecciones y la independencia de las pruebas del marco de trabajo de informes(40). Al finalizar cada funcionalidad se aplicó una prueba unitaria utilizando el módulo unittest.

Al aplicarle la prueba al componente, en la primera iteración, arrojo un resultado de 8 no conformidades en un tiempo de ejecución de 0.130 segundos como se puede apreciar en la imagen.

```
Ran 24 tests in 0.130s

FAILED (failures=4, errors=8)

Process finished with exit code 1
```

Figura 12 Resultado de la primera iteración de las pruebas unitarias.

Las 8 no conformidades son producto de cuando se intentaba obtener el listado de documentos y carpetas compartidas en la máquina virtual como parte del apartado de las políticas de seguridad; luego de la primera iteración se corrigieron estos errores y se realizó otra iteración de pruebas obteniendo como resultado ninguna no conformidad y el tiempo de ejecución aumentó a 0.121 segundos.

```
Ran 24 tests in 0.121s

OK

Process finished with exit code 1
```

Figura 13 Resultado de la segunda iteración de las pruebas unitarias.

Pruebas de Caja Negra

Las pruebas de caja negra son aquellos elementos que se estudian desde el punto de vista de las entradas que reciben, y las respuestas que provee sin tener en cuenta el funcionamiento interno. Esta prueba examina algunos aspectos del modelo, fundamentalmente del sistema, sin tener mucho en cuenta la estructura interna del software. Se centran principalmente en los requisitos funcionales del software. Permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos(41). A

continuación, se muestra la iteración 1 de las pruebas realizadas, el resto de las iteraciones se pueden consultar en el Anexo 5.

Iteración #1 Integración de los Módulos de Antivirus y el gClient.

Tabla 7: Integración de los Módulos de Antivirus y el gClient.

Módulo actual	Módulo integrado	Funcionalidad	Condiciones de ejecución	Escenario de prueba	Resultado previsto
gClient	Inventario de máquinas virtuales	Realizar inventario	El cliente de XILEMA GRHS debe estar instalado en la máquina virtual.	Obtener nombre del antivirus.	Obtener el nombre del antivirus instalado en la máquina virtual
				Obtener fecha de activación.	Obtener fecha de activación del antivirus
				Obtener fecha de expiración.	Obtener fecha de expiración del antivirus.
				Obtener si está activo.	Obtener información del antivirus referente a si está activo.
				Obtener si está actualizado.	Obtener información del antivirus referente a si está actualizado.

3.6.3 Entornos de Pruebas

Computadora Servidor

Tabla 8: Características de la computadora servidor donde se realizan las pruebas

Hardware	Especificaciones
Procesador	I5-4460
Memoria	2GB
Tarjeta Madre	Asus H81M-CT
Ancho de Banda	100 Mbps
Sistema Operativo	CentOS7

Computadora Cliente para los sistemas operativos GNU/Linux y Windows

Tabla 9: Características de la computadora cliente donde se realizan las pruebas.

Hardware	Especificaciones
Procesador	I3-4400
Memoria	2GB
Tarjeta Madre	Fujitsu LifeBook E554
Ancho de Banda	100 Mbps
Sistema Operativo	Nova 5

3.6.4 Resultado de las pruebas

Las pruebas realizadas a la solución desarrollada se dividieron en tres iteraciones. La primera iteración fue probada detalladamente, detectando una serie de no conformidades entre las que se destacan problemas a la hora de detectar si el antivirus está actualizado, problemas con la obtención de la frecuencia de la memoria RAM, la información del monitor

duplicada, contando con un total de 12 no conformidades. Para las pruebas de la segunda iteración, las no conformidades detectadas anteriormente habían sido subsanadas, pero se detectaron 4 nuevas no conformidades entre la que destaca la obtención de información de la placa base. Para la última iteración se resolvieron todas las no conformidades anteriormente detectadas y no fue detectada ninguna nueva no conformidad. Así resulta un módulo funcional con las características descritas en los requerimientos del sistema, cumpliendo así, con el objetivo propuesto.

Figura 14 Resultado de las iteraciones de las pruebas realizadas.

3.7 Conclusiones parciales

En el presente capítulo se realizó la descripción de la implementación del componente para el módulo a través del diagrama de despliegue, el cual facilitó mostrar la disposición de las particiones físicas del sistema y la asignación de los componentes de software a estas particiones. Se realizaron pruebas unitarias y de integración con un total de 16 no conformidades encontradas en 3 iteraciones de pruebas, las cuales fueron resueltas en su totalidad, logrando tener una solución funcional.

Conclusiones

El resultado principal de este trabajo de investigación es la obtención de un módulo capaz de realizar el inventario a máquinas virtuales para ser integrado al cliente del sistema Gestor de Recursos de Hardware y Software. Se cuenta con una documentación completa y bien explicada que se fue fundamentando durante las fases del diseño y de implementación. Con el propósito de darle cumplimiento al objetivo general y basado en la problemática expuesta, se llevaron a cabo satisfactoriamente cada una de las taras que fueron perfiladas al comienzo de la investigación. Se realizó un estudio de las tendencias actuales en el marco de los inventarios de hardware y software, identificando entre las herramientas estudiadas algunas que son capaces de inventariar máquinas virtuales, sin embargo, se encontraron otras características en estas aplicaciones por lo cual fueron descartadas como propuesta de solución.

- La utilización de AUP-UCI como metodología de desarrollo permitió guiar el proceso de desarrollo de software, ya que genera abundante documentación y artefactos necesarios para validar la solución.
- Se definieron los requisitos funcionales y no funcionales de acuerdo a las necesidades del sistema.
- Fueron implementados los componentes para la obtención de información referente al sistema operativo presente en la máquina virtual.
- Se implementaron componentes para la obtención de las especificaciones de hardware asignadas por el propietario a la máquina virtual.
- Se implementaron componentes para la obtención de información referente a los softwares, antivirus y políticas de seguridad presentes en la máquina virtual.
- Fueron realizadas pruebas internas del módulo, obteniendo resultados satisfactorios.

Recomendaciones

En función del constante proceso de mejora y evolución que es inherente a todo sistema de software se recomienda lo siguiente:

- Ser capaz de detectar una máquina virtual antes de que tenga instalado el cliente de XILEMA GRHS.
- Que el sistema XILEMA GRHS sea capaz de detectar las máquinas virtuales que estén funcionales en una computadora, de esta forma tener conocimiento de las máquinas virtuales que son creadas, aunque no tengan el cliente de XILEMA GRHS para máquinas virtuales instalado.

Referencias bibliográficas

1. La importancia de un inventario actualizado - Aranda Software. In: [online]. [Accessed 3 junio 2019]. Available from: https://arandasoft.com/la-importancia-de-un-inventario-actualizado/.

2. What is a Virtual Machine (VM)? - Definition from Techopedia. In: [online]. [Accessed 3 junio 2019]. Available from: https://www.techopedia.com/definition/4805/virtual-machine-vm.

3. Misión | Universidad de las Ciencias Informáticas. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.uci.cu/universidad/mision.

4. GESTOR DE RECURSOS DE HARDWARE Y SOFTWARE. In: [online]. [Accessed 23 enero 2019]. Available from: https://studylib.es/doc/7113174/gestor-de-recursos-de-hardware-y-software.

5. Inventario de hardware y software - NetGrid. In: [online]. [Accessed 23 enero 2019]. Available from: https://www.netgrid.com.co/inventario-de-hardware-y-software/.

6. ¿Qué Es La Virtualización y Cuáles Son Sus Beneficios? In: Información práctica sobre Redes, Linux, Seguridad y Hacking para profesionales de TI. Capacity Academy [online]. 7 agosto 2012. [Accessed 3 junio 2019]. Available from: http://blog.capacityacademy.com/2012/08/07/que-es-la-virtualizacion-y-cuales-son-sus-beneficios/.

7. Hardware and Software Inventory | ManageEngine Desktop Central. In: [online]. [Accessed 3 junio 2019]. Available from: https://www.manageengine.com/products/desktop-central/help/inventory/hardware_software_inventory.html.

8. GLPI: Herramienta para la gestión de un inventario. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.coriaweb.hosting/glpi-herramienta-la-gestion-inventario/.

9. GLPI Gestión de inventario de hardware; ¿Cuál es GLPI; La instalación; Introducción de datos; Ventajas y desventajas. In: [online]. [Accessed 20 mayo 2019]. Available from: http://imparatudos.com/article/glpi-gestin-de-inventario-de-hardware.

10. Discovery | Ivanti. In: [online]. [Accessed 23 enero 2019]. Available from: https://www.ivanti.com/solutions/it-service-management/discovery.

11. Automatiza la gestión de tus equipos con OCS. In: [online]. [Accessed 20 mayo 2019]. Available from: https://hipertextual.com/archivo/2010/09/automatiza-la-gestion-de-tus-equipos-con-ocs/.

12. MANAGERS, Net. Administración de recursos IT: OCS-Inventory NG & GLPI. In: Net Managers [online]. [Accessed 13 noviembre 2018]. Available from: http://www.netmanagers.com.ar/stories/ocs_glpi/.

13. Resumen de funcionalidades de Pandora FMS. In: Pandora FMS Community [online]. [Accessed 23 enero 2019]. Available from: https://pandorafms.org/es/producto/funcionalidades-monitorizacion/.

14. Metodologías de desarrollo de Software - EcuRed. In: [online]. [Accessed 23 enero 2019]. Available from: https://www.ecured.cu/Metodologias_de_desarrollo_de_Software.

15. Metodología UCI. S.l. [no date].

16. CASE: Visual Paradigm Community Edition - UML Diagrams and ERD tools - Grupos de Google. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://groups.google.com/forum/#!topic/uptospnfi/-1R7-IDVdMk.

17. Símbolos y notación BPMN | Lucidchart. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.lucidchart.com/pages/es/simbolos-bpmn.

18. Definición de lenguaje de programación - Qué es, Significado y Concepto. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://definicion.de/lenguaje-de-programacion/.

19. web2py - El lenguaje Python. In: [online]. [Accessed 13 noviembre 2018]. Available from: http://web2py.com/books/default/chapter/36/02/el-lenguaje-python.

20. HTML5. In: Documentación web de MDN [online]. [Accessed 13 noviembre 2018]. Available from: https://developer.mozilla.org/es/docs/HTML/HTML5.

21. JavaScript. In: Documentación web de MDN [online]. [Accessed 13 noviembre 2018]. Available from: https://developer.mozilla.org/es/docs/Web/JavaScript.

22. Entorno de Desarrollo Integrado (IDE). | fergarciac. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

23. PyCharm: uno de los mejores IDE para Python. In: Escuela de Python [online]. 31 enero 2018. [Accessed 3 junio 2019]. Available from: https://www.escuelapython.com/pycharm-uno-de-los-mejores-ide-para-python/.

24. ¿Qué es un framework y para qué se utiliza? | Orix Systems. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.orix.es/que-es-un-framework-y-para-que-se-utiliza.

25. ¿Qué es Django? · Django Girls Tutorial. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://tutorial.djangogirls.org/es/django/.

26. JS.FOUNDATION, JS Foundation-. jQuery. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://jquery.com/.

27. ¿Qué es un gestor de datos y para qué sirve? In: [online]. [Accessed 13 noviembre 2018]. Available from: https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/que-es-un-gestor-de-datos-y-para-que-sirve.

28. PostgreSQL-ES | Emc2Net. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://e-mc2.net/es/postgresql-es#sobre-postgresql.

29. PgAdmin III - Guía Ubuntu. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

30. ARIAS CHAVES, Michael. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. In: InterSedes: Revista de las Sedes Regionales [online]. 2005, Vol. VI, no. 10. [Accessed 23 enero 2019]. Available from: http://www.redalyc.org/resumen.oa?id=66612870011.

31. Estructuración y Especificación de Casos de Uso - alfonsoperezr. In: [online]. [Accessed 23 enero 2019]. Available from: https://sites.google.com/site/alfonsoperezr/investigacion/estructuracin-y-especificacin-de-casos-de-uos.

32. 5.2. Recopilar Requisitos. In: Project Management | Gladys Gbegnedji [online]. 31 octubre 2016. [Accessed 1 febrero 2019]. Available from: https://www.gladysgbegnedji.com/recopilar-requisitos/.

33. 3Diagrama de Clases de Diseño | LENGUAJE DE MODELADO UNIFICADO - UML. In: [online]. [Accessed 25 enero 2019]. Available from: http://stadium.unad.edu.co/ovas/10596_9836/diagrama_de_clases_de_diseo.html.

34. Sparx Systems - Tutorial UML 2 - Diagrama de Despliegue. In: [online]. [Accessed 25 enero 2019]. Available from: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

35. Pattern-Oriented Software Architecture, Volume 1 - A System Of Patterns.pdf - Google Drive. In: [online]. [Accessed 20 febrero 2017]. Available from: https://docs.google.com/file/d/0B1MogsyNAsj9a1BVbi12VzhPX2M/edit.

36. Patrones.pdf. S.l.: s.n.

37. Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides-Design Patterns_ Elements of Reusable Object-Oriented Software -Addison-Wesley Professional (1994).pdf. S.l.: s.n.

38. pep8es.pdf. S.l.: s.n.

39. Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF. S.l.: s.n.

40. 25.3. unittest — Unit testing framework — Python 2.7.16 documentation. In: [online]. [Accessed 8 mayo 2019]. Available from: https://docs.python.org/2/library/unittest.html.

41. Ingenieria.de.software.enfoque.practico.pdf. S.l.: s.n.

1. Bibliografía

1. ARIAS CHAVES, Michael. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. In: InterSedes: Revista de las Sedes Regionales [online]. 2005, Vol. VI, no. 10. [Accessed 23 enero 2019]. Available from: http://www.redalyc.org/resumen.oa?id=66612870011.
2. GAMMA, Erich, HELM, Richard, JOHNSON, Ralph y VLISSIDES, John. Design Patterns Elements of Reusable Object-Oriented Software [online]. S.l.: s.n., [no date]. Available from: https://sophia.javeriana.edu.co/~cbustaca/docencia/DSBP-2018-01/recursos/Erich%20Gamma,%20Richard%20Helm,%20Ralph%20Johnson,%20John%20M.%20Vlissides-Design%20Patterns_%20Elements%20of%20Reusable%20Object-Oriented%20Software%20%20-Addison-Wesley%20Professional%20%281994%29.pdf.
3. JOSE ANTONIO RODRÍGUEZ CASCARET, Leydis Rodríguez Zamora. HERRAMIENTA INFORMÁTICA PARA EL MONITOREO DEL COMPORTAMIENTO DEL TRÁFICO DE LLAMADAS INTERNACIONALES DE LA EMPRESA DE TELECOMUNICACIONES DE CUBA S. A. La Habana, Cuba: Universidad de Ciencias Informáticas, 2016.
4. JS.FOUNDATION, JS Foundation-. jQuery. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://jquery.com/.
5. MANAGERS, Net. Administración de recursos IT: OCS-Inventory NG & GLPI. In: Net Managers [online]. [Accessed 13 noviembre 2018]. Available from: http://www.netmanagers.com.ar/stories/ocs_glpi/.
6. ¿Qué Es La Virtualización y Cuáles Son Sus Beneficios? In: Información práctica sobre Redes, Linux, Seguridad y Hacking para profesionales de TI. Capacity Academy [online]. 7 agosto 2012. [Accessed 3 junio 2019]. Available from: http://blog.capacityacademy.com/2012/08/07/que-es-la-virtualizacion-y-cuales-son-sus-beneficios/.
7. 5.2. Recopilar Requisitos. In: Project Management | Gladys Gbegnedji [online]. 31 octubre 2016. [Accessed 1 febrero 2019]. Available from: https://www.gladysgbegnedji.com/recopilar-requisitos/.
8. PyCharm: uno de los mejores IDE para Python. In: Escuela de Python [online]. 31 enero 2018. [Accessed 3 junio 2019]. Available from: https://www.escuelapython.com/pycharm-uno-de-los-mejores-ide-para-python/.
9. Software de gestión de inventarios [online]. S.l.: s.n., 2018. [Accessed 23 enero 2019]. Available from:

https://es.wikipedia.org/w/index.php?title=Software_de_gesti%C3%B3n_de_inventarios&oldid=109280982. Page Version ID: 109280982

10. PATRONES DE ARQUITECTURA Y DISEÑO DE SOFTWARE. In: DESARROLLO DE PÁGINAS WEB - DISEÑO DE PÁGINAS WEB [online]. 14 agosto 2018. [Accessed 7 mayo 2019]. Available from: https://www.desarrollodepaginasweb.com.mx/patrones-de-arquitectura-de-software/.
11. Máquina virtual [online]. S.l.: s.n., 2018. [Accessed 13 noviembre 2018]. Available from: https://es.wikipedia.org/w/index.php?title=M%C3%A1quina_virtual&oldid=111079887. Page Version ID: 111079887
12. Inventario [online]. S.l.: s.n., 2018. [Accessed 6 diciembre 2018]. Available from: https://es.wikipedia.org/w/index.php?title=Inventario&oldid=111652075. Page Version ID: 111652075
13. PyCharm [online]. S.l.: s.n., 2018. [Accessed 13 noviembre 2018]. Available from: https://en.wikipedia.org/w/index.php?title=PyCharm&oldid=867547074. Page Version ID: 867547074
14. ¿Qué es Django? · Django Girls Tutorial. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://tutorial.djangogirls.org/es/django/.
15. ¿Qué es un framework y para qué se utiliza? | Orix Systems. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.orix.es/que-es-un-framework-y-para-que-se-utiliza.
16. ¿Qué es un gestor de datos y para qué sirve? In: [online]. [Accessed 13 noviembre 2018]. Available from: https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/que-es-un-gestor-de-datos-y-para-que-sirve.
17. ¿Qué es un modelo de base de datos? | Lucidchart. In: [online]. [Accessed 7 mayo 2019]. Available from: https://www.lucidchart.com/pages/es/que-es-un-modelo-de-base-de-datos.
18. ¿Qué es y cómo funciona la virtualización de servidores? In: [online]. [Accessed 13 noviembre 2018]. Available from: https://blogmexico.comstor.com/que-es-y-como-funciona-la-virtualizacion-de-servidores.
19. ¿Qué es y cómo funciona la virtualización de servidores? [online]. S.l.: s.n. [Accessed 13 noviembre 2018]. Available from: https://blogmexico.comstor.com/que-es-y-como-funciona-la-virtualizacion-de-servidores.
20. ▷ Diagrama de paquetes. In: [online]. [Accessed 7 mayo 2019]. Available from: https://diagramasuml.com/paquetes/.

21. 08-Patrones.pdf. S.l.: s.n.
22. 25.3. unittest — Unit testing framework — Python 2.7.16 documentation. In: [online]. [Accessed 8 mayo 2019]. Available from: https://docs.python.org/2/library/unittest.html.
23. Arquitectura de Software. In: SG Buzz [online]. [Accessed 7 mayo 2019]. Available from: https://sg.com.mx/revista/27/arquitectura-software.
24. Automatiza la gestión de tus equipos con OCS. In: [online]. [Accessed 20 mayo 2019]. Available from: https://hipertextual.com/archivo/2010/09/automatiza-la-gestion-de-tus-equipos-con-ocs/.
25. CASE: Visual Paradigm Community Edition - UML Diagrams and ERD tools - Grupos de Google. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://groups.google.com/forum/#!topic/uptospnfi/-1R7-IDVdMk.
26. Definición de lenguaje de programación - Qué es, Significado y Concepto. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://definicion.de/lenguaje-de-programacion/.
27. Diagrama de Clases de Diseño | LENGUAJE DE MODELADO UNIFICADO - UML. In: [online]. [Accessed 25 enero 2019]. Available from: http://stadium.unad.edu.co/ovas/10596_9836/diagrama_de_clases_de_diseo.html.
28. Discovery | Ivanti. In: [online]. [Accessed 23 enero 2019]. Available from: https://www.ivanti.com/solutions/it-service-management/discovery.
29. Entorno de Desarrollo Integrado (IDE). | fergarciac. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.
30. Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides-Design Patterns_ Elements of Reusable Object-Oriented Software -Addison-Wesley Professional (1994).pdf. S.l.: s.n.
31. Estructuración y Especificación de Casos de Uso - alfonsoperezr. In: [online]. [Accessed 23 enero 2019]. Available from: https://sites.google.com/site/alfonsoperezr/investigacion/estructuracin-y-especificacin-de-casos-de-uos.
32. GESTOR DE RECURSOS DE HARDWARE Y SOFTWARE. In: [online]. [Accessed 23 enero 2019]. Available from: https://studylib.es/doc/7113174/gestor-de-recursos-de-hardware-y-software.
33. GLPI Gestión de inventario de hardware; ¿Cuál es GLPI; La instalación; Introducción de datos; ¿Ventajas y desventajas? In: [online]. [Accessed 20 mayo 2019]. Available from: http://imparatudos.com/article/glpi-gestin-de-inventario-de-hardware.

34. GLPI: Herramienta para la gestión de un inventario. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.coriaweb.hosting/glpi-herramienta-la-gestion-inventario/.
35. Hardware and Software Inventory | ManageEngine Desktop Central. In: [online]. [Accessed 3 junio 2019]. Available from: https://www.manageengine.com/products/desktop-central/help/inventory/hardware_software_inventory.html.
36. HTML5. In: Documentación web de MDN [online]. [Accessed 13 noviembre 2018]. Available from: https://developer.mozilla.org/es/docs/HTML/HTML5.
37. Ingenieria.de.software. enfoque. practico.pdf. S.l.: s.n.
38. Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF. S.l.: s.n.
39. Inventario de hardware y software - NetGrid. In: [online]. [Accessed 23 enero 2019]. Available from: https://www.netgrid.com.co/inventario-de-hardware-y-software/.
40. JavaScript. In: Documentación web de MDN [online]. [Accessed 13 noviembre 2018]. Available from: https://developer.mozilla.org/es/docs/Web/JavaScript.
41. La importancia de un inventario actualizado - Aranda Software. In: [online]. [Accessed 3 junio 2019]. Available from: https://arandasoft.com/la-importancia-de-un-inventario-actualizado/.
42. Metodologia UCI. S.l. [no date].
43. Metodologias de desarrollo de Software - EcuRed. In: [online]. [Accessed 23 enero 2019]. Available from: https://www.ecured.cu/Metodologias_de_desarrollo_de_Software.
44. Misión | Universidad de las Ciencias Informáticas. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.uci.cu/universidad/mision.
45. Patrón MVT: Modelo-Vista-Template | Curso de Django | Hektor Profe. In: [online]. [Accessed 7 mayo 2019]. Available from: https://docs.hektorprofe.net/django/web-personal/patron-mvt-modelo-vista-template/.
46. pep8es.pdf. S.l.: s.n.
47. PgAdmin III - Guía Ubuntu. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.guia-ubuntu.com/index.php?title=PgAdmin_III.
48. PostgreSQL-ES | Emc2Net. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://e-mc2.net/es/postgresql-es#sobre-postgresql.
49. Python Unit Testing Framework. In: [online]. [Accessed 8 mayo 2019]. Available from: http://pyunit.sourceforge.net/pyunit.html.
50. Resumen de funcionalidades de Pandora FMS. In: Pandora FMS Community [online]. [Accessed 23 enero 2019]. Available from: https://pandorafms.org/es/producto/funcionalidades-monitorizacion/.

51. Símbolos y notación BPMN | Lucidchart. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.lucidchart.com/pages/es/simbolos-bpmn.
52. Software de gestión de activos IT, software para inventario automático | ManageEngine AssetExplorer. In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.manageengine.com/es/asset-explorer/index.html.
53. Software de gestión de inventarios TI (Hardware y Software). In: [online]. [Accessed 13 noviembre 2018]. Available from: https://www.ciset.es/soluciones/inventario-de-activos-ti.
54. Sommerville. S.l.: s.n., [no date].
55. Sparx Systems - Tutorial UML 2 - Diagrama de Despliegue. In: [online]. [Accessed 25 enero 2019]. Available from: http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.
56. web2py - El lenguaje Python. In: [online]. [Accessed 13 noviembre 2018]. Available from: http://web2py.com/books/default/chapter/36/02/el-lenguaje-python.
57. What is a Virtual Machine (VM)? - Definition from Techopedia. In: [online]. [Accessed 3 junio 2019]. Available from: https://www.techopedia.com/definition/4805/virtual-machine-vm.