Facultad 2

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

Licenciador por tiempo para el sistema Gestor de Recursos de Hardware y Software

Autores: Daniela de la Caridad Hernández de la Portilla

Tutores: MSc.Madelis Pérez Gil
Ing. Fernando Ricardo Romero
Ing. Ramón Guzmán Alemañy

La Habana, junio de 2019

‘‘La verdadera fuerza de una persona no se encuentra en los músculos, sino en la capacidad de vencer las adversidades y superarse, en la capacidad de cumplir los retos, convertirlos en metas y las metas convertirlas en realidades’’.

Anónimo

¿Por dónde comenzar?
¡Algunos dirían que por el principio! Pues entonces a la primera persona que tengo que dar gracias es ***a mi madre*** *que me dio la vida, que es lo más grande que tengo en este mundo y que es la persona que más ha confiado en mí incluso en esos momentos en los que no confiaba ni yo.*
A mi padre*, que es mi modelo a seguir, mi motor impulsor, mi critico más ferviente y mi mayor admirador.*
A ***mi hermana*** *que, aunque ya es casi mi hija es mi mayor amiga, la primera de ellas, la más leal de todas.*
A ***mis abuelos****, a todos ellos, a los que ya no están pero que estarán siempre porque los llevo grabados en el corazón y me acompañan cada día.*
A ***mi familia*** *toda, que, aunque pequeña son lo mejor del mundo y si me hubieran dado a escoger los escogería exactamente igual.*
Agradezco todo el apoyo, todo el amor, las horas de sueño perdidas, los momentos vividos (buenos, malos, regulares, remalos y geniales), todo lo aprendido, pero sobre todas las cosas agradezco lo incondicionales que han sido; se lo agradezco a mis amigos(as). Son muchos, demasiados como para que no me falte nadie, pero ellos lo saben, que los amo, que para mí son familia, y que la familia es sagrada.
A ***mis profesores*** *de la carrera, a mis compañeros (a los que se quedaron por el camino y los que lo terminaron), a todos los que he conocido (que son muchísimos) y todos con los que he convivido (que son un poquito menos), de todos me llevo los mejores recuerdos de los que serán posiblemente los mejores años de mi vida.*
No por último son menos importantes ***mis tutores****, que no pueden faltar porque se merecen todo el reconocimiento y los aplausos. Profesionales de éxito, seres humanos increíbles, pero sobre todas las cosas los mejores súper tutores del mundo mundial. Que sepan que me siento orgullosa (y no lo pongo en mayúsculas porque sería un error ortográfico) de que me hayan guiado (lo que no fue fácil) por este camino tan importante en la vida de un estudiante. Hoy son como mi familia y el cariño que les tengo es real, porque los admiro y los respeto por sobre todas las cosas, por todo muchísimas gracias.*

A todos, ***MUCHAS GRACIAS***

Declaración de autoría

Declaro ser la autora de la presente tesis que tiene por título: “Licenciador por tiempo para el sistema Gestor de Recursos de Hardware y Software” y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los 12 días del mes de junio del año 2019.

Daniela de la
Caridad Hernández
de la Portilla

Firma del Autor

MSc. Madelis
Pérez Gil

Firma del Tutor

Ing. Fernando
Ricardo Romero

Firma del Tutor

Ing. Ramón
Guzmán Alemañy

Firma del Tutor

Datos de contacto

Tutor 1:

MSc. Madelis Pérez Gil: Ingeniera en Ciencias Informáticas 2008. Máster en Gestión de Información de la Facultad de Economía / Universidad de La Habana / Cátedra UNESCO 2014. Profesora Auxiliar del Dpto. de Ingeniería de Software de la Facultad 2. Vicedecana de Formación de la Facultad 2.

Email: mgil@uci.cu
Universidad de las Ciencia Informáticas, La Habana, Cuba

Tutor 2:

Ing. Ramón Guzmán Alemañy: Graduado en el año 2017 como Ingeniero en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas. Se desempeña como especialista A en Ciencias Informáticas en el Departamento de Desarrollo de Aplicaciones del Centro Telemática. Ha desempeñado el rol de desarrollador en el proyecto y hoy es el líder del proyecto XILEMA GRHS.
Universidad de las Ciencias Informáticas, La Habana, Cuba
Email: ralemany@uci.cu

Tutor 3:

Ing. Fernando Ricardo Romero. Graduado en el año 2014 como Ingeniero en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas. Se desempeña como especialista B en Ciencias Informáticas en el Departamento de Desarrollo de Aplicaciones del Centro Telemática. Ha desempeñado el rol de desarrollador en el sistema XILEMA GRHS.
Universidad de las Ciencias Informáticas
Email: fricardo@uci.cu

Agradecimientos

¿Por dónde comenzar? ¡Algunos dirían que por el principio! Pues entonces a la primera persona que tengo que dar gracias es a mi madre que me dio la vida, que es lo más grande que tengo en este mundo y que es la persona que más ha confiado en mí incluso en esos momentos en los que no confiaba ni yo. A mi padre, que es mi modelo a seguir, mi motor impulsor, mi critico más ferviente y mi mayor admirador. A mi hermana que, aunque ya es casi mi hija es mi mayor amiga, la primera de ellas, la más leal de todas. A mis abuelos, a todos ellos, a los que ya no están pero que estarán siempre porque los llevo grabados en el corazón y me acompañan cada día. A mi familia toda, que aunque pequeña son lo mejor del mundo y si me hubieran dado a escoger los escogería exactamente igual.

Agradezco todo el apoyo, todo el amor, las horas de sueño perdidas, los momentos vividos (buenos, malos, regulares, rémalos y geniales), todo lo aprendido, pero sobre todas las cosas agradezco lo incondicionales que han sido; se lo agradezco a mis amigos(as). Son muchos, demasiados como para que no me falte alguien, pero ellos lo saben, que los amo, que para mí son familia, y que la familia es sagrada.

A mis profesores de la carrera, a mis compañeros (a los que se quedaron por el camino y los que lo terminaron), a todos los que he conocido (que son muchísimos) y todos con los que he convivido (que son un poquito menos), de todos me llevo los mejores recuerdos de los que serán posiblemente los mejores años de mi vida.

No por ultimo son menos importantes mis tutores, que no pueden faltar porque se merecen todo el reconocimiento y los aplausos. Profesionales de éxito, seres humanos increíbles, pero sobre todas las cosas los mejores súper tutores del mundo mundial. Que sepan que me siento orgullosa (y no lo pongo en mayúsculas porque sería un error ortográfico) de que me hayan guiado (lo que no fue fácil) por este camino tan importante en la vida de un estudiante. Hoy son como mi familia y el cariño que les tengo es real, porque los admiro y los respeto por sobre todas las cosas, por todo muchísimas gracias.

Dedicatoria

A mis padres, a mis tutores, a todos mis profesores, y a todos mis amigos que son demasiados como para llenar estas páginas.

Resumen

Las licencias de software son unos de los mecanismos más utilizados en la actualidad para gestionar y controlar la comercialización de un software dado. Se utilizan en distintos escenarios y con diferentes fines, pero su mayor función es monetizar el software de la mejor forma posible constituyendo de esta forma una herramienta útil para las empresas comercializadoras y productoras de software. La presente investigación permitió desarrollar el licenciador por tiempo para el sistema Gestor de Recursos de Hardware y Software (XILEMA GRHS) v2.0. El mismo permite que el sistema cuente con una licencia temporal además de la licencia por usuarios concurrentes ya existente, de esta forma el Centro de Telemática adscrito a la Facultad 2 de la Universidad de las Ciencias Informáticas como empresa se vea beneficiado, manteniendo así un mejor control de su producto y logrando un mayor índice de ingresos netos. Se analizan las herramientas, tecnologías y metodologías de desarrollo con el fin de establecer la mejor selección para la construcción del licenciador, seleccionándose Proceso Unificado Ágil (AUP-UCI), como metodología, Python, PostgreSQL y Django para el desarrollo de la aplicación. Se describen los artefactos generados en las fases que establece la metodología de desarrollo de software seleccionada. Durante el desarrollo del licenciador por tiempo para XILEMA GRHS, este fue sometido a pruebas para propiciar al cliente conformidad y unidad en el producto final.

Palabras clave: licencia, licenciante, licencia de software, licencia temporal, monetizar.

Abstract

Software licenses are one of the most used mechanisms today to manage and control the marketing of a given software. They are used in different scenarios and with different purposes, but their main function is to monetize the software in the best possible way, thus constituting a useful tool for marketing companies and software producers. The present investigation allowed to develop the licensor by time for the Hardware and Software Resource Management System (XILEMA GRHS) v2.0, according to acronym in Spanish language. It allows the system to have a temporary license in addition to the existing concurrent user license, in this way the Telematics center assigned to faculty 2 of The University of Informatics Sciences as an independent company is benefited, thus maintaining a better control of its product and achieving a higher rate of net income. The tools, technologies and development methodologies are analyzed in order to establish the best selection for the construction of the licensor, selecting Agile Unified Process (AUP-UCI), as methodology, Python, PostgreSQL and Django for the development of the application. The artifacts generated in the phases established by the selected software development methodology are exposed. During the elaboration of the product it was subjected to tests to propitiate to the client conformity and unit in the developed application.

Keywords: *license, licensor, software license, temporary license, monetize.*

Índice

Índice de tablas

Índice de figuras

Introducción

En la industria de software las últimas tendencias proponen un enriquecimiento de licencias con mayor flexibilidad a permitir nuevos usos controlados del software y resolver apropiadamente los problemas reales de los clientes, según *Luis Daniel Soto, Director de Evangelización en Nuevas Tecnologías en Microsoft*. Estas indican que la protección y el licenciamiento de las soluciones que se desarrollan tienen gran importancia, ya que el uso no autorizado de las mismas puede provocar grandes pérdidas. Estas pueden ser tanto monetarias como intelectuales y en ambos casos afecta de sobremanera a las empresas productoras de software.

Las licencias son una de las principales formas de controlar un sistema informático a la hora de su comercialización, ya que permiten que el software funcione al cien por ciento o sea limitado. Eso depende de las pautas que se definan entre el cliente y la empresa comercializadora de software.(Lugo Rojas & Simón Méndez, 2015). Siendo esta una de las principales razones por las que se licencia un software en la actualidad. Las aplicaciones de licencias también se han propuesto en los últimos años para ayudar a un administrador a controlar de manera eficiente el uso de productos, algunas de las más comunes en el mercado son las licencias temporales y las licencias de usuarios concurrentes. Estas se utilizan sobre todo para una mejor comercialización del producto y son una forma de monetizar el uso de un software determinado.(Salvatore D'Alo, 2019)Y aunque todo indica que en un futuro no muy lejano los software migrarán hacia la nube completamente, el licenciamiento de los software comerciales prevalece en el mercado brindándole al usuario mayor seguridad en el producto y evitando de esta forma la contaminación del sistema, principal inconveniente del software libre.

Los distintos tipos de licencia se utilizan dependiendo de las características del producto a licenciar y del uso que se espera tenga el mismo, permitiendo así a las distintas empresas elegir la licencia o las licencias más adecuadas para controlar su producto. Estas se vinculan además con distintas políticas de seguridad que ayudan a comercializar un producto más completo.

Una de las fortalezas que presentan los softwares privativos es la referida al intercambio que se establece entre el cliente y el productor, siendo una opción que tiene el cliente de

comunicar al productor cualquier dificultad que presente el producto adquirido. Intercambio que posibilita mejorar la funcionabilidad de la aplicación para así adecuarla mejor a las necesidades de los clientes. Aquellas instituciones que no utilizan aplicaciones obtenidas por las vías convencionales de los productores de software están expensas a no contar con todas las prestaciones que ofrece el producto y en consecuencia afrontar todos los riesgos que esto trae.(Lugo Rojas & Simón Méndez, 2015). Como por ejemplo corren el riesgo de someterse a disputas legales por el uso indebido o desautorizado del software en cuestión, además de no contar con las políticas de seguridad correspondientes o con las versiones más actualizadas, limitando de esta forma el funcionamiento pleno del sistema.

En Cuba se desarrollan proyectos informáticos que han adquirido un gran auge y aceptación en el mercado tanto nacional como internacional. Todo esto no fuera posible sin la existencia de los centros productivos dentro de las universidades del país, donde el Centro Universitario Julio Antonio Echeverría (CUJAE), la Universidad de la Habana (UH) (Lozano Díaz, del Toro Gundin, Arencibia Jorgel, & Martínez Rodríguez, 2008)y la Universidad de las Ciencias Informáticas (UCI) figuran como principales partícipes en dicha tarea. En esta última universidad se encuentra el Centro de Telemática (TLM) adscrito a la Facultad 2, el cual desarrolla sistemas y servicios informáticos integrales para las Telecomunicaciones y la Seguridad Informática, altamente comprometido con la Revolución, capaz de integrar los procesos docentes, productivos e investigativos con alto nivel, contando con un personal especializado en dichas áreas (UCI, 2019a).

Entre los productos que desarrolla el Centro TLM está el sistema Gestor de Recursos de Hardware y Software (XILEMA GRHS), el cual es un sistema que realiza el inventario de la información de los componentes de hardware y software en una red de computadoras. Facilita detectar cambios en dichos componentes, clasificarlos en Autorizados o No autorizados, según se haya establecido en la entidad, así como tomar acciones de control (apagar, hibernar, suspender, reiniciar, enviar notificación por correo) ante los cambios no autorizados. Posee una interfaz web que permite la visualización y administración de la información obtenida de las computadoras. (UCI, 2019b)

El sistema XILEMA GRHS, está compuesto por 4 módulos: Gclient, Gupdater, Gadmin y Gserver cuenta actualmente con una licencia por cantidad de computadoras clientes que pueden estar conectadas al servidor. Esta licencia impide que el cliente conecte más

computadoras a su servidor, por lo que conlleva a que deban solicitar una nueva licencia si así lo necesitan. La licencia por usuarios concurrentes o licencia por cantidad de computadoras conectadas al servidor, bajo la cual se mantiene funcional actualmente el sistema XILEMA GRHS no permite que el software sea comercializado periódicamente. Actualmente no existe una licencia que venza en un tiempo determinado, lo cual implica que el software una vez comercializado con una empresa determinada solo pueda generar ganancias si la misma necesita conectar una nueva computadora al servidor y esta requiere de los servicios de XILEMA GRHS. Por tal motivo se necesita un licenciador por tiempo para XILEMA GRHS, lo que conllevaría a una comercialización más eficiente del sistema.

La situación existente lleva a plantearse el siguiente **problema a resolver:** ¿Cómo limitar el uso del sistema XILEMA GRHS una vez comercializado, teniendo en cuenta el factor tiempo?, para resolver este problema se toma como **objeto de estudio:** el proceso de licenciamiento de software. Para dar solución al problema planteado se propone como **objetivo general:** Implementar en el sistema XILEMA GRHS la capacidad de contar con una licencia que se base en el tiempo, para limitar su uso en las diferentes empresas, delimitando como **campo de acción**: el proceso de licenciamiento por tiempo para XILEMA GRHS.

Para dar cumplimiento al objetivo planteado, se definen las siguientes **tareas de investigación:**

1. Revisar la bibliografía para elaborar el marco teórico conceptual en lo referente al desarrollo de los licenciadores por tiempo.
2. Analizar sistemas homólogos, para conocer aspectos regulares en el diseño de los licenciadores por tiempo aplicados al proceso de la gestión de hardware y software.
3. Analizar herramientas informáticas y metodologías de desarrollo de software asumidas por el proyecto XILEMA GRHS para realizar la implementación del sistema.
4. Identificar las principales funcionalidades del sistema para la posterior implementación del mismo, teniendo en cuenta los requisitos definidos anteriormente.

5. Identificar los diferentes tipos de pruebas de software, para su posterior aplicación sobre el sistema desarrollado.

Para facilitar el cumplimiento del objetivo propuesto y de las tareas de investigación se emplean métodos teóricos y empíricos de la investigación científica.

Métodos Teóricos:

Análisis Histórico – Lógico: Se aplica con el objetivo de conocer la existencia de herramientas encargadas de la gestión de licencias de software para aplicaciones web. Este método permite el análisis de la evolución de sistemas similares, de esta manera se profundiza sobre los rasgos que caracterizan a estos sistemas y en los aspectos principales para fundamentar la propuesta de solución a la problemática planteada.

Analítico – Sintético: Se aplica en la recopilación de información requerida durante la realización del estado del arte referido a las licencias temporales y a los sistemas gestores de inventarios; y para el desarrollo del trabajo mediante la revisión de documentos y artículos, de donde se extrajeron los elementos más significativos relacionados con los licenciadores de software. Además del análisis de las diferentes herramientas, metodologías y tecnologías a utilizar en el desarrollo del licenciador.

Métodos Empíricos:

Observación: Este método puede utilizarse en distintos momentos de la investigación. Permite la recogida de la información de cada uno de los distintos conceptos asociados a los licenciadores de software. Permite también investigar el fenómeno en su manifestación externa, dígase de la interacción con distintos sistemas que utilizan las licencias temporales. De esta forma el documento guía la observación siendo lo suficientemente preciso y claro para garantizar que se aplique en observaciones futuras.

Modelación: Es la reproducción simplificada de la realidad que permite descubrir nuevas relaciones y cualidades del objeto. Este método permite la creación de modelos, (propuestas, alternativas, estrategias, etc.). En esta investigación a través del método empírico se realiza el modelado mediante diagramas. Esto permite reflejar la estructura, las relaciones y características del licenciador a desarrollar. En este caso, con el uso de BPMN, se facilita también el diseño de clases necesario para la implementación del licenciador.

El presente trabajo de diploma se encuentra estructurado en tres capítulos que estarán guiados por los siguientes temas a tratar:

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA: Se realiza un estado del arte sobre las herramientas encargadas del licenciamiento de software, así como las herramientas y lenguajes a usar en la implementación.

CAPÍTULO II. PROPUESTA DE SOLUCIÓN: Abarca el modelo de dominio y requisitos con los que debe cumplir la solución propuesta. También se analizan los artefactos generados durante la etapa de Análisis y Diseño de la solución.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA: Se describe todo lo relacionado a los procesos de implementación de la propuesta de solución y se representan las pruebas realizadas al sistema, así como los resultados.

Capítulo I Fundamentos Teóricos

1.1. Introducción

El presente capítulo está centrado en la realización de un análisis para obtener información actualizada de algunos sistemas de gestión de licencias que han sido implementados nacional e internacionalmente y conocer qué aspectos deben ser tomados en cuenta para que cualquier sistema de este tipo esté al nivel que se exige hoy en día. Se valoran las tendencias actuales en el contexto informático partiendo de que continuamente surgen aplicaciones novedosas con mejoras que se van imponiendo en la industria del software. El capítulo trata conceptos generales asociados al tema de las licencias de software. Así como de un estudio de la tendencia actual del licenciamiento de software, así como las herramientas informáticas y tecnologías usadas en el desarrollo del sistema propuesto. Además de la metodología de desarrollo que se utiliza.

1.2. Conceptos asociados

Licencia: Autorización que se concede para explotar con fines industriales o comerciales una patente, marca o derecho.(Española, 2019)

Si se incluyen los derechos de autor y propiedad, la licencia es un contrato entre el desarrollador del software sometido a propiedad intelectual y a derechos de autor y el usuario, en el cual se definen con precisión los derechos y deberes de ambas partes. Es el desarrollador, o aquel a quien éste haya cedido los derechos de explotación, quien elige la licencia según la cual se distribuye el software.(Samuel Emilio 2019)

Software: El software de computadora es el producto que construyen los programadores profesionales y al que después le dan mantenimiento durante un largo tiempo. Incluye programas que se ejecutan en una computadora de cualquier tamaño y arquitectura, contenido que se presenta a medida que se ejecutan los programas de cómputo e información descriptiva tanto en una copia dura como en formatos virtuales que engloban virtualmente a cualesquiera medios electrónicos.(Roger S. Pressman, 2010)

Licenciante: El licenciante o proveedor-licenciante es aquel que provee el software más la licencia al licenciatario, la cual, le permitirá a este último tener ciertos derechos sobre el software. El rol de licenciante lo puede ejercer cualquiera de los siguientes actores:

Autor: El desarrollador o conjunto de desarrolladores que crea el software son por antonomasia quienes en una primera instancia poseen el rol de licenciante al ser los titulares originales del software.

Licencia de software: Es un contrato entre el licenciante (autor/titular de los derechos de explotación/distribuidor) y el licenciatario del programa informático (usuario consumidor /usuario profesional o empresa), para utilizar el software cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas.

La licencia de software es el conjunto de permisos que un desarrollador da para la distribución, uso y/o modificación de la aplicación que desarrolló. Puede indicar en esta licencia también los plazos de duración, el territorio donde se aplica, etc. En inglés: software license.
La licencia de software es un instrumento legal que gobierna el uso o redistribución del software. Un software tiene protección tanto su código fuente como su forma de código objeto. La única excepción es el software en dominio público.(Alegsa, 2016)

En el momento en que el usuario decide descargar, instalar, copiar o utilizar un determinado software, implica que acepta las condiciones que se estipulan en la licencia que trae ese programa, es la autorización que le permita el uso legal de determinado programa, esta licencia es un documento, ya sea electrónico, en papel original o número de serie autorizado por el autor, como obras intelectuales, no son objeto de venta, sino de cesión o licenciamiento de los derechos de la obra.

Específicamente las licencias de software hacen referencia a la libertad o falta de ella que reciben los usuarios de un software desarrollado por parte de su(s) autor(es) para poder usarlo y/o modificarlo de acuerdo a unas condiciones previamente pactadas.

Titular de los derechos de explotación: Es la persona natural o jurídica que recibe una cesión de los derechos de explotación de forma exclusiva del software desde un tercero, transformándolo en titular derivado y licenciante del software.

Distribuidor: Es la persona jurídica a la cual se le otorga el derecho de distribución y la posibilidad de generar sublicencias del software mediante la firma de un contrato de distribución con el titular de los derechos de explotación.

Licenciatario: El licenciatario o usuario licenciatario es aquella persona física o jurídica que se le permite ejercer el derecho de uso más algún otro derecho de explotación sobre un determinado software cumpliendo las condiciones establecidas por la licencia otorgada por el licenciante.

Usuario consumidor: Persona natural que recibe una licencia de software otorgada por el licenciante, la cual, se encuentra en una posición desventajosa ante los términos y condiciones establecidas en ella.

Usuario profesional o empresa: Persona natural o jurídica que recibe una licencia de software otorgada por el licenciante, la cual, se encuentra en igualdad de condiciones ante el licenciante para ejercer sus derechos y deberes ante los términos y condiciones establecidos en la licencia. (Martín Toral, 2011)

Elementos objetivos de una licencia de software temporal:

Plazo: El plazo determina la duración en el tiempo durante la cual se mantienen vigentes los términos y condiciones establecidos en licencia. Las licencias en base a sus plazos se pueden clasificar en, licencias con plazo específico, indefinido y sin especificación de plazo. (Martín Toral, 2011)

Garantía de titularidad: Es la garantía ofrecida por el licenciante o propietario, en la cual, asegura que cuenta con suficientes derechos de explotación sobre el software como para permitirle proveer una licencia al licenciatario.

1.3. Tecnologías actuales para el licenciamiento de software por tiempo

Además de proteger el software y la propiedad intelectual, se necesita aumentar los ingresos de las ventas de los productos. Para el aseguramiento de que el software sólo está disponible para los usuarios adecuados, de acuerdo con los términos que se definan, este

proceso se controla a través de las licencias. Las licencias proporcionan la flexibilidad necesaria para implementar las estrategias del negocio para la distribución del software. Para obtener el máximo beneficio de la estrategia de licencias en una empresa, es necesario un sistema de licencias de software que le proporcione la flexibilidad para adaptar los términos de licencias, de modo que coincidan con la estrategia de negocio y para adaptarse rápidamente a los cambios del mercado y a las necesidades de la empresa.

Soluciones basadas en hardware. Las soluciones basadas en hardware, proporcionan un dispositivo hardware externo junto con el software. El funcionamiento de su software dependerá de la conexión del dispositivo al ordenador del usuario final. En el tiempo de ejecución, el software se comunica con el dispositivo hardware y sólo funciona correctamente si recibe una respuesta verosímil de este.

Soluciones basadas en software. Con las soluciones basadas en software, basta con seguir la instalación del software del equipo del usuario final para que la protección y licencias se asocien a ese equipo en concreto. El software solo funcionará cuando el usuario haya introducido la llave del producto. En el tiempo de ejecución, el servidor comprueba que el software está en el equipo que tiene licencia para ejecutarlo y que se está utilizando de acuerdo con los términos de licencia del usuario.

1.4. Estado del arte

Una de las primeras etapas que debe desarrollarse dentro de una investigación es el estado del arte ya que permite determinar la forma como ha sido tratado el tema, cómo se encuentra el avance del conocimiento en el momento de realizar una investigación y cuáles son las tendencias existentes para el desarrollo de la temática o problemática que se va a llevar a cabo. El estado del arte sirve al investigador como referencia para asumir una postura crítica frente a lo que se ha hecho y lo que falta por hacer en torno a una temática o problemática concreta, para evitar duplicar esfuerzos o repetir lo que ya se ha dicho, y además para localizar errores que ya fueron superados.

NetSupport DNA:

Es una completa solución que permite realizar inventarios de hardware y de software, así como la gestión de licencias. Permite asignar información de licencias a los datos activos

de una computadora. Tiene una interfaz muy amigable que permite que los operadores puedan asignar rápidamente licencias a varios ordenadores. Además, permite monitorear el uso de licencias dentro de una organización. Al introducir el número de licencias para cada aplicación es posible identificar el uso de ellas cada vez que se ejecute el inventario de software.("NetSupport Manager," 2014)

Open-AudIT:

Terminaremos con Open-AudIT, un **entorno que te indica todo lo que hay conectado en tu red**, otorgándote información de cada dispositivo y programa. Los datos sobre la red se insertan mediante una secuencia de comandos Bash Script (Linux) o VBScript (Windows). Pero también la plataforma te otorga información sobre hardware y software instalados en tu equipo.

Open-AudIT se otorga bajo los términos de la Licencia Pública General de Affero de GNU publicada por la Free Software Foundation. Esto le otorga permiso legal para copiar, distribuir y / o modificar Open-AudIT bajo ciertas condiciones. Lea el archivo “licencia” en la distribución de Open-AudIT o lea la versión en línea de la licencia para obtener más detalles. Open-AudIT se proporciona tal cual, sin garantía de ningún tipo, incluyendo la garantía de diseño, comerciabilidad y adecuación para un propósito particular.(Open-AudIT, 2019)

WinAudit:

Comenzaremos con WinAudit, una **herramienta de inventario de programas** que ofrece varias funciones de reporte de hardware y software del sistema. La plataforma genera una lista de todos los entornos instalados en el sistema, pero también te muestra información necesaria como, nombre del proveedor, versión, ID (software y producto) y ubicación, entre otra tanta información. WinAudit realiza una auditoría rápida y exhaustiva del software instalado, información de licencia, configuración de seguridad, inventario de hardware, configuración de red, etc. El informe se puede guardar en varios formatos (texto, PDF, HTML, etc.), enviarse por correo electrónico o exportar a una base de datos a través de ODBC. WinAudit es gratuito, de código abierto y puede ser utilizado o distribuido por cualquier persona. La Licencia Pública de la Unión Europea es una licencia de software

libre y copyleft creada por la Unión Europea para una previsible liberación de programas pertenecientes a las administraciones públicas(EUPL).(WinAuditv3.2.1, 2017)

Network Inventory Advisor:

Network Inventory Advisor es un software profesional y muy pulido de gestión de hardware y software que lleva la usabilidad del software al siguiente nivel. Creemos que prácticamente cualquier persona puede instalar Network Inventory Advisor y, en pocos minutos, crear un inventario completo de todos los activos de hardware y software en la red de la empresa. Esto se debe a que Network Inventory Advisor está diseñado para escanear automáticamente cientos de computadoras en cuestión de minutos, agrupando sus redes internas y las de sus clientes. Lo mejor de todo es que Network Inventory Advisor puede ser probado por un período de hasta 15 días gratis. Tipo de licencia shareware**.** (NetworkInventoryAdvisor, 2019)

Spiceworks Inventory:

Spiceworks Inventory es un sistema de gestión de activos muy sencillo, compatible con Windows, Mac y Linux, dispositivos UPS, almacenamiento y más. Proporciona las características del descubrimiento automático de hardware, el seguimiento y viene con unas sofisticadas capacidades de generación de informes que le permiten crear informes personalizados que se adaptan perfectamente a sus necesidades. La funcionalidad de Spiceworks Inventory puede ampliarse con varios complementos, pero la solución de software de inventario de hardware y software puede hacer prácticamente cualquier cosa que desee. Spiceworks Inventory funciona mejor cuando se utiliza junto con otros productos de Spiceworks, que están destinados principalmente a las empresas grandes. Tipo de licencia: software libre.(SpiceworksInventory, 2019)

Conclusiones del estado del arte

Debido a que algunos de estos sistemas son propietarios y por ende sus costos para uso son elevados, financiar los gastos de la implantación de un sistema de los existentes en el mundo, para la gestión del otorgamiento de licencias de software sería muy difícil para países subdesarrollados como Cuba. Lo antes descrito ligado a que estos sistemas no cubren en su totalidad con las necesidades imperantes en el centro de desarrollo (TLM),

para satisfacer de una mejor forma las necesidades del cliente y así brindarle un mejor servicio, puesto que algunas monitorean el uso de licencias, pero no logran asignarlas, se limitan además a chequear la cantidad de instalaciones de cada sistema informático, con el objetivo de verificar el cumplimiento de las licencias, así como presentan limitaciones en cuanto a la durabilidad del licenciamiento y no gestionan servicios asociados a instalación, soporte, actualización, configuración etc. Además, algunos de estos sistemas funcionan sólo para productos de empresas específicas, por lo que el Centro TLM no tendría acceso a las mismas. Es por ello que se decide desarrollar un licenciador por tiempo para XILEMA GRHS, aunque con el estudio de las herramientas se tomarán en cuenta particularidades de algunas de ellas que se puedan incorporar a la solución propuesta.

1.5 Metodologías de desarrollo de software

La selección de una metodología para el desarrollo de un determinado sistema está condicionada por varios factores que definen la más apropiada, entre ellos se encuentran, el tipo de sistema a desarrollar, el personal participante, los recursos disponibles y las necesidades del cliente, por lo que no existe una metodología estándar a seguir, más bien se necesita que este proceso sea adaptable y configurable de acuerdo al proyecto. Existen dos tipos fundamentales de metodologías para tratar este tema, las que siguen los métodos tradicionales, que son generalmente para software de gran envergadura, y las que proponen procesos ágiles para el desarrollo, más usadas en pequeños softwares y con marcadas diferencias entre sí. Las diferencias se basan fundamentalmente en las siguientes características: tamaño del equipo de desarrollo, tiempo del que se dispone para la implementación del sistema, la cantidad de documentación que se genera, la envergadura del software y las necesidades del cliente.(Roger S. Pressman, 2010)

A pesar de la variedad de metodologías usadas, se ha comprobado que muy pocos proyectos dentro de la Universidad de las Ciencias Informáticas (UCI) la aplican en su totalidad. Es por esto que surge en la misma lo que se llama una variación al Proceso Unificado Ágil (AUP) de Scott Ambler la cual usa técnicas ágiles. Esta variación de AUP que se utiliza en la universidad cuenta con cuatro fases que transcurren de forma consecutiva.(Díaz Romeu, 2016)

1. **Inicio:** El objetivo de esta fase es obtener una comprensión común cliente-equipo de
   desarrollo del alcance del nuevo sistema.

2. **Elaboración:** El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.

3. **Construcción:** Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.

4. **Transición:** El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación.

A partir de que el modelado de negocio propone tres variantes a utilizar en los proyectos (CUN, DPN o MC) y existen tres formas de encapsular los requisitos (CUS, HU, DRP), surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma:

- **Escenario No 1:** Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.

- **Escenario No 2:** Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

- **Escenario No 3:** Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

- **Escenario No 4:** Proyectos que no modelan negocio solo pueden modelar el sistema con HU.

La metodología AUP en su variación UCI se adapta a la configuración y ciclo de vida de la actividad productiva de la UCI. El desarrollo de la documentación es totalmente entendible logrando con su ejecución un producto de software con la calidad requerida. Consta de los principales aspectos positivos de la mayoría de las metodologías ágiles adaptándose correctamente a equipos de desarrollo pequeños; es por esto que se decide utilizar la metodología AUP en su variación UCI. Se trabajará en el Escenario 2, ya que luego de

evaluar el negocio se llegó a la conclusión de que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades. Además, el objetivo general de la investigación está en correspondencia con lo propuesto en este escenario, ya que la gestión y presentación de la información son importantes.

1.6. Tecnologías y herramientas de desarrollo

1.6.1. Marcos de trabajo

Django 1.8

Marco de trabajo web implementado sobre el lenguaje de programación Python, perteneciente a la licencia BSD (licencia de software libre permisiva). Django brinda estructura al código fuente, fomenta 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 marco de trabajo, 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.("Django 1.8 release notes," 2019)Utilizado en el desarrollo del producto final partiendo en un principio de que el Centro TLM define Python como lenguaje de programación y Django como marco de trabajo.

XILEMA-Base-Web 1.0

Es un marco de trabajo desarrollado en el Centro de Telemática (TLM) que está constituido por Django como marco de trabajo base, librerías de JavaScript como son JQuery y Backbone (TLM 2016). Cumple con las pautas de diseño de la universidad y la estrategia marcaria a seguir por el Centro de Telemática y la Universidad de las Ciencias Informáticas.("Xilema-Base-Web," 2016) Por este motivo y con lo anteriormente analizado se decide utilizar como marco de trabajo en la implementación de la solución propuesta.

1.6.2. Lenguaje de programación

Python 2.7.9

Python es un lenguaje de programación multipropósito de alto nivel, simple y sencillo de aprender además es libre. Se propone su utilización pues brinda muchas funcionalidades para aplicaciones web que trabajen a un nivel bien cercano al sistema operativo, además posee mecanismos eficientes para su integración con bases de datos.("Python," 2018). Se decidió utilizar Python porque XILEMA GRHS está implementado sobre este lenguaje, además el licenciador a implementar se va a integrar a este sistema y debe ser desarrollado bajo sus mismas tecnologías. En el Centro de TLM definió este lenguaje de programación para sus productos, lo que beneficia que cualquier persona pueda darle mantenimiento al sistema o programar nuevas funcionalidades cuando sea necesario.

HTML 5

HTML, más que el lenguaje de marcado predominante usado para estructurar el contenido web, es realmente una tecnología web que interrelaciona varios estándares. En esta última revisión del lenguaje se incluyen varias características verdaderamente útiles, como son nuevos elementos semánticos y mejoras en los formularios(MacDonald, 2015).Por todas las mejoras con las que cuenta y teniendo en cuenta que la propuesta de solución formará parte del sistema XILEMA GRHS desarrollado con este lenguaje se decidió utilizar el mismo para su desarrollo.

JavaScript

JavaScript es un potente lenguaje de scripting basado en objetos. El código JavaScript puede ser embebido directamente en un documento HTML e interpretado por un navegador web en el lado del cliente (su uso más extendido). Cuando se combina con el Document Object Model (DOM) definido por un navegador web, JavaScript permite crear o manipular dinámicamente el contenido HTML(Flanagan, 2011).

UML (Lenguaje de Modelado Unificado)

Lenguaje de Modelado Unificado (Unified Modeling Language) es la sucesión de una serie de métodos de análisis y diseño orientados a objetos. Fue creado para forjar un lenguaje de modelado visual común y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento.("¿Qué es UML?," 2008)

1.6.3. Herramientas de desarrollo

Pycharm 2018.2.2

Es un Entorno de Desarrollo Integrado (IDE) desarrollado por la compañía Jetbrain. Es multiplataforma, hay binarios para: Windows, Linux y Mac OS X. Proporciona análisis de código, depuración gráfica y soporte para el desarrollo web con Django, entre otras bondades (islam 2015). Tiene un excelente soporte para Python, además de facilitar los procesos de búsquedas de errores y ejecución de tareas a través de consola(JetBrains, 2018).

PgAdmin III

Es un completo sistema de gestión y diseño de base de datos de la plataforma PostgreSQL para sistemas Unix 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. Está disponible gratuitamente bajo los términos de la licencia PostgreSQL y puede redistribuirse siempre que se cumplan los términos de la licencia .("PgAdmin," 2018)

1.6.4. Herramienta de modelado

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. Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.("Visual Paradigm," 2014)

1.6.5. Sistema gestor de base de datos

PostgreSQL 9.4

Es un sistema de gestión de bases de datos objeto-relacional. Se encuentra distribuido bajo licencia BSD y tiene su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar 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)

1.7. Conclusiones del capítulo

Con el desarrollo del marco teórico se logró organizar y guiar el trabajo, profundizando en el estudio de algunos conceptos fundamentales para el desarrollo de un sistema que permita realizar un licenciador por tiempo para GRHS. Se realizó un análisis de sistemas similares en el ámbito internacional como nacional, donde se llega a la conclusión de que ninguno cubría las necesidades que requiere el problema planteado en esta investigación. Se proponen el lenguaje, las tecnologías, herramientas y metodología de desarrollo de software que se utilizarán en la elaboración del sistema informático.

Capítulo II: Análisis y Diseño

2.1 Introducción

En este capítulo se abordan los aspectos relacionados con el dominio de la investigación y la propuesta del trabajo del licenciador por tiempo para el sistema XILEMA GRHS. Se crea el diagrama de modelo conceptual. Se definen los requerimientos funcionales y los requisitos no funcionales para el correcto funcionamiento del sistema. Se diseñan los diagramas de casos de uso del sistema y los prototipos de interfaz. Se realiza el modelo de BD y se describen cada una de las tablas del modelo.

2.2 Modelo Conceptual

El modelo de dominio o modelo conceptual es una representación que se realiza para entender el proyecto donde se está trabajando, cuando no se tiene una visión clara de los procesos que se realizan en el mismo. Se utiliza para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. Contiene conceptos que estarán asociados tanto a su definición natural como al papel que juegan desde el punto de vista informático. ("EcuRed," 2014)

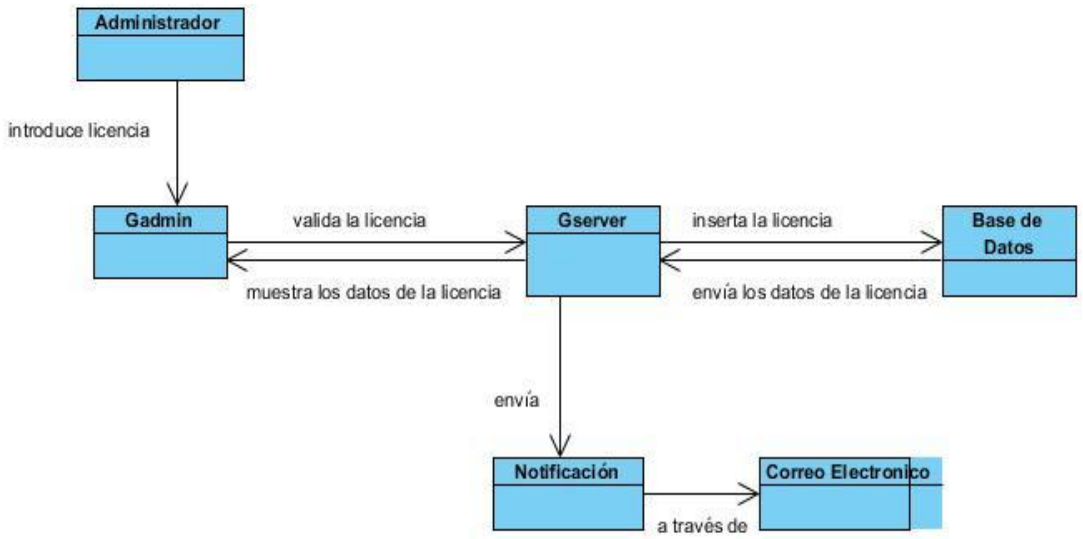


Figura 1 Modelo Conceptual

2.2.1 Descripción del Modelo Conceptual

El sistema XILEMA GRHS cuenta actualmente con una licencia por cantidad de computadoras conectadas a una red, actualmente se introduce la licencia a través del subsistema web denominado Gadmin (Interfaz visual del servidor web), y este a su vez ingresa los datos en Gserver (servidor web) para de esta forma validar la licencia. De esta forma el licenciador por tiempo para el sistema GRHS seguirá el mismo flujo adicionando la opción de generar no solo la licencia por cantidad sino también la licencia por tiempo y además notificará al usuario a través del correo, del tiempo restante de validez de la licencia por tiempo.

2.3 Propuesta de solución

Figura 2 Propuesta de solución

La propuesta de solución consiste en un licenciador por tiempo para XILEMA GRHS ya que la licencia con que cuenta actualmente no satisface todos los requerimientos de la empresa en cuento a la monetización del software dependiendo del factor tiempo. Lo que se pretende que realice el sistema es, siguiendo en esencia el mismo flujo con el que trabaja actualmente XILEMA GRHS, permitiendo de esta forma administrar la licencia temporal de la misma forma que hoy se administra la licencia por usuarios concurrentes. Siguiendo este principio el sistema debe ser capaz de generar la licencia temporal partiendo de la

concatenación y cifrado del (BIOS+RAM+BOAR+ (fecha (año +mes+ número de la semana +día del mes +día de la semana +hora +minutos +segundos)) en una cadena denominada Serial. Esto cifrado además junto a las Siglas de la Entidad y las fechas de inicio y fin del periodo por el cual va a ser válida la licencia, conformarían la licencia generada. Permitiendo que esta sea insertada en el sistema luego de una verificación donde se genera nuevamente el Serial y se compara la hora generada con la hora del servidor.

2.4 Diagramas de secuencia

Los diagramas que se muestran a continuación representan las clases del diseño con las que se tuvo interacción durante la presente investigación, los mismos fueron tomados del expediente de proyecto para el sistema XILEMA GRHS v2.0.

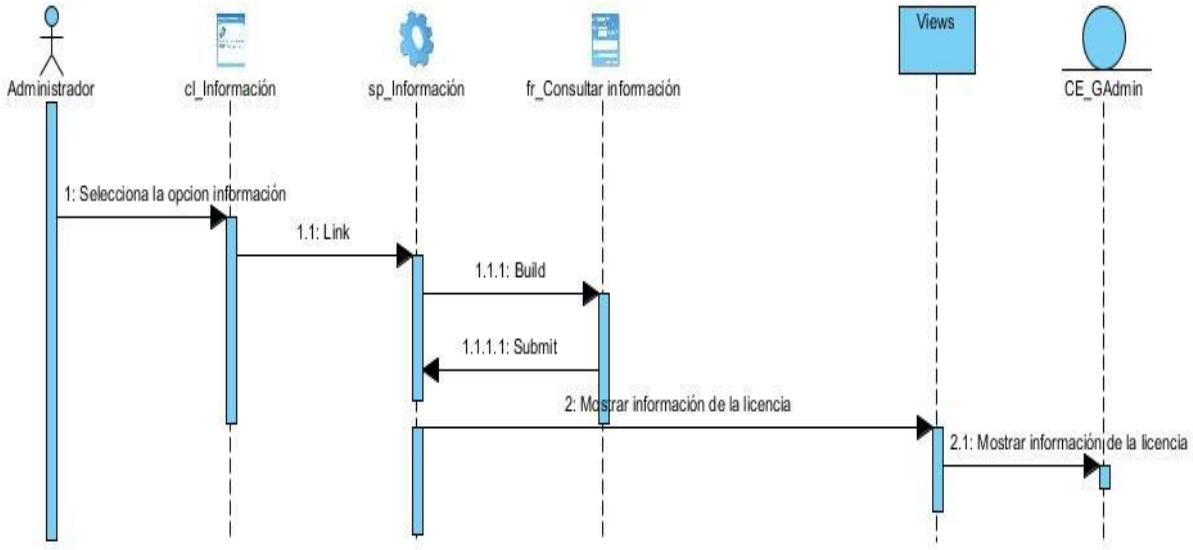


Figura 3 Diagrama de secuencia del RF Consultar información de la licencia registrada

.

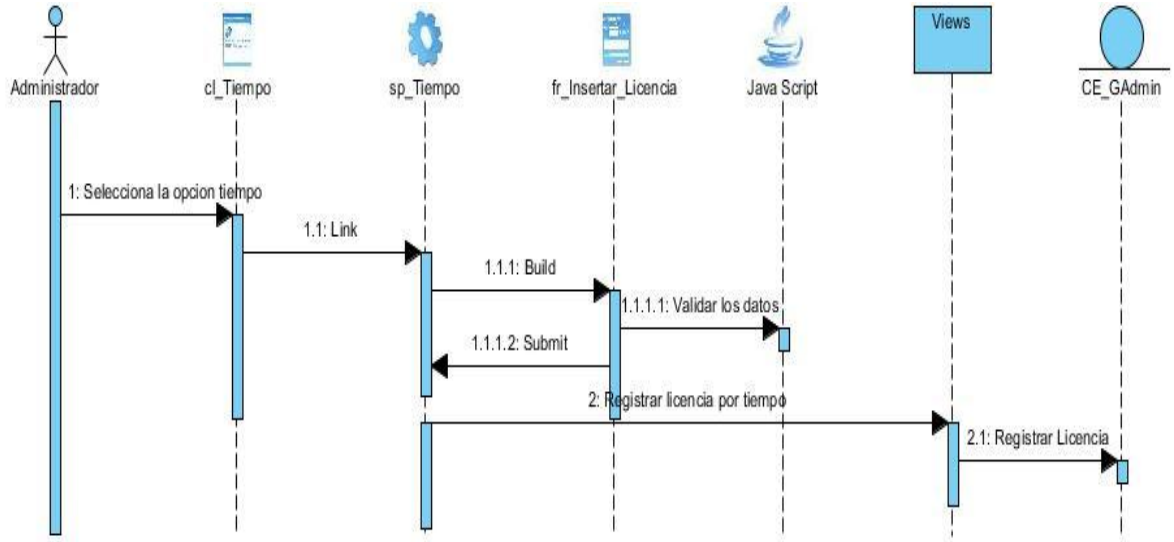


Figura 4Diagrama de secuencia del RF Registrar licencia por tiempo

2.5 Diagrama de clases del diseño

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones. Los diagramas de clase son el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido. (Craig, 1999).

Diagrama de clases del diseño RF Registrar licencia por tiempo:

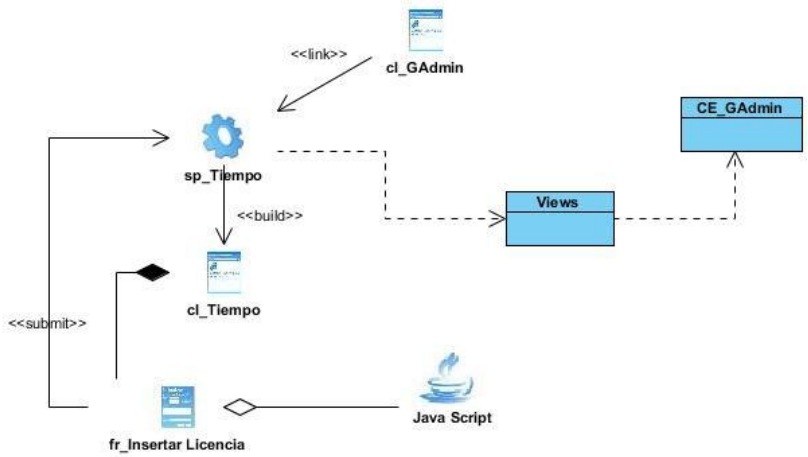


Figura 5 Diagrama de clase del diseño RF Registrar licencia por tiempo

Diagrama de clases del diseño RF Consultar información de la licencia registrada:

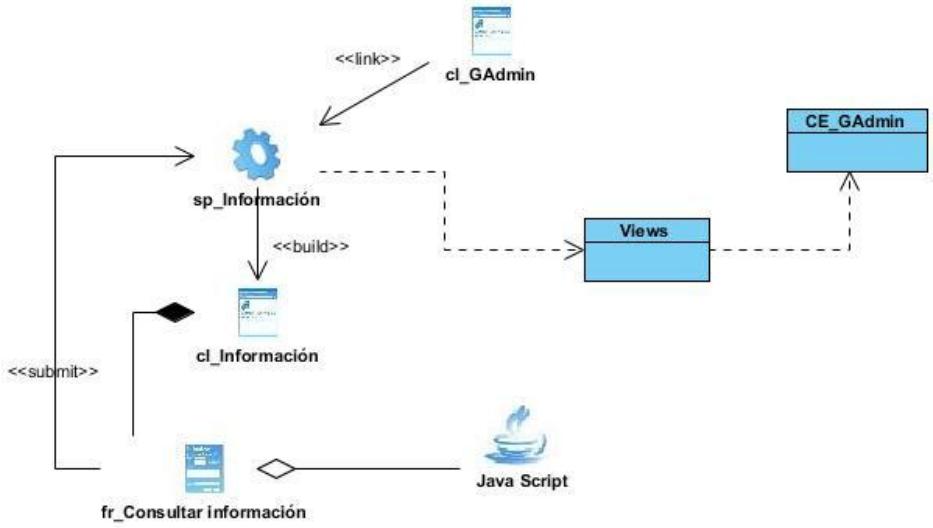


Figura 6 Diagrama de clases del diseño RF Consultar la información de la licencia registrada

2.4 Especificación de requisitos de software

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

2.4.1 Requisitos funcionales

Los requisitos funcionales de un sistema son los que describen las capacidades o funciones que el componente debe cumplir y los servicios que de él se esperan (JavaScript,2018), a continuación, las funcionalidades definidas por el cliente y la descripción del requisito generar licencia el resto de las descripciones se encuentran en Anexo.

- **RF1:** Generar licencia por tiempo.
- **RF2:** Notificar vía correo el vencimiento de la licencia por tiempo.
- **RF3:** Consultar información de la licencia registrada.
- **RF4:** Registrar licencia por tiempo.

Diagrama de casos de uso del sistema:

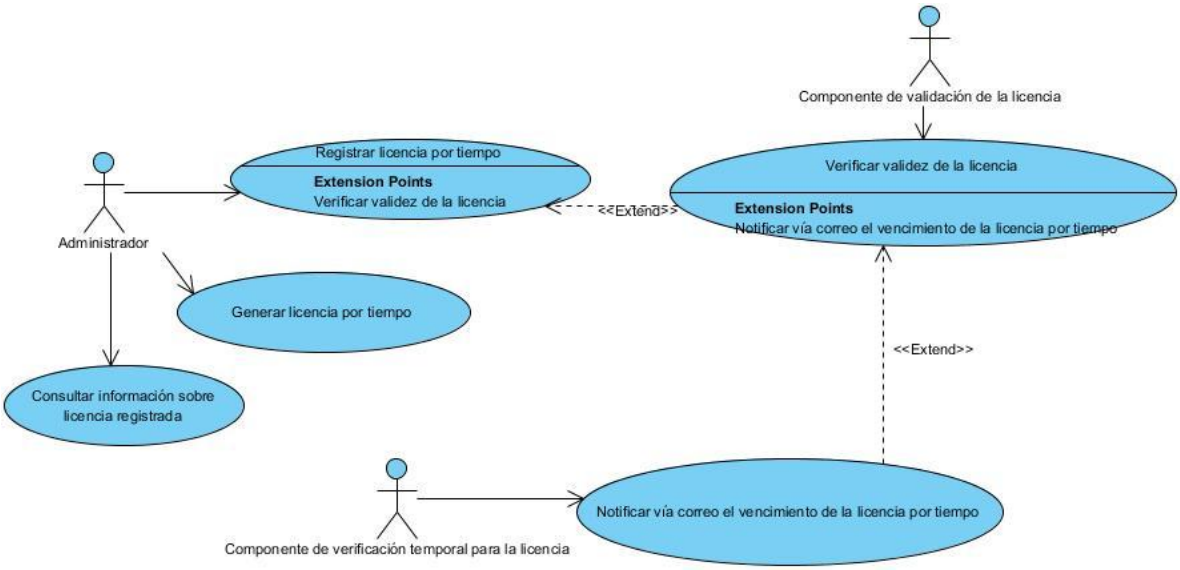


Figura 7 Diagrama de Casos de Uso del Sistema

Descripción de los actores:

Actor	Objetivos
Administrador	Administra y gestiona todas las funcionalidades del sistema (XILEMA GRHS)
Componente de validación de la licencia	Componente del sistema (XILEMA GRHS) encargado de verificar si la licencia cumple las condiciones para considerarse válida
Componente de verificación temporal para la licencia	Componente del sistema (XILEMA GRHS) encargado de verificar el periodo de tiempo por el cual la licencia es válida en el sistema y enviar notificaciones de vencimiento

Especificación de casos de uso (CU):

CU 1. Generar licencia por tiempo

Actores	Administrador	
Prioridad	**Alta**	
Precondiciones	El usuario ha sido autenticado en el sistema XILEMA GRHS 2.0 y posee permisos de administrador del mismo.	
Pos-condiciones	La licencia se generó correctamente	
Flujo de eventos		
Flujo básico Generar licencia		
	Actor	**Sistema**
1.	Selecciona el componente asociado a las fechas inicio y fecha fin	Activa los campos fecha inicio y fecha fin permitiendo su modificación
2.	Introduce los datos	Valida que los datos sean correctos
3.	Selecciona el botón generar	Genera la licencia por tiempo
3	Selecciona el botón exportar	Exporta la licencia a un archivo .txt
Flujos alternos		
1a: Los datos son incorrectos o existen campos vacíos		
	Actor	**Sistema**
1	Introduce datos incorrectos o no llena todos los campos	El sistema muestra un mensaje. “El nombre de la entidad o las fechas no son correctas”.
Flujos alternos		
1b: Error en los datos insertados		
	Actor	**Sistema**

1.	No desea generar una licencia con los datos introducidos en el sistema o estos son incorrectos por lo que selecciona el botón Limpiar	Limpia todos los campos
Requisitos no funcionales	Usabilidad, Eficiencia	
Asuntos pendientes	**NA**	
Prototipo elemental de interfaz gráfica de usuario		

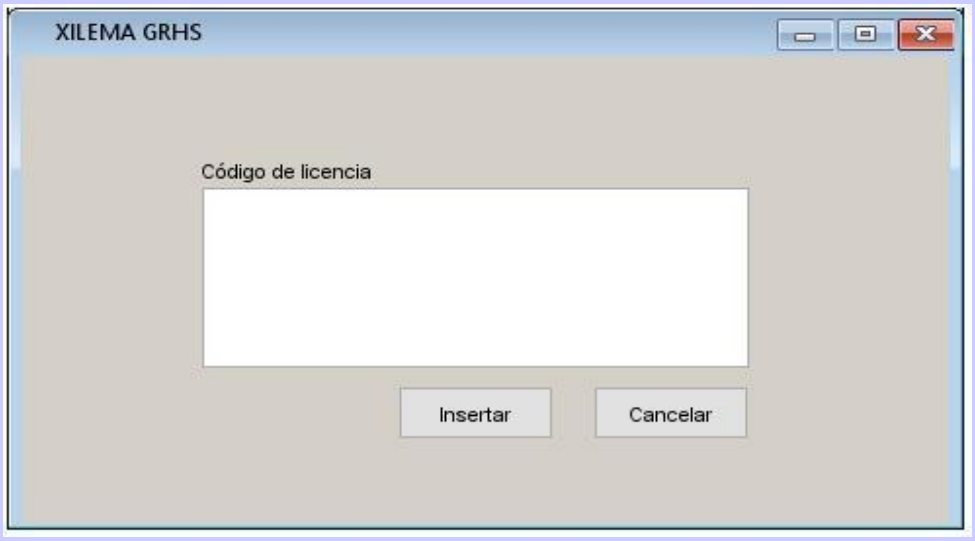
CU 2 Registrar licencia por tiempo

Actores	Administrador	
Prioridad	**Alta**	
Precondiciones	El usuario ha sido autenticado en el sistema XILEMA GRHS 2.0 y posee permisos de administrador del mismo.	
Pos-condiciones	La licencia se registró correctamente	
Flujo de eventos		
Flujo básico Registrar licencia por tiempo		
	Actor	**Sistema**
1	Selecciona el módulo Licencia en el sistema	Despliega un panel que muestra los módulos Tiempo, Cantidad, Información
2	Selecciona el módulo Tiempo	Muestra un cuadro de texto en el que se insertara la licencia
3	Si el usuario no desea insertar la licencia registrada o existe algún error en la misma	Limpia el cuadro de texto en el que se inserta el código de la licencia

	selecciona el botón cancelar	
4	Inserta el código de la licencia generada y selecciona el botón insertar	Inserta la licencia y muestra automáticamente la información de la misma
Flujos alternos		
1a: Los datos son incorrectos o existen campos vacíos al presionar el botón Insertar		
	Actor	**Sistema**
2	Introduce datos incorrectos o el cuadro de texto está vacío	El sistema muestra un mensaje. "Licencia inválida".

Relaciones	**CU extendidos**	Verificar validez de la licencia en el CU Registrar licencia por tiempo
Requisitos no funcionales	Usabilidad, Eficiencia	
Asuntos pendientes	**CU Verificar validez de la licencia**	

Prototipo elemental de interfaz gráfica de usuario



CU 3 Verificar validez de la licencia

Actores	Componente de validación de la licencia
Prioridad	**Alta**
Precondiciones	El usuario ha insertado una licencia en el sistema XILEMA GRHS 2.0
Pos-condiciones	La licencia insertada es válida
Flujo de eventos	
Flujo básico Verificar validez de la licencia	

	Actor	**Sistema**

1	Descifra el código insertado, separando el serial, las fechas de inicio y fin, y las siglas de la entidad	Genera nuevamente el serial insertado en el generador de licencias
2	Compara el serial insertado en la licencia con el que se genera en el servidor y determina que son iguales	Muestra la información de la licencia válida insertada
Flujos alternos		
1a: Los seriales no son iguales		
	Actor	**Sistema**
3	Compara el serial insertado en la licencia con el que se genera en el servidor y determina que no son iguales	El sistema muestra un mensaje. “Licencia inválida”.

Relaciones	**CU extendidos**	Notificar vía correo el vencimiento de la licencia en el CU Verificar validez de la licencia
Requisitos no funcionales	Usabilidad, Eficiencia	
Asuntos pendientes	Notificar vía correo el vencimiento de la licencia	

CU 4 Notificar vía correo el vencimiento de la licencia

Actores	Componente de verificación temporal para la licencia	
Prioridad	**Alta**	
Precondiciones	El sistema XILEMA GRHS 2.0 ha determinado que la licencia es válida	
Pos-condiciones	Se envía correctamente la notificación	
Flujo de eventos		
Flujo básico Notificar vía correo el vencimiento de la licencia		
	Actor	**Sistema**
1	Realiza una resta entre la fecha de vencimiento de la licencia y la fecha en la que se inicializa	Determina si el resultado de la resta coincide con los siguientes valores (15,10,5,3,1) y si es así envía una notificación vía correo electrónico de la proximidad de vencimiento de la licencia
2	Verifica si el resultado de la resta es 0	El sistema XILEMA GRHS se detiene e impide su funcionamiento
Flujos alternos		
1a: Los seriales no son iguales		
	Actor	**Sistema**

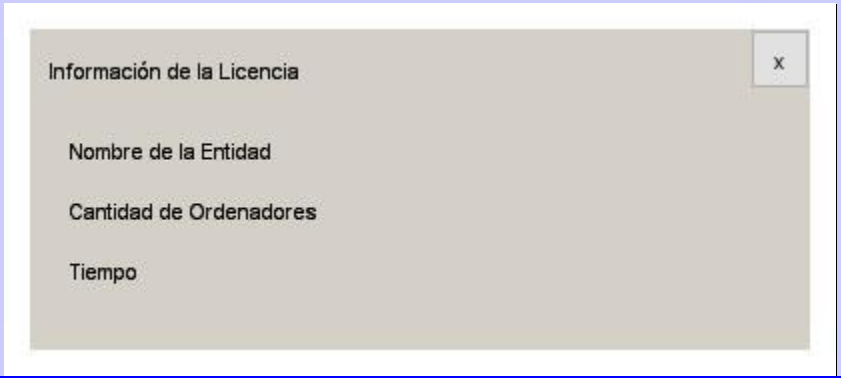
4	Compara el serial insertado en la licencia con el que se genera en el servidor y determina que no son iguales	El sistema muestra un mensaje. “Licencia inválida”.

Relaciones	**CU extendidos**	Notificar vía correo el vencimiento de la licencia en el CU Verificar validez de la licencia
Requisitos no funcionales	Usabilidad, Eficiencia	
Asuntos pendientes	Notificar vía correo el vencimiento de la licencia	

CU 5. Consultar información sobre licencia registrada

Actores	Administrador	
Prioridad	**Baja**	
Precondiciones	La licencia ha sido insertada satisfactoriamente	
Pos-condiciones	Se muestra la información de la licencia insertada	
Flujo de eventos		
Flujo básico Consultar información sobre licencia registrada		
	Actor	**Sistema**
1	Selecciona el moduló Licencia	Despliega un panel con los botones Cantidad, Tiempo, Información
2	Selecciona el botón Información	Muestra un cartel donde se aprecia la información de la licencia registrada (Nombre de la Entidad y Tiempo)
Requisitos no funcionales	Usabilidad, Eficiencia	
Asuntos pendientes	**NA**	

Prototipo elemental de interfaz gráfica de usuario



2.4.2 Requisitos no funcionales

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

Relación de los requisitos no funcionales definidos:

- Usabilidad
- Hardware
- Portabilidad
- Confiabilidad

Funcionalidad, Seguridad, Interoperabilidad, Confiabilidad, Usabilidad, Eficiencia, Mantenibilidad, Portabilidad

Tabla 1 Requisito No Funcional de Confiabilidad

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

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

Tabla 2 Requisito No Funcional de Confiabilidad

Atributo de Calidad	Confiabilidad
Sub-atributos/Sub-características	Cumplimento de fiabilidad
Objetivo	Prevenir el acceso no autorizado, a los datos.
Origen	El Usuario
Artefacto	Canal de Comunicación, Sistema
Entorno	El usuario desea obtener información sobre los datos procesados por el sistema.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Persona no autorizada intenta acceder a los datos del sistema.	

	Bloquear el acceso al sistema
Medida de respuesta	
NA	
2.a Persona con permisos restringidos intenta acceder a información a la que no tiene acceso.	
	Bloquear el acceso a estos datos.
Medida de respuesta	
NA	

Tabla 3 Requisito no funcional de Usabilidad

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)**
NA	NA
Medida de respuesta	
NA	

Tabla 4 Requisito no funcional de Usabilidad

Atributo de Calidad	Usabilidad
Sub-atributos/Sub-características	Operabilidad
Objetivo	Lograr que el sistema sea capaz de realizar todas las operaciones solicitadas por el cliente.

Origen	El usuario
Artefacto	El sistema
Entorno	El sistema está funcionando correctamente.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
NA	NA
Medida de respuesta	
NA	

Tabla 6 Requisito no funcional de Usabilidad

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)**
1.a El usuario realiza una petición al sistema	
	El sistema responde correctamente la petición.
Medida de respuesta	
NA	

Tabla 7 Requisito no funcional de Hardware

Atributo de Calidad	Hardware

Sub-atributos/Sub-características	Utilización de recursos
Objetivo	CPU (4 x 2.33 GHz (Intel Xeon 5140 Core2 1.10 GHz)) RAM (2GB) HDD (30 Gb RAID 5) LAN (1 x NIC (1 Gbit))
Origen	Externo al sistema
Artefacto	Servidor de aplicación
Entorno	Operación normal
Estímulo	Respuesta: Flujo de eventos (Escenarios)
NA	NA

Tabla 8 Requisito no funcional de Hardware

Atributo de Calidad	Hardware
Sub-atributos/Sub-características	Utilización de recursos
Objetivo	CPU (4 x 2.33 GHz (Intel Xeon 5140 Core2 1.10 GHz)) RAM (2GB) HDD (50 Gb RAID 5) LAN (2 x NIC (1 Gbit))
Origen	Externo al sistema
Artefacto	Servidor de almacenamiento de información
Entorno	Operación normal
Estímulo	Respuesta: Flujo de eventos (Escenarios)
NA	NA

Tabla 9 Requisito no funcional de Portabilidad

Atributo de Calidad	Portabilidad
Sub-atributos/Sub-características	Adaptabilidad

Objetivo	Lograr que el sistema sea adaptable a diferentes entornos especificados.
Origen	El usuario
Artefacto	El sistema
Entorno	Funcionando correctamente
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Se prueba el sistema en diferentes entornos.	
	El sistema funciona correctamente.
Medida de respuesta	
NA	

Tabla 10 Requisito no funcional de Portabilidad

Atributo de Calidad	Portabilidad
Sub-atributos/Sub-características	Instabilidad
Objetivo	Lograr que el sistema se pueda instalar en un entorno especificado.
Origen	El usuario
Artefacto	El sistema
Entorno	Funcionando correctamente
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Se instala el sistema en diferentes entornos.	
	El sistema funciona correctamente.
Medida de respuesta	
NA	

2.5 Diagrama de Despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los

elementos y artefactos del software se trazan en esos nodos. Un nodo es un elemento de hardware o software.

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

Figura 8 Diagrama de Despliegue

2.6 Conclusiones del Capítulo

Los artefactos obtenidos durante esta etapa del desarrollo de la solución propuesta dan cumplimiento a los requisitos funcionales. De esta forma se describe lo que el sistema debe hacer en cada momento y los requisitos no funcionales los que definieron las restricciones del mismo. Mediante la representación del modelo de dominio, la autora identificó las entidades relacionados con el sistema y las relaciones entre ellas. El diagrama de casos de uso del sistema posibilitó la agrupación de los distintos requisitos funcionales perfeccionando de esta forma la concepción del sistema. El diagrama de despliegue logró que se obtuvieran de forma concreta y detallada las relaciones existentes entre el hardware y el software. Todos estos artefactos propuestos por la metodología AUP-UCI, posibilitó estructurar todo el proceso de desarrollo para las siguientes actividades de implementación y prueba.

Capítulo III Implementación y pruebas

3.1. Introducción

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

3.2. Arquitectura de software

La arquitectura de software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores. Es definida según la IEEE[1] Estándar 1471-2000 como: la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos, el contexto en el que se implantarán, y los principios que orientan su diseño y evolución.

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

[1] Instituto de Ingeniería Eléctrica y Electrónica

3.3. Patrones de arquitectura

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

Un patrón describe un problema que ocurre una y otra vez en el entorno y describe también el núcleo de la solución al problema, de forma que puede reutilizarse continuamente.

Los patrones de arquitectura expresan los esquemas de organización estructural fundamental para sistemas software. Además, proveen de un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y guías para la organización de las relaciones entre ellos.

3.3.1 Arquitectura cliente servidor

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

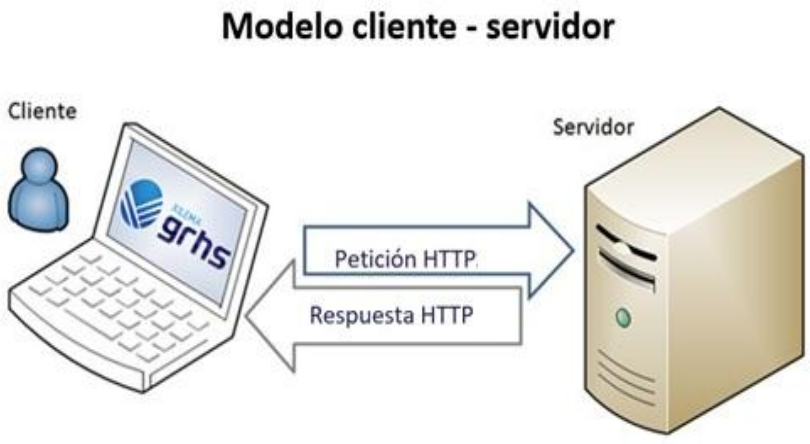


Figura 9 Arquitectura Cliente Servidor

Para el desarrollo de la propuesta de solución para el sistema XILEMA GRHS se contará con un servidor desarrollado con las tecnologías de Python 2.7 y Django 1.8, el cual brindará los datos necesarios para la incorporación al sistema de una licencia temporal. Dichos cambios se podrán observar a través de la interfaz gráfica con que cuenta XILEMA GRHS (Gadmin) permitiendo así la interacción con los clientes.

3.3.2 Patrón arquitectónico Modelo Plantilla Vista

Django se basa en el Modelo Plantilla Vista (MPV), que es una modificación del Modelo Vista Controlador (MVC). El controlador pasa a ser la vista y la vista pasa a denominarse plantilla. En Django, una vista describe los datos que se ofrecen al usuario, pero no necesariamente su aspecto. Una vista habitualmente delega los datos a una plantilla que describe la forma de presentarlos. El modelo es el encargado de las consultas a la base de datos. (JavaScript,2018)

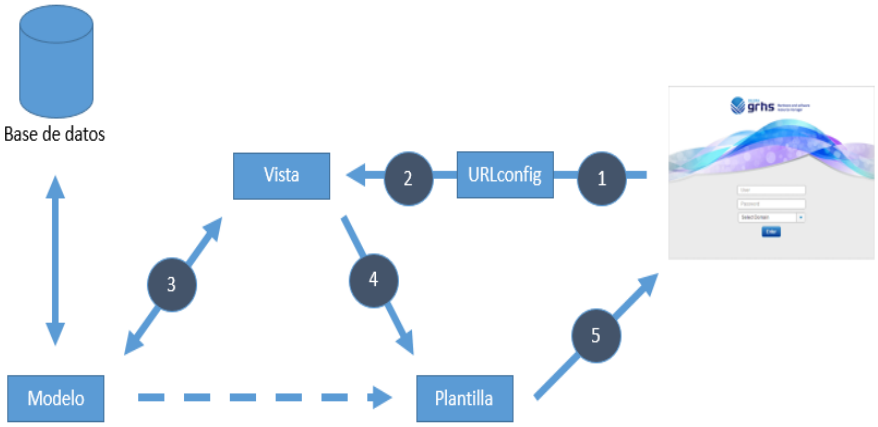


Figura 10 Patrón arquitectónico MPV

Explicación del modelo:

1. El navegador envía una solicitud.
2. El URLConf interpreta la solicitud y ubica la vista apropiada.
3. La vista interactúa con el modelo para obtener los datos.
4. La vista selecciona la plantilla apropiada.
5. La plantilla renderiza la respuesta a la solicitud del navegador.

3.4. Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Representan una descripción de las clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. El uso de patrones posibilita estandarizar el modo en que se realiza el diseño y proporciona reusabilidad, extensibilidad y mantenimiento del código.("JavaScript a fondo,")

3.4.1 Patrones General Responsibility Assignment Software Patterns (GRASP)

Los patrones GRASP, describen la asignación de responsabilidades a objetos. Para el desarrollo del componente fueron utilizados los siguientes patrones de esta clasificación que en su mayoría son implementados de forma nativa por el marco de trabajo Django:(Roger S. Pressman)

- **Experto:** Es un patrón encargado de asignar responsabilidades; es un principio básico que utiliza un diseño orientado a objetos. Clase generador_licenci.py.
- **Alta cohesión**: propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables asumiendo solamente las responsabilidades que deben manejar, evadiendo un trabajo excesivo. Este patrón se pone de manifiesto durante todo el desarrollo del componente, ya que cada clase se encarga solamente de hacer lo que le corresponde, evitando la dependencia entre las clases. Clase generador_licenci.py.
- **Bajo Acoplamiento**: debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas no se puede extraer software de forma independiente y reutilizable. Este patrón está presente ya que cada clase tiene dependencia de hacer lo que le corresponde. Clase cifrado.py.
- **Creador:** el patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos (una tarea muy común). La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. La ventaja de utilizar este patrón es eliminar la dependencia entre las clases logrando una mayor mantenibilidad y reutilización.(Roger S. Pressman). Clase generador.py.

3.5. Pruebas del software

La prueba de software es un elemento crítico para la garantía del correcto funcionamiento del software. Entre sus objetivos están: detectar defectos en el software, verificar la integración adecuada de los componentes y verificar que todos los requisitos se han implementado correctamente. Además de identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente y diseñar casos de prueba que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.(Roger S. Pressman)

3.5.1 Pruebas de caja negra

Las pruebas de caja negra también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa(Roger S. Pressman). Esta prueba intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructura de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y terminación.

3.5.2 Estrategia de prueba

Trazar una estrategia de prueba es primordial para realizar la ejecución de las pruebas de un software. Mediante esta quedan plasmado los niveles de prueba a tratar, los tipos de pruebas que se deben llevar a cabo por cada nivel, los métodos de pruebas a aplicar y las técnicas a utilizar por el método.

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requieran. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados. Una estrategia de prueba de software debe ser suficientemente flexible para promover un uso personalizado de la prueba. Al mismo tiempo, debe ser suficientemente rígida para alentar la planificación razonable y el seguimiento de la gestión conforme avanza el proyecto (Pressman, 2010).

La estrategia seguida para la realización de las pruebas al sistema contiene dos niveles: el primero lo constituyen las pruebas Unitarias y el segundo nivel las pruebas de Aceptación. En el primer nivel fueron realizadas las pruebas de aceptación directamente con el cliente concluidas con la entrega por parte del cliente de un Acta de aceptación, en la que deja constancia de que el sistema que fue desarrollado cumple con las especificaciones técnicas y funcionales que fueron solicitadas al momento de realizarse la solicitud de servicio. En el segundo nivel fue usada la librería PyUnit para Python la cual arroja el tiempo de respuesta del sistema.

3.6. Método de prueba

La metodología propone 2 métodos fundamentales que serán usados en el proceso de desarrollo de las pruebas al sistema, los cuales serán puntualizados a continuación.

Pruebas de caja negra:

Las pruebas de cajas negras o funcionales son realizadas a la interfaz del software y para aplicarlas solo se necesita conocer la funcionalidad del mismo. Estas pruebas se centran en los requisitos funcionales, permitiendo derivar conjuntos de condiciones de entradas que ejercitarán por completo dichos requerimientos.

Para confeccionar los casos de prueba existen distintos criterios:

- **Técnica de la partición de equivalencia**: Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Técnica del análisis de valores límites**: Esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Técnica de grafos de causa-efecto**: Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Para la realización de los casos de prueba del sistema propuesto se selecciona el criterio de Partición de Equivalencia pues permite examinar los valores válidos e inválidos de las entradas existentes en el software.

Prueba de caja blanca:

Estas pruebas, también suelen ser llamadas estructurales o de cobertura lógica. En ellas se pretende investigar sobre la estructura interna del código, exceptuando detalles referidos

a datos de entrada o salida. Realizan un seguimiento del código fuente según se va ejecutando los casos de prueba, determinándose de manera concreta las instrucciones y/o bloques; que han sido ejecutados. A continuación, se describen las técnicas utilizadas para la realización del método.

Detección y corrección de errores: Cuando se encuentra un error (“bug”), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto (Moreno Sánchez, 2013).

Ruta o trayectoria básica: Permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba (Pressman, 2010).

3.7. Tipos de pruebas

Existen diferentes tipos de pruebas que se pueden aplicar para verificar que el sistema cumple con todos los requisitos identificados en el proceso de análisis y comprobar su correcto funcionamiento. A continuación, se explican los tipos de pruebas seleccionados:

Pruebas unitarias: Enfocan los esfuerzos de verificación en la unidad más pequeña del diseño de software. Al usar la descripción del diseño de componente como guía, las rutas de control importantes se prueban para descubrir errores dentro de la frontera del módulo. Se enfocan en la lógica de procesamiento interno y de las estructuras de datos dentro de las fronteras de un componente. Este tipo de pruebas puede realizarse en paralelo para múltiples componentes (Pressman, 2010).

Pruebas funcionales: Son un proceso de control de calidad que consiste en asegurar el cumplimiento de un sistema o componente con requerimientos funcionales. El objetivo principal de las pruebas funcionales es analizar el producto terminado y determinar si cumple con todo lo que debería hacer y si lo hace correctamente. Este tipo de pruebas entran dentro de lo que se llaman pruebas de caja negra: aquí no se centra en cómo se generan las respuestas del sistema, solo se analizan los datos de entrada y los resultados obtenidos (Oterino, 2014).

Pruebas de aceptación: Son realizadas con el cliente y define su aceptación del sistema. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos. Estas pruebas no se realizan durante el desarrollo, sino una vez pasadas todas las pruebas por parte del desarrollador. Pueden realizarse durante un período de semanas o meses, y mediante ellas implementación y pruebas del componente propuesto se descubren errores acumulados que con el tiempo puedan degradar el sistema. La mayoría de los constructores de productos de software usan un proceso llamado prueba alfa y prueba beta para descubrir errores que al parecer sólo el usuario final es capaz de encontrar (Pressman, 2010).

Las pruebas alfa se realizan en el espacio del desarrollador por un grupo representativo de usuarios finales. El software se usa en un escenario natural con el desarrollador y se realizan en un ambiente controlado.

Las pruebas beta se realiza en uno o más espacios del usuario final. A diferencia de la prueba alfa, por lo general el desarrollador no está presente. El cliente registra todos los problemas que se encuentran durante la prueba y los reporta al desarrollador periódicamente. En ocasiones se realiza una variación de la prueba beta, llamada prueba de aceptación del cliente, cuando el software se entrega a un cliente bajo contrato. El cliente realiza una serie de pruebas específicas con la intención de descubrir errores antes de aceptar el software del desarrollador.

3.8.1 Pruebas de Aceptación:

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (Sánchez, 2015).

Se empleó la técnica de prueba alfa, por el cliente y se evaluaron las funcionalidades del sistema basándose en la especificación de requisitos del expediente de proyecto. Los interesados verificaron cada requisito funcional y comprobaron si estos correspondían con los requerimientos planteados.

Al ser concluida las pruebas, se liberó el sistema de administración para PostgreSQL, entregándole al equipo de desarrollo la carta de aceptación en la que consta que el sistema está apto para ser utilizado y cumple con las expectativas planteadas.

Diseños de Casos de Prueba

Un Diseño de Caso de Prueba (DCP) está compuesto por un conjunto de entradas, respuestas (que emite el sistema de acuerdo a esas entradas) y el flujo central que indica el camino del escenario descrito. Estos son desarrollados para verificar el cumplimiento total o parcial de un requisito. Las entradas representan las variables que se pueden especificar y las mismas contienen: V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor de la variable en un determinado caso, ya que es irrelevante.

A continuación, se muestran los casos de pruebas generados para las funcionalidades asociadas al licenciador por tiempo para el sistema XILEMA GRHS.

Escenario	Descripción	Siglas de la Entidad	Serial	Fecha Inicio	Fecha Fin	Respuesta del sistema	Flujo central
EC 1.1 Generar licencia por tiempo satisfactoriamente.	Se accede al generador de licencias. Se llenan los campos requeridos para la licencia por tiempo. Se selecciona el botón Generar.	*V*	V	V	V	El sistema muestra el código de la licencia generada.	1-Acceder al generador de licencias.
EC 1.2 Seleccionar el botón Limpiar.	El usuario selecciona el botón Limpiar en el generador de licencias.	NA	NA	NA	NA	El sistema deshace todo los cambios y limpia los campos.	. 1-Acceder al generador de licencias. 2-Selecciona el botón Limpiar
EC 1.3 Campos vacíos.	El usuario selecciona el botón Generar sin llenar todos los campos o estando todos vacíos.	NA	NA	NA	NA	El sistema muestra un mensaje informando que existen campos vacíos o que las fechas no son correctas.	1- Acceder al generador de licencias. 2-Selecciona el botón Generar.

EC 1.4 Seleccionar el botón Exportar	El usuario selecciona el botón Exportar en el generador de licencias	V	V	V	V	El sistema exporta el código de la licencia generada en un archivo .txt	1- Acceder al generador de licencias. 2-Selecciona el botón Generar. 3-Seleccionar el botón Exportar

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Siglas de la Entidad	Campo de texto	No	Sin espacios entre ellas y sin tildes.
2	Serial	Campo de texto	No	Cadena de 50 caracteres.
3	Fecha Inicio	Numérico	No	Valores numéricos correspondientes a los días (1-31), meses (1-12), años (hasta el 3000)
4	Fecha Fin	Numérico	No	Valores numéricos correspondientes a los días (1-31), meses (1-12), años (hasta el 3000)

Escenario	Descripción	Código de Licencia	Respuesta del sistema	Flujo central
EC 2.1 Registrar licencia por tiempo satisfactoriamente	Se accede al módulo Licencia, en el panel que se despliega se selecciona la opción Tiempo, se ingresa el código de la licencia generada. Selecciona la opción Insertar.	V	Muestra los datos en pantalla de la licencia insertada.	1-Autenticarse en el sistema. 2-Acceder al módulo Licencia. 3-Seleccionar la opción Tiempo.

EC 2.2 Seleccionar el botón Cancelar	Se accede al módulo Licencia, en el panel que se despliega se selecciona la opción Tiempo, se ingresa el código de la licencia generada. Selecciona la opción Cancelar.	NA	Limpia el cuadro de texto.	1-Autenticarse en el sistema. 2-Acceder al módulo Licencia. 3-Seleccionar la opción Tiempo.
EC 2.3 No registrar licencia por tiempo satisfactoriamente o campos vacíos	Se accede al módulo Licencia, en el panel que se despliega se selecciona la opción Tiempo, se ingresan valores aleatorios o se mantiene el campo vacío. Selecciona la opción Insertar.	I	El sistema muestra in mensaje informando que la licencia es invalida.	1-Autenticarse en el sistema. 2-Acceder al módulo Licencia. 3-Seleccionar la opción Tiempo.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Código de Licencia	Campo de texto	No	Cadena de caracteres

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 3.1 Consultar información de	Se accede al módulo Licencia, en el	El sistema muestra correctamente	1-Autenticarse en el sistema.

la licencia correctamente	panel que se despliega se selecciona la opción Información.	la información de la licencia previamente registrada.	2-Acceder al módulo Licencia. 3-Seleccionar la opción Información.

Escenario	**Descripción**	**Respuesta del sistema**	**Flujo central**
EC 4.1 Notificar vía correo el vencimiento de la licencia por tiempo correctamente	Se ingresa en el sistema una licencia que cuente con un periodo de vencimiento comprendido entre los (15, 10, 5, 3,1) días.	El sistema envía la notificación sobre la cercanía de vencimiento de la licencia correctamente.	1-Ingresar en el sistema una licencia válida.

Pruebas unitarias

Las pruebas unitarias están dirigidas a las funciones internas del sistema. La prueba es una verificación técnica del software que los desarrolladores pueden usar para examinar si el código trabaja como se esperaba. Se realizan probando la lógica de la aplicación y comprobando el estado del software en varios puntos, para verificar que los resultados de dicho estado coincidan con los esperados.

En la ejecución de las pruebas unitarias se utilizó la herramienta llamada PyUnit, la cual está contenida en el lenguaje de programación Python. Al aplicarle la prueba al componente, en la primera iteración, arrojó como resultado 4 no conformidades y el tiempo de la ejecución fue de 1.557 segundos, como se observa en la imagen.

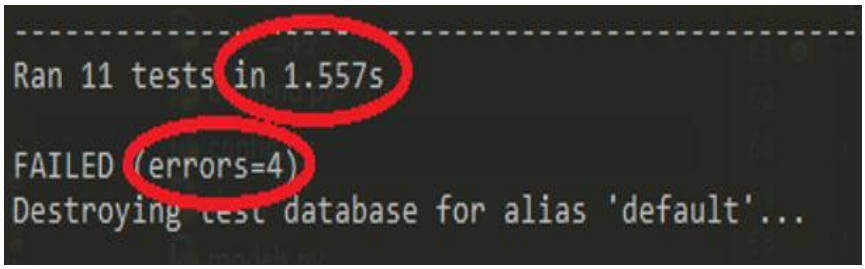


Figura 11 Resultado de la primera iteración

Las 4 no conformidades, eran producto de errores en las interfaces a nivel del servidor para cargar los logos del centro; luego de la primera iteración, se corrigieron estos errores y se realizó otra iteración de pruebas obteniendo como resultado ninguna no conformidad y el tiempo de ejecución aumentó a 1.363 segundos.

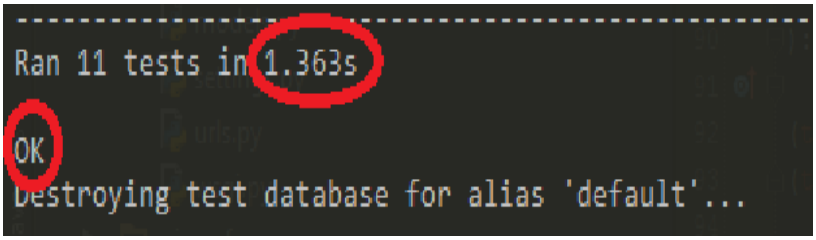


Figura 12 Resultado de la última iteración

3.9 Resultados de las pruebas realizadas

La realización de pruebas al módulo permitió detectar varias no conformidades en las primeras iteraciones siendo estas resueltas. Con la conclusión de esta fase de pruebas, fue posible comprobar que el módulo cumple con las especificaciones que se trazaron en los requisitos definidos. A continuación, se muestra la relación de no conformidades (detectadas y resueltas) por iteración:

Figura 13 Resultado de las pruebas de aceptación

Como se puede apreciar en la figura anterior se realizaron tres iteraciones de pruebas. A lo largo de la primera iteración se detectaron diez no conformidades, cinco asociadas a errores de interfaz y dos a errores ortográficos, todas fueron resueltas en la propia iteración. En la segunda iteración fueron detectadas seis no conformidades asociadas a errores de interfaz siendo satisfactoriamente solucionadas. En una tercera y última iteración no se detectó ninguna no conformidad por lo que la aplicación mostró un buen funcionamiento y se considera terminada.

Conclusiones parciales

En el presente capítulo se llevó a cabo el análisis de la arquitectura, los patrones arquitectónicos y los patrones de diseño que se utilizaron posteriormente en el desarrollo del sistema permitiendo así una mayor rapidez en esta etapa. Las pruebas realizadas permitieron encontrar errores que fueron solucionados en su totalidad.

Conclusiones

A partir del contenido del trabajo presentado, de los antecedentes revisados en la literatura y su análisis, se arriba a las siguientes conclusiones:

1. Las soluciones existentes consultadas, que de alguna manera tributan a la investigación, no satisfacen el problema de la investigación, no obstante, aportaron algunos elementos tenidos en cuenta en el desarrollo de la solución.
2. Con la aplicación de la metodología de desarrollo AUP-UCI posibilitó que se obtuvieran los artefactos generados del proceso ingenieril, proporcionando comprensión a la solución desarrollada, favoreciendo la rigurosidad en su diseño.
3. Con la implementación del sistema, se logró obtener un sistema que permite el licenciamiento por tiempo para XILEMA GRHS.
4. El proceso de pruebas desarrollado, permitió identificar y resolver los errores en la implementación, garantizando el correcto funcionamiento del producto final, obteniéndose una aplicación que cumple con los requisitos definidos y que satisface las necesidades establecidas en los mismos por el cliente.

Recomendaciones

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

- ✓ Sintetizar las dos licencias que se utilizan hoy en el sistema XILEMA GRHS, la licencia temporal (desarrollada en esta tesis) y la licencia por cantidad de computadoras conectadas, en una sola licencia.
- ✓ Incorporar el generador de licencias como un módulo de XILEMA GRHS.

Referencias Bibliográficas

1. 2008. ¿Qué es UML? *El Lenguaje de Modelado Unificado.* [En línea] 2008. [Citado el: 7 de Enero de 2019.] https://www.docirs.com/uml.htm.

2. Díaz Romeu, Ivaniet. 2016. *SISTEMA INFORMÁTICO DE GESTIÓN DE INDICADORES PARA LA FORMACIÓN DE ESTUDIANTES POTENCIALMENTE TALENTOSOS EN LA UCI.* La Habana : s.n., 2016.

3. 2019. Django 1.8 release notes. [En línea] 2019. [Citado el: 9 de Diciembre de 2018.] https://docs.djangoproject.com/en/2.1/releases/1.8/.

4. 2014. EcuRed. [En línea] 22 de Noviembre de 2014. [Citado el: 1 de Febrero de 2018.] http://www.ecured.cu/index.php/Modelo_de_dominio.

5. 2014. Ecured Portable versión 1.5. *Licencias Comerciales de los Software.* [En línea] 20 de Noviembre de 2014. [Citado el: 2 de Enero de 2019.]

6. 2019. ElecKey Release 9. *Software Licensing.* [En línea] 2019. [Citado el: 7 de Enero de 2019.] http://www.sciensoft.com/products/eleckey/.

7. Flanagan. 2011. JavaScript: The Definitive Guide - David Stilson. [En línea] 2011. [Citado el: 1 de Enero de 2019.]

8. JavaScript a fondo. [En línea] [Citado el: 3 de Enero de 2018.] https://desarrolloweb.com/javascript/#librerias.

9. JetBrains. 2018. [En línea] 2018. [Citado el: 29 de Diciembre de 2018.]

10. 2014. Kaspersky Small Office Security. *Kaspersky Lab.* [En línea] 2014. [Citado el: 3 de Diciembre de 2018.] http://www.kaspersky.es/software-antivirus-domestico/small-office-security.

11. Lugo Rojas, Yosley y Simón Méndez, Arian. 2015. *LICENCIAMIENTO DEL GESTOR DE RECURSOS DE HARDWARE Y SOFTWARE.* La Habana : s.n., 2015.

12. MacDonald. 2015. Html5 The Missing Manual Matthew Macdonald. [En línea] 2015. [Citado el: 20 de Diciembre de 2018.]

13. Martín Toral, Diego. 2011. Prácticas del Módulo Aplicaciones Ofimáticas. *Bubok.* [En línea] 2011. [Citado el: 12 de Enero de 2019.]

14. Meneses Hernandez, Duarny y Carbonell Figueredo, Ciro Alejandro. 2015. Modulo de presentacion de resultados de inventarios de hardware y software en el sistema GRHS. [En línea] 2015.

15. Metodología de desarrollo para la Actividad productiva de la UCI. [En línea] [Citado el: 2 de Diciembre de 2018.]

16. ModelN/sDashboard. [En línea]

17. 2014. NetSupport Manager. [En línea] 2014. [Citado el: 5 de Enero de 2019.] http://www.netsupportmanager.com/es/features.asp.

18. 2018. PgAdmin. [En línea] 2018. [Citado el: 4 de Diciembre de 2018.]

19. 2018. PostgreSQL. [En línea] 2018. [Citado el: 2 de Diciembre de 2018.]

20. PRESSMAN, R.S. 2010. *Ingeniería del software. Un enfoque práctico.* 2010.

Pressman, Roger S. Ingenieria de Software:Un enfoque practico. 7, Vol. 7.

2018. Python. [En línea] 2018. [Citado el: 17 de Diciembre de 2018.] https://www.python.org/downloads/release/python-279/.

21. *Revista Cubana de Informática Médica.* Castro Márquez, Carlos Luis y Delgado García, Alejandro. 2014. 1, La Habana : s.n., 2014, Vol. 6.

22. 2014. Visual Paradigm. [En línea] 2014. [Citado el: 3 de Diciembre de 2018.]

23. 2016. XILEMA-Base-Web. *Wiki TLM.* [En línea] 2016. [Citado el: 12 de Diciembre de 2018.]

24. Sánchez, Tamara Rodríguez. 2015. Metodología de desarrollo para la Actividad productiva de la UCI. La Habana. Cuba : s.n., 2015.

27. Rumbaugh James, Jacobson Ivar y Booch, Grady. El Proceso Unificado de Desarrollo de Software. Manual de Referencia. s.l. : Addison Wesley.

28. Oterino, A.M. del C.G. 2014. ¿Pruebas de integración, funcionales, de carga?¿Qué diferencias hay? 2014.

29. Española, R. R. A. d. l. L. (2019). Definicion de Licencia.

30. Lozano Díaz, I. A., del Toro Gundin, B. J., Arencibia Jorgel, R., & Martínez Rodríguez, A. (2008). Producción científica de la Universidad de La Habana en el Web of Science, 2000 - 2006. ACIMED, 18, 0-0.

31. NetworkInventoryAdvisor. (2019). from https://www.network-inventory-advisor.com/es/software-license-audit.html

32. Open-AudIT. (2019). Introducing Open-AudIT. from https://www.open-audit.org/

33. Roger S. Pressman, P. D. (2010). INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO. SÉPTIMA EDICIÓN.

34. Salvatore D'Alo, A. D. B., Alessandro Donatelli, Claudio Marinelli. (2019). Planning assignment of software licenses.

35. Samuel Emilio , A. M. P. (2019). Las licencias de software.

36. SpiceworksInventory. (2019). from https://www.spiceworks.com/free-pc-network-inventory-software/

37. UCI. (2019). Centro TLM.

38. UCI. (2019). GRHS 1.0.

39. WinAuditv3.2.1. (2017). from https://www.portablefreeware.com/?id=610

Bibliografía

1. 2008. ¿Qué es UML? *El Lenguaje de Modelado Unificado.* [En línea] 2008. [Citado el: 7 de Enero de 2019.] https://www.docirs.com/uml.htm.
2. Díaz Romeu, Ivaniet. 2016. *SISTEMA INFORMÁTICO DE GESTIÓN DE INDICADORES PARA LA FORMACIÓN DE ESTUDIANTES POTENCIALMENTE TALENTOSOS EN LA UCI.* La Habana : s.n., 2016.
3. 2019. Django 1.8 release notes. [En línea] 2019. [Citado el: 9 de Diciembre de 2018.] https://docs.djangoproject.com/en/2.1/releases/1.8/.
4. 2014. EcuRed. [En línea] 22 de Noviembre de 2014. [Citado el: 1 de Febrero de 2018.] http://www.ecured.cu/index.php/Modelo_de_dominio.
5. 2014. Ecured Portable versión 1.5. *Licencias Comerciales de los Software.* [En línea] 20 de Noviembre de 2014. [Citado el: 2 de Enero de 2019.]
6. 2019. ElecKey Release 9. *Software Licensing.* [En línea] 2019. [Citado el: 7 de Enero de 2019.] http://www.sciensoft.com/products/eleckey/.
7. Flanagan. 2011. JavaScript: The Definitive Guide - David Stilson. [En línea] 2011. [Citado el: 1 de Enero de 2019.]
8. JavaScript a fondo. [En línea] [Citado el: 3 de Enero de 2018.] https://desarrolloweb.com/javascript/#librerias.
9. JetBrains. 2018. [En línea] 2018. [Citado el: 29 de Diciembre de 2018.]
10. 2014. Kaspersky Small Office Security. *Kaspersky Lab.* [En línea] 2014. [Citado el: 3 de Diciembre de 2018.] http://www.kaspersky.es/software-antivirus-domestico/small-office-security.
11. Lugo Rojas, Yosley y Simón Méndez, Arian. 2015. *LICENCIAMIENTO DEL GESTOR DE RECURSOS DE HARDWARE Y SOFTWARE.* La Habana : s.n., 2015.
12. MacDonald. 2015. Html5 The Missing Manual Matthew Macdonald. [En línea] 2015. [Citado el: 20 de Diciembre de 2018.]
13. Martín Toral, Diego. 2011. Prácticas del Módulo Aplicaciones Ofimáticas. *Bubok.* [En línea] 2011. [Citado el: 12 de Enero de 2019.]
14. Meneses Hernandez, Duarny y Carbonell Figueredo, Ciro Alejandro. 2015. Modulo de presentacion de resultados de inventarios de hardware y software en el sistema GRHS. [En línea] 2015.
15. Metodología de desarrollo para la Actividad productiva de la UCI. [En línea] [Citado el: 2 de Diciembre de 2018.]

16. ModelN/sDashboard. [En línea]

17.2014. NetSupport Manager. [En línea] 2014. [Citado el: 5 de Enero de 2019.] http://www.netsupportmanager.com/es/features.asp.

18.2018. PgAdmin. [En línea] 2018. [Citado el: 4 de Diciembre de 2018.]

19.2018. PostgreSQL. [En línea] 2018. [Citado el: 2 de Diciembre de 2018.]

20.PRESSMAN, R.S. 2010. *Ingeniería del software. Un enfoque práctico.* 2010.

21.Pressman, Roger S. Ingenieria de Software:Un enfoque practico. 7, Vol. 7.

22.2018. Python. [En línea] 2018. [Citado el: 17 de Diciembre de 2018.] https://www.python.org/downloads/release/python-279/.

23.Revista Cubana de Informática Médica. Castro Márquez, Carlos Luis y Delgado García, Alejandro. 2014. 1, La Habana : s.n., 2014, Vol. 6.

24.2014. Visual Paradigm. [En línea] 2014. [Citado el: 3 de Diciembre de 2018.]

25.2016. XILEMA-Base-Web. *Wiki TLM.* [En línea] 2016. [Citado el: 12 de Diciembre de 2018.]

26. Española, R. R. A. d. l. L. (2019). Definicion de Licencia.

27.Lozano Díaz, I. A., del Toro Gundin, B. J., Arencibia Jorgel, R., & Martínez Rodríguez, A. (2008). Producción científica de la Universidad de La Habana en el Web of Science, 2000 - 2006. ACIMED, 18, 0-0.

30.NetworkInventoryAdvisor. (2019). from https://www.network-inventory-advisor.com/es/software-license-audit.html

31.Open-AudIT. (2019). Introducing Open-AudIT. from https://www.open-audit.org/

32.Roger S. Pressman, P. D. (2010). INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO. SÉPTIMA EDICIÓN.

33.Salvatore D'Alo, A. D. B., Alessandro Donatelli, Claudio Marinelli. (2019). Planning assignment of software licenses.

34.Samuel Emilio , A. M. P. (2019). Las licencias de software.

35.SpiceworksInventory. (2019). from https://www.spiceworks.com/free-pc-network-inventory-software/

36.UCI. (2019). Centro TLM.

35.UCI. (2019). GRHS 1.0.

36.WinAuditv3.2.1. (2017). from https://www.portablefreeware.com/?id=610 27. 37.Sánchez, Tamara Rodríguez. 2015. Metodología de desarrollo para la Actividad productiva de la UCI. La Habana. Cuba : s.n., 2015.

38. Rumbaugh James, Jacobson Ivar y Booch, Grady. El Proceso Unificado de Desarrollo de Software. Manual de Referencia. s.l.: Addison Wesley.

39. Oterino, A.M. del C.G. 2014. ¿Pruebas de integración, funcionales, de carga? ¿Qué diferencias hay? 2014.