FACULTAD 2

Personalización del Gestor de Recursos de Hardware y Software para la Universidad de las Ciencias Informáticas

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

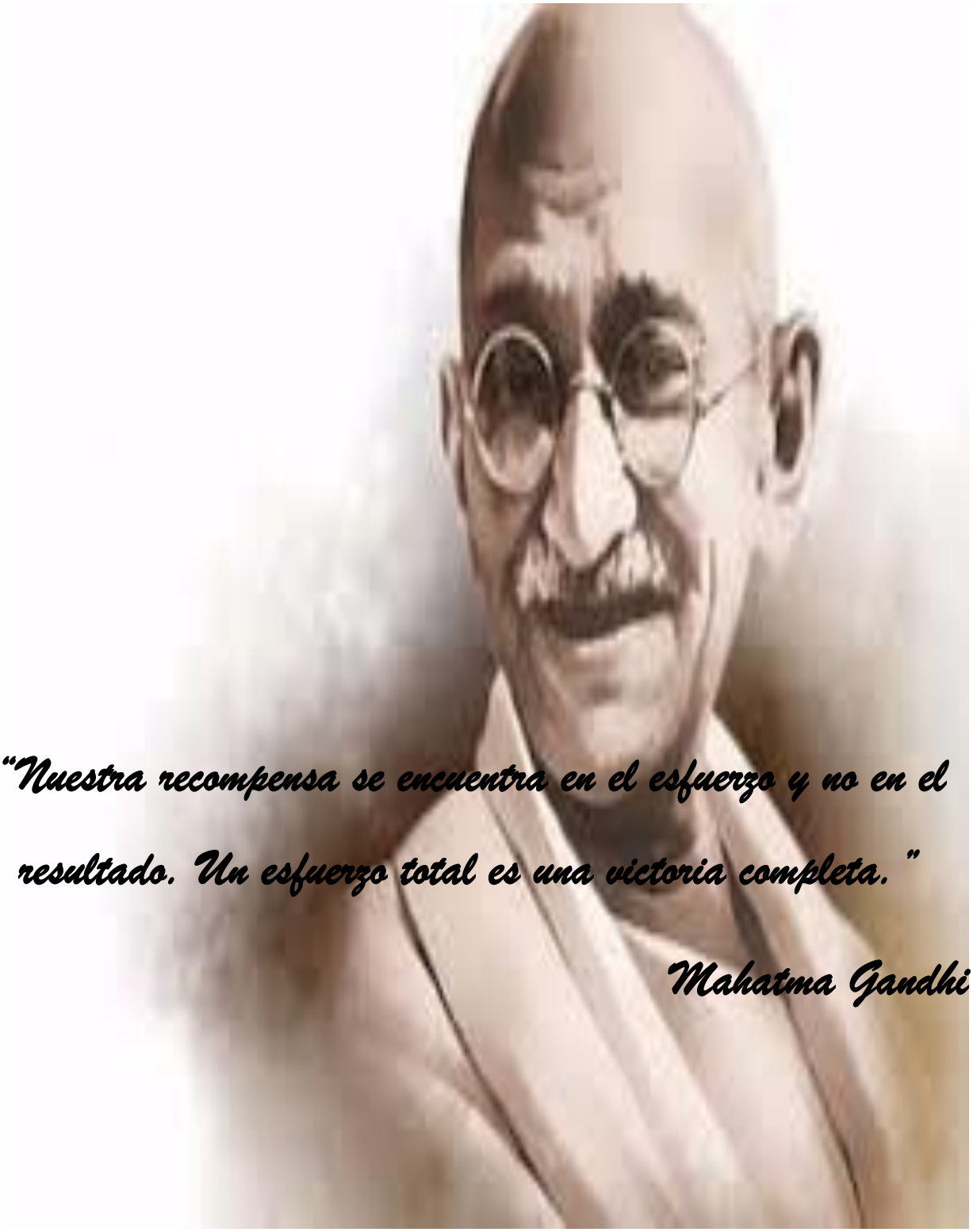
Autor: Reybel Domínguez Fuentes

Tutor: Ing. Jorge Armando Túnez González

Co-Tutora: Ing. Dannicel Tersilla Hidalgo

Consultante: MSc. Lianet Salazar Labrada

La Habana, 2018



"Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa."

Mahatma Gandhi

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

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

_____

Reybel Domínguez Fuentes

(Autor)

_____ _____

Ing. Jorge Armando Túñez González **Ing. Dannicel Tersilla Hidalgo**

(Tutor) **(Co-Tutora)**

Datos de contacto:

Autor:

Reybel Domínguez Fuentes

Universidad de las Ciencias Informáticas, La Habana, Cuba

E-mail: rdominguez@estudiantes.uci.cu

Tutor:

Jorge Armando Túñez González

Universidad de las Ciencias Informáticas, La Habana, Cuba

E-mail: janunez@uci.cu

Co-Tutora:

Dannicel Tersilla Hidalgo

Universidad de las Ciencias Informáticas, La Habana, Cuba

E-mail: dannicel@uci.cu

Agradecimientos

En primer lugar, quiero agradecer a mis padres por haber dado todo en estos cinco años y contribuir profundamente a mi formación como Ingeniero, por apoyarme en todas mis decisiones tomadas, comprenderme y ayudarme.

A mi madre por haberme dado la vida, sin ti hoy no pudiera estar aquí para agradecerte todos los sacrificios que has hecho por mí. Esta victoria te la debo a ti.

A mi hermana menor por existir, por siempre estar a mi lado, espero que veas en mí un ejemplo de superación. Te quiero mucho.

A todos mis familiares en general, que de una forma u otra contribuyeron durante todo este tiempo. Gracias

A mi novia Brenda, por todo su apoyo incondicional durante momentos decisivos, por aguantar mis berrinches, por toda tu ayuda brindada, por todo el amor y el cariño que me das, fuiste determinante en este último año. Gracias

A mis tutores y consultante por toda la ayuda brindada durante el desarrollo de este trabajo de diploma.

Quiero hacer un agradecimiento especial a mis compañeros de aula Wendy y Ramón, por toda la amistad brindada durante estos cinco años, por todos los buenos momentos vividos y las memorias que quedarán.

Dedicatoria

Este trabajo de diploma va dedicado a mis padres, sin ellos este bello momento nunca hubiera sido posible.

A ellos que son la luz de mis días, gracias por ser los mejores.

Resumen

El sistema de control interno en Cuba es de vital importancia para el correcto cumplimiento de los objetivos propuestos por las instituciones y velar por el cuidado de los activos fijos tangibles. Es necesario su aplicación en todos los sectores y actividades productivas en el ámbito nacional. El cuidado de los medios computacionales es una tarea fundamental para salvaguardar la economía del país y la información que estos activos son capaces de brindar. En respuesta a esto, en la Universidad de las Ciencias Informáticas se ha desarrollado el Gestor de Recursos de Hardware y Software, XILEMA GRHS, para velar y cumplir con un control sobre los activos de cómputo con que se cuentan.

En el presente trabajo, se desarrolla una personalización del Gestor de Recursos de Hardware y Software para la Universidad de las Ciencias Informáticas, que permitirá determinar la localización de cada uno de los ordenadores en el inmueble de la institución. Se adicionó un componente capaz de recopilar datos referentes a las características de hardware, software y el cumplimiento de las políticas de seguridad. Información que podrá ser accesible a todos los usuarios que dispongan de la aplicación cliente de XILEMA GRHS. Se incluyen nuevos mecanismos para determinar el estado en que se encuentran las computadoras. En la propuesta se utiliza como metodología de desarrollo de software AUP-UCI, los lenguajes de programación Python, HTML y JavaScript, además la herramienta de administración de base de datos pgAdmin para PostgreSQL y la base de datos SQLite III.

Palabras Clave: activos fijos, control, localización, XILEMA GRHS.

Abstract

The internal control system in Cuba is of vital importance for the correct fulfillment of the objectives proposed by the institutions and to ensure the care of tangible fixed assets. Is of necessary application in all the sectors and productive activities in the national scope. The care of the computational media is a fundamental task to safeguard the economy of the country and the information that these assets are able to provide. In response to this, the Hardware and Software Resources Manager, XILEMA GRHS, has been developed at the University of Information Sciences to watch over and comply with control over the computing assets that are counted. In response to this, the Hardware and Software Resources Manager, XILEMA GRHS, has been developed at the University of Information Sciences to watch over and comply with control over the computing assets that are counted.

In the present work, a customization of the Resource Manager of Hardware and Software for the University of Computer Science is developed, which will allow to determine the location of each of the computers in the building of the institution. A component capable of collecting data referring to the characteristics of hardware, software and compliance with security policies was added. This aforementioned information may be accessible to all users who have the XILEMA GRHS client application. New mechanisms are included to determine the state of computers. In the proposal is used as software development methodology AUP-UCI, Python, HTML and JavaScript programming languages, also the tool of database management pgAdmin for PostgreSQL and SQLite III database.

Keywords: fixed assets, control, location, XILEMA GRHS.

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

En los últimos años los sistemas de información masiva han sufrido cambios cuya vertiginosidad y complejidad no admite precedentes. La sociedad actual se encuentra totalmente inmersa en una revolución tecnológica basada en la informática y liderada por Internet, en la que se relacionan los medios de comunicación, la educación y las tecnologías, quedando conformada las Tecnologías de las Información y las Comunicaciones (TICS). Este elemento fundamental encuentra su principal impulso en el acceso y procesamiento de información y tributa a que empresas, instituciones, pueblos y hasta países desarrollen aceleradamente todas sus habilidades tecnológicas para así alcanzar sus verdaderos potenciales. Sin lugar a dudas, la base de todo lo tecnológico que hoy se conoce, se centra en la computadora como elemento principal de interacción de la información.

"La computadora no es más que una máquina electrónica que […] permite almacenar y tratar información" (RAE 2017), que se puede adquirir como conocimiento a través del aprendizaje. El conocimiento es una de las capacidades más importantes del ser humano, constituye una herramienta que permite tomar decisiones ajustadas a lo que se aprende y se puede deducir. Esto aplica para empresas e instituciones, que han desarrollado su máximo potencial, a través del intercambio cognoscitivo entre individuos basándose en la computación. La informática aplicada a entidades, ha propiciado la solución de problemas mediante el empleo de programas y algoritmos informáticos, evidenciado en la informatización de procesos empresariales como la gestión de los activos fijos tangibles.

Las empresas utilizan y tienen como base a los activos fijos, para poder brindar bienes y servicios. Los AFT (Activos Fijos Tangibles) son aquellos bienes de larga duración que son palpables y cuyo tiempo de vida útil estimado no es menor de un año (Definiciones-de 2010). Los ordenadores dentro de esta categoría, conforman uno de los medios más importantes en el que a través de una organizada gestión de los AFT deben ser protegidos, para poder garantizar toda la información que manejan.

Actualmente, se han desarrollado sistemas de planificación de recursos empresariales para gestionar, administrar y controlar los procesos y recursos en entidades. Los mismos son catalogados sistemas ERP (del inglés *Enterprise Resourse Planning*) y constituyen una herramienta global de planificación de recursos y de gestión de la información. Permiten conocer el estado actual de las entidades y realizar un inventario de todos los activos fijos tangibles (Ángelo Benvenuto Vera 2012).

A nivel mundial la organización y control de los AFT se rige por la Norma Internacional de Contabilidad (NIC). Esta norma constituye un conjunto de estándares que establecen la información que

debe presentarse y la forma en que debe aparecer en los estados financieros[1]. Su objetivo es reflejar la esencia económica de las operaciones del negocio y presentar una imagen fiel de la situación financiera de una empresa (Accid 2005).

En Cuba, los AFT requieren de un control a partir de un sistema de modelos que muestran su destino y uso, atendiendo a lo establecido en la Resolución 60/2011[2] de la Contraloría General de la República[3] acerca del sistema de control interno. Esta resolución plantea un conjunto de normas de obligada observancia para el país a tono con la política económica y social de necesaria implementación. El cumplimiento del Sistema de Control Interno en base a los requerimientos del desarrollo económico y administrativo del país, es un tema primordial en las empresas, organismos y entidades (Contraloría General de la República de Cuba 2011).

Muchas organizaciones gestionan sus recursos empresariales a través de los sistemas ERP. Este es el caso de la Universidad de las Ciencias Informáticas (UCI) la cual utiliza el sistema Assets NS para gestionar todos los recursos institucionales de forma integrada. Assets NS dispone de métodos para la administración, planificación de inventarios, análisis y consultas que permiten obtener y conocer con exactitud la situación actual de la entidad. La gestión de inventario de activos se aplica en la universidad con el propósito de registrar todos los medios básicos con que se cuentan, entre ellos los ordenadores (Assets NS 2018). Sin embargo, el sistema no es capaz de recoger las especificaciones de los recursos de cada computadora como memoria RAM, placa base, disco duro, entre otros bienes costosos que resultan de gran relevancia para la institución y pueden ser extraídos o desviados fácilmente.

La UCI cuenta con varios centros destinados a la producción de software, entre los que se encuentra el Centro de Telemática, orientado al ámbito de telecomunicaciones y seguridad informática (UCI 2018). En dicho centro se desarrolló el Gestor de Recursos de Hardware y Software (XILEMA GRHS). Esta aplicación permite realizar un inventario de todos los ordenadores incluyendo los recursos de hardware y software de cada uno de ellos. Además, es posible auditar que cada uno de estos medios registrados en el inventario cumpla con las políticas de seguridad informática que rigen la universidad. En el registro de inventario que recoge la plataforma se obtienen datos que son capaces de identificar cada computadora. La localización

[1] **Estado financiero:** Informe o registro de tipo formal que suelen utilizar las empresas, personas y entidades, para tener constancia de las diferentes actividades económicas que realizan las mismas.

[2] **Resolución 60/2011:** Forma parte de un conjunto de medidas tomadas por el Estado y el Gobierno cubano con el objetivo de lograr una cultura del control. Constituye una regulación aplicable al sistema organizacional cubano.

[3] **Contraloría General de la República:** Organización creada en el año 2009. Forma parte del proceso de institucionalización del país, así como el fomento de la gestión gubernamental.

es uno de los datos registrados y se determina a partir de la subred donde se encuentre el ordenador, que a su vez está asociada a un área de responsabilidad material (Rodríguez Adrián E Mena 2015).

Actualmente el proceso de asociación subred-área en XILEMA GRHS es realizado de forma manual por el administrador local, provocando la inserción de información incorrecta debido a errores humanos. Además, existe falta de información cuando no encuentra una relación entre una subred y un área de responsabilidad material. Por otra parte, las subredes no se encuentran distribuidas en atención a la relación subred-área, puesto que existen varias áreas de responsabilidad material contempladas dentro de una misma subred. Por consiguiente, en el sistema se muestran localizaciones duplicadas, lo que afecta la fiabilidad de la información mostrada en XILEMA GRHS.

Los ordenadores disponen de una vida útil determinada y pueden quedar obsoletos debido a su desgaste físico. Al ser dados de baja quedan almacenados en el sistema XILEMA GRHS con estado inactivo, lo cual no necesariamente significa que una computadora se encuentre de baja, por lo que no existe un mecanismo viable para determinar la disponibilidad física de cada uno de estos medios. Esta problemática tributa a que un determinado local, tenga asociado computadoras adicionales cuando en realidad son inexistentes, lo que dificulta realizar un correcto control de los bienes distribuidos por locales, ya que la información brindada por el sistema no es completamente fiable.

La información manejada por XILEMA GRHS es objeto de cuidadosa protección. El sistema se encuentra protegido de acceso no autorizado, a través de un nivel de acceso definido por diferentes roles con permisos asignados, como administradores de áreas, jefes de tecnología, directivos entre otros cargos. La falta de estandarización y legibilidad de los datos dificulta la realización de reportes estadísticos para la toma de decisiones. Los reportes se generan de acuerdo a diferentes criterios como por ejemplo la localización y el estado.

XILEMA GRHS también permite conocer qué políticas de seguridad informática cumple cada ordenador registrado en el sistema. Los usuarios corrientes que tienen asignado una computadora para su trabajo diario son los máximos responsables del correcto cumplimiento de las mismas. Al no disponer de acceso a XILEMA GRHS, resulta engorroso revisar manualmente que el ordenador cumpla con todas las políticas de seguridad establecidas y conocer las características de hardware y software del mismo.

Según la problemática anteriormente planteada se define como **problema a resolver:** ¿Cómo garantizar la fiabilidad y accesibilidad de la información referente a los ordenadores?

El problema antes mencionado se centra en el **objeto de estudio:** Los datos ofrecidos por XILEMA GRHS y el sistema Assets NS.

Teniéndose como **objetivo general:** Desarrollar una personalización de XILEMA GRHS para la UCI que permita la reducción de inconsistencias en la asignación de localizaciones y el acceso local de los usuarios al inventario de hardware, software y el cumplimiento de las políticas de seguridad.

Para ello se identifica como **campo de acción**: Los datos referentes a la asignación de localizaciones y el acceso de usuarios a la información de los componentes de hardware, software y políticas de seguridad.

Para dar solución al objetivo planteado se proponen como **tareas de la investigación:**

- Diagnóstico del estado actual de la gestión de inventarios para conocer sus características específicas.
- Estudio del proceso de inventariado de los activos informáticos en XILEMA GRHS para identificar y analizar los problemas que presenta.
- Análisis de los mecanismos que utilizan sistemas homólogos a XILEMA GRHS en la obtención de datos para identificar cuáles podrían ser utilizados en la solución.
- Diagnóstico de los sistemas de gestión integral de tipo ERP para evaluar qué ventajas podrían brindar.
- Análisis de la información brindada por los sistemas de tipo ERP con respecto a los activos informáticos para determinar la información complementaria que pueden ofrecer.
- Análisis de las herramientas y tecnologías para determinar cuáles de ellas podrían ser de mayor utilidad en la implementación del sistema.

Para dar cumplimiento a las tareas propuestas se tuvo en cuenta el estudio de varios métodos de investigación:

Métodos Teóricos de la Investigación:

- **Histórico-Lógico:** Es utilizado para analizar la evolución histórica de las soluciones similares de proceso de inventariado de hardware y software, las tendencias actuales y su utilización en Cuba para determinar si resuelven el problema planteado.
- **Analítico-Sintético:** Se emplea para analizar la documentación existente relacionada con el tema y extraer los elementos más importantes y necesarios de manera que permita sintetizar todo lo obtenido.
- **Modelación:** Es aplicado para modelar los procesos del negocio en cuestión y los elementos necesarios para la implementación.

Métodos Empíricos de la Investigación:

- **Entrevista:** Es una técnica usada para obtener la información necesaria de parte de los especialistas de la Dirección de Informatización de la Universidad de Ciencias Informáticas, y qué puntos específicos del Gestor de Hardware y Software se quieren perfeccionar.
- **Observación:** Se refleja durante todo el transcurso de la investigación, fundamentalmente para observar el proceso de obtención de datos de XILEMA GRHS y tomar experiencias de aplicaciones similares.

En aras de estructurar el proceso de desarrollo del presente trabajo de una manera coherente y organizada, se dividió el mismo en un total de tres capítulos:

Capítulo 1. Fundamentación teórica de la investigación: Se brinda una descripción de los conceptos asociados al proceso de inventariado de hardware y software. Además, se muestra un estudio del estado del arte de las principales herramientas utilizadas para la gestión de recursos empresariales y el inventario de activos de cómputo. También se proporciona una descripción de las herramientas, tecnologías y metodología de desarrollo de software a utilizar para dar solución al problema planteado.

Capítulo 2. Propuesta de solución: Se presenta una explicación de la solución propuesta y una descripción de los procesos del modelo del sistema. Se especifican las funcionalidades de la herramienta y propiedades del producto y se brinda una descripción de los casos de uso.

Capítulo 3. Implementación y pruebas: Se especifica la arquitectura del sistema, los patrones de diseño empleados y se muestran los resultados de las pruebas realizadas al software obtenido.

Capítulo 1: Fundamentación Teórica de la Investigación

1.1 Introducción

El presente capítulo está destinado a brindar una breve descripción de los conceptos asociados al proceso de inventariado de hardware y software. Además, se realiza un estudio para analizar el proceso de obtención de datos de herramientas similares a XILEMA GRHS y sistemas ERP, con el objetivo de conocer qué mecanismos e información complementaria podrían ser de ayuda para dar solución a la problemática planteada. También se define la metodología de desarrollo que guía la solución y el conjunto de herramientas y tecnologías a utilizar para el posterior desarrollo del software.

1.2 Gestión de inventario y control de bienes

La gestión de inventarios es el proceso que garantiza a una organización la total disponibilidad de los productos que necesita manteniendo los costos lo más bajo posible, por lo que se convierte en una actividad clave para salvaguardar los bienes y la economía de las entidades («Inventory Management Definition & Example InvestingAnswers» 2018). Este proceso tiene estrecha relación con el control de recursos empresariales, que no es más que la comprobación o inspección de los diferentes bienes pertenecientes a una empresa.

1.3 Gestor de Recursos de Hardware y Software

Gestor de Recursos de Hardware y Software (XILEMA GRHS), es una aplicación web desarrollada por el Centro de Telemática de la Universidad de las Ciencias Informáticas. Su principal función es obtener un inventario de todos los medios informáticos conectados a una red de computadoras. Envía notificaciones de incidencias mediante correos electrónicos. Todas sus aplicaciones están diseñadas usando la ingeniería basada en componentes. Está escrita en tecnologías libres como Python, Django y PostgreSQL. Para la comunicación entre aplicaciones usa el protocolo AMQP[4] y HTTP/HTTPS[5] en conjunto con REST[6]. Está diseñado para las plataformas Microsoft Windows y GNU/Linux. Permite su aplicación en varios entornos con características específicas por su flexibilidad de configuración. La modularidad de sus aplicaciones

[4] **AMQP:** *(Advanced Message Queuing Protocol),* es un protocolo de estándar abierto que se encuentra en la capa de aplicaciones de un sistema de comunicación.

[5] **HTTP/HTTPS:** *(Hypertext Transfer Protocol),* es un protocolo de comunicación que permite las transferencias de información a través de un navegador.

[6] **REST:** *(Representational State Transfer),* es un estilo de arquitectura para diseñar aplicaciones en red.

permite adicionar nuevas funcionalidades y desactivar algunas de las existentes (Rodríguez Adrian E Mena 2015).

XILEMA GRHS maneja datos referentes a las computadoras que conforman un amplio y detallado inventario, algunos de los datos más importantes son las características de hardware y software, el cumplimiento de políticas de seguridad, y las localizaciones de cada uno de los medios.

1.3.1 Obtención de datos de XILEMA GRHS

Para el proceso de obtención de todos los datos antes mencionados, XILEMA GRHS utiliza su aplicación Gclient quien le envía información a Gserver sobre cada uno de los clientes donde fue instalada. A continuación, se detalla cómo se adquiere y determina información relevante para el usuario.

Localización:

Entre los datos obtenidos por XILEMA GRHS podemos destacar la localización de cada uno de los medios tecnológicos. Gclient envía la dirección IP de cada uno de los clientes en la que Gserver tiene configurado de forma predeterminada todas las subredes asociadas a cada uno de los locales con que cuenta el inmueble. La Figura 1, describe de forma más detallada cómo es obtenida la localización de cada uno de los medios.

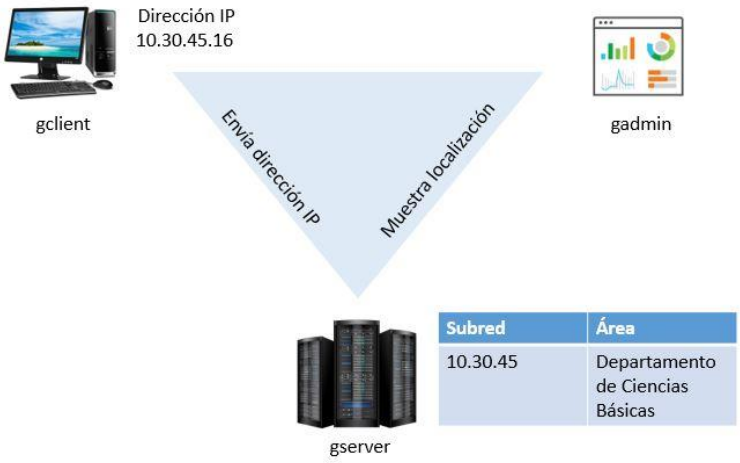


Figura 1: Relación subred-área

Fuente: Elaboración propia

Los encargados de definir en Gserver qué subredes pertenecen a las determinadas áreas, son los administradores locales. Actualmente, en XILEMA GRHS existen una gran cantidad de activos que no cuentan con ninguna localización establecida por los administradores, también existen muchas

localizaciones que se encuentran duplicadas ya que existen muchas subredes que abarcan más de un área. Cuando se produce un caso de este tipo, el sistema solamente seleccionará como localización la primera que detecte. Por ejemplo: La subred 10.22.32 está asociada al área Cátedra de Ajedrez, pero resulta que dicha subred abarca, además, el área de Cátedra de Matemática y Librería. En este caso los activos informáticos que se encuentren en las dos últimas áreas mencionadas, se registrarán en el sistema con la localización de Cátedra de Ajedrez.

Políticas de Seguridad:

El acceso al sistema se encuentra restringido para la mayoría de los usuarios, debido a la gran importancia que tiene toda la información almacenada en el sistema. Entre los datos que se muestran, se puede contemplar un diagnóstico del cumplimiento de las políticas de seguridad de cada uno de los activos. A continuación, se listan los datos que se muestran:

- Antivirus
- Carpetas compartidas
- Protección de datos
- Cortafuegos
- Bloqueo de pantalla
- Grupos de administradores

Los usuarios que cuentan con una estación de trabajo, les resulta complejo conocer si su computadora cumple con las políticas de seguridad que rigen a la universidad, ya que tendrían que revisar cada una de ellas y confirmar que están en orden.

En versiones inferiores a la 2.0 del Gestor de Recursos de Hardware y Software, se contaba con una interfaz de acceso enfocada al cliente, lo que permitía a los usuarios conocer un diagnóstico en tiempo real del cumplimiento de las políticas de seguridad y adherirse a ellas. El componente formaba parte de la aplicación Gclient de XILEMA GRHS y fue retirado por cuestiones de seguridad informática. El puerto 8418 por el cual se accedía a la interfaz permanecía abierto, lo que representaba una brecha de seguridad informática vulnerable a ataques desde la red.

Estado:

Cada uno de los agentes registrados en Gserver, contiene el parámetro Estado. Fue incluido con el objetivo de verificar el tiempo de inactividad de cada uno de los colectores. Hoy en día, existen muchos agentes registrados en el sistema con estado inactivo, que, en efecto, no es más que un conjunto de ordenadores que culminaron su vida útil y fueron dados de baja. El sistema no proporciona ningún

mecanismo e interfaz para determinarlo. Actualmente, existen locales que se encuentran registrados en el sistema con medios adicionales cuando en realidad son inexistentes, viéndose afectada la gestión de inventario de equipos de cómputo.

1.4 Sistemas similares al Gestor de Recursos de Hardware y Software

OCS Inventory NG:

OCS Inventory NG es una herramienta para la automatización de control de inventario realizado al hardware de varias computadoras en una estación de trabajo. De origen francés, fue creada en el año 2001. OCS transmite información tanto sobre las características de hardware de cada estación de trabajo, como sobre el software instalado allí. Todos estos datos se pueden ver en una interfaz web con funciones de exportación. Los diccionarios de software también se pueden definir para agrupaciones (como las actualizaciones de Windows). Con respecto a la implementación remota, OCS puede administrar instalaciones de software en estaciones de trabajo Windows, Mac o Linux, por medio de paquetes creados por los administradores. Está basado principalmente en tecnologías como Perl y MySQL. No es una herramienta que se considere activa, puesto que no realiza ningún tipo de acciones ante incidencias, como tampoco envía notificaciones a determinados usuarios (OCS Inventory 2018).

Al igual que XILEMA GRHS el acceso a la consola de administración por parte de los usuarios se realiza definiendo diferentes roles con los permisos asignados para cada uno. En la consola de administración es donde se puede encontrar toda la información colectada por la aplicación, que funciona a través de agentes instalados en ordenadores clientes. Los agentes no disponen de ninguna interfaz para conocer sus características específicas en materia de seguridad informática y propiedades del hardware y software.

Entre la información disponible de los activos en OCS Inventory, no se ha podido encontrar un mecanismo que defina las localizaciones de cada uno de los medios. OCS Inventory además de ser una herramienta para el control de los ordenadores, está enfocado en el descubrimiento por red vía IPDiscover. En el **Anexo I** se puede encontrar un listado más detallado del inventario realizado por OCS Inventory.

Loginventory:

Es una herramienta que realiza inventario de hardware y software en la que no es necesario la instalación de agentes para la obtención de la información, aunque también dispone de esta opción. Es un software que, para un correcto funcionamiento, requiere de altos requisitos de hardware en servidores. Permite realizar inventario de routers, switches, impresoras y hubs. Es una herramienta gratuita siempre y cuando la red no exceda los 20 ordenadores (LOGINventory 2018).

Loginventory es una herramienta que se encuentra centralizada en un solo servidor, donde sólo puede ser visualizada por el administrador del área, que debe tener instalado la aplicación de consola de administración. Aunque la aplicación permite administrar los roles de usuario con acceso al sistema y los permisos asignados a cada grupo de usuarios, no es accesible desde cualquier computadora de la red. En caso de utilizar agentes para la gestión de inventarios, Loginventory no proporciona una interfaz informativa para que cada uno pueda conocer detalles característicos en referencia al hardware, software y seguridad del dispositivo. Además, permite establecer una estructura jerárquica de todo el inventario realizado en el cual permite asignar localizaciones de forma manual a grupos específicos de medios.

Luego de un análisis de los parámetros analizados en el inventario de Loginventory, no se pudo constatar que dicha aplicación obtuviera la localización de los activos de TI[7] que en ella se gestionan de forma automática.

1.5 Sistemas de Planificación de Recursos Empresariales

Los sistemas de planificación de recursos empresariales permiten gestionar diferentes áreas empresariales de forma integrada. Sus siglas ERP, provienen del término en inglés *Enterprise Resource Planning*. Por lo general este tipo de sistemas está compuesto de módulos como Recursos Humanos, Ventas, Contabilidad y Finanzas, Compras y Producción que brindan información cruzada e integrada de todos los procesos del negocio. Este software debe ser parametrizado y configurado para responder a las necesidades específicas de cada organización. Una vez implementado, un ERP permite a los empleados de una empresa administrar los recursos de todas las áreas, simular distintos escenarios y obtener información consolidada en tiempo real, permitiendo tomar futuras decisiones. Disímiles ventajas son brindadas por sistemas de este tipo, pero entre las más valiosas se puede destacar, la posibilidad de disponer de información confiable, precisa y oportuna (Ángelo Benvenuto Vera 2012).

La información es el factor principal por la cual diferentes entidades optan por la implementación de sistemas ERP. Los datos se perciben, se integran y generan la información necesaria para producir el conocimiento que finalmente permite tomar decisiones favorables, tributando a salvaguardar el presupuesto de la entidad.

Antecedentes:

El origen de los sistemas ERP data de la década de los 70, cuando se comenzó a utilizar un software llamado MRP (*Material Requirement Planning*), cuyo objetivo era planificar todos los requerimientos de

[7] **TI:** Tecnologías de la información.

materia prima dentro de las organizaciones. A principios de la década de los 80 aparecen los sistemas de planificación de recursos de fabricación MRP II (*Manufacturing Resource Planning*), con el cual se pretendía contrastar la disponibilidad de recursos necesarios para la ejecución de las órdenes de producción planificadas. Durante los años 90 hubo intentos de integración de la gestión de la empresa, iniciativas como la denominada BRP (del inglés *Business Resources Planning*), sin embargo, este proceso puede considerarse característico de la década de los 90, en la que termina por imponerse la denominación ERP (Farro, M. 2007).

Assets NS Sistema de Gestión Integral:

Assets NS es un Sistema de Gestión Integral estándar, que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos Fijos, Útiles y Herramientas y Recursos Humanos. Todos sus módulos trabajan en estrecha relación, generando automáticamente al módulo de Contabilidad los comprobantes de operaciones por cada una de las transacciones efectuadas, esto permite que se pueda trabajar bajo el principio de Contabilidad al Día. Cuenta con más de 400 clientes en toda Cuba, siendo uno de los sistemas ERP más utilizados en el país (Assets NS 2018).

La Universidad de las Ciencias Informáticas gestiona muchos de sus procesos con el sistema Assets NS, como es el caso de la gestión de activos fijos tangibles. Este sistema, provee los mecanismos necesarios para llevar un correcto control y gestión de los AFT. Es posible inventariar todos los medios contemplados en la institución desde útiles y herramientas hasta los ordenadores como uno de los activos más importantes.

Los datos que maneja este sistema sobre los ordenadores están enmarcados en una breve descripción sobre los mismos sin especificidades, mencionando solamente algunos de los componentes que identifican a estos medios básicos, a pesar de que estos están compuestos de varios recursos de gran valor económico. Assets, también brinda otras informaciones valiosas como la localización de cada uno de los activos en el inmueble de la universidad. La localización está dada por Área de responsabilidad material y Centro de costo que determinan de manera precisa su ubicación. Los datos son introducidos y modificados en el sistema de forma manual.

A continuación, se muestra un ejemplo de los datos que brinda el sistema Assets NS sobre cada uno de los activos:

Tabla 1: Datos obtenidos del sistema Assets

rótulo	nombre_de_activo	área_responsabilidad_material	centro_de_costo	activo
MB00101443	P8H61-MLX/i3 3.30MGhz/1TB/DDR3-1333 4GB/DVDRW/MCar	LABORATORIO 305 DOC – 1	Centro de Telemática (SITEL)	True

SAP (System Applications Products in Data Processing):

SAP es una herramienta que posee diversidad de módulos para áreas específicas de empresas como manufactura, ventas o incluso en servicio al cliente. El proceso de implementación de SAP puede extenderse más de lo considerado, debido a los cambios que genera la preparación de los equipos, usuarios y procesos que la empresa posea; esto se debe a la complejidad con la que está construido el programa. Es un producto privado y el precio de compra es sumamente alto. El tiempo de implementación y ejecución de la herramienta es elevado (Boeder y Groene 2014).

La herramienta SAP requiere de una adaptación adecuada en el entorno donde se desea desplegar y debe ser preparada para obtener el máximo provecho del sistema. Dispone de varios módulos para la administración empresarial como Finanzas, que, a su vez, contiene varios sub-módulos como AA (Assets Accounting – Contabilidad de Activos), en el que se pueden administrar de forma integrada los activos fijos tangibles con que cuenta la institución. SAP gestiona la localización de los AFT, una vez adaptada al inmueble, es decir, debe contar con un registro de toda la infraestructura institucional (Áreas de responsabilidad, Centro de costos y pequeños locales) para poder gestionar la localización de los medios.

Odoo:

Odoo es un sistema integrado de gestión empresarial (ERP) de código abierto y sin coste de licencias que es capaz de cubrir las necesidades de las áreas de grandes, medianas y pequeñas empresas. Este sistema ERP ha sido creado por la compañía belga Odoo S.A. (antiguamente OpenERP S.A. y fundada en 2004) y se declara como alternativa a otros sistemas de código propietario como SAP o Microsoft Dynamics.

La aplicación ERP Odoo, entre las disímiles funcionalidades que propone se encuentra la Gestión de Inventario que funciona a través de entradas realizadas por los usuarios. Odoo debe ser cuidadosamente desplegado en las entidades para poder aprovechar al máximo todas las potencialidades que brinda. La infraestructura con que cuenta la institución debe ser llevada a la aplicación para así poder gestionar de una

forma eficaz los activos fijos. Entre la información gestionada por Odoo se encuentra la localización que es administrada manualmente por los usuarios (Peñas López, Ana 2016).

1.6 Conclusiones del estudio realizado

Luego de realizado un estudio de los diferentes sistemas similares al Gestor de Recursos de Hardware y Software y sistemas de tipo ERP, se obtuvieron los siguientes resultados: Los sistemas analizados no cuentan con un mecanismo de asignación automática de localizaciones según el inmueble donde se encuentre desplegado. Loginventory, cuenta con asignación de localizaciones que permiten realizar consultas y obtener información, pero estas deben ser asignadas a grupos de medios manualmente por el usuario. La infraestructura para todas las instituciones, empresas entre otras entidades, suelen estar conformadas de forma muy particular, no existe un estándar o patrón por el cual aplicaciones de este tipo se suelen regir para su estructuración.

Todas las empresas e instituciones están conformadas por una diferente distribución del inmueble, es una de las características que las diferencian. No podríamos afirmar que un software de tipo ERP desplegado para la Universidad de las Ciencias Informáticas podría funcionar de la misma forma en cualquier otra institución. Los sistemas ERP se caracterizan por contar con implementaciones que proporcionan grandes gastos de tiempo en preparación del personal y del ambiente de despliegue. Son sistemas privativos, y en su mayoría de alto costo adquisitivo. Actualmente, en la Universidad de las Ciencias Informáticas se encuentra ya desplegado y en pleno funcionamiento el sistema Assets NS, de tipo ERP, el cual se puede complementar con XILEMA GRHS, para así solucionar algunos de los problemas actuales que presenta. El sistema Assets NS, está adaptado a la infraestructura del inmueble de la universidad, por lo tanto, cuenta con un registro de todos los Centros de Costo y Áreas de responsabilidad material con que cuenta la institución.

Entre las aplicaciones analizadas similares a XILEMA GRHS, no se pudo constatar que contaran con un componente específico relacionado con los agentes, para mostrar toda la información de hardware, software y políticas de seguridad informática a todos los usuarios pertenecientes a la entidad.

A partir de lo anteriormente expuesto, se propone integrar la aplicación XILEMA GRHS con el sistema ERP Assets NS desplegado en la Universidad de las Ciencias Informáticas, para solucionar errores presentados en la asignación de localizaciones. Además, se propone desarrollar un componente capaz de mostrar en cada cliente vía web todas las características de hardware, software y políticas de seguridad.

1.7 Metodología de desarrollo de software

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

- Desarrollo dirigido por pruebas
- Modelado ágil
- Gestión de cambios ágil
- Refactorización de base de datos para mejorar la productividad

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

Fases AUP:

- Inicio: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
- Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
- Transición: El sistema se lleva a los entornos de producción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción (Charles Edeki 2013).

1.7.1 Variación de AUP para la UCI

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide utilizar una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad de software que se produce, de ahí la importancia de aplicar buenas prácticas que se centran en el desarrollo de productos y servicios de calidad (Sánchez 2017).

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) la metodología AUP-UCI mantiene la fase de Inicio en la que se llevan a cabo actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial con el cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo del proyecto.

La fase restante se unifica en Ejecución, en la cual se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el sistema, se obtienen los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto, además de realizar una capacitación a los usuarios que lo utilizarán. Por último, se agrega una fase de Cierre, en la que se analizarán tantos los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

La adaptación de la metodología AUP para la UCI, tiene un alcance para todo tipo de proyecto de la Universidad. Además, es la metodología utilizada en la creación del Gestor de Recursos de Hardware y Software. Para el desarrollo de este proyecto se necesita una metodología ágil, ya que se cuenta con un equipo de desarrollo pequeño que tiene un ciclo de vida de corta duración. La refactorización es una técnica de ingeniería de software para reestructurar el código fuente sin que la aplicación sufra muchos cambios, esta es una ventaja que brinda la metodología AUP-UCI porque se corresponde a las necesidades del cliente, ya que el proyecto será una personalización de un software de tipo modular.

Entre los escenarios propuestos por la metodología AUP-UCI se seleccionó el escenario número dos que propone realizar una descripción de requisitos por casos de uso del sistema y un modelo conceptual, ya que brindan la posibilidad de realizar una descripción detallada de cómo el cliente (Actor) opera con el sistema en desarrollo, describiendo todos los pasos a seguir de las actividades relacionadas con los requerimientos que debe cumplir.

1.8 Herramientas y tecnologías a utilizar

1.8.1 Marcos de Trabajo:

Django 1.8:

Framework web implementado sobre el lenguaje de programación Python, perteneciente a la licencia BSD *(Licencia de Software Libre Permisiva)*. Django brinda estructura al código fuente, fomentando las buenas prácticas de desarrollo web, lo que promueve un código legible y fácil de mantener. La implementación del patrón de diseño MTV (modelo-plantilla-vista), es una característica propia que contiene el framework, la cual contribuye a la organización de las distintas partes de la aplicación y a modificar éstas

sin afectar cualquier otra pieza del software. Su alta escalabilidad le posibilita manejar el crecimiento continuo de trabajo de manera fluida sin perder calidad en los servicios (Holovaty y Kaplan-Moss 2009). Django es el marco de trabajo con el cual está diseñado XILEMA GRHS.

Xilema-Base-Web:

Xilema-Base-Web es un marco de trabajo desarrollado en el Centro de TLM que está constituido por Django como framework base, librerías de javascript como son Jquery y Backbone, además cuenta con las pautas de diseño de la Universidad (TLM 2016).

Garantiza el correcto cumplimiento de la estrategia marcaria del Centro de Telemática y de la Universidad de las Ciencias Informáticas, además es utilizado como base en la estructura del Gestor de Recursos de Hardware y Software.

1.8.2 Lenguajes de programación y modelado

Python 2.7.6:

Es un lenguaje que favorece un código legible. Se trata de un lenguaje interpretado o de script, multiplataforma y orientado a objetos. Python es un lenguaje que posee una sintaxis simple, clara y sencilla. Tiene estructuras de datos de alto nivel y una solución de programación orientada a objetos, pero eficaz.

Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de cadenas, números, archivos. Contiene una gran biblioteca de módulos que facilitan realizar tareas de programación web. También posee bibliotecas capaces de apoyar el desarrollo del sistema vinculado con los SGBD (sistemas gestores de base de datos) PostgreSQL y MySQL; es el caso de Psycopg2 y MySQLdb respectivamente (Guido van Rossum 2009). Es el lenguaje de programación con el cual está escrito la aplicación XILEMA GRHS.

HTML:

El lenguaje de marcas de hipertexto, HTML o (HyperText Markup Language) se basa en el metalenguaje SGML (Standard Generalized Markup Language) y es el formato de los documentos de la World Wide Web. El World Wide Web Consortium (W3C) es la organización que desarrolla los estándares para normalizar el desarrollo y la expansión de la Web y la que publica las especificaciones relativas al lenguaje HTML (Lapuente y Lapuente 2013).

JavaScript:

JavaScript es un lenguaje de programación de scripts (secuencia de comandos) orientado a objetos. Actualmente es principalmente utilizado en internet, junto con las páginas web (HTML o XHTML). Javascript está directamente incluido en la página web (o en un archivo externo) y mejora una página HTML, añadiendo interacción del usuario, animación y ayudas a la navegación (Asensio 2014).

UML:

El Lenguaje Unificado de Modelado (Unifield Modeling Lenguaje UML), es un lenguaje estándar para escribir planos de software, UML se puede utilizar para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. UML establece un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan (José Enrique González Cornejo 2008).

1.8.3 Herramientas de desarrollo

PyCharm 2017.2.2:

Es un IDE multiplataforma, utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica y soporte para el desarrollo web con Django, entre otras bondades. PyCharm es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la Community que es gratuita y orientada a la educación y al desarrollo puro en Python y la Professional, que incluye más características como el soporte a desarrollo web (Islam 2015).

pgAdmin III:

Es la herramienta que permite la gestión y administración de base de datos SQL de la plataforma de código abierto PostgreSQL. La aplicación se puede utilizar en Linux, FreeBSD, Solaris, Mac OS X y Windows. Está diseñada para responder a las necesidades de todos los usuarios, desde escribir consultas SQL sencillas hasta el desarrollo de bases de datos complejas. La conexión del servidor puede hacerse a través de TCP/IP o sockets de dominio Unix (en plataformas * nix), y puede ser encriptado SSL para la seguridad. No se requieren conductores adicionales para comunicarse con el servidor de base de datos. Es un software libre publicado bajo la Licencia PostgreSQL (pgAdmin - Official Page 2018).

Bootstrap 2.0:

Bootstrap es un enfoque de diseño web destinado a la elaboración de sitios web para proporcionar una visualización óptima para una experiencia de navegación fácil y con un mínimo de cambio de tamaño, paneo y desplazamiento a través de una amplia gama de dispositivos.

Un sitio diseñado con Bootstrap adapta su diseño a las condiciones de observación mediante el uso de fluidos, las proporciones basadas en cuadrículas, imágenes flexibles y CSS3 en los medios (Arias 2014).

1.8.4 Herramienta de Modelado

Visual Paradigm 8.0 Enterprise Edition:

Visual Paradigm for UML es una herramienta CASE que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software (Visual Paradigm 2014).

Es una útil herramienta que ayuda a comprender el funcionamiento del sistema a desarrollar, puede ser utilizado para modelar diagramas de ingeniería de software como Casos de uso, Paquetes, Despliegue, Clases y Entidad-Relación.

1.8.5 Sistema gestor de base de datos

PostgreSQL 9.4:

Es un gestor de base de datos objeto-relacional que utiliza un modelo cliente/servidor y usa multiproceso lo que posibilita que, si existiera una falla en algún proceso, esto no afectará al resto de ellos permitiendo la estabilidad del sistema. Esto ofrece como ventaja que un fallo en uno de los procesos no afecte al resto, garantizando así que el sistema continúe funcionando (PostgreSQL 2018).

XILEMA GRHS cuenta con una base de datos PostgreSQL, que permite realizar toda la gestión de la información referente al inventariado de hardware y software, así como las configuraciones llevadas a cabo en el sistema.

SQLite III:

SQLite es una base de datos integrada. En lugar de ejecutar de forma independiente como un proceso independiente, coexiste simbióticamente dentro de la aplicación a la que sirve, dentro de su espacio de proceso. Una ventaja de tener un servidor de base de datos dentro de su programa es que no se requiere configuración o administración de red. Esto reduce los gastos generales relacionados con las llamadas de

red, simplifica la administración de la base de datos y facilita la implementación de su aplicación (Allen y Owens 2011).

Gclient representa la aplicación agente de XILEMA GRHS, y es desplegada en cada estación de trabajo para la obtención de los datos que luego serán enviados a Gserver. Gclient cuenta con una base de datos SQLite integrada a la aplicación el cual permite guardar todos los datos colectados.

Conclusiones del capítulo

El desarrollo de la fundamentación teórica organizó el trabajo, profundizando en el estudio de los términos necesarios que se consideran fundamentales para el desarrollo de la propuesta de solución lo cual tributó a una mayor comprensión acerca de la temática abordada. El estudio de los sistemas homólogos a nivel nacional e internacional evidenció la carencia de mecanismos similares que dieran solución a la problemática planteada, por lo tanto, se determinó integrar XILEMA GRHS al sistema Assets. Además de desarrollar un componente en la aplicación cliente para mostrar el inventario de hardware, software y políticas de seguridad informática. La selección de la metodología de desarrollo de software favorece el adecuado diseño del sistema y la realización de cada una de las fases del ciclo de vida del software. Se caracterizan las herramientas informáticas para el desarrollo de la solución mediante un estudio de las mismas que permitió conocer sus principales características e identificar las de mayor utilidad para la implementación del software.

Capítulo 2. Propuesta de solución

2.1 Introducción

En este capítulo se describen las características de la propuesta de solución, se realiza una descripción detallada de los casos de usos que se proponen para dar cumplimiento al objetivo general dado por la situación problemática. Se describen las características funcionales con la cuales debe contar la aplicación y las propiedades del producto.

2.2 Propuesta de solución

La solución propuesta está diseñada para desarrollar una personalización del Gestor de Recursos de Hardware y Software para la UCI, la cual permitirá a los usuarios vinculados a cualquier entorno de trabajo de la universidad, conocer el estado actual del cumplimiento de las políticas de seguridad además de los componentes de hardware y software. También permitirá aumentar la fiabilidad de los datos mostrados por la plataforma al vincular los provenientes del sistema Assets y se solucionarán problemas actuales en cuanto a errores de inserción de los mismos. Además, XILEMA GRHS será capaz de conformar nuevas localizaciones a través de un análisis comparativo de la información brindada por ambos sistemas. La personalización del sistema XILEMA GRHS, tributará al desarrollo de una aplicación robusta y confiable.

La solución planteada está compuesta por un servicio web WSDL[8] (Web Services Description Language) el cual será consumido a través del módulo Suds, incorporado como librería de Python. El servicio contiene el método +obtenerActivoDadoRotulo, el cual requiere el parámetro rótulo (MB) y devuelve todos los datos relacionados con el activo. La información obtenida del sistema Assets NS será guardada en una nueva tabla en la base de datos llamada Agents_Assets, la cual contendrá los campos Nombre del activo, Centro de Costo, Área de responsabilidad material y Estado que conforman los datos obtenidos de cada uno de ellos. A través del uso de hilos y tareas multiprocesos, el sistema debe permitir actualizar toda la información proveniente del sistema Assets cada un período de tiempo determinado por el usuario. Para un mejor entendimiento de lo antes expuesto se presenta la siguiente figura:

[8] **WSDL (Web Services Description Language):** Es un formato XML que se utiliza para describir servicios web (WS).

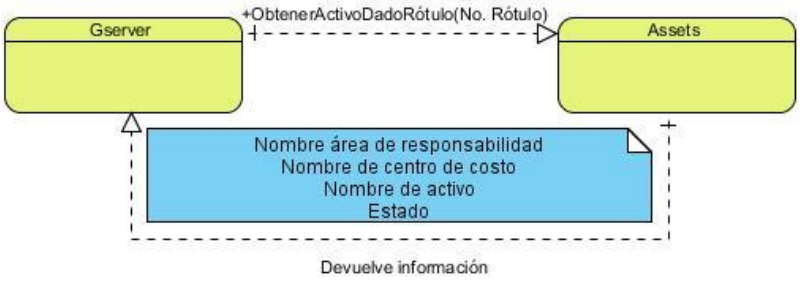


Figura 2: Diagrama de comunicación entre XILEMA GRHS y el sistema Assets

Se dispondrá de la función Unificar localización, en la cual a través de expresiones regulares se conformarán nuevas localizaciones, en la que el usuario podrá confirmar la nueva propuesta y así ser reescrita en el sistema.

El estado del activo se determinará por tres criterios que se detallan a continuación:

Baja: Se refiere a que el ciclo de vida del activo informático ha culminado. El sistema proporcionará la interfaz y los mecanismos necesarios para determinar el estado.

Activo: Se refiere a que el agente esté activamente funcional. El sistema proporciona la interfaz necesaria para determinar este estado.

Inactivo: Se refiere al tiempo de inactividad del agente. El sistema proporciona la interfaz necesaria para determinar este estado.

Se contará con un componente que permitirá que usuarios sin permiso para acceder a la plataforma, identifiquen su computadora en una página local y puedan ver todas las características de hardware y software con que cuenta, además de poder visualizar un diagnóstico del cumplimiento de las políticas de seguridad informática. La interfaz será accesible vía web a través del puerto 8418.

Se decidió reutilizar esta interfaz como base para el desarrollo de una página web la cual se mostrará de forma local. Mediante el uso del módulo SocketServer incorporado en el lenguaje de programación Python, se hizo uso de la clase TCPServer la cual se configuró de forma tal que el puerto 8418 permanecerá abierto sólo localmente para el cliente en que se ejecute, no será visible ni accesible en caso de un escaneo de red. Además, se agregaron datos de las características específicas sobre el hardware y software con que cuenta cada estación de trabajo, y se perfeccionó la interfaz mediante el uso de hojas de estilos CSS gracias a la aplicación de la biblioteca Bootstrap en su versión 3.0, de acuerdo a las pautas de diseño del centro.

XILEMA GRHS está compuesto por cuatro subsistemas: Gclient, Gserver, Gadmin y Gupdater. Gclient se encarga de recopilar la información de los clientes, detectar los cambios ocurridos en el hardware y el software, detectar las incidencias ante cambios no autorizados y tomar acciones a partir de las incidencias detectadas. Gserver recopila la información de todos los inventarios, almacena las configuraciones y guarda la información de todos los clientes. Gadmin es la consola de administración de XILEMA GRHS, se utiliza para establecer las configuraciones, mostrar reportes de los inventarios y mostrar la información de los clientes. Por otra parte, Gadmin se encarga de comunicar las configuraciones y órdenes a la aplicación Gserver. Una vez que se hayan establecido las configuraciones para XILEMA GRHS se comporta como un conjunto de aplicaciones autónomas y no necesitan de la interacción con el usuario.

Gserver, obtendrá a través de un servicio toda la información referente a las localizaciones para llevar a cabo todo el proceso mencionado anteriormente. A su vez, Gclient mostrará en cada cliente una página web, en la cual se visualizará un diagnóstico del cumplimiento de las políticas de seguridad y las características de hardware y software correspondientes al agente.

A continuación, se muestra una gráfica que representa cómo quedará estructurado el software a desarrollar:

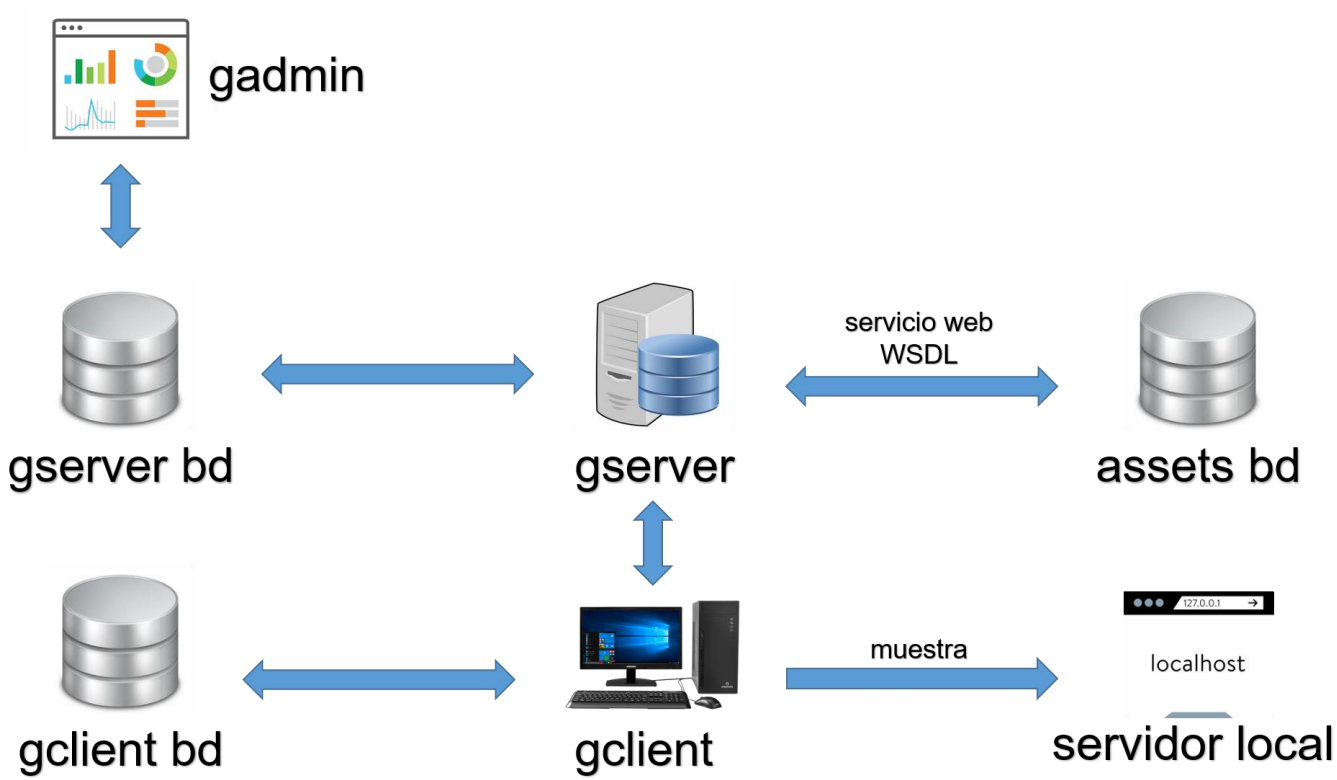


Figura 3: Propuesta de solución

2.2.1 Preprocesamiento de datos

El propósito del preprocesamiento de datos es principalmente corregir las inconsistencias de los datos que serán la base de análisis en procesos de minería de datos. En el caso de las fuentes de datos estructuradas, el propósito no es distinto y pueden ser aplicadas diversas técnicas estadísticas y de aprendizaje computacional. Con el preprocesamiento de datos se pretende que los datos que van a ser utilizados en tareas de análisis o descubrimiento de conocimiento conserven su coherencia (Rodríguez y Rodríguez 2013).

El preprocesamiento es una tarea necesaria para la preparación de los datos que serán utilizados para el posterior análisis. La justificación de este proceso preliminar, generalmente, radica en que los datos vienen con ruido por diferentes razones, entre las cuales se encuentran:

- Datos incompletos: valores faltantes para algunos atributos o sólo se tienen los datos agregados y no se cuenta con el detalle.
- Ruido: errores en los datos.
- Inconsistencias: contiene discrepancias en los datos. Por ejemplo, edad de un empleado = 30 y fecha de nacimiento = 03/07/1998.

En muchas ocasiones, la naturaleza y severidad de los problemas dependen del control de los operarios humanos de las aplicaciones que nutren las fuentes originales de datos. Debido a los efectos de estos problemas en los resultados del análisis de datos, se ha establecido como meta identificarlos y rectificarlos.

Las tareas y técnicas de preprocesamiento de datos pueden mejorar la calidad de los datos, ayudando a mejorar la precisión y eficiencia de los procesos de análisis de datos, de ahí que el preprocesamiento de datos se convierta en un paso preliminar importante. Detectando anomalías, corrigiéndolas a tiempo y reduciendo los datos que serán analizados se puede ayudar para que la toma de decisiones sea mucho más eficaz (Rodríguez y Rodríguez 2013).

La estandarización de datos facilita que los datos de origen sean internamente coherentes; es decir, que cada tipo de datos tenga el mismo tipo de contenido y formato (IBM 2012). Los datos obtenidos del sistema Assets, presentan inconsistencia de acuerdo a los datos de XILEMA GRHS. La estandarización de ambos sistemas es distinta, por lo que presentan diferencias de acuerdo a la estructura de las localizaciones. Para una comparación es necesario que presenten iguales caracteres, por lo que sería útil realizar la tarea de preprocesamiento de datos, utilizando la técnica de suavizado de datos.

Para este proceso, fue necesario identificar los patrones que permitirán marcar una pauta para la estandarización general de toda la información recogida por XILEMA GRHS, para que la aplicación sea

capaz de realizar un análisis comparativo entre ambas localizaciones. Para llevar a cabo dicha tarea se utilizará el módulo RE de Python con el cual es posible crear objetos de tipo patrón y generar objetos tipo *matcher,* que son los que contienen la información de la coincidencia en la cadena. En la siguiente tabla se detallan las expresiones regulares utilizadas como patrones para detectar localizaciones:

Tabla 2: Expresiones regulares utilizadas

Uso	**Expresión regular**

Docentes	`re.compile('DOC - [1-9]+')`
	`re.compile('DOC [1-9]+')`
	`re.compile('DOCENTE [1-9]+')`
	`re.compile('DOC[1-9]+')`

Facultad	`re.compile('FAC - [1-9]+')`
	`re.compile('FAC [1-9]+')`
	`re.compile(FACULTAD [1-9]+')`
	`re.compile('FAC[1-9]+')`

Centros	`re.compile('TELEMÁTICA')`
	`re.compile('SITEL')`
	`re.compile('TLM')`
	`re.compile('CESIM')`
	`re.compile('MÉDICA')`
	`re.compile('VERTEX')`
	`re.compile('INTERACTIVO')`
	`re.compile('CENID')`
	`re.compile('CISED')`
	`re.compile('IDENTIFICACIÓN Y SEGURIDAD DIGITAL')`
	`re.compile('CEIGE')`
	`re.compile('INFORMATIZACIÓN DE ENTIDADES')`
	`re.compile('CENTRO DE SOPORTE')`
	`re.compile('CIGED')`
	`re.compile('GESTIÓN DOCUMENTAL')`
	`re.compile('CEGEL')`
	`re.compile('GOBIERNO ELECTRÓNICO')`

Aulas	`re.compile('AULA [1-9]+')` `re.compile('AULA[1-9]+')`
Laboratorios	**`re.compile('LAB[1-9]+')`** **`re.compile('LABORATORIO [1-9]+')`**
Espacios adicionales	`re.sub(' +', ' ', STRING.upper)`
Guiones repetidos	`re.sub('_+', '_', cadena)`

El uso de mayúsculas en la expresión regular está consignado a definir una estandarización de los datos, los cuales al ser obtenidos son transformados al estilo upperCase, para mitigar incompatibilidades en la escritura.

Se describen los siguientes escenarios:

- Para las localizaciones que no coincidan con las expresiones regulares definidas anteriormente se determinará su localización a partir del Área de responsabilidad material y el Centro de Costo al que pertenezca.
- En caso de que el sistema detecte que ambas localizaciones son nulas, se mostrará una notificación indicando que el No. de medio básico introducido es incorrecto.
- Para las localizaciones registradas en XILEMA GRHS que sean nulas, automáticamente acogerá la localización generada por el sistema Assets, y se proporcionará como nueva propuesta de localización.
- Para las localizaciones que presenten diferencias en cuanto al Laboratorio, Aula u Oficina, se mantendrá la misma localización excepto el parámetro que no coincida, el cual será sustituido por el obtenido del sistema Assets.
- El usuario siempre tendrá la posibilidad de determinar si la nueva localización generada por el sistema es válida.

Para los locales ubicados en docentes, se determinó utilizar la nomenclatura designada por XILEMA GRHS que se conforma de la siguiente manera:

- Doc#_Fac#_Lab#
- Doc#_Fac#_Aula#
- Doc#_Fac#_Nombre_Oficina
- Doc#_Fac#_CentroNombre_OficinaNombre
- Doc#_Fac#_CentroNombre_Aula#
- Doc#_Fac#_CentroNombre_Lab#

En el caso de los locales pertenecientes a centros de desarrollo ubicados en docentes, el campo facultad puede ser variable.

Para las oficinas de Rectorado, Biblioteca, Cátedras, Transporte, Nodo, Comedores, entre otras instalaciones de la universidad se utilizará la siguiente nomenclatura:

- NombreAreaResponsabilidad_NombreCentrodeCosto

2.3 Modelo Conceptual

En el modelo conceptual se describe cómo se relacionan los diferentes conceptos en una misma temática. Se utiliza para representar un problema de manera gráfica a través del diagrama de entidad relación, en su estado lógico (Visual Paradigm 2016).

A continuación, se muestra el modelo conceptual en el que describe desde una forma general la relación entre el sistema Assets, la universidad y cómo se encuentra reflejada en el Gestor de Recursos de Hardware y Software. La figura comienza con la entidad UCI, y trabaja con los sistemas Assets y GRHS, los que gestionan muchos medios básicos e integran datos entre sí. GRHS contiene el módulo que gestiona todos los agentes inventariados en el sistema, la localización a la que pertenecen cada uno de ellos y los resultados de la integración con Assets.

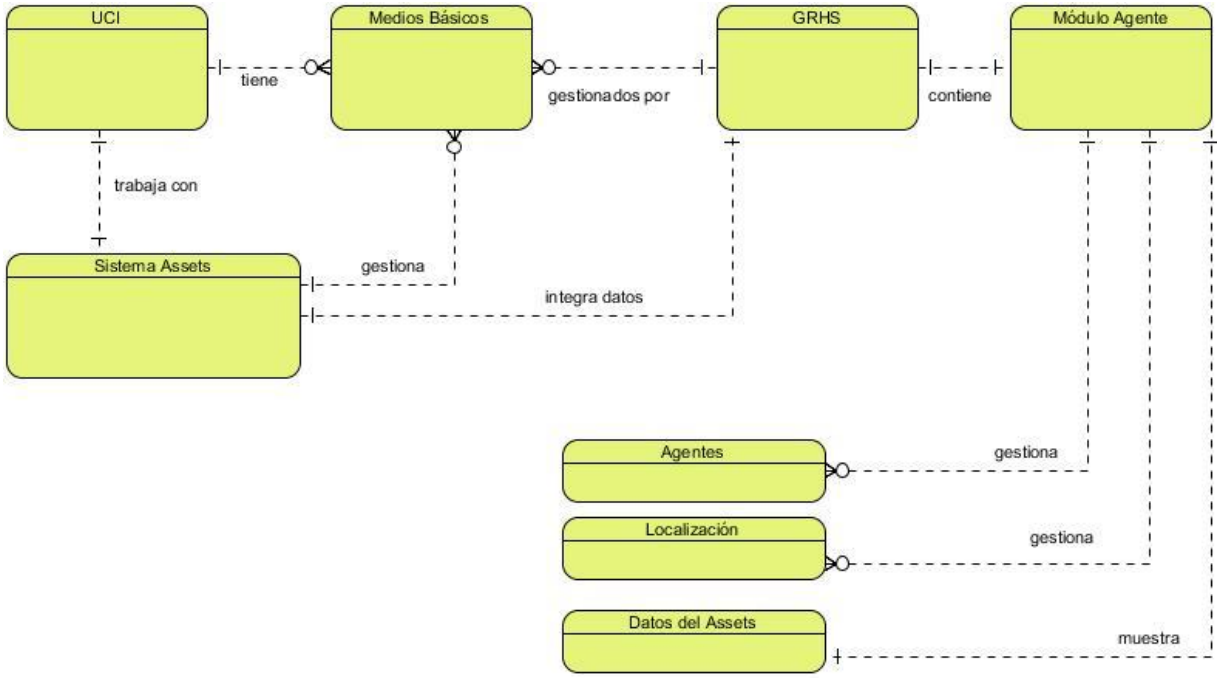


Figura 4: Modelo conceptual

2.4 Requisitos

La ingeniería de requisitos se refiere a establecer qué debe hacer el sistema, sus propiedades emergentes deseadas y esenciales, y las limitaciones en el funcionamiento del sistema y los procesos de desarrollo de software. Por lo tanto, se puede pensar en la ingeniería de requisitos como el proceso de comunicación entre los clientes y usuarios de software y los desarrolladores de software. La ingeniería de requisitos no es simplemente un proceso técnico. Los requisitos del sistema están influenciados por los

gustos, disgustos y prejuicios de los usuarios, y por cuestiones políticas y de organización. Estas son características humanas fundamentales, y las nuevas tecnologías, como casos de uso, escenarios y métodos formales, no ayudan mucho a resolver estos espinosos problemas (Sommerville 2010).

2.5 Requisitos funcionales del sistema

Los requisitos funcionales para un sistema describen lo que el sistema debería hacer. Estos requisitos dependen del tipo de software que se está desarrollando, los usuarios esperados del software y la evaluación general tomada por la organización al redactar los requisitos. Cuando se expresan como requisitos del usuario, los requisitos generalmente se describen de una manera bastante abstracta. Sin embargo, los requisitos del sistema funcional describen la función del sistema en detalle, sus entradas, salidas y excepciones (Sommerville 2010). Los requisitos funcionales se describirán a través del escenario propuesto por la metodología de desarrollo AUP-UCI descripción de requisitos por casos de uso del sistema.

A continuación, se listan los requisitos funcionales con los que debe cumplir el sistema.

1. Determinar la disponibilidad física de un agente.
2. Mostrar estadísticas.
3. Mostrar nombre de activo.
4. Mostrar nombre de área de responsabilidad.
5. Mostrar nombre de centro de costo.
6. Mostrar localización de XILEMA GRHS y sistema Assets.
7. Filtrar por localizaciones.
8. Unificar localización.
9. Adicionar URL.
10. Modificar URL.
11. Eliminar URL.
12. Mostrar URL.
13. Guardar en base de datos la fecha del último acceso al sistema Assets.
14. Programar tarea de actualización automática.
15. Actualizar datos con sistema Assets.
16. Adicionar intervalo de tiempo.
17. Modificar intervalo de tiempo.
18. Eliminar intervalo de tiempo.
19. Mostrar intervalo de tiempo.
20. Mostrar especificaciones de hardware del cliente.

21. Mostrar especificaciones de software del cliente.
22. Mostrar especificaciones de las políticas de seguridad.

2.6 Casos de Uso

Los casos de uso son una técnica que se basa en escenarios para la obtención de requerimientos que se introdujeron por primera vez en el método Objectory[9]. Actualmente se han convertido en una característica fundamental de la notación de UML, que se utiliza para describir modelos de sistemas orientados a objetos. En su forma más simple, un caso de uso identifica el tipo de interacción y los actores involucrados (Sommerville 2010).

2.6.1 Actores involucrados

Los actores en el proceso se representan como figuras delineadas, y cada clase de interacción se representa como una elipse con su nombre. En la siguiente tabla se detallan los actores involucrados en el sistema.

Tabla 3: Descripción de los actores involucrados en el sistema

Actor	Descripción
Administrador	Representa rol específico con acceso a XILEMA GRHS con todos los permisos asignados.
Jefe de Tecnología	Representa un rol específico con acceso a XILEMA GRHS con algunos de los permisos asignados.

2.6.2 Diagrama de casos de uso

Un diagrama de casos de uso es una excelente representación del contexto del sistema. Sirve como herramienta de comunicación que resume el comportamiento de un sistema y sus actores. La figura 5 presenta una muestra de casos de uso para el software a desarrollar, en los cuales se agruparon los 15 requisitos funcionales identificados en 7 casos de uso.

[9] **Objectory**: Es un proceso organizado para la construcción industrial de software. Este proceso de diseño está guiado por casos de uso.

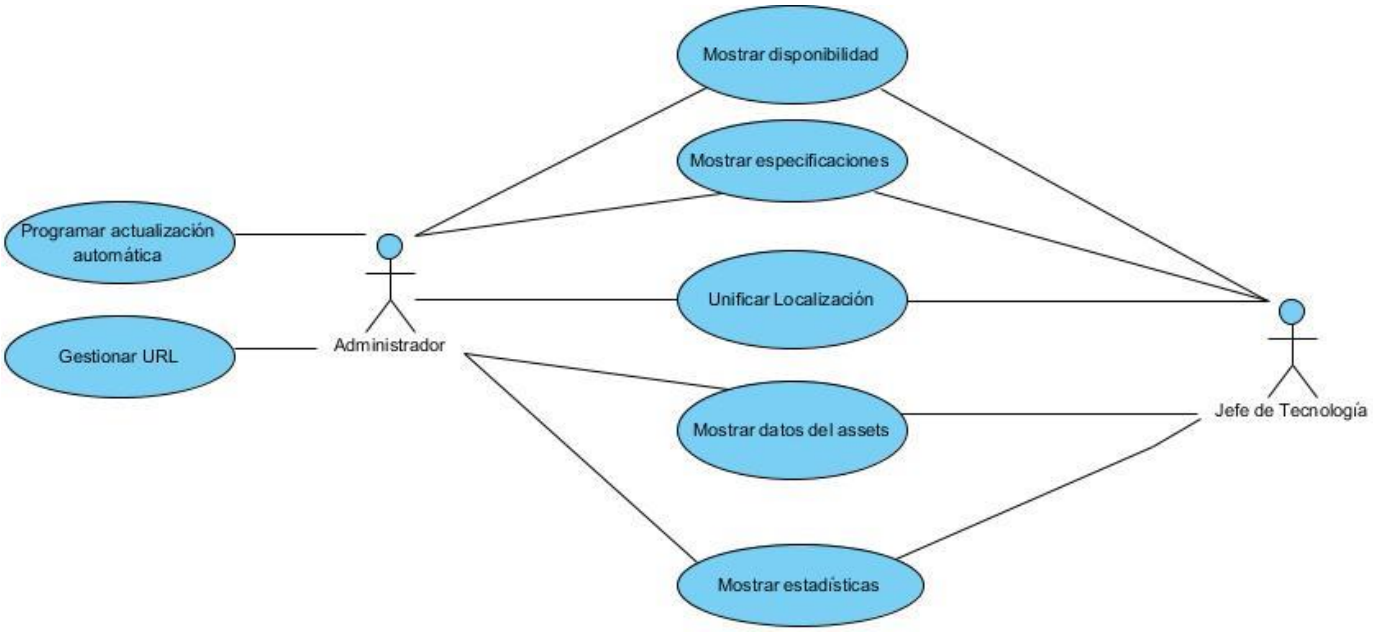


Figura 5: Diagrama de casos de uso

En la siguiente tabla se evidencia la relación entre los requisitos funcionales agrupados por casos de uso.

Tabla 4: Relación Caso de Uso - Requisitos

Caso de Uso	Requisito Funcional
Mostrar disponibilidad	• Mostrar disponibilidad física de un agente.
Mostrar especificaciones	• Mostrar especificaciones de Hardware del cliente. • Mostrar especificaciones de Software del cliente. • Mostrar especificaciones de las Políticas de Seguridad.
Unificar localización	• Mostrar localización de XILEMA GRHS y sistema Assets. • Filtrar por localizaciones. • Unificar Localización.
Mostrar datos del Assets	• Mostrar Nombre de Activo. • Mostrar Nombre de Área de responsabilidad. • Mostrar Nombre de Centro de Costo.

	• Actualizar datos con sistema Assets.
Mostrar estadísticas	• Mostrar estadísticas.
Programar actualización automática	• Programar tarea de actualización automática. • Adicionar intervalo de tiempo. • Modificar intervalo de tiempo. • Eliminar intervalo de tiempo. • Mostrar intervalo de tiempo. • Guardar en base de datos la fecha del último acceso al sistema Assets.
Gestionar URL	• Adicionar URL. • Modificar URL. • Eliminar URL. • Mostrar URL

2.6.3 Especificación de casos de uso

A continuación, se presenta la especificación del caso de uso Mostrar especificaciones, el resto se pueden encontrar en el **Anexo II**.

Descripción textual del requisito Mostrar especificaciones:

La información referente a cada cliente es guardada en una base de datos que contiene toda la información del activo que será enviada al servidor, incluyendo las políticas de seguridad. El usuario podrá acceder a toda esta información a través de una página web que estará disponible a través de la siguiente dirección: http://localhost:8418.

Tabla 5: Descripción de caso de uso Mostrar especificaciones

Objetivo	Mostrar especificaciones
Actores	Usuarios
Resumen	El caso de uso inicia cuando el usuario necesita conocer las especificaciones con que cuenta la PC cliente donde se encuentra. Una página local deberá iniciarse mostrando todas las especificaciones de la computadora incluyendo las políticas de seguridad. El sistema proporciona las interfaces necesarias para su visualización, será accesible por la dirección http://localhost:8418, finalizando así el caso de uso.
Complejidad	Alta

Prioridad	Alta
Precondiciones	Tener instalado la aplicación cliente de XILEMA GRHS.
Postcondiciones	NA
Flujo de eventos	
Flujo básico Mostrar especificaciones	

	Actor	**Sistema**
1.	Accede a la dirección del navegador http://localhost:8418	
2.		Se muestra una interfaz amigable donde el usuario será capaz de revisar todas las características de su PC y un diagnóstico del cumplimiento de las políticas de seguridad. Permite realizar varias acciones: - Mostrar especificaciones de Hardware. Ver Sección 1: Mostrar especificaciones de Hardware. - Mostrar especificaciones de Software. Ver Sección 2: Mostrar especificaciones de Software. - Mostrar especificaciones de las Políticas de Seguridad. Ver Sección 2: Mostrar especificaciones de las Políticas de Seguridad.
3.		Termina el caso de uso.

Sección 1: “Mostrar especificaciones de Hardware”		
Flujo básico <Mostrar especificaciones>		
	Actor	**Sistema**
1.	Selecciona el enlace Hardware.	
2.		Se muestran todas las características de hardware colectadas por Gclient: Motherboard, Dispositivos de Almacenamiento, RAM, Microprocesador, Monitor, Impresora, Scanner, Ratón y Teclado.
3.		Termina el caso de uso.

Sección 2: “Mostrar especificaciones de Software”		
Flujo básico <Mostrar especificaciones>		
	Actor	**Sistema**
1.	Selecciona el enlace Software.	
2.		Se muestran especificaciones de software colectadas por Gclient: Sistema Operativo, programas instalados.

3.		Termina el caso de uso.

Prototipo elemental de interfaz gráfica de usuario

Sección 2: "Mostrar especificaciones de Políticas de Seguridad"

Flujo básico <Mostrar especificaciones>

	Actor	Sistema
1.	Selecciona el enlace Políticas de seguridad.	
2.		Se muestran especificaciones de políticas de seguridad colectadas por Gclient: Carpetas compartidas, Antivirus, Cortafuegos, Bloqueo de Pantalla, Protección de datos.
3.		Termina el caso de uso.

Prototipo elemental de interfaz gráfica de usuario

Relaciones	**CU incluidos**	NA
	CU extendidos	NA
Requisitos no funcionales	Comprensibilidad, Usabilidad	
Asuntos pendientes	NA	

Prototipo elemental de interfaz gráfica de usuario

2.7 Requisitos no funcionales del sistema

Los requisitos no funcionales, como su nombre lo indica, son requisitos que no están directamente relacionados con las funciones específicas entregadas por el sistema. Pueden relacionarse con las propiedades emergentes del sistema, como la fiabilidad o el tiempo de respuesta, el rendimiento del sistema, la seguridad, la disponibilidad y otras propiedades. Alternativamente, pueden definir restricciones en el sistema tales como las representaciones de datos utilizadas en las interfaces del sistema.

Los requisitos no funcionales surgen a través de las necesidades del usuario, debido a restricciones presupuestarias, políticas organizacionales, necesidad de interoperabilidad con otros sistemas de software o hardware, o debido a factores externos tales como regulaciones de seguridad o legislación de privacidad (Sommerville 2010).

A continuación, se presenta una descripción de todos los requisitos no funcionales con que debe cumplir el sistema a desarrollar.

Tabla 6: Descripción de requisito no funcional Usabilidad. Comprensibilidad

Atributo de Calidad	Usabilidad
Sub-atributos/Sub-características	Comprensibilidad
Objetivo	Lograr que el sistema sea entendible fácilmente para los usuarios.
Origen	El usuario

Artefacto	El sistema
Entorno	El sistema está funcionando correctamente.
Estímulo	Respuesta: Flujo de eventos (Escenarios)
El sistema debe estar adaptado a la estrategia marcaria y las pautas de diseño que define el Centro de Telemática.	
	El sistema se acoge a la marca registrada Xilema, y a la estrategia marcaria que contempla el framework, Xilema-Base-Web.
Medida de respuesta	
NA	

Tabla 7: Descripción de requisito no funcional Funcionabilidad. Precisión

Atributo de Calidad	Funcionabilidad
Sub-atributos/Sub-características	Precisión
Objetivo	Verificar que el sistema arroje resultados o efectos acordes a las necesidades para las cuales fue creado.
Origen	Cliente del producto final.
Artefacto	El sistema.
Entorno	El sistema está funcionando correctamente.
Estímulo	Respuesta: Flujo de eventos (Escenarios)
El usuario verifica que el sistema XILEMA GRHS, muestre correctamente toda la información colectada por el sistema Assets.	
	El sistema muestra una tabla con los campos ofrecidos por el sistema Assets.
Medida de respuesta	
NA	

Tabla 8: Descripción de requisito no funcional Funcionabilidad. Interoperabilidad

Atributo de Calidad	Funcionabilidad
Sub-atributos/Sub-características	Interoperabilidad
Objetivo	Lograr que el sistema sea capaz de interactuar con el sistema externo Assets.
Origen	El sistema
Artefacto	El sistema, Canal de comunicación (Servicio Web)
Entorno	El usuario desea actualizar datos provenientes del sistema externo.

Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
Variable	Datos
Medida de respuesta	
2-3 segundos	

Tabla 9: Descripción de requisito no funcional Funcionabilidad. Tolerancia a Fallos

Atributo de Calidad	Funcionabilidad
Sub-atributos/Sub-características	Tolerancia a Fallos
Objetivo	Lograr que el sistema sea capaz de recuperarse ante fallos.
Origen	Interno del sistema.
Artefacto	El sistema, Canal de comunicación (Servicio Web)
Entorno	Operación Normal.
Estímulo	Respuesta: Flujo de eventos (Escenarios)
Interrupción de las comunicaciones de red en la PC	
	El sistema notifica que no se ha podido establecer conexión con el servicio web y continúa operando normalmente, intenta establecer conexión cada un intervalo de tiempo definido o a petición del usuario.
Medida de respuesta	
2 segundos	
Ocurrencia de una excepción	
	Se notifica al usuario, el sistema continúa funcionando de forma normal.
Medida de respuesta	
2 segundos	

Tabla 10: Descripción de requisito no funcional Mantenibilidad. Cambiabilidad

Atributo de Calidad	Mantenibilidad
Sub-atributos/Sub-características	Cambiabilidad
Objetivo	Facilitar la correcta modificación del sistema en caso de ser necesario, en aspectos como dirección URL del servicio web y el tiempo de intervalo entre actualizaciones automáticas, puesto que son aspectos que podrían estar sujetos a cambios.
Origen	El usuario
Artefacto	El sistema
Entorno	El sistema está funcionando correctamente.
Estímulo	Respuesta: Flujo de eventos (Escenarios)

El usuario realiza una nueva configuración en el sistema	
	El sistema asimila la nueva configuración correctamente.
Medida de respuesta	
NA	

Tabla 11: Descripción de requisito no funcional Mantenibilidad. Ensayabilidad

Atributo de Calidad	Mantenibilidad
Sub-atributos/Sub-características	Ensayabilidad
Objetivo	Permitir que el software modificado sea validado
Origen	El desarrollador
Artefacto	El sistema
Entorno	Funcionando correctamente
Estímulo	Respuesta: Flujo de eventos (Escenarios)
El desarrollador realiza pruebas al sistema	
	El sistema responde correctamente.
Medida de respuesta	
NA	

Tabla 12: Descripción de requisito no funcional Portabilidad. Adaptabilidad

Atributo de Calidad	Portabilidad
Sub-atributos/Sub-características	Adaptabilidad
Objetivo	Lograr que el sistema sea adaptable a diferentes entornos especificados. Se debe contar con un servicio web que proporcione la información de una base de datos del sistema Assets en su versión 2.1
Origen	El usuario
Artefacto	El sistema
Entorno	Funcionando correctamente
Estímulo	Respuesta: Flujo de eventos (Escenarios)
Se prueba el sistema en varios entornos	
	El sistema responde correctamente.
Medida de respuesta	
2 segundos	

Tabla 13: Descripción de requisito no funcional Eficiencia. Comportamiento en el tiempo

Atributo de Calidad	Eficiencia
Sub-atributos/Sub-características	Comportamiento en el tiempo
Objetivo	Lograr que el sistema ejecute correctamente las peticiones solicitadas por el usuario en el menor tiempo posible.
Origen	El usuario
Artefacto	El sistema
Entorno	Funcionando correctamente
Estímulo	Respuesta: Flujo de eventos (Escenarios)
El usuario realiza una actualización con el sistema assets.	
	El sistema actualiza los datos correctamente.
Medida de respuesta	
3 segundos	

2.8 Diagrama de paquetes

Un diagrama de paquetes en el Lenguaje Unificado de Modelado (UML) representa las dependencias entre los paquetes que componen un modelo. Permite organizar los elementos modelados con UML, facilitando de esta forma el manejo de los modelos de un sistema complejo. Los diagramas de paquetes permiten dividir un modelo para agrupar y encapsular sus elementos en unidades lógicas individuales, en general, pueden tener una interfaz (métodos de clases e interfaces exportadas) y una relación de estas interfaces (clases internas que implementan dichas interfaces). Los paquetes pueden estar anidados unos dentro de otros, y unos paquetes pueden depender de otros paquetes. Se pueden utilizar para planear la arquitectura del sistema a nivel macro (Demián Gutiérrez 2009).

Para la confección del diagrama de paquetes, se dividió la aplicación en tres unidades lógicas, que consisten en el Módulo Agentes, Módulo Configuración, y Cliente. El Módulo de Agentes, contiene anidado los paquetes Estadísticas, Assets, Agentes y Localizaciones, los cuales se encargan de la manipulación y gestión de los datos que corresponden a los activos almacenados en XILEMA GRHS y el sistema Assets. El paquete Módulo Configuración, contiene anidado el paquete Configuración Assets, donde se encuentran todas las referentes a la conexión con el sistema. En el paquete Cliente se maneja toda la información referente a las características de hardware y software de forma individual. Todos los paquetes se encuentran conectados con las bases de datos que brindarán toda la información necesaria.

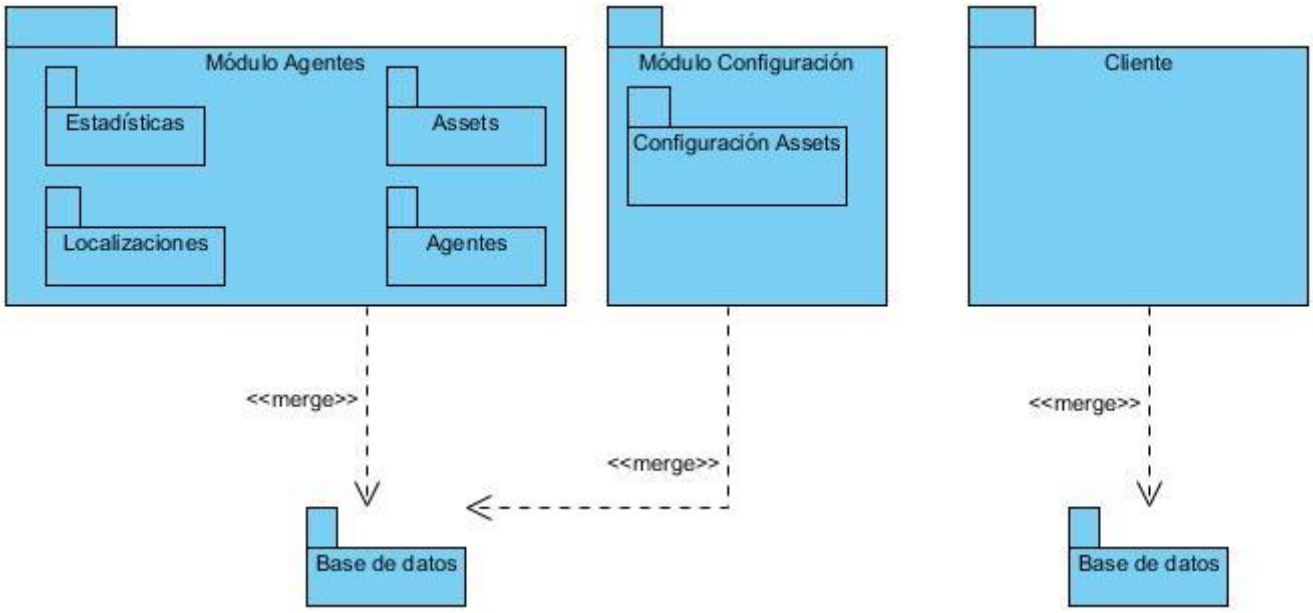


Figura 6: Diagrama de paquetes

2.9 Diagrama de clases

Un diagrama de clases en UML, representa los propósitos fundamentales del lenguaje, porque separa los elementos de diseño de la codificación del sistema. UML ha sido establecido como un modelo estandarizado para describir un enfoque de programación orientado a objetos. Dado que las clases son el bloque de construcción de los objetos, los diagramas de clase son los bloques de construcción de UML. Los componentes de creación de diagramas en un diagrama de clase pueden representar las clases que realmente van a ser programadas, los objetos principales, o las interacciones entre clases y objetos (Lucid Hart 2016).

El diagrama de clase está compuesto de tres partes:

- **Sección superior:** Nombre de la clase. Esta sección siempre es necesaria sin importar si está hablando del clasificador o de un objeto.
- **Sección media:** Atributos de la clase. Los atributos describen las variables que describen las cualidades de la clase. Esto solamente es necesario al describir una instancia específica de una clase.
- **Sección inferior:** Operaciones de la clase. Mostrado en formato de lista, cada operación tiene su propia línea. Las operaciones describen cómo una clase puede interactuar con los datos.

Como define el framework Django, los modelos conforman las tablas en las bases de datos y son los encargados de gestionar toda la información. Son representados en formas de clases y sus atributos componen los campos de las tablas y pueden contener métodos capaces de manipular los datos y estar relacionados con otras. En el siguiente diagrama, se aprecian diferentes clases las cuales se heredan de la Models, contienen sus propias operaciones y a su vez están relacionadas con otras.

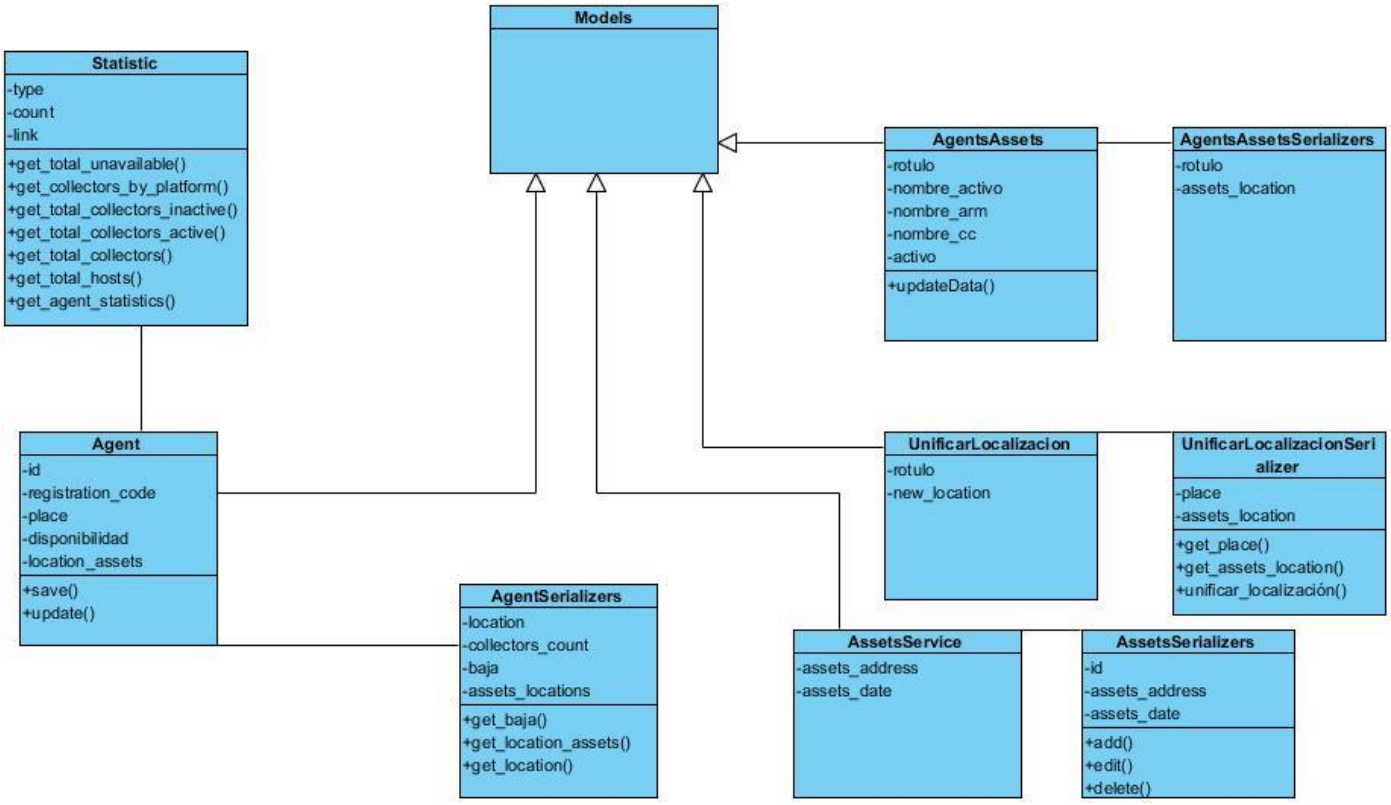


Figura 7: Diagrama de clases

La descripción de cada una de las clases modeladas en el diagrama se puede encontrar en el **Anexo III**

2.10 Diagrama de despliegue

Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema. Describen la topología del sistema y la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP (Sarmiento 2014).

En la Figura 8 se muestra el diseño de diagrama de despliegue para XILEMA GRHS, en el cual se puede apreciar cómo la estación de trabajo está compuesta por los componentes Gclient, Gupdater y Web Browser. Gclient envía a través de http/https toda la información obtenida en la computadora local, utilizando

el navegador web para mostrarla al usuario. Luego de ser enviada toda la información a Gserver a través del protocolo AMQP, es guardada en la base de datos, lo que servirá para consultar en el sistema Assets y así obtener todos los datos correspondientes a través de un conjunto de protocolos TCP/IP.

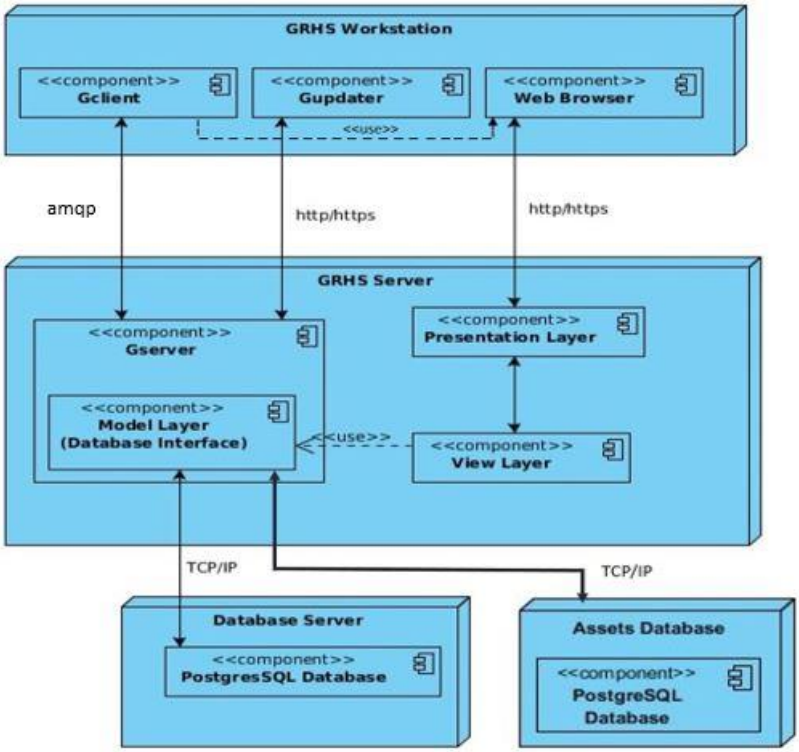


Figura 8: Diagrama de despliegue

Conclusiones del capítulo

En el capítulo culminado se realiza una descripción de la propuesta de solución, que permitió alcanzar un mayor grado de comprensión acerca del sistema a desarrollar y se abordaron temas de importancia como el preprocesamiento de datos ajustado al funcionamiento del sistema. Se identificaron 22 requisitos funcionales que darán solución a la problemática planteada, los cuales fueron agrupados en 7 casos de uso para un mejor apoyo en la implementación del sistema a desarrollar. Además, para una mayor comprensión desde diferentes perspectivas se confeccionaron los artefactos de ingeniería de software que propone la metodología AUP-UCI en su fase de Ejecución.

Capítulo III: Implementación y Pruebas

3.1 Introducción

Las fases de implementación y prueba complementadas en la fase de ejecución y cierre de la metodología AUP-UCI son de vital importancia. Es donde se concreta lo planificado durante la etapa de diseño y se transforman las descripciones de los requisitos funcionales y no funcionales en artefactos de ingeniería de software, para lograr una mejor comprensión de la implementación del sistema propuesto.

En este capítulo se aborda el tema referente a la implementación del sistema, para ello se describen los patrones arquitectónicos y de diseño empleados. Se presenta, además, el diagrama de base de datos para un mejor entendimiento del software desarrollado y se realizan las pruebas al software para su validación.

3.2 Arquitectura de software

La arquitectura de software es, a grandes rasgos, una vista de la estructura del sistema que incluye los componentes principales del mismo. Esta estructuración representa un diseño de alto nivel del sistema que tiene dos propósitos primarios: satisfacer los atributos de calidad (desempeño, seguridad, modificabilidad), y servir como guía en el desarrollo. La arquitectura de software es de gran importancia ya que la manera en que se estructura un sistema tiene un impacto directo sobre la capacidad de este para satisfacer lo que se conoce como los atributos de calidad del sistema (GUTIÉRREZ, L. E 2010).

3.2.1 Patrón arquitectónico MTV (Model-Template-View)

El patrón MTV conjunto con Django fueron diseñados para promover el acoplamiento débil y la estricta separación entre las piezas de una aplicación, lo que facilita realizar cambios en un lugar particular de la aplicación sin afectar otras piezas. El patrón MTV tiene como objetivo hacer más fluida la comunicación entre desarrolladores, ya que permite organizar el trabajo de los mismos dividiendo la aplicación en tres entidades separadas: el modelo, la vista y la plantilla (Holovaty y Kaplan-Moss 2009).

El modelo

El modelo define los datos almacenados, se escribe en forma de clases de Python, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros, puede contener métodos también. Todo esto permite indicar y controlar el comportamiento de los datos.

La vista

La vista se presenta en forma de funciones en Python, su propósito es determinar qué datos serán visualizados, entre otras funciones. El ORM[10] de Django permite escribir código Python en lugar de SQL para hacer las consultas que necesita la vista. La vista también se encarga de tareas conocidas como el envío de correo electrónico, la autenticación con servicios externos y la validación de datos a través de formularios. Lo más importante a entender con respecto a la vista es que no tiene nada que ver con el estilo de presentación de los datos, sólo se encarga de los datos, la presentación es tarea de la plantilla.

La plantilla

La plantilla es básicamente una página HTML con algunas etiquetas extras propias de Django, en sí no solamente crea contenido en HTML (también XML, CSS, Javascript, y CSV).

A continuación, se muestra una Figura que ayudará a comprender mejor el funcionamiento del patrón arquitectónico Modelo-Vista-Plantilla.

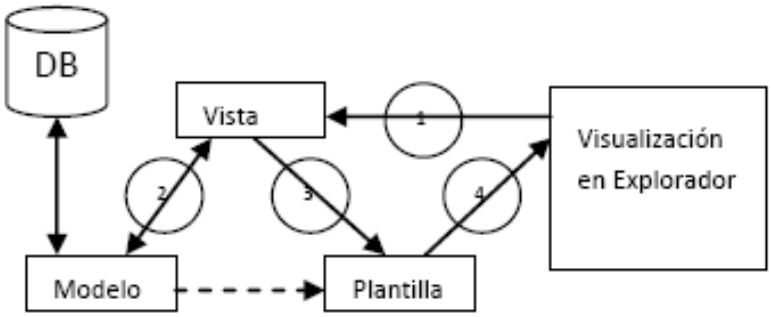


Figura 9: Patrón arquitectónico MVT

En la figura 9 se puede observar como el navegador envía una solicitud a través de una URL, en la cual es interpretada por URLConf y ubica la vista apropiada. La vista requerida interactúa con el modelo para obtener los datos deseados. La vista llama a la plantilla que a su vez renderiza la respuesta a la solicitud del navegador.

[10]**ORM**: Object-Relational mapping, es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación.

3.2.2 Patrón arquitectónico Broker

Se puede definir como una arquitectura orientada a la transferencia de mensajes entre aplicaciones que se comportan como emisores – receptores. Son mensajes que han sido formalmente definidos entre las diferentes aplicaciones que se comunican. El patrón arquitectónico Broker libera al cliente de tener que mantener información sobre donde se proporciona un servicio y como obtenerlo, de forma tal que el cliente puede acceder a los mismos de forma más fácil y efectiva. Además, actúa como un mediador en la comunicación de las aplicaciones, minimizando el grado de interacción entre ellas, obteniendo un efectivo desacoplamiento (Gomaa 2011).

Actualmente la arquitectura de XILEMA GRHS en la comunicación entre Gclient y Gserver se apoya en el patrón arquitectónico Broker, ya que cuentan con un intermediario encargado de la comunicación entre las dos aplicaciones como es el caso de RabbitMQ, basado en el protocolo AMQP.

En la siguiente Figura se puede apreciar la representación de aplicaciones que conforman XILEMA GRHS (gclient, gserver) adaptado al concepto que se define anteriormente para el patrón arquitectónico Broker.

Figura 10: Patrón arquitectónico Broker

3.2.3 Arquitectura basada en componentes

El desarrollo basado en componentes es una aplicación de la técnica *divide & conquer* para manejar la complejidad. Es una rama de la Ingeniería de Software en la cual se trata con énfasis la descomposición del software en componentes funcionales. Esta descomposición permite convertir componentes pre-existentes en piezas más grandes de software.

El proceso de construcción de una pieza de software con componentes ya existentes, da origen al principio de reutilización del software, mediante el cual se promueve que los componentes sean implementados de una forma que permita su utilización funcional sobre diferentes sistemas en el futuro («Arquitectura Basada en Componentes» 2015).

El Gestor de Recursos de Hardware y Software, cuenta con varios componentes, que representan los módulos que conforman la aplicación. Para el desarrollo del sistema se trabajó en los componentes Agentes y Configuración.

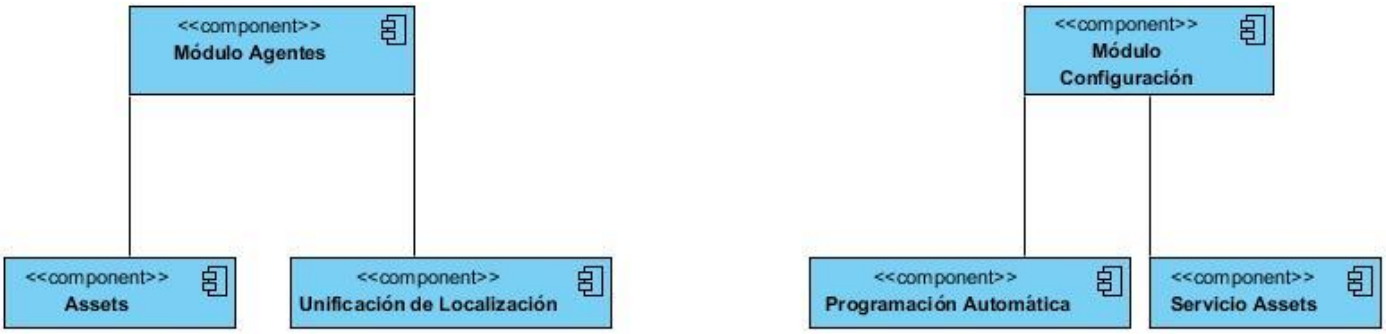


Figura 11: Diagrama de componentes

3.3 Patrones de diseño utilizados en el desarrollo del sistema propuesto

El término de patrón fue dado por primera vez en el año 1977 por el arquitecto Christopher Alexander, quien dio en su libro A Pattern Language la siguiente definición: Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma. En la ingeniería de software, un patrón constituye el apoyo para la solución a los problemas más comunes que se presentan durante las diferentes etapas del ciclo de vida del software (Guerrero, Suárez y Gutiérrez 2013).

3.3.1 Patrones GRASP

Los Patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Larman 2003).

Para el diseño de la aplicación se empleó el uso de los Patrones Generales de Software para Asignar Responsabilidades (GRASP), así como de los patrones de diseño.

Los patrones GRASP utilizados fueron:

Alta cohesión: Se aplica en la gran mayoría de las clases del diseño, ya que en cada una sólo se implementan las funcionalidades que le corresponden. En el siguiente fragmento de código se puede apreciar la clase AssetsService, que contiene el método update, únicamente utilizado para la manipulación de los datos de la propia clase.

```
class AssetsService(models.Model):
    asset_address = models.CharField(max_length=500)
    access_date = models.DateTimeField(null=True, blank=True)


    def update(self, force_insert=False, force_update=False, using=None):
        return super(AssetsService, self).save(force_insert, force_update, using)
```

Bajo acoplamiento: El objetivo es que las clases estén lo menos ligadas entre sí como sea posible. De tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto, potenciando la reutilización y disminuyendo la dependencia. Se puede evidenciar su utilización donde la clase ScheduleUpdate solamente se relaciona con su clase serializadora ScheduleUpdateSerializer.

```
class ScheduleUpdate(models.Model):
    time_interval = models.IntegerField()


class ScheduleUpdateSerializer(serializers.ModelSerializer):

    class Meta:
        model = ScheduleUpdate
        fields = ('id', 'time_interval')
```

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases, de forma tal que una instancia de un objeto sólo pueda ser creada por el objeto que contiene la información necesaria para ello. En el siguiente fragmento de código se puede apreciar la instancia obj de la clase AssetsService utilizada para acceder a todos los atributos con que cuenta.

```
class AssetsSerializer(serializers.ModelSerializer):
    access_dates=serializers.SerializerMethodField('get_access_dates')
    class Meta:
        model = AssetsService

    def get_access_dates(self, obj):
        if obj.access_date:
            return obj.access_date.strftime('%y/%m/%d %H:%M:%S')
```

Controlador: El patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa. Este sugiere que la lógica de negocios debe estar separada de la capa de presentación, para aumentar la reutilización de código y a la vez tener un mayor control.

Ejemplo: El patrón controlador se ve evidenciado en la vista updateData, en la cual se gestionan los datos provenientes del sistema assets y son almacenados en base de datos a través de las clases modelo.

Experto: Se mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener.

El patrón experto se evidencia en el objeto t, que contiene la información propia del módulo threading y a través de la clase Scheduled_Update, obtiene todos los datos para ejecutar la función deseada.

```
t = threading.Thread(target=hilo)
t.setDaemon(True)
t.start()
```

3.3.2 Patrones de diseño

Los patrones de diseño, describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos (Larman 2003). A continuación, se detallan los patrones utilizados en el diseño del sistema.

Command Pattern: Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma. Se evidencia en el siguiente fragmento de código, en el cual se aprecia cómo se encapsula la información generada en un objeto tipo JsonResponse.

```
res.save()
return JsonResponse({'msg':'ok'})
```

Active Record: Encapsula el acceso a una base de datos relacional. Se encuentra evidenciado en la implementación de modelos en Django.

Identity Field: Consiste en que las tablas de las bases de datos relacionales se encuentren identificadas por una llave, (normalmente la llave primaria). Se evidencia en las llaves primarias creadas por la clase Models de Django.

Front Controller (Controlador Frontal): Django posee una implementación de Controlador Frontal que despacha las peticiones hacia métodos o clases, que en la práctica son páginas controladoras. Antes del despacho, la petición es procesada por varios filtros.

3.4 Modelo de base de datos

El modelo entidad-relación es una útil herramienta que permite el entendimiento y la comprensión entre los distintos tipos de usuarios con diferentes perspectivas. Es una herramienta conceptual que no es técnica y permite que esté libre de ambigüedades. El modelo Entidad-Relación es uno de los más utilizados

y fue introducido por Chen en 1976. Está basado en una percepción de un mundo real que consiste en una colección de objetos básicos, denominados entidades, y de relaciones entre estos objetos.

Para la confección del diagrama Entidad-Relación se tuvieron en cuenta las nuevas tablas creadas agents_agentsassets, configurations_apps_scheduleupdate, configuration_apps_assetsservice y además la tabla modificada agents_agent, que representan entidades en el diagrama. A continuación, se muestra el DER (Entity Relationship Diagrams, Diagrama Entidad Relación) diseñado para el desarrollo del software:

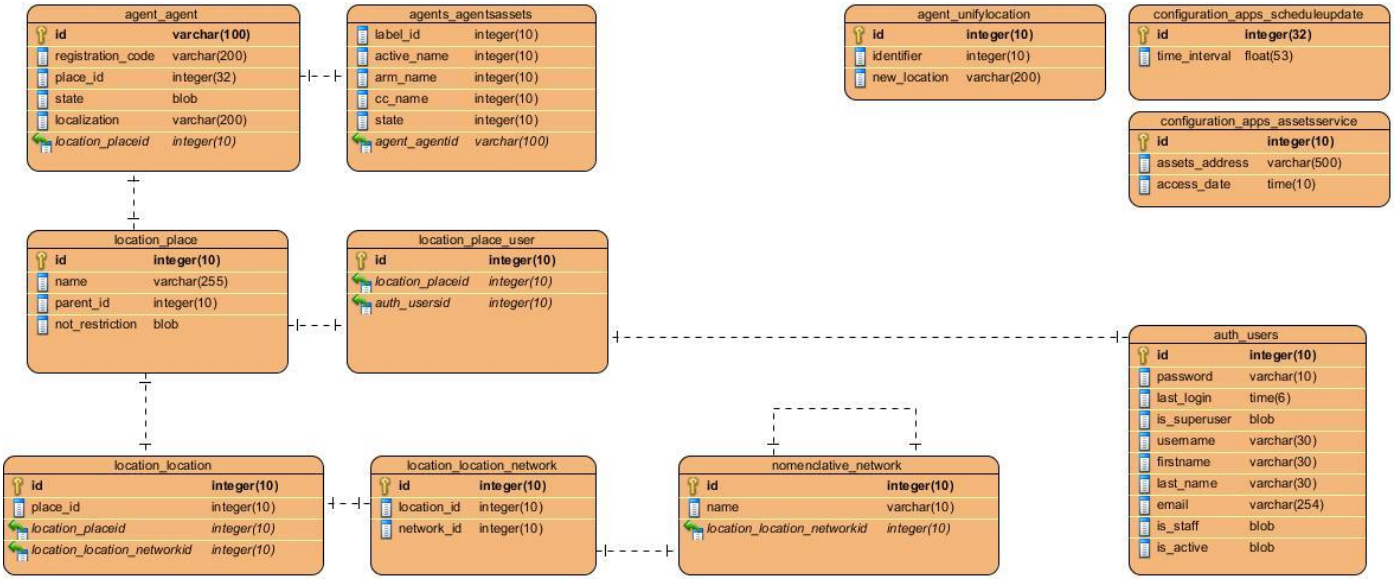


Figura 12: Diagrama Entidad-Relación

3.5 Estándares de codificación

En la implementación del sistema a desarrollar, se utilizará un estándar de codificación, que certificará legibilidad y organización al código de la misma, simplificando esfuerzos a la hora de darle mantenimiento y seguimiento al sistema. En estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible.

Para lograr este objetivo, se utilizó la Guía de estilo para el código Python (PEP 8). Esta guía posee una gran cantidad de convenciones para escribir código legible, dentro de las cuales se destacan:

- Usar cuatro espacios por indentación.
- Nunca mezclar tabulaciones y espacios.
- Limitar todas las líneas a un máximo de caracteres.
- Separar funciones de alto nivel y definiciones de clase con dos líneas en blanco, mientras que las

definiciones de métodos dentro de una clase son separadas por una línea en blanco.

- Codificación UTF-8 en todos los módulos.
- Las importaciones deben estar en líneas separadas.
- Evitar espacios en blanco innecesarios.

Se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado. También se debe inicializar cada variable en su declaración a menos que su valor inicial sea vacío (Calleja 2014).

3.6 Pruebas Unitarias. PyUnit

Las pruebas unitarias son uno de los pilares de las metodologías ágiles para el desarrollo de software. Todas las funcionalidades deben pasar las pruebas unitarias antes de ser liberadas o publicadas. Por otra parte, las pruebas deben ser definidas antes de realizar el código. Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del mismo.

PyUnit es un módulo de Python utilizado para la realización de las pruebas unitarias al código. La arquitectura de PyUnit es simple y efectiva. Las pruebas se implementan por herencia de una clase base, TestCase, que admite el comportamiento del dispositivo de prueba. El módulo PyUnit (unitest.py), no solo contiene el código base para la prueba de unidad en construcción, sino que también actúa como un corredor de prueba en el que permite ejecutarlas desde la consola (Hamill 2004).

Se realizan tres iteraciones para 11 pruebas unitarias a las clases y los métodos con que cuenta el sistema, sobre los que se detectan cinco errores que se resuelven de manera satisfactoria. En la figura 13, se muestran los resultados obtenidos durante la aplicación de dichas pruebas. Los resultados mostrados se corresponden a la última iteración, mostrando todas las pruebas realizadas en las iteraciones anteriores.

```
ok
1 items had no tests:
    __main__
11 items passed all tests:
   1 tests in __main__.AgentAssets_test
   1 tests in __main__.AssetsService_test
   1 tests in __main__.URL_test
   1 tests in __main__.get_access_date_test
   1 tests in __main__.get_baja_test
   1 tests in __main__.get_location_test
   1 tests in __main__.get_total_unavailable_test
   1 tests in __main__.hilo_test
   1 tests in __main__.save_location_test
   1 tests in __main__.schedule_test
   1 tests in __main__.unificar_loc_test
11 tests in 12 items.
11 passed and 0 failed.
Test passed.
```

Figura 13: Pruebas unitarias

3.7 Pruebas de Aceptación

Las pruebas de aceptación son creadas en base a los casos de uso. El cliente debe especificar uno o diversos escenarios para comprobar que un caso de uso ha sido correctamente implementado. Las pruebas de aceptación son consideradas como “pruebas de caja negra” (“*Black box system tests*”). Los clientes son responsables de verificar que los resultados de éstas sean correctos y en caso de que fallen varias, deben indicar el orden de prioridad de resolución. Un caso de uso no se puede considerar terminado hasta tanto pase correctamente todas las pruebas de aceptación. Dado que la responsabilidad es grupal, es recomendable publicar los resultados, de manera que todo el equipo esté al tanto de esta información.

El ejemplo que se expone a continuación, pertenece al caso de uso Unificar localización, en el que se realiza una descripción de todas las variables tanto de entrada como salida, para una mejor comprensión.

Descripción de las variables:

Tabla 14: Descripción de las variables. Casos de prueba

No.	Nombre de variable	Clasificación	Descripción
1	id_agent	Numérico (integer)	Contiene el id del agente seleccionado. Se debe seleccionar un agente para generar nueva localización.
2	location_place	Cadena (string)	Contiene la localización del agente registrada en XILEMA GRHS.
3	location_asset	Cadena (string)	Contiene la localización del agente obtenida del sistema Assets.

Descripción General:

Se ejecutaron pruebas para el caso de uso con diferentes entradas al sistema, con el objetivo de determinar que los resultados obtenidos fueran los esperados bajo cualquier situación, y así, dar por cumplidos los requerimientos del cliente.

A continuación, se presenta una muestra de los casos de prueba que se realizaron para el caso de uso Unificar Localización.

Tabla 15: Caso de prueba. Unificar Localización

Escenario	**Descripción**	**Id_agent**	**location_place**	**location_asset**	**Respuesta del Sistema**	**Flujo Central**
EC 1.1: Unificar Localización	El escenario está destinado a unificar localizaciones a partir de la selección de localización de XILEMA GRHS y el	V	V	V	El sistema conforma nueva localización a partir de ambas localizaciones. Se muestra la nueva localización.	El usuario selección en el módulo Agentes, el submenú Unificar localización. Presiona el botón Adicionar
		1	Doc1_Fac2_Lab305	LABORATORIO 306 DOC – 1 Centro de Telemática		

	sistema Assets					Identificador y rellena el campo Identificador, selecciona el botón Aceptar y se muestra la nueva localización generada.
EC 1.2 Localización inexistente.	El escenario está destinado a Unificar localizaciones con datos inválidos a partir de ambos sistemas.	V	I	I	El sistema no genera ninguna localización y muestra el mensaje de validación: Error, no se ha podido generar una localización.	
		1	<<null>>	<<null>>		
EC 1.3	El escenario está destinado a Unificar localizaciones con datos inválidos seleccionados de XILEMA GRHS y datos validos seleccionados del sistema Assets.	V	I	V	El sistema acogerá la localización propuesta por la variable 2. Se muestra la nueva localización.	
		1	<<null>>	CISED LABORATORIO 206 D-3 COMPONENTES (CISED)		
EC 1.4	El escenario	V	V	I	El sistema acogerá la	

	está destinado a Unificar localizaciones con datos válidos seleccionados de XILEMA GRHS y datos inválidos seleccionados del sistema Assets.	1	VRP_Doc6_CentroSoporte_Lab205	<<null>>	localización propuesta por la variable 1. Se muestra la nueva localización.	

Las celdas de la tabla contienen V, I, o N/A; V indica válido, I indica inválido y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

Se realizaron un total de 16 casos de pruebas los cuales se pueden encontrar en el **Anexo IV**.

3.7.1 Análisis de los resultados de las pruebas:

Para validar la implementación del software desarrollado, se realizaron tres iteraciones de prueba, en las cuales se detectaron 5 no conformidades, y todas fueron resueltas. La siguiente Figura muestra los resultados obtenidos durante las iteraciones.

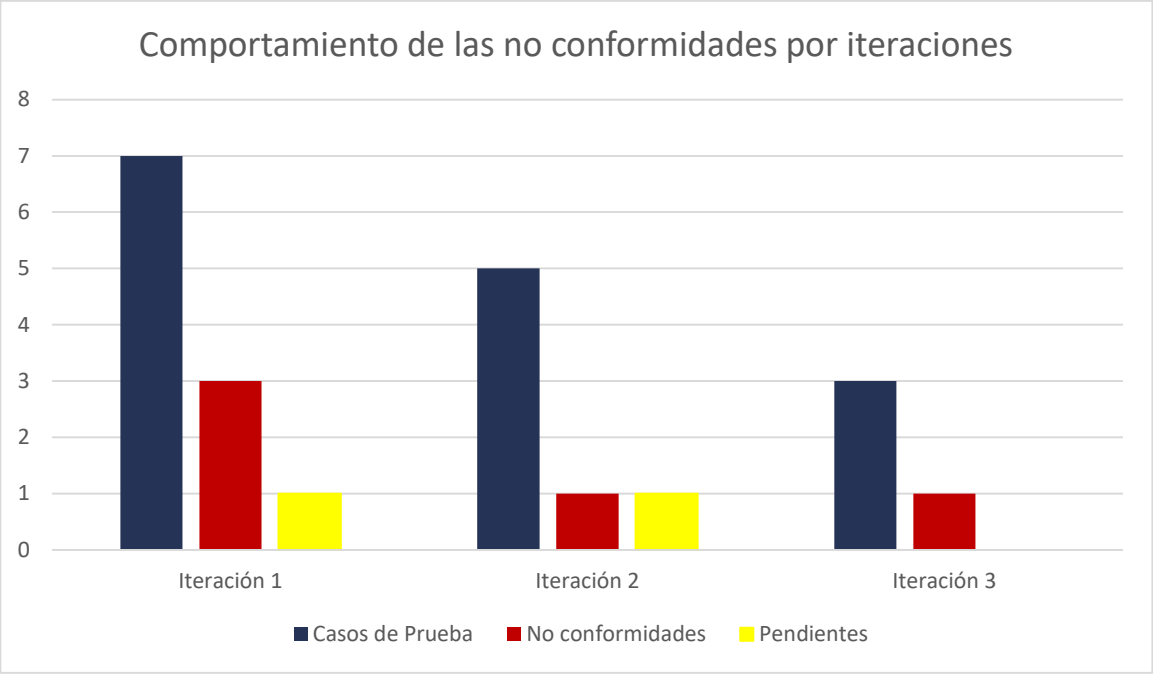


Figura 14: Diagrama de comportamiento de no conformidades

En la siguiente tabla se detalla una descripción de las no conformidades detectadas por el cliente.

Tabla 16: Descripción de no conformidades

No.	Descripción de no conformidades	No. de Iteración
1	No se validan los datos intervalo de tiempo.	1
2	El sistema no muestra notificación de error, cuando no se han podido actualizar los datos con el sistema Assets	1
3	Existen guiones bajos duplicados en la conformación de nuevas localizaciones.	2
4	Es posible adicionar más de una URL, lo cual es incorrecto, ya que XILEMA GRHS está solamente conectado a un solo sistema (Assets).	2
5	El servidor al reiniciarse deja de comprobar las fechas actuales con la última fecha de acceso al sistema, por lo que la funcionalidad de Actualización automática deja de ser efectiva.	3

Conclusiones del capítulo

En el capítulo concluido se determinó los patrones arquitectónicos que fueron utilizados en el desarrollo y cumplimiento de la propuesta de solución. Además, se identificaron los Patrones de Asignación de Responsabilidades (GRASP) y los Patrones de Diseño los cuales permitieron guiar la implementación

del software. Se confeccionó además el modelo entidad – relación para una mayor comprensión de las funcionalidades del sistema. Las pruebas de caja blanca y caja negra realizadas, permitieron arrojar resultados a través de la ejecución de casos de pruebas confeccionados para detectar no conformidades en el sistema.

Conclusiones Generales

Con la realización del presente trabajo de diploma se desarrolló una integración entre XILEMA GRHS y el sistema Assets que facilitó la obtención de información de las computadoras. Además, se desarrolló un componente capaz de mostrar a cada cliente, de forma local, un diagnóstico del cumplimiento de las políticas de seguridad informática, así como las características de hardware y software con que cuenta cada estación de trabajo. Se puede concluir afirmando que:

- El estudio realizado al proceso de obtención de datos del Gestor de Recursos de Hardware y Software, proyectó resultados que permitieron comprender la necesidad actual de desarrollar una personalización de XILEMA GRHS para suprimir los problemas encontrados.
- Se realizó un estudio del estado del arte que evidenció la carencia de sistemas que contaran con mecanismos que respondieran a las necesidades específicas del cliente. Además, se fundamentaron las herramientas y tecnologías las cuales posibilitaron la implementación del software deseado y así cumplir con el objetivo de la presente investigación.
- La realización de la propuesta de solución, permitió obtener un diseño que a través de artefactos de ingeniería de software fue posible modelar y describir las funcionalidades del sistema, a modo de facilitar el desarrollo del presente trabajo de diploma.
- A partir de la implementación, se desarrolló una personalización de XILEMA GRHS para la Universidad de las Ciencias Informáticas en aras de satisfacer las necesidades específicas del cliente.
- La ejecución de pruebas permitió validar la solución y asegurar el correcto funcionamiento del software desarrollado.

Recomendaciones

Se recomienda continuar con la investigación acerca de la contribución que podrían aportar sistemas de tipo ERP para aplicaciones destinadas al proceso de inventariado de hardware y software. Se propone incorporar al sistema de notificaciones vía correo electrónico que contiene XILEMA GRHS, nuevas alertas que permitan dar a conocer al usuario cuando exista algún cambio en el sistema Assets. Se sugiere, además, incorporar nuevas funcionalidades al componente desarrollado en la aplicación gclient para mostrar todas las características de hardware, software y el cumplimiento de las políticas de seguridad en sistemas operativos que carecen de interfaz visual.

Referencias Bibliográficas

ACCID, 2005. *Norma Internacional de Contabilidad NIC-NIF*. S.l.: Grupo Planeta (GBS). ISBN 9788496426627. Mayo, 2018

ALLEN, G. y OWENS, M., 2011. *The Definitive Guide to SQLite*. S.l.: Apress. ISBN 9781430232261.

ÁNGELO BENVENUTO VERA, 2012. Implementación de sistemas ERP, su impacto en la gestión de la empresa e integración con otras TIC. , vol. 4, no. 3.

ANTAL,BARZAN TONY, 2010. *IT Inventory and Resource Management with OCS Inventory NG 1.02*. S.l.: Packt Publishing Ltd. ISBN 9781849511117.

ARIAS, M.A., 2014. *Responsive Design con Bootstrap* [en línea]. IT Campus Academy: s.n. [Consulta: 29 marzo 2018]. ISBN 978-1495492099. Disponible en: https://books.google.com.cu/books?id=J8lJDQAAQBAJ&printsec=frontcover&dq=bootstrap&hl=es-419&sa=X&ved=0ahUKEwid5or2puLaAhXndN8KHTVODF4Q6AEIMzAB#v=onepage&q=bootstrap&f=false.

Arquitectura Basada en Componentes. *Scribd* [en línea], 2015. [Consulta: 14 junio 2018]. Disponible en: https://es.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes.

ASENSIO, R.M.-B., 2014. *Lenguaje-de-programacion-JavaScript* [en línea]. S.l.: s.n. [Consulta: 3 enero 2018]. Disponible en: http://www.um.es/docencia/barzana/DAWEB/Lenguaje-de-programacion-JavaScript-1.pdf.

ASSETS NS, 2018. *ASSETS: Sistema de Gestión Integral* [en línea]. 2018. S.l.: s.n. [Consulta: 25 enero 2018]. Disponible en: http://www.assets.co.cu/assets.asp.

BOEDER, J. y GROENE, B., 2014. *The Architecture of SAP ERP: Understand how successful software works*. S.l.: Tredition. ISBN 9783849576622.

CALLEJA, M.A., 2014. Estándares de Codificación. [en línea], [Consulta: 16 marzo 2018]. Disponible en: http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf.

CHARLES EDEKI, 2013. Agile Unified Process. En: Ph.D, American Intercontinental University, Department of Information Technology, 160 Parkside Ave, 9M, Brooklyn NY, 11226, *INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND MOBILE APPLICATIONS*, vol. 1, pp. 13-17. ISSN 2321-8363.

CONTRALORÍA GENERAL DE LA REPÚBLICA DE CUBA, 2011. NOTA INFORMATIVA - 4 Res. 60-11 Sobre las Normas del Sistema de Control Interno.pdf. [en línea]. [Consulta: 29 abril 2018]. Disponible en: http://bvs.sld.cu/revistas/infd/n1311/4%20Res.%2060-11%20Sobre%20las%20Normas%20del%20Sistema%20de%20Control%20Interno..pdf.

DEFINICIONES-DE, 2010. Qué significa activo fijo - Información, significado y ejemplos de oraciones con activo fijo. En: Diccionario Enciclopédico, *Definiciones-de.com*.

DEMIÁN GUTIÉRREZ, 2009. UML. Diagramas de Paquetes. [en línea]. Universidad de los Andes, Venezuela: s.n., [Consulta: 3 abril 2018]. Disponible en: www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf.

FARRO, M., 2007. *Estudio de los sistemas de gestión de recursos empresariales (ERP) en el Perú orientado al pymes* [en línea]. Tesis para optar el título de Licenciado en Ingeniería Industrial y de Sistemas. Piura, Perú: Facultad de Ingeniería, Universidad de Piura. Disponible en: https://hdl.handle.net/11042/1220.

GOMAA, H., 2011. *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*. S.l.: Cambridge University Press. ISBN 9781139494731. pág. 280

GUERRERO, C.A., SUÁREZ, J.M. y GUTIÉRREZ, L.E., 2013. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *Información tecnológica*, vol. 24, no. 3, pp. 103-114. ISSN 0718-0764. DOI 10.4067/S0718-07642013000300012.

GUIDO VAN ROSSUM, 2009. *El Tutorial de Python* [en línea]. Fred L. Drake, Jr. S.l.: s.n. [Consulta: 25 enero 2018]. Disponible en: http://docs.python.org.ar/tutorial/pdfs/TutorialPython3.pdf.

GUTIÉRREZ, L. E, 2010. *Arquitectura Software. Investigación Aplicada a la Construcción de Marcos de Trabajo-Bucaramanga*. 2010. S.l.: Ltda.

HAMILL, P., 2004. *Unit Test Frameworks: Tools for High-Quality Software Development*. S.l.: O'Reilly Media, Inc. ISBN 9780596552817.

HOLOVATY, A. y KAPLAN-MOSS, J., 2009. *The definitive guide to Django: Web development done right* [en línea]. S.l.: Apress. ISBN 9781430219378. Disponible en: https://books.google.com.cu/books?hl=es&lr=&id=h2tR8p-4a9QC&oi=fnd&pg=PR27&dq=The+definitive+guide+to+Django:+Web+development+done+right&o

ts=Voj7c49JOb&sig=esaAgFarLPD8u6AvgwADtnaEcJU&redir_esc=y#v=onepage&q=The%20definitive%20guide%20to%20Django%3A%20Web%20development%20done%20right&f=false.

IBM, 2012. IBM Knowledge Center - Establecer datos coherentes mediante estandarización. [en línea]. [Consulta: 29 abril 2018]. Disponible en: https://www.ibm.com/support/knowledgecenter/es/SSZJPZ_9.1.0/com.ibm.swg.im.iis.qs.ug.doc/topics/c_Conforming_output_data.html?view=embed.

Inventory Management Definition & Example InvestingAnswers. [en línea], 2018. [Consulta: 25 enero 2018]. Disponible en: http://www.investinganswers.com/financial-dictionary/financial-statement-analysis/inventory-management-5999.

ISLAM, Q.N., 2015. *Mastering PyCharm*. S.l.: Packt Publishing Ltd. ISBN 9781783551323.

JOSÉ ENRIQUE GONZÁLEZ CORNEJO, 2008. El Lenguaje de Modelado Unificado (UML). .

LAPUENTE, M.J.L. y LAPUENTE, C.L., 2013. HTML. [en línea]. [Consulta: 29 abril 2018]. Disponible en: http://www.hipertexto.info/documentos/html.htm.

LARMAN, C., 2003. *UML y Patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* S.l.: Pearson Education. ISBN 9788420534381.

LOGINVENTORY, 2018. About Us - LOGINventory. [en línea]. [Consulta: 26 enero 2018]. Disponible en: https://www.loginventory.de/aboutus/?lang=en.

LUCID HART, 2016. Diagrama de clase. *Lucidchart* [en línea]. [Consulta: 29 abril 2018]. Disponible en: https://www.lucidchart.com/pages/es/diagrama-de-clase.

OCS INVENTORY, 2018. OCS Inventory NG. [en línea]. [Consulta: 26 marzo 2018]. Disponible en: https://www.ocsinventory-ng.org/en/.

PEÑAS LÓPEZ, ANA, 2016. *Implantación del ERP Odoo en una PYME dedicada al Comercio Minorista* [en línea]. ESCUELA DE INGENIERIAS INDUSTRIALES, Valladolid: UNIVERSIDAD DE VALLADOLID. [Consulta: 5 marzo 2018]. Disponible en: https://uvadoc.uva.es/bitstream/10324/16892/1/TFG-I-381.pdf.

PGADMIN - OFFICIAL PAGE, 2018. Introduction — pgAdmin III 1.22.2 documentation. [en línea]. [Consulta: 29 abril 2018]. Disponible en: https://www.pgadmin.org/docs/pgadmin3/1.22/introduction.html.

POSTGRESQL, 2018. PostgreSQL: The world's most advanced open source database. [en línea]. [Consulta: 9 marzo 2018]. Disponible en: https://www.postgresql.org/.

RAE, 2017. Computadora. *Diccionario de la Lengua Española* [en línea]. Edicion del Tricentenario. Real Academia Española. Disponible en: http://dle.rae.es/?id=A4hIGQC.

RODRÍGUEZ ADRIAN E MENA, 2015. *GESTOR DE RECURSOS DE HARDWARE Y SOFTWARE*. 2015. S.l.: s.n.

RODRÍGUEZ, C.H. y RODRÍGUEZ, J.E.R., 2013. Preprocesamiento de datos estructurados. *Revista Vínculos*, vol. 4, pp. 27-48.

SÁNCHEZ, T.R., 2017. *Metodología de desarrollo para la actividad productiva de la UCI. AUP-UCI* [en línea]. noviembre 2017. S.l.: s.n. [Consulta: 27 enero 2018]. Disponible en: Metodologia_AUP_UCI.pdf.

SARMIENTO, J., 2014. UML: Diagrama de Despliegue. [en línea]. Bogotá: La Empresa: Disponible en: http://umldiagramadespliegue.com.

SOMMERVILLE, Ian, 2010. *Ingeniería de Software*. 7ma Edición. New York: Pearson Addison Wesley. ISBN 84-7829-074-5.

TLM, 2016. *Xilema Base Web*. 2016. S.l.: s.n.

UCI, 2018. Centro de Telemática (TLM) | Universidad de las Ciencias Informáticas. [en línea]. [Consulta: 29 abril 2018]. Disponible en: https://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo/centro-de-telematica-tlm.

VISUAL PARADIGM, 2014. Tutorial Visual Paradigm - PracticaVP. [en línea]. [Consulta: 3 marzo 2018]. Disponible en: http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

VISUAL PARADIGM, 2016. Entity Relationship Diagram - Data Modeling - UML Diagramming Software. [en línea]. [Consulta: 29 abril 2018]. Disponible en: https://www.visual-paradigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html.

Bibliografía

1. LOGINVENTORY. About Us - LOGINventory. [online]. Enero 2018. [Accessed 26 January 2018]. Available from: https://www.loginventory.de/aboutus/?lang=en

2. CHARLES EDEKI. Agile Unified Process. . September 2013. Vol. 1, no. 3, p. 13–17. Ph.D, American Intercontinental University, Department of Information Technology, 160 Parkside Ave, 9M, Brooklyn NY, 11226

3. Arquitectura Basada en Componentes. *Scribd* [online]. 2015. [Accessed 14 June 2018]. Available from: https://es.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes

4. FERNÁNDEZ, Llorente, Alberto. *Arquitectura Cliente-Servidor*. 2005. Google-Books-ID: 4GwjcgAACAAJ

5. GUTIÉRREZ, L. E. *Arquitectura Software. Investigación Aplicada a la Construcción de Marcos de Trabajo-Bucaramanga*. 2010. Ltda. Colombia

6. ASSETS NS. *ASSETS: Sistema de Gestión Integral* [online]. 2018. [Accessed 25 January 2018]. Available from: http://www.assets.co.cu/assets.asp

7. UCI. Centro de Telemática (TLM) | Universidad de las Ciencias Informáticas. [online]. 2018. [Accessed 29 April 2018]. Available from: https://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo/centro-de-telematica-tlm

8. RAE. Computadora. *Diccionario de la Lengua Española* [online]. Edicion del Tricentenario. Real Academia Española, 2017. Available from: http://dle.rae.es/?id=A4hIGQC

9. ASALE, RAE-. control. *Diccionario de la lengua española* [online]. [Accessed 12 January 2018]. Available from: http://dle.rae.es/?id=AeYZ09VVersión electrónica del Diccionario de la lengua española, obra lexicográfica académica por excelencia.

10. MONTAÑO, Nicolás. *Descripción de actividades. Asignación de Activos en la herramienta SAP* [online]. Universidad de los Andes, 2015. [Accessed 30 April 2018]. Available from: https://planeacion.uniandes.edu.co/dmdocuments/PRO-27-2-01-

02_Asignaci%C3%B3n_de_activos.pdf

11. LUCID HART. Diagrama de clase. *Lucidchart* [online]. 15 March 2016. [Accessed 29 April 2018]. Available from: https://www.lucidchart.com/pages/es/diagrama-de-clase

12. HUMBERTO, COSSIO TABOADA MIGUEL and OVIDIO, HUANCA APAZA FLOREN. *Diagrama de Despliegue*. LA PAZ – BOLIVIA : Universidad Salesiana de Bolivia, Ingeniería de Sistemas, 2009.

13. RAE. DLE: inventario - Diccionario de la lengua española - Edición del Tricentenario. [online]. [Accessed 12 January 2018]. Available from: http://dle.rae.es/?id=M2v6jgO

14. JOSÉ ENRIQUE GONZÁLEZ CORNEJO. El Lenguaje de Modelado Unificado (UML). Enero 2008.

15. CAROLINA PLASENCIA ASOREY. El Sistema de Control Interno: garantia del logro de los objetivos. *MEDISAN* [online]. July 2010. Vol. 14. [Accessed 23 March 2018]. Available from: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1029-30192010000500001

16. GUIDO VAN ROSSUM. *El Tutorial de Python* [online]. Fred L. Drake, Jr., [no date]. [Accessed 12 January 2018]. Available from: http://python.org.ar/pyar/Tutorial

17. VISUAL PARADIGM. Entity Relationship Diagram - Data Modeling - UML Diagramming Software. [online]. 2016. [Accessed 29 April 2018]. Available from: https://www.visual-paradigm.com/VPGallery/datamodeling/EntityRelationshipDiagram.html

18. SAMARA, Tarek. *ERP and Information Systems: Integration Or Disintegration*. John Wiley & Sons, 2015. ISBN 9781848218963.

19. CALLEJA, Manuel Arias. Estándares de Codificación. [online]. 2014. [Accessed 16 March 2018]. Available from: http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf

20. FARRO, M. *Estudio de los sistemas de gestión de recursos empresariales (ERP) en el Perú orientado al pymes* [online]. Tesis para optar el título de Licenciado en Ingeniería Industrial y de

Sistemas. Piura, Perú : Facultad de Ingeniería, Universidad de Piura, 2007. Available from: https://hdl.handle.net/11042/1220Universidad de Piura

21. RODRÍGUEZ ADRIAN E MENA. *GESTOR DE RECURSOS DE HARDWARE Y SOFTWARE*. 2015.

22. TLM, UCI. Gestor de Recursos de Hardware y Software. [online]. [Accessed 12 January 2018]. Available from: http://www.informaticahabana.cu/sites/default/files/ponencias/SEG14.pdf

23. ORDOÑES, Yoanni, AVILÉS, Ernesto, PÉREZ, Julio and RODRÍGUEZ, Odaysa. GRHS: Gestor de Recursos de Hardware y Software. In : . 9 April 2014.

24. CENTER FOR HISTORY AND NEW MEDIA. Guía rápida. [online]. Available from: http://zotero.org/support/quick_start_guide

25. LAPUENTE, María Jesús Lamarca and LAPUENTE, Chusa Lamarca. HTML. [online]. 2013. [Accessed 29 April 2018]. Available from: http://www.hipertexto.info/documentos/html.htm

26. IBM. IBM Knowledge Center - Establecer datos coherentes mediante estandarización. [online]. 11 October 2012. [Accessed 29 April 2018]. Available from: https://www.ibm.com/support/knowledgecenter/es/SSZJPZ_9.1.0/com.ibm.swg.im.iis.qs.ug.doc/topics/c_Conforming_output_data.html?view=embed

27. PEÑAS LÓPEZ, ANA. *Implantación del ERP Odoo en una PYME dedicada al Comercio Minorista* [online]. ESCUELA DE INGENIERIAS INDUSTRIALES, Valladolid : UNIVERSIDAD DE VALLADOLID, 2016. [Accessed 5 March 2018]. Available from: https://uvadoc.uva.es/bitstream/10324/16892/1/TFG-I-381.pdf

28. ÁNGELO BENVENUTO VERA. Implementación de sistemas ERP, su impacto en la gestión de la empresa e integración con otras TIC. . 2012. Vol. 4, no. 3.

29. SOMMERVILLE, lan. *Ingeniería de Software*. 7ma Edición. New York : Pearson Addison Wesley, 2010. ISBN 84-7829-074-5.

30. PGADMIN - OFFICIAL PAGE. Introduction — pgAdmin III 1.22.2 documentation. [online]. 2018. [Accessed 29 April 2018]. Available from: https://www.pgadmin.org/docs/pgadmin3/1.22/introduction.html

31. ASALE, RAE-. inventario. *Diccionario de la lengua española* [online]. [Accessed 12 January 2018]. Available from: http://dle.rae.es/?id=M2v6jgOVersión electrónica del Diccionario de la lengua española, obra lexicográfica académica por excelencia.

32. INVESTINGANSWERS. Inventory Management Definition & Example | InvestingAnswers. [online]. 2018. [Accessed 29 April 2018]. Available from: http://www.investinganswers.com/financial-dictionary/financial-statement-analysis/inventory-management-5999

33. ANTAL,BARZAN TONY. *IT Inventory and Resource Management with OCS Inventory NG 1.02*. Packt Publishing Ltd, 2010. ISBN 9781849511117. Google-Books-ID: jgysqOVMqi8C

34. UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA (ÚLTIMO). LENGUAJE DE MODELADO UNIFICADO UML. [online]. December 2016. [Accessed 29 April 2018]. Available from: http://stadium.unad.edu.co/ovas/10596_9839/qu_es_uml.html

35. ASENSIO, Rafael Menéndez-Barzanallana. *Lenguaje-de-programacion-JavaScript* [online]. 2014. [Accessed 3 January 2018]. Available from: http://www.um.es/docencia/barzana/DAWEB/Lenguaje-de-programacion-JavaScript-1.pdf

36. ISLAM, Quazi Nafiul. *Mastering PyCharm*. Packt Publishing Ltd, 2015. ISBN 9781783551323. Google-Books-ID: MPh_CwAAQBAJ

37. SÁNCHEZ, Tamara Rodríguez. *Metodología de desarrollo para la actividad productiva de la UCI. AUP-UCI* [online]. November 2017. [Accessed 27 January 2018]. Available from: Metodologia_AUP_UCI.pdf

38. ESCUELA DE POSTGRADO. INSTITUTO TECNOLÓGICO DE BUENOS AIRES. METODOLOGÍA PARA SELECCIÓN DE SISTEMAS ERP. [online]. [Accessed 12 January 2018]. Available from: http://www.ucla.edu.ve/dac/departamentos/informatica-II/metodologia-para-

seleccion-de-sistemas-erp.PDF

39. NIC - Normas Internacionales de Contabilidad - Normas Internacionales de Información Financiera. [online]. [Accessed 29 April 2018]. Available from: http://www.normasinternacionalesdecontabilidad.es/nic/nic.htm

40. ACCID. *Norma Internacional de Contabilidad NIC-NIF*. Grupo Planeta (GBS), 2005. ISBN 9788496426627. Mayo, 2018

41. CONTRALORÍA GENERAL DE LA REPÚBLICA DE CUBA. NOTA INFORMATIVA - 4 Res. 60-11 Sobre las Normas del Sistema de Control Interno.pdf. [online]. 1 March 2011. [Accessed 29 April 2018]. Available from: http://bvs.sld.cu/revistas/infd/n1311/4%20Res.%2060-11%20Sobre%20las%20Normas%20del%20Sistema%20de%20Control%20Interno..pdf

42. OCS INVENTORY. OCS Inventory NG. [online]. Enero 2018. [Accessed 26 March 2018]. Available from: https://www.ocsinventory-ng.org/en/

43. APPRISE. Our Story - Apprise. [online]. Enero de 2018. [Accessed 25 January 2018]. Available from: http://www.apprise.com/company/our-story.asp

44. GUERRERO, Carlos A., SUÁREZ, Johanna M. and GUTIÉRREZ, Luz E. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *Información tecnológica*. January 2013. Vol. 24, no. 3, p. 103–114. DOI 10.4067/S0718-07642013000300012.

45. AYALA, Jose Luis Condori. Phython - DjangoFramework de desarrollo web para perfeccionistasBasado en el Modelo MTV. *Revista de Información, Tecnología y Sociedad*. 2012. P. 36.

46. CENTRO DE TELEMATICA, UCI. *Plugin para Xilema Base Web*. 18 March 2015.

47. POSTGRESQL. PostgreSQL: The world’s most advanced open source database. [online]. 2018. [Accessed 9 March 2018]. Available from: https://www.postgresql.org/

48. RODRÍGUEZ, Claudia Hernandez and RODRÍGUEZ, Jorge Enriquez Rodríguez.

Preprocesamiento de datos estructurados. *Revista Vínculos*. 2013. Vol. 4, p. 27–48.

49. CLEMENTE, Alexis López, BELTRÁN, Dauvildo Hurtado and GONZÁLEZ, Nancy Jiménez. 169: *Propuesta de acciones para solucionar las deficiencias en cuanto al control de los activos fijos tangibles en Cuba* [online]. Observatorio de la Economía Latinoamericana : Filial Universitária de Salud Município de Yaguajay, 2012. [Accessed 18 January 2018]. Available from: http://www.eumed.net/cursecon/ecolat/cu/2012/

50. DEFINICIONES-DE. Qué significa activo fijo - Información, significado y ejemplos de oraciones con activo fijo. *Definiciones-de.com. 2010. Diccionario Enciclopédico*

51. ARIAS, Miguel A. Responsive Design con Bootstrap [online]. IT Campus Academy, 2014. [Accessed 29 March 2018]. ISBN 978-1495492099. Available from: https://books.google.com.cu/books?id=J8lJDQAAQBAJ&printsec=frontcover&dq=bootstrap&hl=es-419&sa=X&ved=0ahUKEwid5or2puLaAhXndN8KHTVODF4Q6AEIMzAB#v=onepage&q=bootstrap&f=false

52. MORAN, ING LUIS ARRIAZA. Sistemas de información gerencial. [online]. 2001. [Accessed 20 February 2018]. Available from: www.academia.edu/download/36710601/PORTAFOLIO_SIG.pdfUniversidad Don Bosco, Facultad de Ingeniería, Escuela: Computación

53. AMAYA,JAIRO AMAYA. *Sistemas de información gerenciales: Hardware, software, redes, Internet, diseño*. ECOE EDICIONES, 2010. ISBN 9789586486354. Google-Books-ID: nZzFAQAAQBAJ

54. GOMAA, Hassan. *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*. Cambridge University Press, 2011. ISBN 9781139494731. pág. 280

55. BOEDER, Jochen and GROENE, Bernhard. *The Architecture of SAP ERP: Understand how successful software works*. Tredition, 2014. ISBN 9783849576622. Google-Books-ID: hcsZAwAAQBAJ

56. HOLOVATY, Adrian and KAPLAN-MOSS, Jacob. *The definitive guide to Django: Web development done right* [online]. Apress, 2009. ISBN 9781430219378. Available from: https://books.google.com.cu/books?hl=es&lr=&id=h2tR8p-4a9QC&oi=fnd&pg=PR27&dq=The+definitive+guide+to+Django:+Web+development+done+right&ots=Voj7c49JOb&sig=esaAgFarLPD8u6AvgwADtnaEcJU&redir_esc=y#v=onepage&q=The%20definitive%20guide%20to%20Django%3A%20Web%20development%20done%20right&f=false

57. ALLEN, Grant and OWENS, Mike. *The Definitive Guide to SQLite*. Apress, 2011. ISBN 9781430232261.

58. VISUAL PARADIGM. Tutorial Visual Paradigm - PracticaVP. [online]. 2014. [Accessed 3 March 2018]. Available from: http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf

59. SARMIENTO, Johana. *UML: Diagrama de Despliegue* [online]. Bogotá: La Empresa, 2014. Available from: http://umldiagramadespliegue.com

60. DEMIÁN GUTIÉRREZ. UML. Diagramas de Paquetes. In : [online]. Universidad de los Andes, Venezuela, 2009. [Accessed 3 April 2018]. Available from: www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf

61. LARMAN, Craig. *UML y Patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Pearson Education, 2003. ISBN 9788420534381.

62. HAMILL, Paul. *Unit Test Frameworks: Tools for High-Quality Software Development*. O'Reilly Media, Inc., 2004. ISBN 9780596552817. Google-Books-ID: 2ksvdhhnWQsC

63. TLM. *Xilema Base Web*. 2016.