Centro de Telemática, Facultad 2

Universidad de las Ciencias Informáticas

Módulo de Gestión de Inventarios e Incidencias de Hardware en Windows

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

Autores: Alfredo Carús Llera

Janier Sánchez Santana

Tutor: Ing. Jenny De La Rosa Pasteur

La Habana, Cuba

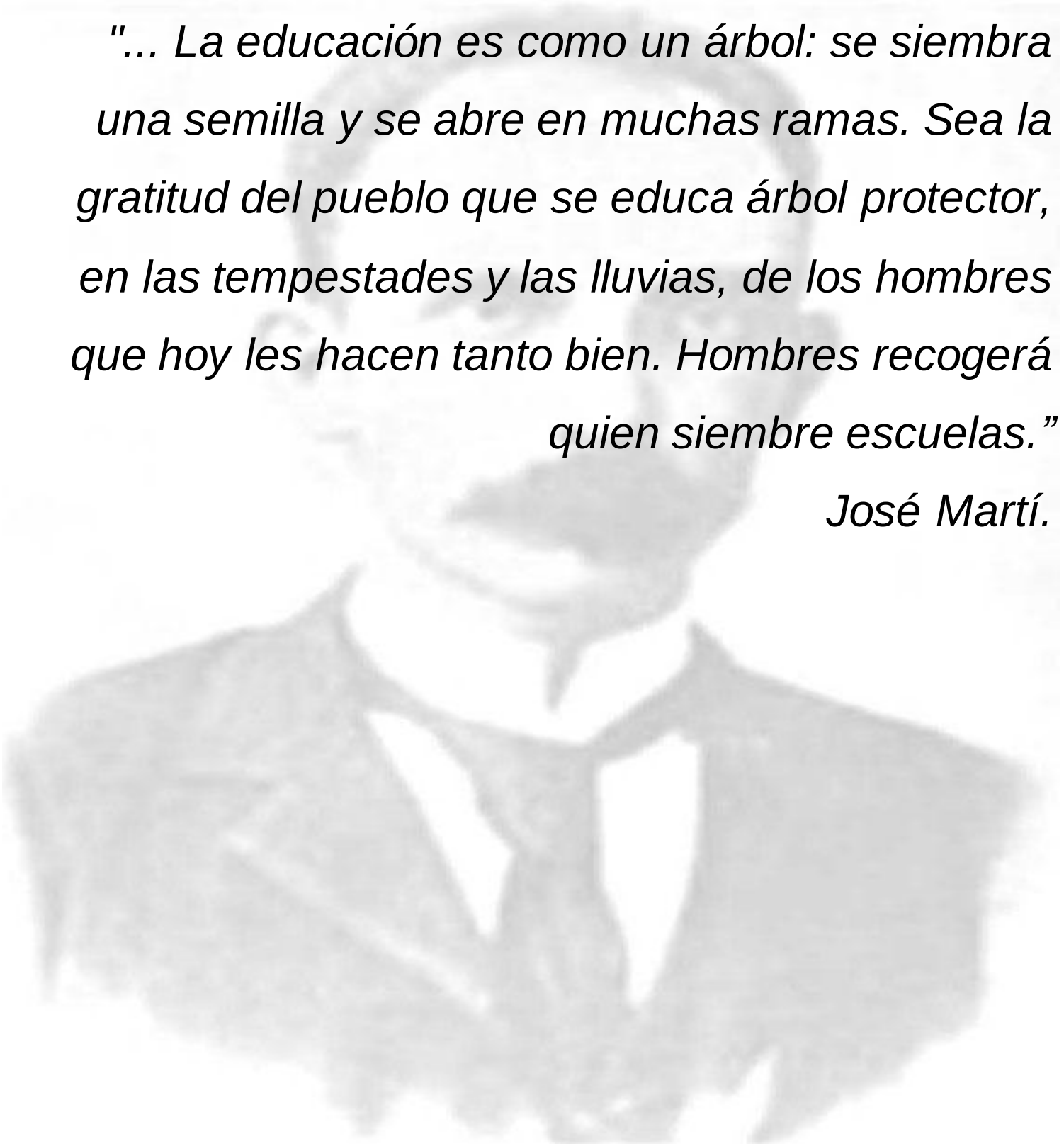
"Año 54 de la Revolución"

Junio, 2012

Pensamiento:

"... La educación es como un árbol: se siembra una semilla y se abre en muchas ramas. Sea la gratitud del pueblo que se educa árbol protector, en las tempestades y las lluvias, de los hombres que hoy les hacen tanto bien. Hombres recogerá quien siembre escuelas."

José Martí.



Declaración de Autoría:

Declaración de autoría

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

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

Alfredo Carús Llera

_____

Janier Sánchez Santana

_____

Ing. Jenny De La Rosa Pasteur

_____

Dedicatoria:

Alfredo Carús Llera

Este trabajo de diploma se lo dedico a mi madre Magaly Llera Cabrera y a mi abuelo Gerardo Llera Echazabal.

Dedicatoria:

Janier Sánchez Santana

Le dedico el presente trabajo de diploma a mi madre Regla de la C. Santana Damas y a mi padre José I. Sánchez Fernández.

Agradecimiento:

Alfredo Carús Llera

A mi madre Magaly, a mi abuelo Gerardo y mi hermanita Dimeisy Carús Llera por no dejar de estar nunca a mi lado en los momentos en los que más los necesitaba.

A mis tíos Gerardito Dagoberto por ser junto a mi abuelo unos padres excelentes.

A mis primas Eylin y Yeni que las quiero mucho.

A mi novia Alilsa por apoyarme todo el tiempo.

A mis amigos José Alberto (pepito), Antonio, Aldo.

A mi compañero de tesis Janier por acompañarme en este último año de la carrera.

A mi tutora Jenny De La Rosa Pasteur que supo tener la paciencia y dedicación para que saliera lo mejor posible.

A todas las personas que fueron parte de mi vida y me ayudaron en cada momento que necesité.

Agradecimiento:

Janier Sánchez Santana

Un agradecimiento muy especial a mi mama y mi papa por confiar en mí y apoyarme hasta el último momento.

A mis hermanos Hanny y Ledhuan por darme un excelente ejemplo.

A mi novia Beatriz por estar a mi lado durante estos cinco años en los buenos y malos momentos.

A mis amigos Vladito, Jorgito y Mario

A mi compañero de tesis Alfredo con el que compartí este trabajo de diploma.

A mi tutora Jenny por creer en nosotros cuando nadie lo hizo, por su inagotable paciencia y dedicación.

A todas las personas que han aportado a mi formación como ingeniero.

Resumen:

Resumen

Mantener un control riguroso sobre los componentes de hardware de una computadora es un aspecto importante para todo tipo de entidad. Llevar a cabo el seguimiento y control de estos medios de manera manual resulta una tarea engorrosa, por lo que se hace necesaria una herramienta capaz de automatizar estas acciones. En la actualidad existen varias aplicaciones que se encargan de recopilar toda esta información de un inventario, pero algunas son privativas o no cumplen con algunas funcionalidades. Con el fin de satisfacer las necesidades del cliente respecto al tema (que por políticas de privacidad pide no se revele su identidad y por tanto en el resto del documento se hará alusión al mismo de esta forma) se ha realizado un estudio de las aplicaciones que realizan inventarios de hardware, además de estudiar las herramientas y metodologías de desarrollo de software para posteriormente pasar a su desarrollo. Como resultado se obtendrá el Módulo de Gestión de Inventarios e Incidencias de Hardware en Windows capaz de monitorear los dispositivos USB atendiendo a las memorias permitidas (memorias que se autoriza o no su conexión o desconexión), realizar inventarios de la mayoría de los componentes de hardware, enviar los resultados al servidor y actualizar el inventario y las configuraciones obtenidas del servidor para así lograr una aplicación que cumpla con las expectativas del cliente.

Palabras Clave:

Inventario, privativas, incidencias, trazas

Índice de Contenido:

Índice de Contenidos:

Índice de Contenido:

Índice de Figuras:

Índice de Tablas:

Introducción

Las Tecnologías de la Informática y las Comunicaciones se encuentran hoy en día en constante desarrollo. Dada la necesidad de la sociedad de estar al nivel de dicho avance, tanto profesionales como estudiantes están motivados a buscar métodos y soluciones para automatizar sus tareas y realizarlas de la forma más sencilla, en el menor tiempo y con el menor costo posible. Actualmente vivimos en un mundo totalmente informatizado donde las redes de computadoras juegan un papel fundamental, los administradores de redes y hasta los mismos usuarios desconocen o no controlan como es debido a la amplia variedad de computadoras que se encuentran en las mismas, los inventarios que se hacen son escasos y de forma manual, además de que toda la documentación se archiva en papeles.

Una manera de controlar los activos tangibles en una red de ordenadores, es la utilización de sellos de seguridad, que al ser componentes físicos no generan reportes ni alertas de intrusión y la seguridad en este caso depende totalmente de la conducta y los valores de las personas que los supervisan, no siendo siempre los más adecuados. Los componentes de seguridad físicos como las puertas, yales o sellos ya no son suficientes cuando se trata de largos períodos de tiempo de utilización de las computadoras, sin embargo en la mayoría de los casos son los responsables de la seguridad de los recursos.

Debido a la mencionada forma de control de los recursos, el robo y las violaciones de las políticas de seguridad son muy frecuentes. El hurto o cambio de un componente de hardware que cumple con una función específica en una red de computadoras puede ocasionar pérdidas de información así como pérdidas monetarias, además del posible inhabilitamiento de los servicios que el mismo brinde. Es por esta razón que se impone la necesidad de implementar herramientas que se encarguen de brindar información de las computadoras e informen al administrador de red sobre los cambios realizados.

En la actualidad el cliente, utiliza la herramienta OCS Inventory para controlar los componentes de hardware de las computadoras. A pesar de ser este software bueno para el control del hardware, no realiza funcionalidades necesarias para el cliente como el monitoreo de dispositivos USB, o la configuración de las memorias autorizadas, que consiste en autorizar determinadas memorias flash para una o varias computadoras.

Introducción:

A partir de la situación problemática anterior se define el siguiente **problema científico a resolver**: ¿Cómo automatizar el control de recursos de hardware de una computadora con sistema operativo Windows?

Definiendo como **objeto de estudio** para esta investigación los procesos de inventario de hardware en una red de computadoras. Como **campo de acción**: los procesos para la obtención de información de hardware en una computadora con sistema operativo Windows.

El objetivo general del trabajo es el siguiente: Desarrollar un módulo para el sistema Gestión de Recursos de Hardware y de Software que permita controlar el inventario de los recursos de hardware de una computadora con sistema operativo Windows a través de la red, y notifique al Módulo de Alarmas y Acciones (MAAI) ante Incidencias sobre las incidencias detectadas.

Objetivos específicos:

- Elaborar el marco teórico de la investigación.
- Identificar los procesos del negocio, requisitos, casos de uso, y clases del diseño que facilite el entendimiento de las funcionalidades.
- Implementar un módulo que se ejecute sobre el sistema operativo Windows para obtener el inventario de hardware de la computadora y enviar las incidencias al Módulo de Alarmas y Acciones ante Incidencias.
- Diseñar las pruebas al módulo para certificar su correcto funcionamiento.

Tareas de la investigación:

- Estudio de las tendencias y herramientas actuales de inventario de hardware en Windows.
- Valoración crítica de las herramientas que permiten el inventario de hardware en Windows.
- Selección de las herramientas y metodologías de desarrollo que más se adapten para el desarrollo del módulo.
- Entendimiento del proceso de inventario de hardware a través del Modelo de Negocio.
- Levantamiento de los requisitos funcionales y no funcionales que deberá contener el módulo.
- Descripción del comportamiento del sistema a través de los casos de uso identificados.
- Identificación de las clases mediante el diseño de los casos de uso.

- Desarrollo de las funcionalidades de inventario de hardware del módulo.
- Desarrollo de las funcionalidades de monitoreo de dispositivos del módulo.
- Desarrollo de las funcionalidades de comparación de hardware y reporte de incidencias del módulo.
- Ejecución de las pruebas para evaluar y validar la calidad del producto.

La **idea a defender** del presente trabajo es la siguiente: Si se desarrolla un módulo que realice el inventario de recursos de hardware de una computadora y detecte cambios, identifique incidencias según configuraciones, así como la notificación de las mismas a MAAI, y envío de resultados al servidor, entonces se obtendrá un control sobre los recursos de hardware.

Los **métodos científicos** utilizados en el trabajo son los siguientes:

Métodos Teóricos:

- Inductivo-Deductivo: Se utilizó para el planteamiento del objetivo, la idea a defender y la extracción de las ideas fundamentales.
- Analítico-Sintético: Se utilizó en la revisión de documentos y artículos, de donde se extrajeron ideas y elementos importantes vinculados con la investigación. Permitió ampliar más sobre el tema, estudiando sus particularidades, obteniendo ideas centrales y relacionándolas como un todo.
- Modelación: Se utilizó para representar de forma simple las propiedades, características y funcionalidades detectadas en la investigación de la aplicación.

El documento está estructurado en cuatro capítulos que incluyen desde la fundamentación teórica hasta las pruebas realizadas al producto con el fin de entregarlo al usuario final.

Capítulo 1. Fundamentación Teórica: En este capítulo se realiza un estudio sobre las aplicaciones existentes encargadas de inventariar hardware y software en una red de computadoras. Se describen las metodologías de desarrollo de software y herramientas que serán utilizadas para la correcta realización del trabajo.

Capítulo 2. Características del Sistema: En el presente capítulo se pretende determinar los objetivos del sistema, basándose en la descripción detallada de los procesos presentes en el negocio con el fin de

establecer una visión general de lo que el sistema debe hacer. Realizando un levantamiento de requisitos y una descripción detallada de los casos de uso que guiarán el desarrollo del software.

Capítulo 3. Diseño del sistema: En este capítulo se pretende documentar todo el proceso de diseño de manera detallada, se realiza la descripción de la arquitectura base del sistema a partir de las tecnologías seleccionadas para la construcción del mismo. Además se tienen en cuenta los patrones de diseño que aportan soluciones concretas a problemas específicos, con el objetivo de obtener un buen diseño de clases.

Capítulo 4. Implementación y Prueba: En este último capítulo se implementa el sistema en términos de componentes tomando como entrada los resultados obtenidos en la etapa de diseño, para entender la forma en que el sistema está estructurado se representan los diagramas de componentes de los bundles (submódulos) de inventario y monitoreo. Además se definen las pruebas a las cuales será sometido el software con el fin de garantizar su correcto funcionamiento.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se realiza un estudio del arte de las herramientas existentes que gestionan inventario de hardware, así como se enuncian los principales conceptos teóricos que constituyen la base de la investigación efectuada. Además se analizan las características fundamentales de las tecnologías, metodologías y herramientas de desarrollo que se utilizarán en el desarrollo de la nueva herramienta.

1.2 Conceptos Fundamentales

1.2.1 Inventario:

<<La palabra inventario proviene del latín “inventarium”, y significa obtener una lista de bienes pertenecientes a una persona o comunidad, hecho con orden y precisión. También se denomina inventario al documento escrito donde consta la anotación de dichos bienes.
Los inventarios sirven para saber los bienes existentes, y son muy útiles a la hora de evaluar los progresos o pérdidas patrimoniales que ocurren en un periodo de tiempo.>> (1)

1.3 Sistemas de obtención de inventarios de hardware

En este epígrafe se realiza un análisis a los sistemas de obtención de inventarios de hardware ya existentes, sistemas que cuentan con una arquitectura cliente-servidor.
<<Por lo general, en los sistemas de obtención de inventarios los datos se registran en documentos, fuente que representan las actividades y acontecimientos ocurridos durante el flujo de operaciones de la organización. Estos sistemas pueden pasar por un flujo que permita su procesamiento electrónico y con ello tratar de satisfacer las necesidades de información de la organización. El proceso de diseño de los sistemas de información comprende tanto el diseño de uno nuevo como el rediseño de un sistema que se encuentre en operación.>>(2) A continuación se realiza un análisis de estas aplicaciones con el objetivo de

considerar si es favorable utilizar una de estas herramientas para solucionar el problema a resolver o desarrollar una nueva tomando como ejemplo las mejores funcionalidades de las existentes.

1.3.1 SIHS

<<SIHS (Sistema de Inventario de Hardware y Software) es un sistema cliente-servidor implementado en Cuba que posibilita el control de los recursos de hardware y software de una red de computadoras. El sistema está compuesto por un servidor el cual muestra los recursos mediante una aplicación web sobre los sistemas operativos Windows y Linux. Los clientes SIHS alertan de los cambios realizados en el hardware a nivel de correo electrónico cuando se realiza sin autorización en las máquinas clientes y monitorean en tiempo real los puertos USB. Almacena toda la información localmente permitiendo enviarla posteriormente si no hay conexión al servidor. Para adicionar una nueva funcionalidad es necesaria la modificación del código fuente, por no poseer un diseño modular que pueda ser extendido mediante la creación de plugins utilizando lo implementado y explotando las características dinámicas del lenguaje utilizado. Después de un análisis de estas características se concluyó no utilizarlo. >> (3)

1.3.2 OCS Inventory NG

<<OCS Inventory es una herramienta que permite realizar un inventario diario de todos los equipos de una red. El servidor de administración utiliza Apache, MySQL y Perl. OCS Inventory utiliza un agente, el cual realiza los inventarios en los clientes y un servidor de administración, que centraliza los resultados del mismo, permitiendo ver estos resultados. Las comunicaciones entre los agentes y el servidor de gestión se realiza mediante los protocolos HTTP / HTTPS. Todos los datos tienen el formato de compresión Zlib XML para reducir el promedio de tráfico de la red. OCS Inventory es liberado bajo la licencia GNU General PublicLicense, versión 2.0 (GNU GPLv2). Siempre que se realiza un inventario este se envía al servidor generando tráfico en la red, en ocasiones de manera innecesaria, ya que pueden ocurrir o no cambios en el nuevo inventario, además de no permitir realizar tareas programadas como pudiera ser la realización de un inventario a una hora especificada. >>(4) A pesar de las funcionalidades que brinda esta herramienta, su arquitectura modular y su licencia GPL se descartó la posibilidad de extender nuevas funcionalidades sobre esta herramienta ya que OCS Inventory solo permite crear plugins que no son más que paquetes

con comandos programados para ejecutar una acción determinada, lo cual no permite crear funcionalidades e integrarlas al sistema agente. En caso de que se detecte alguna incidencia, el administrador solo se entera mediante una notificación en la aplicación web. En caso de que ocurra un fallo eléctrico o de red los agentes no pueden enviar los nuevos inventarios al servidor; en tal caso el sistema no es capaz de ejecutar las tareas que quedaron pendientes apenas las condiciones lo permitan.

1.3.3 NetSupport DNA

<<NetSupport DNA (Dynamic Network Administration) es una completa solución modular que ofrece inventario de hardware, software y gestión de licencias. Presenta avisos detallados plenamente personalizables, medición y control de uso de aplicaciones e Internet. Permite la actualización automática y por consulta de distribución de software, a través de una LAN o una WAN. NetSupport DNA proporciona una puerta de enlace de comunicaciones integrada que le permite interactuar con sus activos con toda seguridad, por Internet, en cualquier lugar, todo ello sin necesidad de Virtual Private Network (VPN) ni cambios en su red existente o en la configuración de cortafuegos. El licenciamiento de NetSupport se basa en el número de estaciones de trabajo donde se instalará el programa además su renovación se realiza cada año, pagando el 20% del precio vigente por cada licencia adquirida.>>(5) A pesar de todas las ventajas que brinda esta herramienta se descartó su utilización ya que presenta algunas desventajas como por ejemplo solamente se notifica en la consola de administración web la ocurrencia de un cambio en el hardware y no hay forma de conocer si fue modificado un inventario en la cache del agente cliente por lo que no se garantiza la integridad de los datos, una característica a tener en cuenta en la realización de la nueva aplicación.

1.3.4 Cacic

<<El Controlador automático y colector de informaciones computacionales (Cacic), implementado en Brasil, fue desarrollado con el propósito de garantizar el control de activos informáticos en redes estructurales, basado en un agente preparado para obtener un diagnóstico completo de cambios producidos en componentes de hardware de cada computadora de la red y proporciona la información a una estación o

servidor. Es capaz de soportar múltiples plataformas, principalmente Linux y Windows de 32 bits, facilita el proceso de obtención de capital, está integrado con la tecnología Intel AMT/vPro – plugin, realiza la captura automatizada de los datos de hardware y software y está disponible bajo la licencia GPL. Esta herramienta presenta como inconvenientes para ser utilizada que la única vía de notificar cambios en el hardware es por correo, la comunicación entre el cliente y el servidor no se realiza de manera segura, se utiliza el protocolo http y no el https y no hay forma de conocer si fue modificado un inventario en la caché del agente cliente por lo que no se garantiza la integridad de los datos además de que no se le puede agregar más funcionalidades.>>(6)

1.4 Tecnologías y herramientas:

1.4.1 Lenguaje de programación:

El objetivo de esta investigación es implementar un sistema de inventario de hardware por lo que es necesaria la selección de un lenguaje de programación.

Cada lenguaje de programación cumple con un determinado propósito el cual se debe tener en cuenta a la hora de hacer la selección. Para elegir un lenguaje de programación primero se debe analizar cada una de sus características y asegurar que este sea el más adaptable al desarrollo del sistema.

<<Por esta razón, se enfatiza en los lenguajes de tercera generación o de alto nivel. En este tipo de lenguaje de programación las instrucciones enviadas para que el ordenador ejecute ciertas órdenes son similares al lenguaje humano. Dado que el ordenador no es capaz de reconocer estas órdenes, es necesario el uso de un intérprete que traduzca el lenguaje de alto nivel a un lenguaje de bajo nivel que el sistema pueda entender.>> (7)

<<Para el desarrollo del sistema se ha seleccionado Python ya que sus características se adaptan a las necesidades, por ejemplo permite dividir el programa en módulos, es un lenguaje de programación con una variada colección de módulos estándar que se pueden emplear en el desarrollo de aplicaciones,

Capítulo 1. Fundamentación Teórica:

Python posee, a diferencia de otros lenguajes, una sintaxis limpia y elegante que permite trabajar de una manera más cómoda y eficiente, a la vez nos ahorra una cantidad considerable de tiempo en el desarrollo de aplicaciones. Además es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad. También, Python es gratuito, incluso para propósitos empresariales.

Este lenguaje se define por un grupo de características de las cuales se destacan las siguientes:

Multiplataforma:

Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje, siempre y cuando exista un intérprete programado para él.

Interactivo:

Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.

Orientado a Objetos:

La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

Funciones y librerías:

Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de cadenas, números, archivos. Además, existen muchas librerías que se pueden importar en los programas para

tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip.>>(8)

1.4.2 Herramienta de desarrollo (IDE)

<<Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) permite editar, compilar, ejecutar y depurar programas de una forma cómoda y ágil. Está compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones.>>(9)

<<Eclipse es un IDE que presenta una arquitectura abierta y basada en plugins, lo que le permite integrar diversos lenguajes sobre un mismo IDE e introducir otras aplicaciones accesorias. Soporta a los proyectos durante todo el ciclo de vida de desarrollo. Eclipse es soportado en los principales sistemas operativos: Linux y Windows. >>(10)

<<Eclipse en combinación con el plugin PyDev constituye un entorno de trabajo para aplicaciones escritas en Python. PyDev es un plugin que posibilita desarrollar aplicaciones utilizando Python, Jython y IronPython en Eclipse. Está publicado bajo licencia Eclipse Public License-v1.0 y es gratuito. Incluye características como completado de código, resaltado de sintaxis, análisis de sintaxis, análisis de código, consola interactiva.>>(11)

1.4.3 Modelamiento con BPMN

<<Los procesos empresariales son complejos, dinámicos y necesitan ser mejorados constantemente. Están interrelacionados unos con otros y se desarrollan de manera interna a la organización o interactuando con otras organizaciones (clientes, proveedores, socios). En cualquiera de los casos

requieren ser gestionados de forma eficaz obteniendo una optimización del rendimiento y control sobre los procesos.

BPMN (Business Process Modeling Notation) es un estándar desarrollado por la BPMI (Business Process Management Initiative), el cual ha sido acogido por la OMG (Object Management Group, Inc.). BPMN tiene como objetivo principal proveer una notación que sea realmente entendible por todos los usuarios relacionados con el negocio: los analistas del negocio que crean los primeros bosquejos de los procesos; los desarrolladores técnicos responsables de implementar la tecnología que caracterizará aquellos procesos; y finalmente la gente de negocio que administrará y monitoreará los mismos procesos. De esta manera se espera crear una forma estandarizada para unir la brecha entre el proceso de diseño del negocio y el proceso de implementación. Es importante señalar el alcance de BPMN, ya que está orientado a soportar sólo conceptos de modelos que son aplicables a procesos tradicionales de negocio. Esto significa que otros tipos de modelos realizados por las organizaciones están fuera del alcance de la notación. Por ejemplo, no se incluye la estructura organizacional y recursos, los modelos de datos e información, la estrategia, y las reglas de negocio.

La implantación de proyectos BPMN aporta los siguientes beneficios:

- Reducción de plazos en los procesos.
- Optimización de costes.
- Integridad y calidad de procesos.
- Integración de terceras partes en los procesos.
- Consolidación de la información derivada de la gestión de los procesos.

BPMN ha sido elegido para el desarrollo del sistema propuesto, por ser capaz de descomponer la actividad global de una empresa u organización en un conjunto de ‘Procesos’, que pueden ser analizadas con detalle y cuyas acciones repetitivas puedan ser automatizadas, tanto en lo concerniente a los sistemas como a las personas que intervienen, para optimizar tiempos, oportunidades y costos, sin perder

la capacidad de adaptación constante y conservando la coexistencia de métodos seguros para facilitar la intervención activa y fundamental de las personas en los procesos.>> (12)

1.4.4 Metodología de Desarrollo de Software:

Proceso Unificado de Desarrollo (Rational Unified Process, RUP):

<<No fuera posible un completo desarrollo de la informática si no estuviera ligada en gran parte a la ingeniería de software, dado que para la realización de proyectos de manera eficaz es necesaria su división en módulos y así lograr un manejo fácil del mismo. Una metodología de desarrollo de software en ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. No existe una metodología que englobe todos los tipos de desarrollo de software por lo que se debe elegir la que más se ajuste a las características específicas del proyecto. >> (13) Teniendo en cuenta lo antes planteado, se decide seleccionar a RUP como metodología de desarrollo ya que está concebida para proyectos de gran escala y controla todo el ciclo de desarrollo de software de manera eficiente y promete como producto terminal un sistema que cumpla con los parámetros establecidos y funcione correctamente.

<<El Proceso Unificado de Desarrollo, es un marco de trabajo extensible. RUP es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye una metodología estándar utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Posee una forma disciplinada de asignar tareas y responsabilidades que definen quién hace qué, cuándo y cómo. Su objetivo es garantizar que la producción de software sea de alta calidad y que satisfaga las necesidades de sus usuarios finales, dentro de un calendario y con el presupuesto previsible. Se utiliza como lenguaje de modelado UML con el que se puede especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos. Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software.

RUP establece actividades y criterios que conducen a un sistema desde su máximo nivel de abstracción (la idea del cliente), hasta su nivel más concreto (un programa ejecutándose en las instalaciones del

cliente). UML ofrece la notación gráfica necesaria a RUP para representar los sucesivos modelos que se obtienen en este proceso. >> (14)

1.4.5 Herramienta Case:

<<Se puede definir a las herramientas Ingeniería de Software Asistida por Computación (CASE) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. La realización de un nuevo software requiere que las tareas sean organizadas y completadas en forma correcta y eficiente. Las herramientas CASE fueron desarrolladas para automatizar esos procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejorados en el ciclo de desarrollo de software.>> (15)

<<Visual Paradigm es una herramienta CASE, la misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta de software libre de probada utilidad para el analista.

Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos, además permite dibujar todos los tipos de diagramas de clases y generar documentación.>> (16)

1.5 Conclusiones

Durante este capítulo se realizó un estudio de las herramientas de gestión de inventario de hardware ya existentes concluyendo que las mismas no eran factibles para el cliente debido a las diferentes deficiencias que presentan. Se llevó a cabo el estudio sobre las principales herramientas a utilizar en el desarrollo del trabajo de diploma. Como lenguaje de programación será utilizado Python, ya que es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, como tecnología IDE a Eclipse ya que presenta una arquitectura abierta y basada en plugins. Debido a la plataforma de seguridad que ofrecen a los sistemas que las usan, se seleccionó como

herramienta CASE Visual Paradigm, ya que permite una rápida construcción de diagramas que son aspectos claves en todo el proceso de desarrollo del sistema. Se seleccionó a RUP como metodología de desarrollo de software con el fin de asegurar la producción del software con calidad que satisfaga las necesidades de los usuarios finales. Las herramientas y metodologías mencionadas anteriormente incluyen los requisitos necesarios para el desarrollo del sistema y la agilización del proceso por lo que permitirán guiar el desarrollo del trabajo de diploma.

Capítulo 2: Características del Sistema

2.1 Introducción

Es necesario, para que una aplicación o software cumpla con la calidad requerida, que exista un total entendimiento entre los actores del negocio y los desarrolladores del sistema para posteriormente dar solución por la vía más óptima al problema real que afecte o interactúe directamente con la sociedad. Con el objetivo de establecer una visión general de lo que el sistema debe realizar, se desarrolla el siguiente capítulo donde se pretende determinar los objetivos del sistema; basándose en la descripción detallada de los procesos presentes en el negocio, de los requerimientos de software del sistema y definición de los casos de uso.

2.2 Propuesta de sistema

Se propone el Módulo de Gestión de Inventarios e Incidencias de Hardware en Windows (GIIHW), que será el encargado de realizar inventarios de hardware y monitoreo de dispositivos USB, el inventario realizado se compara con el inventario que se almacena en la caché(espacio de memoria permanente en la PC cliente) y de existir diferencia entre estos se verifica que los cambios detectados estén comprendidos dentro de las configuraciones de incidencias(fichero que contiene lo que constituye o no una incidencia) y los periodos de cambios(fichero que contiene un rango de tiempo para cada incidencia permitiendo que en ese periodo se realicen cambios autorizados) para posteriormente generar un reporte de incidencias, las cuales se envían al servidor a través del Sistema de Administración de Recursos y Acciones (SARA) y se notifican al Módulo de Alarma y Acciones ante Incidencias (MAAI), encargado este de realizar una acción o generar una alarma mediante: correo electrónico, SMS o mensajería instantánea, este inventario se actualiza en la caché para una posterior comprobación. En caso de no existir ninguna diferencia entre estos dos inventarios (el que se realiza y el almacenado en caché) se envía al servidor solamente las trazas a través de SARA, donde se registra la fecha y hora de este último inventario realizado.

Capítulo 2. Características del sistema

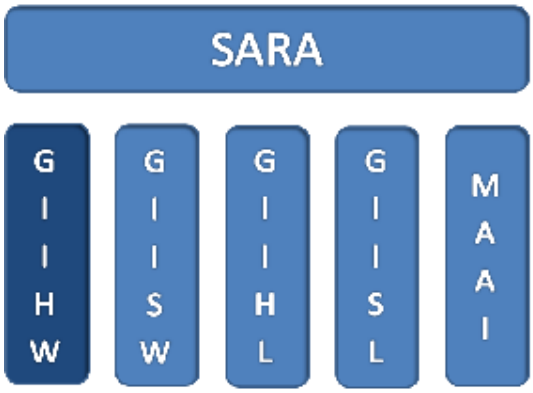


Figura 1: Propuesta del sistema

El sistema de Gestión de Recursos de Hardware y Software (GRHS) está conformado por SARA que es la aplicación encargada de la integración del resto de los módulos en el sistema cliente y de la comunicación con el servidor. El módulo que se desarrollará es el señalado en un color más oscuro y será el encargado de controlar el inventario de los recursos de hardware de una computadora en Windows, y notificar a MAAI sobre las incidencias detectadas.

SARA: Sistema de Administración de Recursos y Acciones

GIIHW: Gestión de Inventarios e Incidencias de Hardware en Windows

GIISW: Gestión de Inventarios e Incidencias de Software en Windows

GIIHL: Gestión de Inventarios e Incidencias de Hardware en Linux

GIISL: Gestión de Inventarios e Incidencias de Software en Linux

MAAI: Módulo de Alarmas y Acciones ante Incidencias

2.3 Modelo de Negocio

Capítulo 2. Características del sistema

En el presente el negocio de la entidad se realiza con la utilización de la herramienta OCS Inventory por lo que el análisis se realizará a partir de esta condición.

A continuación se presenta un modelo detallado del negocio utilizando BPMN, para proveer una notación estándar que sea fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio, en este caso de los procesos y subprocesos identificados en dicho negocio que posteriormente serán automatizados.

2.3.1 Descripción de los Procesos de Negocio

2.3.1.1 Diagrama del Proceso de Negocio Inventariar Hardware con OCS Inventory

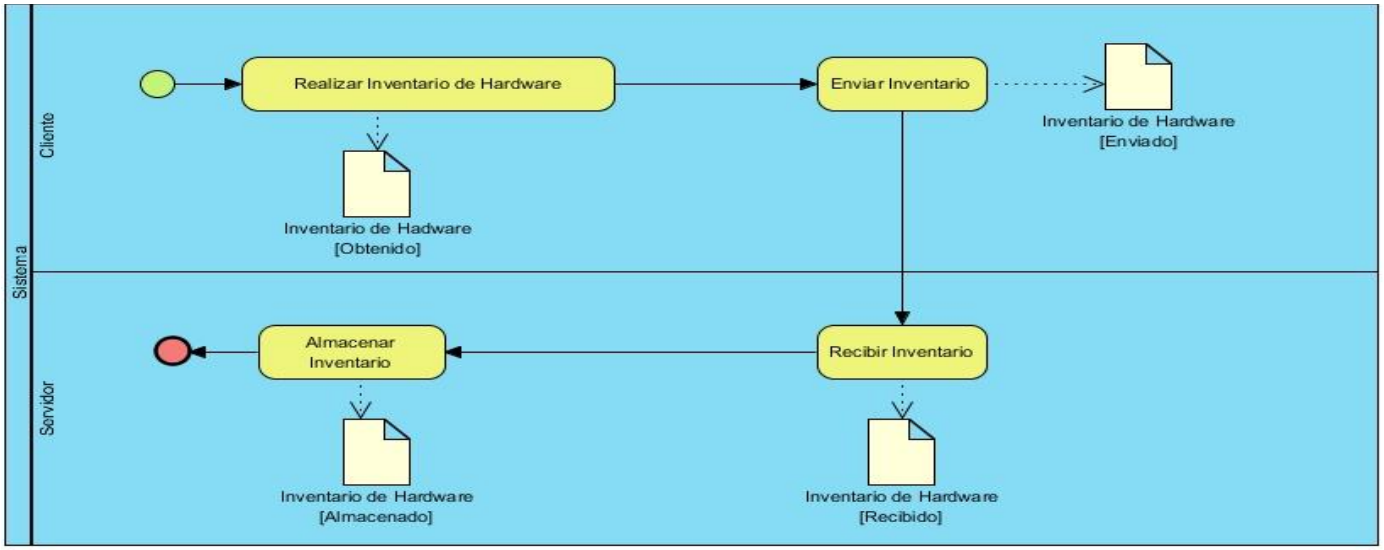


Figura 2: Diagrama del Proceso de Negocio Inventariar Hardware

2.3.1.2 Descripción del proceso Inventariar Hardware de la herramienta OCS Inventory

Realizar Inventario de Hardware: Se obtiene un inventario de los componentes de hardware de la computadora con sus datos característicos.

Enviar Inventario: El cliente establece una conexión con el servidor para enviarle la información del inventario realizado.

Recibir Inventario: El servidor recibe un inventario con la información del hardware que contienen los ordenadores.

Almacenar Inventario: En el servidor de base de datos se almacenan los inventarios asociados a todas las computadoras de la red, pudiendo el administrador a través de una consola acceder a todos estos datos.

2.4 Especificación de los requisitos de software

Un requisito no es más que una condición o capacidad que debe cumplir o poseer un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formalmente impuesto. Son una especificación de qué se debería implementar, son descripciones de cómo se debe comportar el sistema, o de un atributo o propiedad del mismo. Puede ser una restricción en el proceso de desarrollo de un sistema.

2.4.1 Requisitos funcionales

RF1. Realizar inventario de placa base:

Consiste en obtener la siguiente información:

1. Máxima capacidad de memoria
2. Manufacturera
3. Número de serie
4. Versión
5. Producto

RF2. Realizar inventario de memoria:

Consiste en obtener la siguiente información:

1. Tipo
2. Tamaño
3. Velocidad

4. Localizador
5. Manufacturera
6. Ancho de datos
7. Número de serie

RF3. Realizar inventario de disco duro:

Consiste en obtener la siguiente información:

1. Número de serie
2. Capacidad
3. Removible
4. Modelo

De las particiones:

5. Nombre
6. Punto de montaje
7. Sistema de archivos
8. Uso
9. Libre
10. Porciento de uso

RF4. Realizar inventario de lector de CD-ROM:

Consiste en obtener la siguiente información:

1. Número de serie
2. Tipo
3. Modelo

RF5. Realizar inventario de microprocesador:

Consiste en obtener la siguiente información:

1. Familia
2. Versión
3. Voltaje
4. Velocidad máxima

5. Vendedor
6. Número de serie

RF6.Realizar inventario de puertos:

Consiste en obtener la siguiente información:

1. Cantidad de puertos series
2. Cantidad de puertos paralelos
3. Cantidad de puertos USB

RF7.Realizar inventario de memoria flash:

Consiste en obtener la siguiente información:

1. Modelo
2. Número de serie
3. Driver

RF8.Realizar inventario de monitor:

Consiste en obtener la siguiente información:

1. Número de serie

RF9.Realizar inventario de teclado:

Para teclado USB se debe obtener los siguientes datos:

1. ID del vendedor
2. ID del producto

Para teclado PS/2 se debe obtener los siguientes datos:

1. Tipo de interfaz

RF10.Realizar inventario de ratón:

Para ratón USB se debe obtener los siguientes datos:

1. ID del vendedor
2. ID del producto

Para ratón PS/2 se debe obtener los siguientes datos:

1. Tipo de interfaz

RF11.Monitorear eventos de dispositivo:

Capítulo 2.Características del sistema

Consiste en el monitoreo de conexión y desconexión de los dispositivos USB conectados a la computadora.

RF12.Comprobar inventario:

Consiste en la comparación del inventario tomado y el último inventario realizado guardado en la caché, para detectar cambios en el hardware o dispositivos, y comprobación con los períodos de cambios y la configuración de incidencias, en caso de las memorias flash comprobar con la lista de memorias permitidas existentes en la configuración, para determinar una incidencia.

RF13.Notificar incidencia:

Consiste en la notificación de una incidencia a MAAI donde se envían los siguientes datos:

1. Nivel de incidencia
2. Tipo
3. Componente
4. Descripción

RF14.Guardar inventario:

Consiste en guardar el último inventario realizado en caché.

RF15.Leer inventario:

Consiste en leer el último inventario de caché.

RF16.Guardar configuración de incidencia:

Consiste en guardar en caché la configuración de incidencia recibida desde el servidor.

RF17.Leer configuración de incidencia:

Consiste en leer la configuración de incidencia guardada en caché.

RF18.Guardar períodos de cambio:

Consiste en guardar en caché la información de los períodos de cambio recibida desde el servidor.

RF19.Leer períodos de cambio:

Consiste en leer los períodos de cambio guardados en caché.

RF20.Enviar inventario:

Consiste en enviar al servidor el inventario de hardware, que puede enviarse completo en caso de no estar en el servidor, o enviar solo la parte del inventario que cambió.

Se adiciona al inventario los siguientes datos:

1. Fecha y hora
2. Usuario logueado
3. ID del agente

RF21.Enviar incidencia:

Consiste en enviar al servidor los datos de la incidencia encontrada en la comprobación del inventario dado el inventario realizado:

1. Nivel de incidencia
2. Tipo
3. Componente
4. Descripción
5. Fecha y hora
6. Usuario logueado
7. ID del agente

RF22.Enviar traza:

Consiste en enviar al servidor los datos del inventario cuando no se encuentra ningún cambio en el mismo:

1. Fecha y hora
2. Usuario logueado
3. Id agente
4. Descripción
5. Sistema operativo
6. Tipo de inventario

RF23.Actualizar configuraciones:

Cuando se realiza un cambio de las configuraciones de incidencia, configuraciones de periodos de cambio o configuraciones de monitoreo en el servidor, el cliente recibe las nuevas configuraciones, las cuales almacena en caché.

RF24.Actualizar inventario:

Cuando inicia la aplicación se solicita al servidor el último inventario almacenado en la base de datos y se guarda en caché, para evitar reportar cambios ya antes notificados durante la ejecución de otro sistema operativo.

RF25.Autorizar memorias flash:

Consiste en leer de las configuraciones de incidencias las memorias que pueden ser adicionadas y las compara con las nuevas memorias detectadas, en caso de no existir en dicha lista lanza la incidencia.

2.4.2 Requisitos no funcionales

2.4.2.1 Usabilidad

RnF1. Tipo de aplicación informática:

La aplicación será de escritorio.

RnF2. Finalidad:

La aplicación tiene el objetivo principal mantener un inventario de todos los componentes del hardware de las computadoras en Windows, así como informar las incidencias.

RnF3. Ambiente

Características del hardware del cliente:

- 128 MB de RAM.
- 1 procesador de 2.0 GHz o superior.
- 500 MB de espacio en el disco duro.

Características de software en el cliente:

- Sistema operativo: Windows XP y Windows 7.
- Python 2.7.
- WMI en los sistemas operativos Windows XP y Windows 7.

2.4.2.2 Eficiencia

RnF4. Tiempo de respuesta: la aplicación iniciará en 60 segundos.

2.4.2.3 Soporte

RnF5. La codificación se regirá mediante el estilo de codificación TLM-GRHS-0120_55 EstandarPythonv1.0 definido por el proyecto para garantizar un mejor soporte.

2.4.2.4 Restricciones de diseño

RnF6. Se harán tratamientos de excepciones para todo el código incluyendo el importado de bibliotecas externas para garantizar conocer cuando no están instaladas algunas de sus dependencias.

RnF7. El módulo se implementará usando la herramienta Eclipse.

RnF8. Se recopilará la información de hardware mediante WMI.

2.5 Modelo de Casos de Uso del Sistema

La mejor forma para efectuar de manera iterativa e incremental un buen desarrollo de software es encontrar los casos de uso correctos y definir su nivel de prioridad para que el producto final tenga la calidad requerida y se realice con menos tiempo y costo. El principal objetivo de los casos de uso es responder a los requisitos funcionales del futuro producto. A partir del proceso de modelado del negocio y la especificación de requisitos funcionales fueron identificados los siguientes casos de uso:

2.5.1 Casos de uso del Módulo de Gestión de Incidencias e Inventarios de Hardware en Windows

Diagrama de Casos de Uso del Sistema

Capítulo 2. Características del sistema

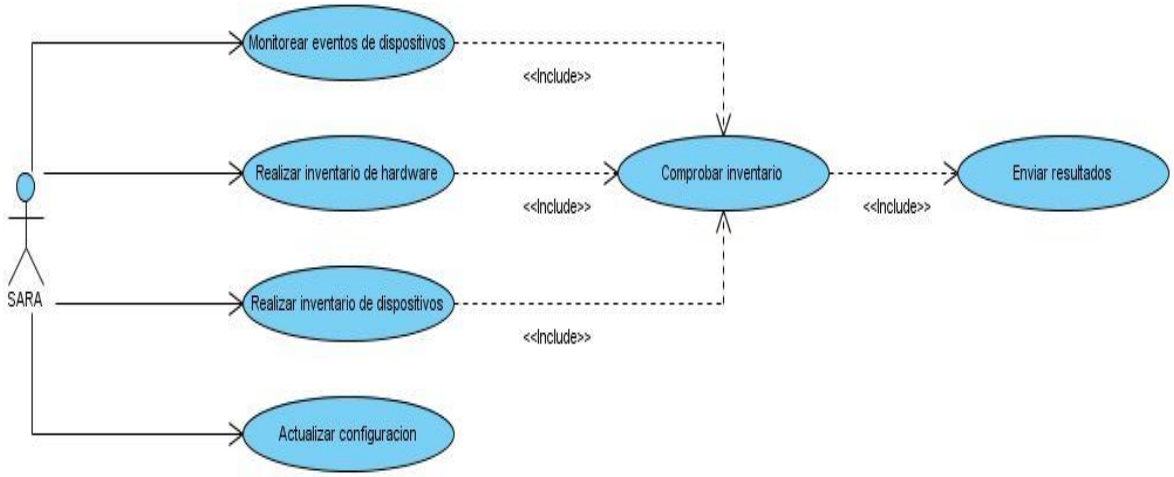


Figura 3: Diagrama de caso de uso del Módulo de Gestión de Inventarios e Incidencias de Hardware en Windows

Descripción detallada de Casos de Uso del Sistema Agente

CU 1. Monitorear de eventos de dispositivos

Objetivo	Consiste en el monitoreo de conexión o desconexión de los dispositivos conectados a la computadora.
Actores	SARA: (Inicia) Realiza el monitoreo de eventos de los dispositivos.
Resumen	En este caso de uso se realiza el monitoreo de eventos de los dispositivos de la PC.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	La computadora está encendida. El sistema debe estar instalado en la PC.
Postcondiciones	Se realiza el monitoreo de eventos de los dispositivos.
Flujo de eventos	
Flujo básico Monitorear eventos de dispositivos.	

	Actor	Sistema
1.	Solicita el monitoreo de eventos de los dispositivos.	
2.		Monitoreo de eventos en los puertos USB.
3.		Compara el monitoreo de eventos de los dispositivos actual con la información almacenada. Ver caso de uso: Comprobar inventario.
4.		Termina caso de uso.
Relaciones	**CU Incluidos**	Comprobar monitoreo de eventos de dispositivos: Paso 3 del flujo básico Monitorear eventos de dispositivos. Comprobar monitoreo de eventos de dispositivos en el caso de uso Comprobar inventario.

Tabla 1: Caso de uso monitorear de eventos de dispositivos

CU 2. Realizar inventario de hardware

Objetivo	Consiste en realizar un inventario de hardware a la computadora.	
Actores	SARA: (Inicia) Realiza un inventario de hardware.	
Resumen	En este caso de uso se realiza un inventario de hardware a la computadora.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	La computadora está encendida. El sistema debe estar instalado en la computadora.	
Postcondiciones	Se realiza el inventario de hardware a la computadora.	
Flujo de eventos		
Flujo básico Realizar inventario de hardware		
	Actor	**Sistema**

1.	Solicita el inventario de hardware de la computadora.	
2.		Obtiene la información de la placa base.
3.		Obtiene la información de la memoria.
4.		Obtiene la información del disco duro.
5.		Obtiene la información del lector de CD-ROM.
6.		Obtiene la información del microprocesador.
7.		Obtiene la información de los puertos.
8.		Compara el inventario de hardware actual con el último inventario almacenado. Ver caso de uso: Comprobar inventario.
9.		Termina caso de uso.
Relaciones	**CU Incluidos**	Comprobar inventario: Paso 13 del Flujo Básico Realizar inventario de hardware. Comprobar inventario en el caso de uso Comprobar inventario.

Tabla 2: Caso de uso realizar inventario de hardware

CU 3. Realizar inventario de dispositivos

Objetivo	Consiste en realizar un inventario de los dispositivos de la computadora.
Actores	SARA: (Inicia) Realiza un inventario de los dispositivos.
Resumen	En este caso de uso se realiza un inventario de los dispositivos a la computadora.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	La computadora está encendida. El sistema debe estar instalado en la computadora.
Postcondiciones	Se realiza el inventario de los dispositivos a la computadora.

Flujo de eventos		
Flujo básico Realizar inventario de dispositivos		
	Actor	**Sistema**
1.	Solicita el inventario de los dispositivos de la computadora.	
2.		Obtiene la información de la memoria flash.
3.		Obtiene la información del monitor.
4.		Obtiene la información del teclado.
5.		Obtiene la información del ratón.
6.		Compara el inventario de los dispositivos actual con el último inventario almacenado. Ver caso de uso: Comprobar inventario.
7.		Termina caso de uso.
Relaciones	**CU Incluidos**	Comprobar inventario: Paso 6 del Flujo Básico Realizar inventario de dispositivos. Comprobar inventario en el caso de uso Comprobar inventario.

Tabla 3: Caso de uso realizar inventario de dispositivos

CU 4. Actualizar configuración

Objetivo	Consiste en actualizar las configuraciones de incidencias, períodos de cambios y monitoreo.
Actores	SARA: (Inicia) Actualiza las configuraciones de incidencias, períodos de cambios y monitoreo.
Resumen	En este caso de uso se actualizan las configuraciones de incidencias, períodos de cambios y monitoreo.
Complejidad	Alta
Prioridad	Crítico

Precondiciones	La computadora está encendida. El sistema debe estar instalado en la computadora.	
Postcondiciones	Se actualizaron las configuraciones de incidencias, períodos de cambio y monitoreo.	
Flujo de eventos		
Flujo básico Actualizar configuración		
	Actor	**Sistema**
1.	Solicita guardar las configuraciones de incidencias, períodos de cambio y monitoreo.	
2.		Obtiene periodos de cambios del servidor.
3.		Obtiene configuración de incidencias del servidor.
4.		Obtiene configuración de monitoreo del servidor
5.		Actualiza los periodos de cambios.
6.		Actualiza la configuración de incidencias.
7.		Actualiza la configuración de monitoreo.
8.		Termina caso de uso.

Tabla 4: Caso de uso actualizar configuración

CU 5. Comprobar inventario

Objetivo	Consiste en comparar el último inventario almacenado en la caché con la información actual del hardware.
Actores	SARA: (Inicia) Solicita comprobar el inventario.
Resumen	En este caso de uso se compara el último inventario almacenado en la caché con la información actual.
Complejidad	Alta

Prioridad	Crítico	
Precondiciones	La computadora está encendida. El sistema debe estar instalado en la computadora.	
Postcondiciones	Se comprobó correctamente el inventario.	
Flujo de eventos		
Flujo básico Comprobar inventario		
	Actor	**Sistema**
1.		Obtiene el último inventario almacenado en caché.
2.		Compara el último inventario almacenado en caché con el nuevo inventario, donde se detecta un cambio.
3.		Obtiene la configuración de incidencias.
4.		Comprueba que el cambio detectado se encuentra registrado en la configuración de incidencias y en caso de las memorias flash comprueba que el número de serie no se encuentre registrado en la configuración de incidencias
5.		Obtiene la información de los periodos de cambio.
6.		Comprueba que la incidencia no se encuentra incluida en el periodo de cambios.
7.		Envía notificación de la incidencia detectada a MAAI.
8.		Guarda inventario en la caché.
9.		Envía la incidencia al servidor. Ver caso de uso

		Enviar resultado.
10.		Termina caso de uso.
Flujos alternos		
1ª El inventario no se encuentra almacenado en la caché.		
	Actor	**Sistema**
1ª1		Almacena el inventario actual en caché.
1ª2		Envía inventario al servidor. Ver caso de uso Enviar Resultado.
1ª3		Termina el caso de uso.
Flujos alternos		
2ª No existe diferencia entre el último inventario almacenado en la caché y el actual.		
	Actor	**Sistema**
2ª1		Enviar trazas. Ver caso de uso Enviar Resultado.
2ª2		Termina el caso de uso.
Flujos alternos		
3ª La incidencia no está incluida.		
	Actor	**Sistema**
3ª1		Enviar inventario. Ver caso de uso Enviar resultado.
3ª2		Guarda inventario en la caché.
3ª3		Termina el caso de uso.
Flujos alternos		
4ª La incidencia está incluida en el período de cambios.		
	Actor	**Sistema**
4ª1		Enviar inventario. Ver caso de uso Enviar

		resultado.
4ª2		Guarda inventario en la caché.
4ª3		Termina el caso de uso.
Relaciones	**CU Incluidos**	Enviar inventario: Paso 1ª2 del Flujo alterno El inventario no se encuentra almacenado en la caché. Enviar inventario en el CU Enviar resultado. Enviar traza: Paso 2ª1 del Flujo alterno No existe diferencia entre el último inventario almacenado en la caché y el actual. Enviar traza en el CU Enviar resultado. Enviar inventario: Paso 3ª1 del Flujo alterno La incidencia no está incluida. Enviar inventario en el CU Enviar resultado. Enviar inventario: Paso 4ª1 del Flujo alterno La incidencia está incluida en el período de cambios. Enviar inventario en el CU Enviar resultado. Enviar incidencia: Paso 9 del Flujo básico Comprobar inventario. Enviar incidencia en el CU Enviar resultado.

Tabla 5: Caso de uso comprobar inventario

CU 6. Enviar Resultados

Objetivo	Consiste en enviar el resultado obtenido.
Actores	SARA: (Inicia) Obtiene y envía el resultado al servidor.
Resumen	Consiste en enviar el inventario, trazas o incidencias al servidor.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	La computadora está encendida.

	El sistema debe estar instalado en la computadora. Se ha obtenido el resultado de la comprobación de inventario.	
Postcondiciones	Se envió el resultado obtenido al servidor.	
Flujo de eventos		
Flujo básico Enviar resultados		
	Actor	**Sistema**
1.		Registra fecha y hora.
2.		Serán ejecutadas algunas de las siguientes acciones: • Enviar inventario: ir a la Sección 1. • Enviar incidencia: ir a la Sección 2. • Enviar traza: ir a la Sección 3.
3.		Termina caso de uso.
Sección 1: "Enviar inventario"		
	Actor	**Sistema**
1.		Envía el inventario al servidor para su posterior almacenamiento.
Sección 2: "Enviar incidencias"		
	Actor	**Sistema**
1.		Envía las incidencias al servidor para su posterior almacenamiento.
Sección 3: "Enviar trazas"		
	Actor	**Sistema**
1.		Envía las trazas al servidor para su posterior almacenamiento.

Tabla 6: Caso de uso enviar resultados

2.6 Conclusiones

Con este capítulo se elaboró una propuesta del sistema que se va a desarrollar. Se realizó un estudio detallado de los procesos del negocio correspondientes a la institución la cual utiliza la herramienta OCS Inventory para gestionar los recursos de hardware con el objetivo de lograr un mejor entendimiento. Se identificaron los actores del sistema, requisitos funcionales y no funcionales, siendo cruciales para el análisis y diseño del mismo. Además se generan los casos de uso acorde con las funcionalidades de la aplicación.

Capítulo 3: Diseño del Sistema

3.1 Introducción

En el presente capítulo se realiza una descripción detallada del modelo de diseño para el Módulo de Gestión de Inventarios e Incidencias de Hardware en Windows para que exista un mayor entendimiento por parte de los desarrolladores. A partir de los requisitos funcionales y casos de uso del sistema, se analiza la posibilidad que existe de darles solución satisfaciendo los requisitos significativos de la arquitectura. Se determinan los patrones de diseño a utilizar para mejorar las actividades de implementación teniendo en cuenta que estos aportan soluciones concretas a problemas específicos.

3.2 Arquitectura de Software

<<En los inicios de la informática, la programación se consideraba un arte y se desarrollaba como tal, con el tiempo se han ido descubriendo y desarrollando formas y guías generales con el fin de facilitar los problemas a resolver. A estas, se les ha denominado Arquitectura de Software, porque, a semejanza de los planos de un edificio o construcción, estas indican la estructura, funcionamiento e interacción entre las partes del software. La Arquitectura de Software tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad.>> (17)

3.2.1 Estilos Arquitectónicos

<<Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante composición de los estilos fundamentales. En tal sentido, los

estilos conjugan elementos o “componentes” como lo son los conectores, configuraciones y restricciones. Un estilo arquitectónico es un conjunto coordinado de restricciones arquitectónicas que restringe los roles/rasgos de los elementos arquitectónicos y las relaciones permitidas entre esos elementos dentro de la arquitectura que se conforma a ese estilo.

La descripción de un estilo se puede formular en lenguaje natural o en diagramas, pero lo mejor es hacerlo en un lenguaje de descripción arquitectónica o en lenguajes formales de especificación.>>(18)

3.2.2 Arquitectura Cliente-Servidor

<<La arquitectura cliente/servidor agrupa conjuntos de elementos que efectúan procesos distribuidos y cómputo cooperativo.

Cliente:

Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes Local Area Network (LAN) o Wide Area Network (WAN). La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

Servidor:

Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LAN o WAN, para proveer de múltiples servicios a los clientes y ciudadanos tales como: impresión, acceso a bases de datos, fax y procesamiento de imágenes.

En el modelo cliente/servidor se pueden encontrar las siguientes características:

1. El cliente y el servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.

2. Las funciones del cliente y el servidor pueden estar en plataformas separadas, o en la misma plataforma.

3. Un servidor da servicio a múltiples clientes en forma concurrente.

4. Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los clientes o de los servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.

5. La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.

6. Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final. >>(19)

Para la realización del sistema GRHS se definió utilizar la arquitectura Cliente-Servidor puesto que permite mantener centralizada la información de todos los recursos de hardware de las computadoras donde se encuentren los clientes, para su posterior consulta por parte del administrador a través de una aplicación web. El módulo que se propone se encuentra en el cliente, brindando la información necesaria al servidor.

3.2.3 Arquitectura basada en capas

<<La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada.

Capítulo 3. Diseño del sistema

El estilo de arquitectura basado en capas se identifica por las siguientes características:

• Describe la descomposición de servicios de forma que la mayoría de la interacción ocurre solamente entre capas vecinas.

• Las capas de una aplicación pueden residir en la misma máquina física (misma capa) o puede estar distribuido sobre diferentes computadores (n-capas).

• Los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces muy bien definidas.

• Este modelo ha sido descrito como una “pirámide invertida de re-uso” donde cada capa agrega responsabilidad y abstracción a la capa directamente sobre ella.>> (20)

Para la realización del módulo de GIIHW se decidió utilizar la arquitectura basada en capas ya que permite separar las clases según sus responsabilidades, y de esta forma una modificación en una capa no afecte a las demás. Bajo esta arquitectura se definió la capa “Service” que contiene todo el negocio de la aplicación, y la capa “Dataaccess” que garantiza la obtención de los datos de los recursos de hardware, así como el paquete “Domain” que contiene las clases persistentes.

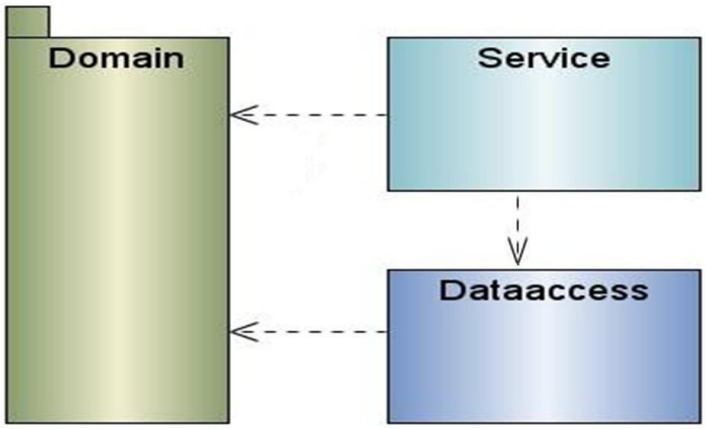


Figura 4: Arquitectura basada en capas

Capítulo 3. Diseño del sistema

3.3 Modelo de diseño

<<El modelo de diseño es una realización del diseño del sistema. Es un modelo físico y concreto. Se centra en cómo los requisitos funcionales y no funcionales tienen impacto en el sistema a desarrollar. Dentro de sus propósitos están: crear una entrada apropiada y un punto de partida para la implementación, descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo. Adquirir comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnología de distribución y concurrencia. Esto se representa por colaboraciones en el modelo de diseño y denota realización de caso de uso del diseño.>>(21)

3.3.1 Diagrama de Paquetes

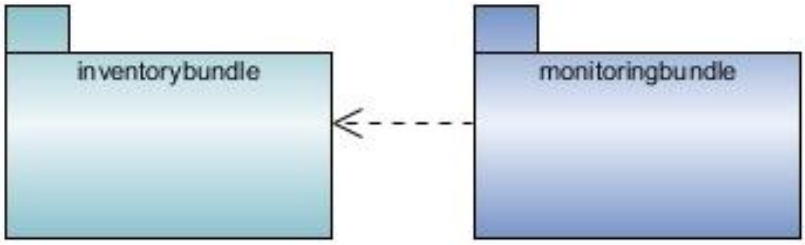


Figura 5: Diagrama de paquetes

En el diagrama de paquetes se muestra como están agrupados los componentes del sistema, en el caso de inventorybundle contiene dos capas, la de servicio y la de acceso a datos al igual que monitoringbundle. El sistema se dividió en dos paquetes ya que permite una mejor organización y cada subsistema es el encargado de realizar una tarea en específico.

3.3.2 Diagrama de Clases del Diseño

<<Un diagrama de Clases del Diseño es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Una clase es una descripción de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica.

El diagrama general de las clases del diseño representa la interacción entre las clases de las diferentes capas que integran el sistema.>> (22)

El sistema muestra la división del módulo en dos bundles, el de inventario y el de monitoreo donde las clases del diseño y la relación que existe entre ellas ayudan a comprender fácilmente el sistema para su posterior implementación.

3.3.2.1 Diagrama de clases del diseño bundle de inventario

Capítulo 3. Diseño del sistema

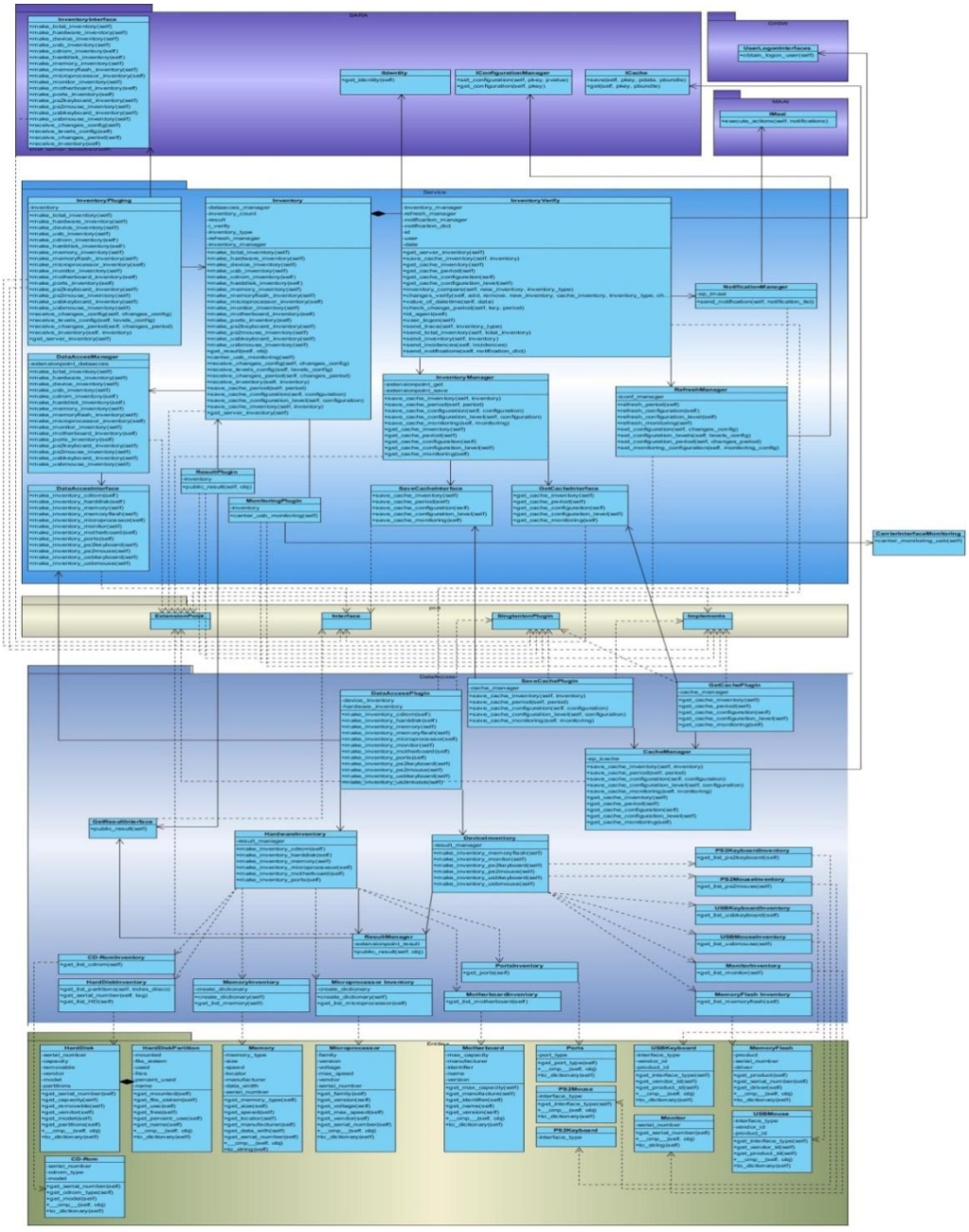


Figura 6: Diagrama de clases del diseño del bundle Inventario

En la figura anterior se muestra el bundle de inventario mediante el cual se realizará el inventario de los componentes de hardware mediante la interfaz que se le brinda a SARA InventoryInterface y a través de la clase Inventory que contendrá un objeto de la clase DataAccessManager que ordenaría realizar un inventario mediante las clases HardwareInventory y DeviceInventory y sus respectivas clases entidades, posteriormente en la clase InventoryVerify se compara el inventario realizado con el que se obtiene de caché a través de la clase CacheManager y se envían las trazas, el inventario y las incidencias según sea el caso.

3.3.2.2 Diagrama de clases del diseño bundle de monitoreo

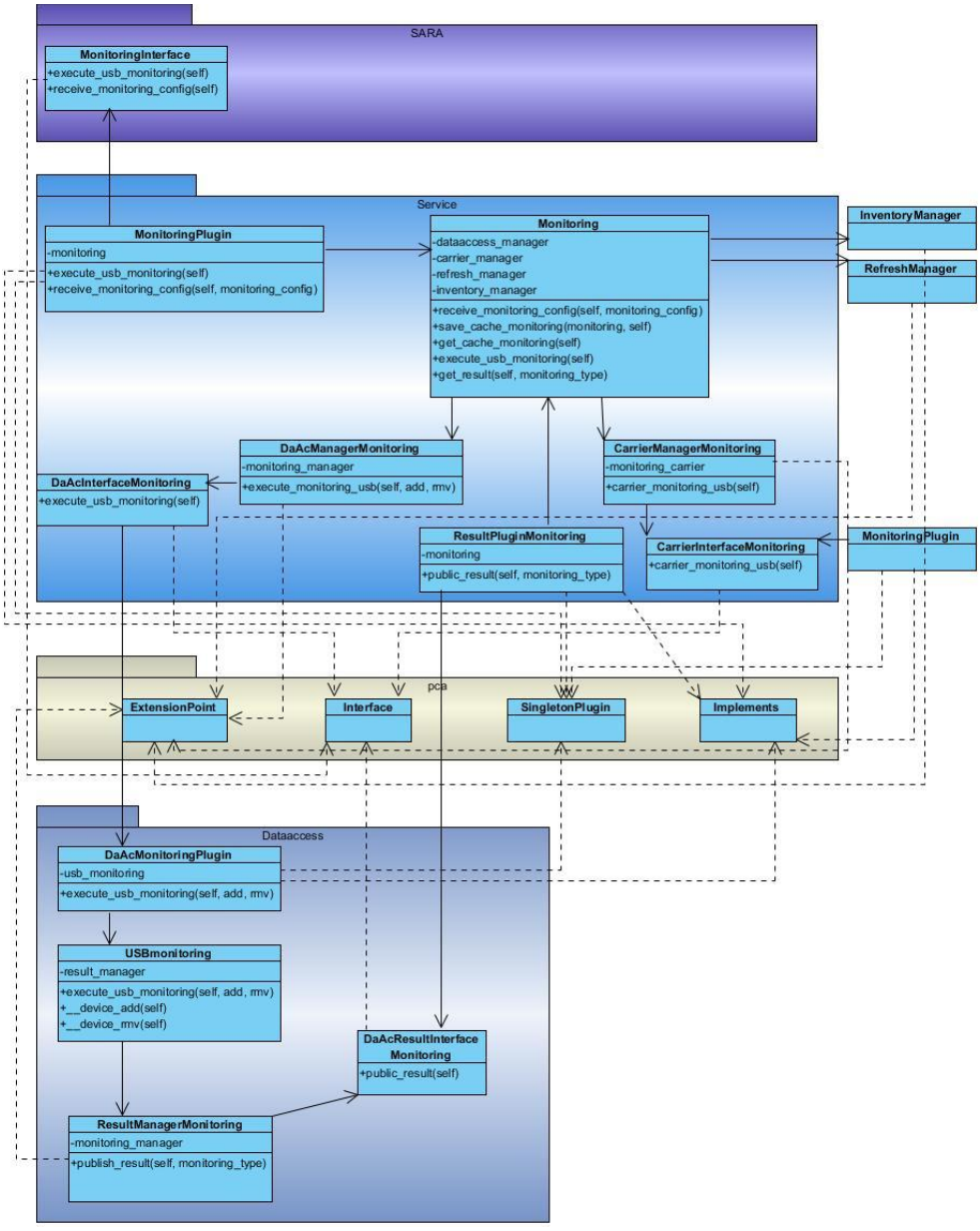


Figura 7: Diagrama de clases del diseño del bundle Monitoreo

En la figura anterior se muestra el bundle de monitoreo mediante el cual se realizará el monitoreo de los dispositivos USB mediante la interfaz que se le brinda a SARA MonitoringInterface y a través de la clase Monitoring que contendrá un objeto de la clase DaAcManagerMonitoring que ordenaría realizar el monitoreo mediante la clase USBMonitoring. Posteriormente cuando se detecte la conexión o desconexión de algún dispositivo USB se manda a hacer un inventario a través del plugin MonitoringPlugin que implementa a la interfaz CarrierInterfaceMonitoring perteneciente al bundle de inventario.

3.4 Patrones de diseño

<<Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan.

Es evidente que a lo largo de multitud de diseños de aplicaciones hay problemas que se repiten o que son análogos, es decir, que responden a un cierto patrón. Sería deseable tener una colección de dichos patrones con las soluciones más óptimas para cada caso. Los patrones de diseño no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables. Han revolucionado el diseño orientado a objetos y todo buen arquitecto de software debería conocerlos.

Patrones GRASP:

GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades).

El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se quiere diseñar eficazmente el software orientado a objetos.

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

Capítulo 3. Diseño del sistema

Una de las actividades más complicadas en Orientación de Objetos consiste en elegir las clases adecuadas y decidir cómo estas clases deben interactuar. Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento.

- **Experto:** La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada.>> (23). Este patrón se pone de manifiesto en las clases del dominio.

- <<**Creador:** El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar la clase responsable de crear una nueva instancia de determinada clase.>> (23). Este patrón se pone de manifiesto en las clases encargadas de crear instancias de las clases del dominio como son HardDiskInventory y MemoryInventory.

- <<**Bajo Acoplamiento:** El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y recurre a ellas. El Bajo Acoplamiento soporta un diseño de clases más independientes, que reducen el impacto de cambios, y permite que sean más reutilizables. >> (23). Este patrón se pone de manifiesto en todas las clases del diseño.

- <<**Alta Cohesión:** La cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos, y auto-identificable. Una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo. >> (23).Este patrón se pone de manifiesto en todas las clases del diseño.

Patrones GoF:

Patrones de creación:

- **Singleton:** <<Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. >> (23). Este patrón se pone de manifiesto en las clases Inventory y Monitoring.

Patrones estructurales:

- **Facade:** <<Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar. >> (23). Este patrón se pone manifiesto en las clases InventoryInterface y MonitoringInterface.

Patrones de comportamiento:

- **Mediator:** <<Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente. >> (23). Este patrón se pone de manifiesto entre las clases interfaces y los plugins que las implementan.
- **Observer:** <<Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos. >> (23). Este patrón se pone de manifiesto entre las clases interfaces y los plugins que las implementan.

3.5 Conclusiones

En este capítulo se definió una arquitectura base flexible teniendo en cuenta las funcionalidades del sistema. Se realizó un estudio y se aplicaron patrones de diseño para asignar responsabilidades a las clases, fundamentales para la fase de implementación y proporcionar soluciones a problemas concretos. Se mostraron los diagramas de paquete, diagramas de clases del sistema, tanto del bundle de inventario como el del monitoreo y las descripciones de cada una de las clases. Lo antes mencionado permite una introducción a la implementación del sistema y una organización del código de manera lógica de forma tal que facilite el trabajo del equipo de desarrollo.

Capítulo 4: Implementación y Prueba

4.1 Introducción

El presente capítulo tiene como objetivo implementar y asegurar la calidad del software. Es en esta fase donde se obtienen los componentes y subsistemas en un modelo de implementación, tomando como punto de partida el modelo de diseño elaborado en la fase anterior, alcanzando un sistema maduro en términos de componentes, llevando a cabo la implementación de las clases y sus relaciones. Dando paso posteriormente a la fase de pruebas, en la cual se le realiza un profundo estudio al software con el fin de garantizar que el mismo cumple con cada funcionalidad que este debe brindar.

4.2 Diagrama de despliegue

<<Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación, muestra las relaciones físicas entre los componentes de hardware y software en el sistema final, es decir la configuración de los elementos de procesamiento en tiempo de ejecución.>> (24). En el caso de la aplicación, cuenta con la PC cliente que es donde se encuentra el agente instalado, con un servidor de caché donde almacenan las configuraciones y los inventarios y un servidor web al cual se le envían los resultados.

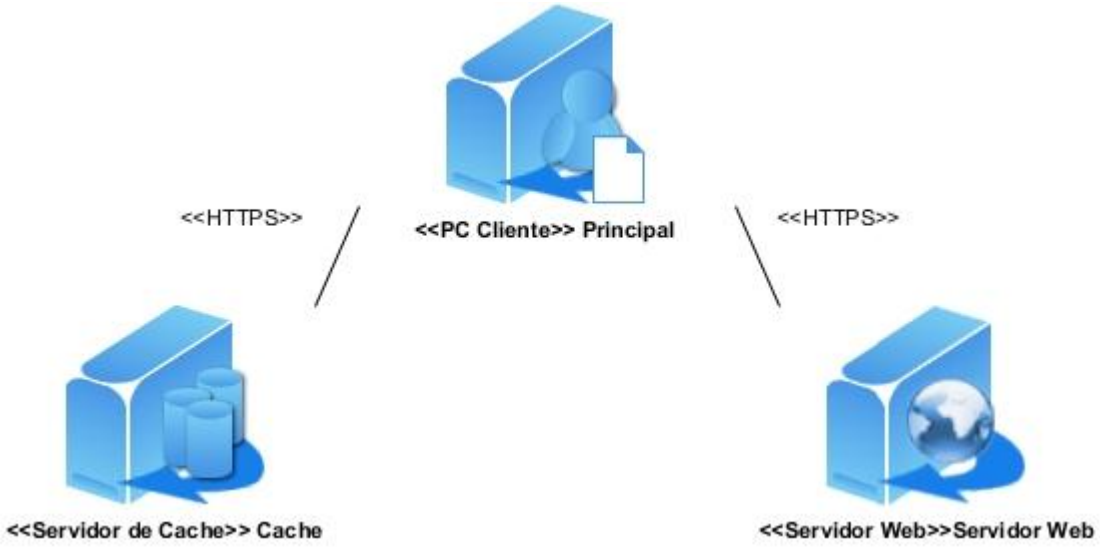


Figura 8: Diagrama de despliegue

Nodo	Descripción
PC Cliente	PC donde se ejecuta el módulo GIIHW
Servidor de Caché	PC donde se almacenan las configuraciones y los inventarios.
Servidor Web	PC donde se encuentra el Sistema Servidor, del cual se obtiene la información correspondiente a cada módulo.

Tabla 7: Diagrama de despliegue

4.3 Diagrama de componentes

<<Un diagrama de componentes muestra la organización y las dependencias entre un conjunto de componentes y sus relaciones de manera gráfica a través del uso de nodos y arcos entre estos.

Normalmente los diagramas de componentes contienen:

- Componentes
- Interfaces
- Relaciones de dependencia, generalización, asociaciones y realización.
- Paquetes o subsistemas
- Instancias de algunas clases

Visto de otro modo un diagrama de componentes puede ser un tipo especial de diagrama de clases que se centra en los componentes físicos del sistema. >> (25)

Diagrama de componente del bundle de inventario

Capítulo 4.Impementación y Prueba

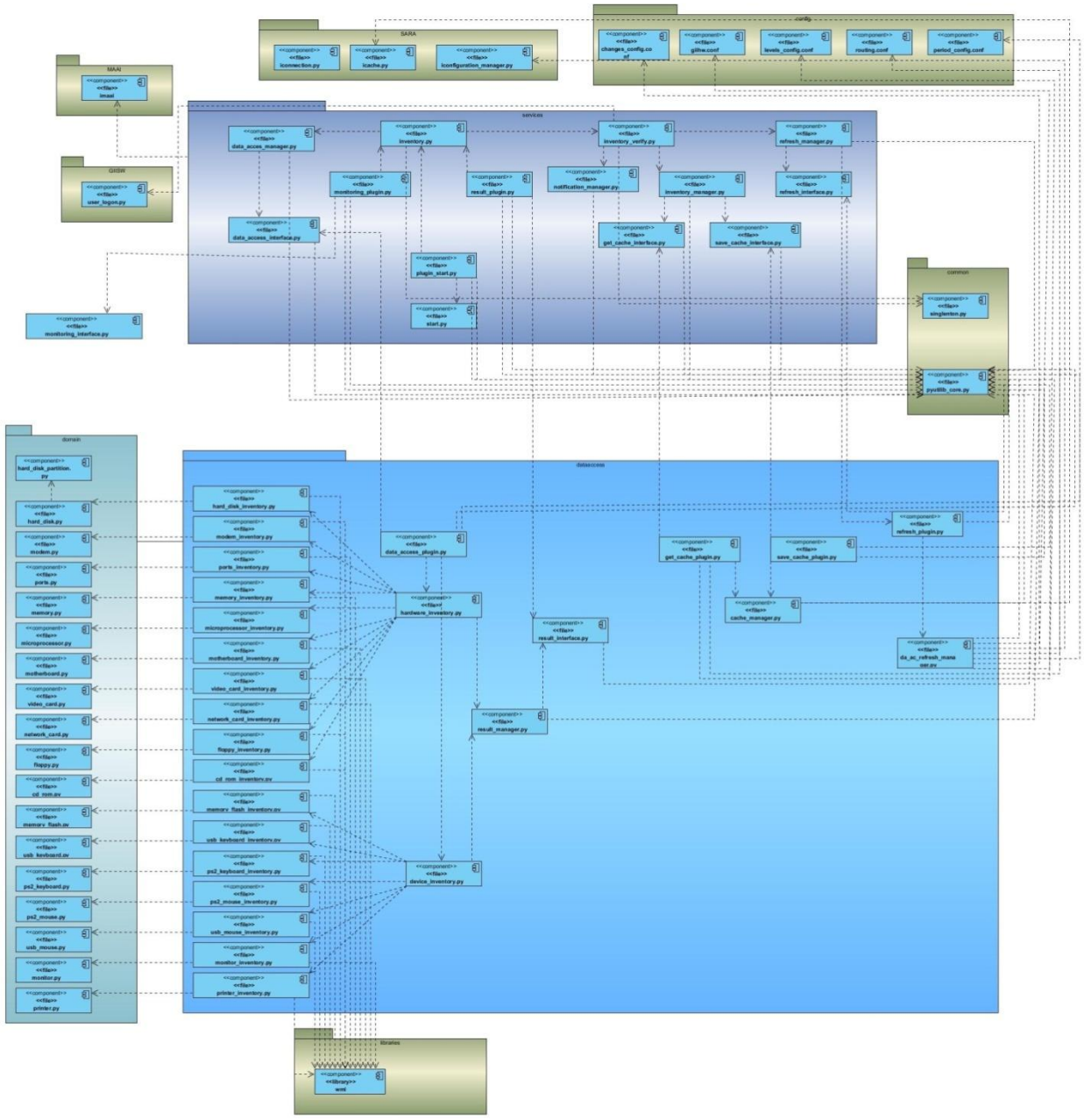


Figura 9: Diagrama de componentes bundle de inventario

En la figura anterior se muestran los diferentes componentes existentes en el bundle de inventario, como la biblioteca WMI y el paquete common que contiene los modulos pyutilib_core y singleton. También muestra las clases como pueden ser inventory, inventory_verify y data_access_manager. Ademas se muestran los ficheros de configuración y los componentes pertenecientes a otros módulos como imaai y user_logon pertenecientes a MAAI y GIISW respectivamente.

Diagrama de componentes del bundle de monitoreo

Capítulo 4.Impementación y Prueba

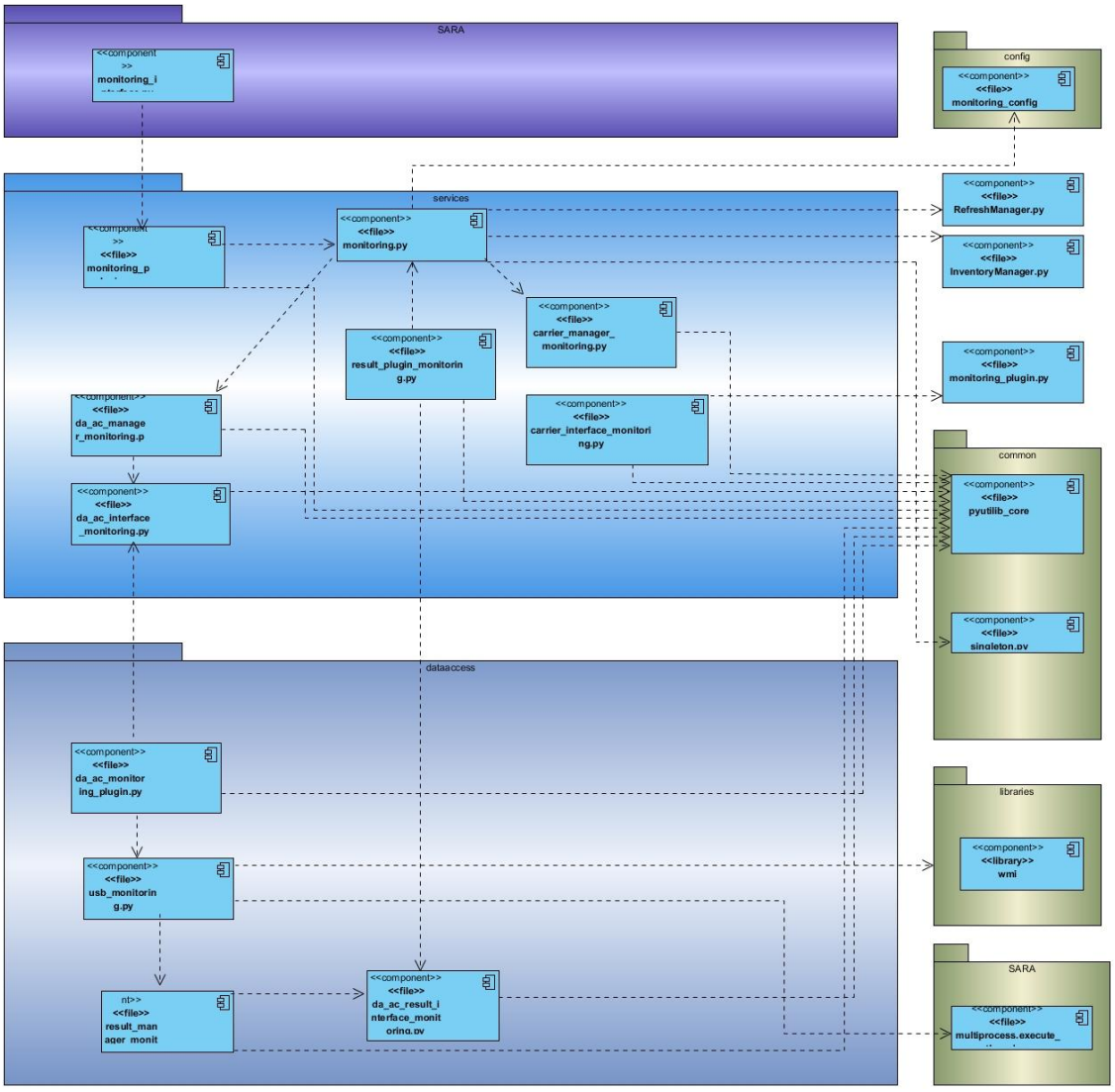


Figura 10: Diagrama de componentes bundle de monitoreo

En la figura anterior se muestran los diferentes componentes existentes en el bundle de monitoreo, como la biblioteca WMI y el paquete common que contiene los modulos pyutilib_core y singleton. También muestra las clases como pueden ser monitoring, usb_monitoring y carrier_manager_monitoring. Ademas

se muestran los ficheros de configuración y los componentes pertenecientes a otros módulos como moltiprocess_execute_thread perteneciente a SARA.

4.4 Niveles de prueba

<<Las pruebas son aplicadas para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo.

Se distinguen los siguientes niveles de pruebas:

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

4.5 Tipos de pruebas

Cada tipo de prueba tiene un objetivo específico y una técnica que lo soporte.

La siguiente lista muestra los tipos de pruebas:

- Funcionalidad:
- Usabilidad
- Fiabilidad
- Rendimiento
- Compatibilidad

4.6 Métodos de Prueba

Son dos fundamentales: el método de la caja negra y de la caja blanca.

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

En la prueba de la caja blanca del software se comprueba los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado.

4.6.1 Técnicas de pruebas de caja negra

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

- Técnica de la Partición de Equivalencia
- Técnica del Análisis de Valores Límites
- Técnica de Grafos de Causa-Efecto

4.6.2 Técnicas de pruebas de caja blanca

- Pruebas de flujo de control
- Pruebas de flujo de datos
- Pruebas de bifurcación
- Pruebas de caminos básicos

4.7 Estrategias de prueba

La estrategia de prueba describe el enfoque y los objetivos generales de las actividades de prueba. Incluye los niveles de prueba (unidad, integración, etc) a ser diseccionados y el tipo de prueba a ser ejecutadas (funcional, stress, etc), además del método de prueba (caja blanca o caja negra) y la técnica que se utiliza.

Capítulo 4.Impementación y Prueba

La estrategia de prueba trazada para el módulo GIIHW se basa en dos niveles de prueba:

a. Nivel de Unidad con el tipo de Funcionalidad, el método de caja blanca y la técnica PyUnit.

La prueba de unidad está enfocada a los elementos testeables más pequeños del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. Se aplica el tipo de prueba de funcionalidad con el método de prueba de caja blanca que requiere del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código. Mediante los métodos de prueba de la caja blanca, el ingeniero de software puede obtener casos de prueba que garanticen que se ejerciten por lo menos una vez todos los caminos independientes para cada módulo, todas las decisiones lógicas en sus vertientes verdaderas y falsa y las estructuras internas de datos para asegurar su validez, además que se ejecuten todos los bucles en sus límites y con sus límites operacionales. Para la prueba de caja blanca se utilizó el framework PyUnit.

Capítulo 4.Impementación y Prueba

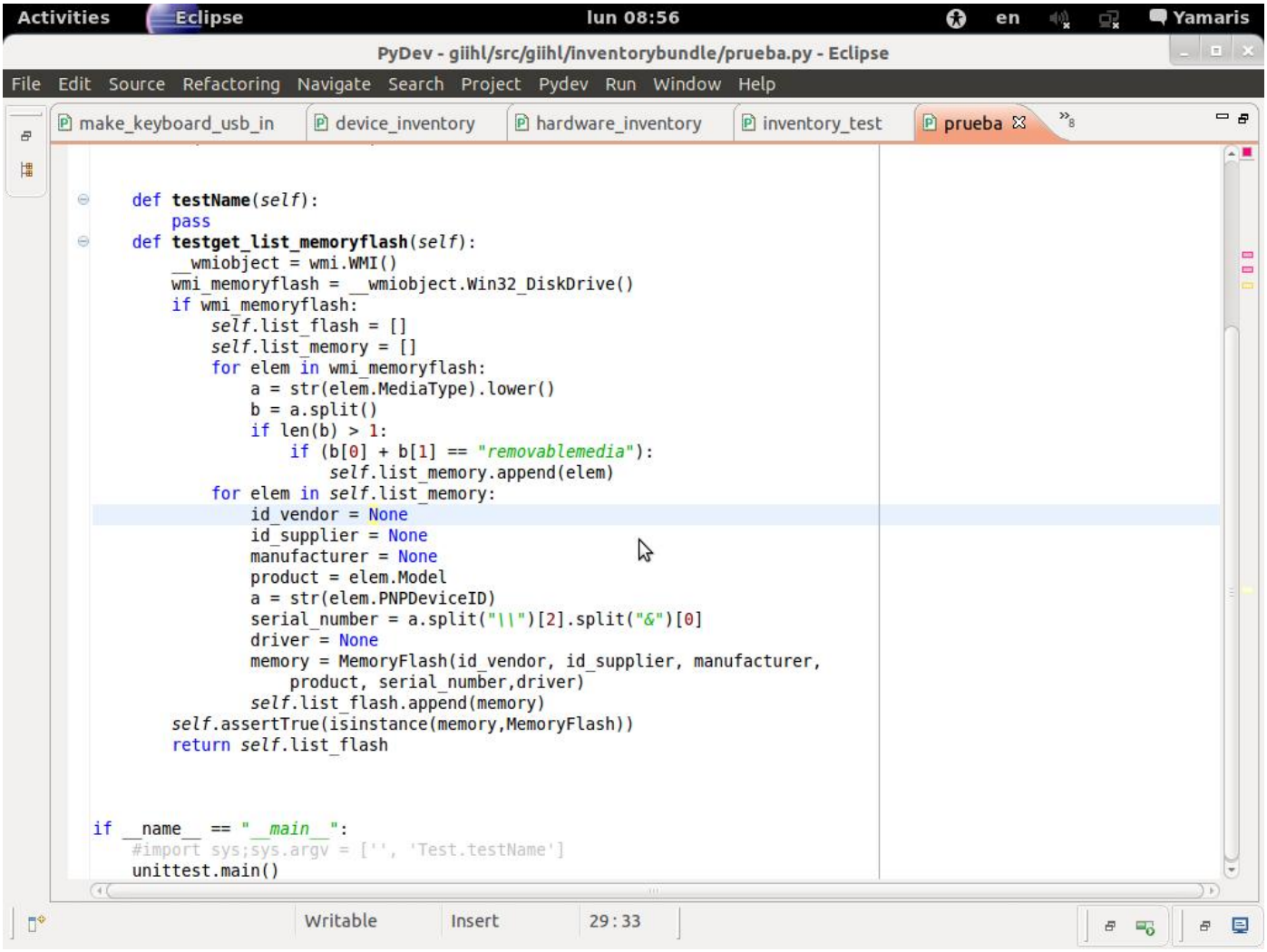


Figura 11: Técnica de PyUnit

b. Nivel de Integración con el tipo de Funcionalidad, el método de caja negra y la técnica Partición de equivalencia.

Capítulo 4.Impementación y Prueba

La prueba de integración comprueba la correcta unión de los componentes entre sí a través de sus interfaces, y si cumplen con la funcionalidad establecida, a pesar de que cada módulo funcione bien por separado es necesario probarlos conjuntamente, un módulo puede tener un efecto adverso o inadvertido sobre otro módulo. Las subfunciones, cuando se combinan, pueden no producir la función principal deseada, la imprecisión aceptada individualmente puede crecer hasta niveles inaceptables al combinar los módulos, los datos pueden perderse o malinterpretarse entre interfaces.

Por lo tanto, es necesario probar el software ensamblando todos los módulos probados previamente. Este es el objetivo de la pruebas de integración. A menudo hay una tendencia a intentar una integración no incremental; es decir, a combinar todos los módulos y probar todo el programa en su conjunto. El resultado puede ser un poco caótico con un gran conjunto de fallos y la consiguiente dificultad para identificar el módulo (o módulos) que los provocó. Dicha prueba de integración se realiza con el objetivo de validar la integración del módulo del presente trabajo con el resto de los módulos con que el mismo se relaciona.

Se aplica el tipo de prueba de funcionalidad con el método de prueba de caja negra que se llevan a cabo sobre el propio software, a través de la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. 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. Esta prueba se realiza mediante la técnica de partición de equivalencia a la cual fue sometida la aplicación, pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.>> (26)

Las tablas que a continuación se muestran reflejan las pruebas de integración que se les realizaron al software.

MGIIHW: Módulo de Gestión de Inventarios e Incidencias de Hardware en Windows.

SARA: Sistema de Administración de Recursos y Acciones.

MAAI: Módulo de Alarmas y Acciones ante Incidencias.

MGIISW: Módulo de Gestión de Inventarios e Incidencias de Software en Windows.

Módulo actual	Módulo integrado	Funcionalidad	Condiciones de ejecución	Escenario de prueba	Resultado previsto	Resultado real
MGIIHW	SARA	SARA manda a realizar el inventario de hardware.	La computadora está encendida. El sistema debe estar instalado en la computadora. Ambos módulos funcionan correctamente.	Realizar inventario de hardware correctamente.	Se espera que llegue la orden de realizar el inventario de hardware.	Llega la orden de realizar el inventario de hardware correctamente.
MGIIHW	SARA	SARA manda a actualizar configuraciones.	La computadora está encendida. El sistema debe estar instalado en la computadora. Ambos módulos funcionan correctamente.	Actualizar configuraciones correctamente.	Se espera que lleguen las configuraciones.	Llegan las configuraciones correctamente.

MGIIHW	SARA	MGIIHW envía los resultados al servidor.	La computadora está encendida. El sistema debe estar instalado en la computadora. Ambos módulos funcionan correctamente.	Enviar resultados correctamente.	Se espera que lleguen los resultados al servidor correctamente.	Se envían correctamente los resultados.
MGIIHW	MAAI	MGIIHW notifica las incidencias a MAAI.	La computadora está encendida. El sistema debe estar instalado en la computadora. Ambos módulos funcionan correctamente.	Notificar las incidencias correctamente.	Se espera que llegue a MAAI las incidencias con los siguientes datos: Nivel de incidencia Tipo Componente Descripción.	Las incidencias son notificadas correctamente a MAAI.

MGIIHW	MGIISW	MGIIHW obtiene usuario logueado de MGIISW.	La computadora está encendida. El sistema debe estar instalado en la computadora. Ambos módulos funcionan correctamente.	Obtener usuario logueado correctamente.	Obtener correctamente el usuario logueado.	Se obtuvo correctamente el usuario logueado.

Tabla 8: Pruebas de integración

4.8 Conclusiones

En este capítulo se describió la distribución del sistema en nodos, especificados en el diagrama de despliegue, además se presentaron los diagramas de componentes observándose las interfaces, componentes, paquetes o subsistemas y las relaciones entre sí. Se puede observar el desarrollo de las pruebas de integración realizadas a la aplicación, mostrando que dichas funcionalidades actúan correctamente sin presentar dificultades en su funcionamiento ya que realizan un correcto tratamiento de los datos con los cuales trabaja. Por lo que se llega a la conclusión de que se ha obtenido un satisfactorio resultado mediante las pruebas realizadas a la aplicación.

Conclusiones Generales

Para dar solución al problema planteado, ¿Cómo automatizar de forma integrada la obtención de la información, la gestión de los eventos y la detección de las incidencias de hardware en una computadora sobre el sistema operativo Windows?, se analizaron las aplicaciones de obtención de inventario de hardware existentes, se realizó el estudio de las herramientas y tecnologías de desarrollo que se utilizaron en la confección de la solución, donde fueron seleccionadas las que resultaron más adecuadas para elaborar el sistema. Se levantaron requisitos los cuales fomentaron las bases de los casos de uso que fueron útiles para elaborar el diseño del software apoyado también por los patrones de diseño que aportan soluciones concretas a problemas específicos, dejando todo listo para pasar a la fase de implementación. Como resultado se obtuvo una herramienta con las funcionalidades requeridas tales como realizar inventarios de hardware y la comparación entre los mismos, monitoreo de conexión y desconexión de dispositivos USB, notificación de incidencias al Módulo de Alarmas y Acciones ante Incidencias y el envío de inventarios, incidencias y trazas al servidor, dando cumplimiento al objetivo general de la investigación así como a los específicos.

Referencias Bibliográficas:

1. REAL ACADEMIA ESPAÑOLA. (s.f.). Obtenido de http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=inventario
2. Wiley, John.*Introduction to Client / Server Systems: A Practical Guide for Systems Professionals*
3. Ruíz, C. D. (Mayo 2010). Plataforma de Gestión de Servicios Telemáticos en GNU/Linux.
4. OCS Inventory Team. OCS Inventory NG | Home. 2011. [cited 6 December 2011]. Available from world wide web: <http://www.ocsinventory-ng.org/en/>
5. NetSupport. NetSupport DNA | Home. 2011. [cited 6 December 2011]. Available from world wide web: <http://www.netsupportdna.com/es/>
6. Alberto Fernades Francisco, Jarbas Teixeira. CACIC | Home. 2011. [cited 6 December 2011] Available from world wide web: <http://www.latinuxpress.com/books/drafts/cacic/caps/09.html>
7. HispaNetwork Publicidad y Servicios. glosarios.net | Home. 2011. [cited 6 December 2011] Available from world wide web: <http://tecnologia.glosario.net/terminos-tecnicos-internet/lenguaje-de-alto-nivel-985.html>
8. Alvarez, Miguel Angel. Lenguaje de programación de propósito general. [En línea] http://www.desarrolloweb.com/articulos/1325.php
9. editorbfb. (15 de febrero de 2011). ProgramacionDesarrollo.com. Obtenido de http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/
10. Eclipse. [En línea] http://www.webmaster-mexico.com/book/export/html/30
11. Rodríguez, L. S. (15 de diciembrede 2009). Panda3d, manual.
12. Quelopana, A. (Diciembre 2009). UNA PROPUESTA METODOLÓGICA PARA MODELAR PROCESOS DE NEGOCIO DE DECISIÓN BASADA EN UNA EXTENSIÓN A BPMN. Chile.
13. Booch Grady, Jacobson Ivar, Rumbaugh James. El Proceso Unificado de Desarrollo de Software. Capítulo 1 Páginas 1-12. [En línea] Addison Wesley , 2000
14. Rational Unified Process. [En línea] http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Referencias Bibliográficas:

15. Montalvo, F. J. (Diciembre de 2011). CONSTRUCCIÓN DE UNA HERRAMIENTA CASE.
16. Sierra, Maria. Trabajando con Visual Paradigmfor. [En línea] http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf
17. ArquiSoft. (s.f.). Grupo de Investigación Arquitecturas de Software. Obtenido de http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/index.php?id=81&type=1
18. Estilos Arquitectónicos. Definición. [En línea] http://www.buenastareas.com/ensayos/Estilos-Arquitect%C3%B3nicos/1019429.html.
19. Application Architecture Guide 2.0, J.D. Meier, Alex Homer, David Hill, Jason Taylor, PrashantBansode, Lonnie Wall, Rob Boucher Jr, AkshayBogawat. [En línea] Microsoft Corporation., 2008 .
20. Peláez, J. (26 de Mayo de 2009). Arquitectura basada en capas. Obtenido de http://www.juanpelaez.com/geek-stuff/arquitectura/arquitectura-basada-en-capas/
21. Honorio, R. M. (2008). APLICACIÓN ESTOCASTICA AL SISTEMA DE INVENTARIOS. LA PAZ – BOLIVIA.
22. Daniel Riesco. UML Diagrama de Clases y de Objetos. [En línea] 12 de 04 de 2010. http://sel.unsl.edu.ar/licenciatura/ingsoft2/UML-DiagramaClaseObjeto.pdf.
23. Gracia, J. (27 de Mayo de 2005). Diseño de Software Orientado a Objetos. Obtenido de http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php
24. Lanvin, D. F. (Junio 2004). Desarrollo de una metodologia para un nuevo paradigma de desarrollo de software.
25. Nieves, M. D. (Sep 2005). Maestria en matematica aplicada e informatica para la administracion. Holguin.
26. Gutiérrez, J. (s.f.). Introducción al Proceso de Pruebas. Obtenido de http://www.lsi.us.es/~javierj/cursos_ficheros/02.SR.pdf

Bibliografía:

Bibliografía

- Jacobson, I.; Booch, G. y Rumbaugh, J.; “El Proceso Unificado de Desarrollo de software”, Addison-Wesley, 2000
- REAL ACADEMIA ESPAÑOLA. (s.f.). Obtenido de http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=inventario
- Wiley, John.*Introduction to Client / Server Systems: A Practical Guide for Systems Professionals*
- Ruíz, C. D. (Mayo 2010). Plataforma de Gestión de Servicios Telemáticos en GNU/Linux.
- OCS Inventory Team. OCS Inventory NG | Home. 2011. [cited 6 December 2011]. Available from world wide web: <http://www.ocsinventory-ng.org/en/>
- NetSupport. NetSupport DNA | Home. 2011. [cited 6 December 2011]. Available from world wide web: <http://www.netsupportdna.com/es/>
- Alberto Fernades Francisco, Jarbas Teixeira. CACIC | Home. 2011. [cited 6 December 2011] Available from world wide web: <http://www.latinuxpress.com/books/drafts/cacic/caps/09.html>
- HispaNetwork Publicidad y Servicios. glosarios.net | Home. 2011. [cited 6 December 2011] Available from world wide web: <http://tecnologia.glosario.net/terminos-tecnicos-internet/lenguaje-de-alto-nivel-985.html>
- Alvarez, Miguel Angel. Lenguaje de programación de propósito general. [En línea] http://www.desarrolloweb.com/articulos/1325.php
- editorbfb. (15 de febrero de 2011). ProgramacionDesarrollo.com. Obtenido de http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/
- Eclipse. [En línea] http://www.webmaster-mexico.com/book/export/html/30
- Rodríguez, L. S. (15 de diciembrede 2009). Panda3d, manual.
- Quelopana, A. (Diciembre 2009). UNA PROPUESTA METODOLÓGICA PARA MODELAR PROCESOS DE NEGOCIO DE DECISIÓN BASADA EN UNA EXTENSIÓN A BPMN. Chile.
- Booch Grady, Jacobson Ivar, Rumbaugh James. El Proceso Unificado de Desarrollo de Software. Capítulo 1 Páginas 1-12. [En línea] Addison Wesley , 2000

Bibliografía:

- Rational Unified Process. [En línea] http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf
- Montalvo, F. J. (Diciembre de 2011). CONSTRUCCIÓN DE UNA HERRAMIENTA CASE.
- Sierra, Maria. Trabajando con Visual Paradigmfor. [En línea] http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf
- ArquiSoft. (s.f.). Grupo de Investigación Arquitecturas de Software. Obtenido de http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/index.php?id=81&type=1
- Estilos Arquitectónicos. Definición. [En línea] http://www.buenastareas.com/ensayos/Estilos-Arquitect%C3%B3nicos/1019429.html.
- Application Architecture Guide 2.0, J.D. Meier, Alex Homer, David Hill, Jason Taylor, PrashantBansode, Lonnie Wall, Rob Boucher Jr, AkshayBogawat. [En línea] Microsoft Corporation., 2008 .
- Peláez, J. (26 de Mayo de 2009). Arquitectura basada en capas. Obtenido de http://www.juanpelaez.com/geek-stuff/arquitectura/arquitectura-basada-en-capas/
- Honorio, R. M. (2008). APLICACIÓN ESTOCASTICA AL SISTEMA DE INVENTARIOS. LA PAZ – BOLIVIA.
- Daniel Riesco. UML Diagrama de Clases y de Objetos. [En línea] 12 de 04 de 2010. http://sel.unsl.edu.ar/licenciatura/ingsoft2/UML-DiagramaClaseObjeto.pdf.
- Gracia, J. (27 de Mayo de 2005). Diseño de Software Orientado a Objetos. Obtenido de http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php
- Lanvin, D. F. (Junio 2004). Desarrollo de una metodologia para un nuevo paradigma de desarrollo de software.
- Nieves, M. D. (Sep 2005). Maestria en matematica aplicada e informatica para la administracion. Holguin.
- Gutiérrez, J. (s.f.). Introducción al Proceso de Pruebas. Obtenido de http://www.lsi.us.es/~javierj/cursos_ficheros/02.SR.pdf

Glosario de Términos:

Activos Informáticos: Refiérase a dispositivos electrónicos conectados a la red (computadoras, impresoras, modem, router, etc.)

Administrador: Personal encargado de la seguridad y el control de los activos informáticos.

Bundle: Conjunto de plugin, interfaces y services que resuelven un problema específico.

Configuraciones de incidencias: Fichero que contiene todos los cambios que pueden ocurrir en el inventario y si constituye o no una incidencia.

Configuraciones de Monitoreo: Fichero que contiene si se va a realizar o no monitoreo de conexión o desconexión

Configuraciones de periodos de cambio: Fichero que contiene todos los cambios que pueden ocurrir en el inventario y un periodo de tiempo para cada uno de los cambios.

Correo: Servicio que permite la transferencia mensajes y paquetes.

GNU GPLv2: Licencia creada por la Free Software Fundation orientada a proteger la libre distribución, modificación y uso del software.

HTTP (Protocolo de transferencia de Hipertextos): Conjunto de reglas para transferir textos.

HTTPS: Es la versión del protocolo HTTP pero de forma segura, usando un cifrado basado en SecureSourcesLayer (SSL).

Incidencia: Información que se envía al servidor cando se detecta algún cambio en el inventario y este se encuentra comprendido en las configuraciones de incidencias y no se encuentra comprendido entre los períodos de cambio.

LAN: Interconexión de computadoras a una red en un área pequeña.

Linux: Conjunto de Sistemas Operativos libres.

MySQL: Sistema de Gestión de Base de Datos Relacional.

Glosario de Términos:

Perl: lenguaje de programación diseñado por Larry Wall en 1987.

Python: Es un lenguaje de alto nivel creado por Guido van Rossum a principios de los años 90.

Traza: Información que se envía al servidor cuando se realiza un inventario que contiene la hora, el usuario logueado, ect…

Unix: es un sistema operativo portable, multitarea y multiusuario.

WAN: Interconexión de computadoras. Diseñada para brindar red a un área mucho más grande que la de la LAN.

Windows: Sistema Operativo privativo hecho por Microsoft.

XML: Estándar de codificación de información.