UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

TÍTULO: LICENCIAMIENTO DEL GESTOR DE RECURSOS DE HARDWARE Y SOFTWARE

Autor(es):

Yosley Lugo Rojas
Arian Simón Méndez

Tutor:

Ing. Yaniel Alfredo Velázquez Bruceta

Co-Tutor:

Profesor Titular. Mariano Héctor Jiménez Milián Dr.C.

LA HABANA, JUNIO DE 2015

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los _____ días del mes de _____ de 2015.

Autores:

_____ _____

Yosley Lugo Rojas Arian Simón Méndez

Tutor:

_____

Ing. Yaniel Alfredo Velázquez Bruceta

Co-Tutor:

_____

Profesor Titular. Mariano Héctor Jiménez Milián. Dr.C.

RESUMEN

La Universidad de las Ciencias Informáticas (UCI) se encuentra desarrollando un producto nombrado Gestor de Recursos de Hardware y Software (GRHS), implementado bajo la arquitectura cliente – servidor. GRHS consta de tres partes, una aplicación para instalar en los ordenadores de los clientes denominada Gclient, que es la encargada de obtener las propiedades de las piezas y programas instalados, así como, detectar los cambios, determinar las incidencias y tomar algunas acciones automáticas cuando ocurren las mismas. Otra aplicación para instalar en el servidor de la entidad donde se despliegue, denominada Gserver, que no es más que el centralizador de inventarios que recibe la información enviada por los clientes, envía órdenes a las computadoras inventariadas y notifica las alertas a los interesados cuando ocurren incidencias. Por último y no menos importante Gadmin, que es la interfaz de administración encargada de visualizar y gestionar toda la información recolectada.

Este software creado con un fin comercial no cuenta con un soporte de licenciamiento, el que resulta necesario para su comercialización. La investigación desarrollada constituyó un perfeccionamiento del software GRHS al aportar un licenciamiento por cantidad de ordenadores que proporciona mayor control para la empresa que lo comercializa, lo que se corresponde con las exigencias de las empresas comercializadoras actuales, de la cual Segurmática es un ejemplo de ello.

Palabras claves: licenciamiento, licenciamiento por cantidad de ordenadores.

ÍNDICE

INTRODUCCIÓN **5**

CAPÍTULO 1. FUNDAMENTOS TEÓRICO-METODOLÓGICOS EN QUE SE SUSTENTAN LOS MÉTODOS DE LICENCIAMIENTO **10**

1.1 Licenciamiento de software 10

1.2 Software similares para la realización de inventarios 12

1.3 Software licenciados por cantidad de ordenadores 13

1.4 El programa a licenciar 15

1.5 Metodologías de desarrollo del software 16

1.6 Lenguaje, tecnología y herramientas seleccionadas 17

1.7 Criptografía 21

1.8 Conclusiones del capítulo 23

CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL MÓDULO DE LICENCIAMIENTO **24**

2.1 Propuesta del módulo de licenciamiento por cantidad de ordenadores 24

2.2 Modelo de dominio 27

2.3 Requisitos funcionales y no funcionales del software 28

2.4 Definición de los casos de uso 30

2.5 Prototipo de interfaz para la generación de licencias 34

2.6 Fundamentación del uso de patrones de diseño 35

2.6 Diagrama de clases del diseño 36

2.7 Conclusiones del capítulo 38

CAPÍTULO 3. IMPLEMENTACIÓN Y VALIDACIÓN DEL MÓDULO DE LICENCIAMIENTO **39**

3.1 Diagrama de despliegue 39

3.2 Diagrama de componentes 40
3.3 Validación de los resultados obtenidos 42
3.4 Conclusiones del capítulo 49
CONCLUSIONES 50
RECOMENDACIONES 51
REFERENCIAS BIBLIOGRÁFICAS 52
BIBLIOGRAFÍA 56
ANEXOS 60

INTRODUCCIÓN

El vertiginoso desarrollo de la ciencia y la técnica en la última década, en especial el avance significativo de las tecnologías de la información y las comunicaciones (TIC) representan nuevos retos para la sociedad. Esta situación define en los diferentes sectores un conjunto de objetivos dentro de los que figuran como principales la superación profesional, la rapidez y optimización de los servicios; imponiendo la urgente necesidad de transformar, perfeccionar y lograr una organización de manera efectiva y con calidad en correspondencia con los cambios del entorno. Todo lo cual es posible mediante la utilización de los software creados con estas finalidades.

Los software al crearse se conciben con mecanismos de protección para que la información que se maneje sea confiable y robusta. Estos mecanismos posibilitan el control de los software para que se usen con el fin por el que se crearon y no sean modificados. Han surgido muchas vías con este objetivo, como son los protocolos de seguridad; dentro de ellos podemos encontrar el logueo o autenticación que permite que solo los usuarios registrados puedan acceder a la aplicación con sus respectivos permisos para poder modificar, eliminar, insertar datos al sistema, entre otras operaciones. Este hecho no da la posibilidad de que el usuario pueda modificar las funcionalidades que contemple la aplicación para adecuarla a sus necesidades, para ello existe la licencia del producto.

Las licencias son una de las principales formas a la hora de controlar un sistema informático, ya que permiten que el software funcione al cien por ciento o sea limitado. Eso depende a la hora de su comercialización, de las pautas que se definan con el cliente.

Una de las fortalezas que presentan los software 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.

En Cuba se desarrollan proyectos informáticos que con el decurso de los años 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 de Estudios Universitarios de Ingenierías y Arquitectura (CUJAE), la Universidad de la Habana (UH) y la Universidad de las Ciencias Informáticas (UCI) figuran como principales partícipes en dicha tarea.

La UCI desde sus inicios siempre ha estado a la vanguardia en el desarrollo y comercialización de software, gracias a los distintos centros con que cuenta, como es el caso del Centro de Identificación y Seguridad Digital (CISED), el Centro de Informática Médica (CESIM), el Centro de Telemática (TLM), entre otros; los cuales están distribuidos en las diferentes facultades con que cuenta la universidad.

Situación Problemática

El Centro de TLM de la UCI, dispone en la actualidad de un producto que permite realizar el inventario de hardware y software en una red de computadoras, el cual se denomina GRHS (Gestor de Recursos de Hardware y Software). El sistema permite mantener un control sobre los recursos de hardware presentes en las computadoras de la red que han sido inventariadas, así como conocer el software que se utiliza en las estaciones de trabajo de la organización.

El software para realizar la tarea de inventario del hardware y el software de la red de computadoras, se apoya en un servidor (Gserver, Servidor de GRHS) y un cliente (Gclient, Cliente de GRHS). La instalación de dicho sistema es llevada a cabo directamente por el personal responsable de administrar la red de computadoras en la entidad donde se ejecutará la aplicación.

GRHS en la actualidad posee un grupo de desventajas, entre las que se destaca la seguridad del software. La implementación del sistema no incluye un control sobre su uso, lo que permite que la empresa que lo utilice pueda hacerlo para un número ilimitado de ordenadores, incluso hacer un uso no controlado de este producto en empresas con las cuales no se ha comercializado el software.

Dada la situación anterior se plantea como **problema a resolver:** ¿Cómo contribuir a la seguridad de GRHS mediante un licenciamiento que se gestione desde el servidor del sistema?

Basado en lo anterior, se define como **objeto de estudio:** Proceso de licenciamiento de los software informáticos.

En vista a darle solución al problema planteado se define como **objetivo general:** Implementar un mecanismo para contribuir a la seguridad de la plataforma GRHS mediante un licenciamiento, desde el servidor del sistema, limitando la cantidad de clientes.

Definiendo como **campo de acción:** El licenciamiento por cantidad de ordenadores que se gestione desde el servidor del sistema.

Se define como **idea a defender:** El software GRHS acompañado mediante un licenciamiento por cantidad de ordenadores garantiza en su comercialización mayor control sobre su uso.

Con el fin de resolver el problema de la investigación y darle cumplimiento al objetivo planteado con anterioridad de forma sistemática y ascendente, se realizaron las siguientes **tareas científicas:**

1. Determinación de los fundamentos teóricos y metodológicos en que se sustentan los procesos de licenciamiento de los software informáticos en particular, las licencias por cantidad de ordenadores para que sirvan de base en la elaboración del módulo de licenciamiento por cantidad de ordenadores.
2. Análisis de diferentes software que presenten licencias, en particular los que contemplan licencias por cantidad de ordenadores, permitiendo identificar el comportamiento de los mismos.
3. Selección de la metodología, lenguaje, tecnología y herramientas a utilizar para el desarrollo de los procesos de licenciamiento por cantidad de ordenadores.
4. Modelación de los procesos fundamentales relacionados con las licencias por cantidad de ordenadores, para ilustrar el desarrollo de los mismos.
5. Elaboración de los requisitos funcionales y no funcionales de las licencias por cantidad de ordenadores, determinando las cualidades que logren el correcto funcionamiento del módulo de licenciamiento.
6. Implementación de los requisitos del módulo de licenciamiento, permitiendo la obtención del producto.
7. Validación de los resultados obtenidos para lograr un software de calidad.

Durante la presente investigación se hace uso de los siguientes **métodos de investigación científica:**

Dentro de los **métodos teóricos:**

1. **Histórico–Lógico:** Para comprobar cómo ha evolucionado el proceso de licenciamiento de los software informáticos, posibilitando el análisis de los tipos de licencias para comprender lógicamente cuáles son sus tendencias actuales**.**
2. **Analítico-Sintético:** Para la comprensión de los conceptos básicos relacionados con las licencias de software.
3. **Modelación:** Para la representación de los procesos fundamentales relacionados con las licencias de software por cantidad de ordenadores mediante gráficas, diagramas y figuras.

Dentro de los **métodos empíricos:**

1. **Intercambio con especialistas del Centro de Identificación y Seguridad Digital (CISED) y la empresa Segurmática:** Para obtener su valoración sobre la concepción teórico-metodológica con respecto a las licencias de software, en especial las referidas a cantidad de ordenadores y gestionadas desde el servidor del sistema

Impacto económico, político y social

El impacto que se espera alcanzar con la aplicación de Licenciamiento de GRHS del Centro de TLM de la Facultad 2, se fundamenta a partir de que:

- ✓ Todo software debe estar licenciado para que sus desarrolladores puedan tener control sobre el mismo a la hora de su comercialización.
- ✓ Un software con licenciamiento representa un avance para el desarrollo del Centro de TLM y de la UCI, teniendo en cuenta que actualmente no existe ningún tipo de software con estas características.
- ✓ El software licenciado posibilita el intercambio con el cliente y la dependencia del mismo hacia el productor.

La comercialización del software permitirá ingresos a la universidad, de ahí el impacto económico que se pretende alcanzar. Las licencias de software tienen un gran impacto político y social debido a que las mismas responden a lo expresado en el artículo 131 de los lineamientos para el desarrollo económico vigentes en el país. (1)

Estructura del documento

La estructura del documento consta de introducción, tres capítulos, conclusiones, recomendaciones, referencia bibliográfica, bibliografía y anexos.

CAPÍTULO 1: FUNDAMENTOS TEÓRICO-METODOLÓGICOS EN QUE SE SUSTENTAN LOS MÉTODOS DE LICENCIAMIENTO.

En este capítulo se exponen los fundamentos teóricos y metodológicos en que se sustentan los métodos de licenciamiento, la metodología de desarrollo del software seleccionada, el lenguaje, la tecnología y las herramientas a utilizar, tendencias actuales de los software que presentan licencias en particular aquellas por cantidad de ordenadores gestionadas desde el servidor del sistema, y un estudio sobre los algoritmos criptográficos.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL MÓDULO DE LICENCIAMIENTO.

En este capítulo se describe la propuesta del módulo de licenciamiento por cantidad de ordenadores. Se modela el diagrama de modelo de dominio, se exponen los requerimientos funcionales y no funcionales identificados, se modela el diagrama de casos de uso del sistema y se describe cada uno de los casos de uso, se muestra el prototipo de interfaz de la solución propuesta, se fundamenta el uso de patrones de diseño y se modela el diagrama de clases del diseño.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DEL MÓDULO DE LICENCIAMIENTO.

En este capítulo se modelan los diagramas de despliegue y de componentes, se analizan los resultados obtenidos realizándose la prueba de caja negra para la validación de interfaces y se muestra gráficamente las no conformidades detectadas durante el proceso de prueba.

CAPÍTULO 1. FUNDAMENTOS TEÓRICO-METODOLÓGICOS EN QUE SE SUSTENTAN LOS MÉTODOS DE LICENCIAMIENTO

A partir de la búsqueda y estudio de la información existente sobre el licenciamiento de programas informáticos, se exponen los temas relacionados con el proceso de licenciamiento de software para su mejor compresión. También con el desarrollo de este capítulo, se presentan una serie de conceptos a los que se hará referencia en el resto del trabajo. Se abordarán diferentes puntos referentes a los tipos de licencias, análisis de software existentes, lenguaje, tecnología y las herramientas seleccionadas para el desarrollo del módulo de licenciamiento por cantidad de ordenadores conjunto con un estudio sobre los algoritmos criptográficos.

1.1 Licenciamiento de software

El tema de las licencias de software puede ser muy complejo. El negocio del software de basa en licencias binarias. La propiedad intelectual de los distribuidores de software comercial nace del código fuente. Las licencias de software se crean con diversos fines empresariales y para afrontar diversos tipos de relaciones (como distribuidor/cliente y partner/partner). Los desarrolladores de software tanto comercial como no comercial utilizan decenas de licencias que abarcan una gran variedad de términos y condiciones. (2)

Definición de licencia de software

Una **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.

Las licencias de software pueden establecer entre otras cosas: la cesión de determinados derechos del propietario al usuario final sobre una o varias copias del programa informático, los límites en la responsabilidad por fallos, el plazo de cesión de los derechos, el ámbito geográfico de validez del contrato e incluso pueden establecer determinados compromisos del usuario final hacia el propietario, tales como la no cesión del programa a terceros o la no reinstalación del programa en equipos distintos al que se instaló originalmente. (2)

Tipos de licencias y sus características

- ✓ Propietaria: Uso en una computadora por el pago de un precio.
- ✓ Shareware: Uso limitado en tiempo o capacidades, después pagar un precio.
- ✓ Freeware: Usar y copiar ilimitado, el precio es cero.
- ✓ Software libre: Usar, copiar, estudiar, modificar, redistribuir. Código fuente incluido.

Propietaria

Son licencias muy restrictivas y prohíben totalmente la distribución/copia del programa. Sólo permiten ejecutar el programa a las personas que han comprado la licencia que, por supuesto, es personal e intransferible. También prohíben la modificación o adaptación del programa y para evitarlo suelen distribuir el programa sin el código fuente.

El ejemplo más claro de licencia propietaria es la que usa Microsoft con su sistema operativo Windows. En esta licencia llamada EULA (End User License Agreement) es destacable una parte del texto en la que se comunica al usuario que el producto se entrega tal cual (as is), con todos los fallos que pueda tener y sin garantías. También advierte que Microsoft no se hace responsable de los daños que produzca el software, por ello el usuario no podrá pedir responsabilidades si se pierden todos los datos de la empresa por un fallo de Windows, por ejemplo. El usuario tampoco podrá pedir que se arregle un error que ha detectado en el programa, puesto que se ha aceptado el producto con ese error, con sus virtudes y con sus defectos.

Shareware

Es otra modalidad de comercialización todavía más extendida, el programa se distribuye con limitaciones, bien como versión de demostración o evaluación, con funciones o características limitadas o con un uso restringido a un límite de tiempo establecido (por ejemplo 30 días). Así, se le da al usuario la oportunidad de probar el producto antes de comprarlo y, más tarde, adquirir la versión completa del programa.

Un ejemplo muy claro de este tipo es el software antivirus, estas compañías suelen permitir la descarga de sus productos de evaluación que sólo son válidos para un determinado número de días. Una vez superado el máximo, el programa se bloquea y es necesario comprar el producto si deseas seguir utilizándolo.

Freeware

Es todo aquel programa que se distribuya gratuitamente, con ningún costo adicional. Uno de los principales ejemplos es la suite de navegador y cliente de correo y noticias de Mozilla, distribuido también bajo licencia GPL (Software Libre). (2)

1.2 Software similares para la realización de inventarios

NetSupport Manager

Potente funcionalidad de control remoto y diagnóstico del sistema desde cualquier dispositivo Windows, Mac, Android o iOS. El usuario puede obtener una vista en tiempo real del hardware y del software instalado en cada estación de trabajo. NetSupport Manager reúne más de 80 elementos de información, específicamente sobre el hardware o el entorno de cada ordenador.

Además, también se proporciona un inventario de software completo de las aplicaciones instaladas actualmente y de todas las revisiones de sistema operativo instaladas.

NetSupport Manager lleva 20 años proporcionando el control remoto más seguro del mercado. La última versión sigue ampliando las cuestiones de seguridad al incluir desde registros de actividad hasta confirmación del usuario para cifrados de 256 bits, soporte para tarjeta inteligente, integración de AD y muchas otras características. NetSupport Manager es el producto elegido por instituciones militares y financieras de todo el mundo. (3)

Everest Ultimate Edition

Es una completa utilidad de software de diagnóstico de ordenadores que ayuda durante la instalación, la optimización o solucionar problemas del ordenador, proporcionando toda la información que se te ocurra sobre el sistema, desde los dispositivos de hardware y los controladores instalados a las métricas de seguridad del sistema operativo y de estabilidad.

Más que la información del sistema, Everest Ultimate Edition también ofrece capacidades integrales de control y comparación de hardware con informes en tiempo real. Aproveche estas potentes herramientas

para comparar el rendimiento de su computadora a otras computadoras y prevenir el sobrecalentamiento, problemas de energía y fallos de hardware. (4)

Speccy

Speccy te ofrecerá estadísticas detalladas sobre cada pieza de hardware de tu ordenador, incluyendo compatibilidad con la CPU, la placa base, la memoria RAM, las tarjetas gráficas, los discos duros, las unidades ópticas, el audio. Además, Speccy agrega las temperaturas de tus diferentes componentes, por lo que puedes ver fácilmente si hay algún problema.

A primera vista, Speccy puede parecerse a una aplicación para los administradores de sistemas y los usuarios avanzados. Lo es sin duda, pero también puede ayudar a los usuarios normales en su vida informática cotidiana. (5)

1.3 Software licenciados por cantidad de ordenadores

Kaspersky Small Office Security

Las empresas más pequeñas se enfrentan a los mismos riesgos de seguridad que las empresas de mayor tamaño, pero no suelen disponer del tiempo ni de los recursos necesarios para configurar y gestionar complejas soluciones de seguridad. Kaspersky Small Office Security ofrece tecnologías de protección para empresas diseñadas para garantizar la facilidad de instalación, configuración y uso. El programa protege sus equipos de escritorio y servidores de archivos con Windows, así como smartphones y tablets Android; para mantener a buen recaudo sus transacciones bancarias online, los datos de su empresa y la información que le confían sus clientes.

- ✓ Seguridad excepcional: la protección en varios niveles protege su negocio frente a las amenazas de malware más recientes.
- ✓ Protección en Internet: protege su empresa frente a ataques de phishing y exploits.
- ✓ Seguridad para dispositivos móviles: las tecnologías antimalware y antirrobo protegen los dispositivos Android.
- ✓ Protección de las transacciones bancarias online: la tecnología Pago Seguro añade otro nivel de seguridad para transacciones bancarias online.

- ✓ Protección de los datos de la empresa: el cifrado de datos y las copias de seguridad local u online protegen sus datos empresariales más importantes, aunque se dañe el dispositivo o incluso en caso de pérdida o robo.
- ✓ Administrador de contraseñas: genera contraseñas seguras y ayuda a los usuarios a mantenerse protegidos. (6)

Así como Kaspersky Small Office Security ofrece su licencia por cantidad de computadoras y por tiempo, también lo hacen los demás productos de esta gran compañía que es Kaspersky pudiendo mencionar:

- ✓ Kaspersky Endpoint Security for Business Select
- ✓ Kaspersky Endpoint Security for Business Advanced
- ✓ Kaspersky Total Security for Business

Segurmática Antivirus

Es un software totalmente cubano orientado a la protección contra el accionar de los programas malignos en sistemas operativos de Microsoft Windows que contribuye a elevar la seguridad informática del sistema operativo de la computadora. Actualmente poseen dos variantes: Segurmática Antivirus Personal y Segurmática Antivirus Corporativo. (7)

Segurmática Antivirus Personal:

Programa antivirus que puede ser instalado en diferentes sistemas operativos a partir de Windows 2000 y es compatible con Windows 7 (plataforma de 32 bits). Es un antivirus que protege el sistema de archivo de la computadora contra cualquier amenaza registrada en su base de datos y posee además un avanzado método de detección genérico. (7)

Segurmática Antivirus Corporativo:

Es el término que se utiliza para referirse al escenario basado en una estructura cliente-servidor, donde los programas clientes son del tipo Segurmática Antivirus y el servidor es el Servidor Corporativo al cual se conectan éstos.

El Servidor Corporativo es un software orientado a la administración centralizada del programa Segurmática Antivirus (el programa antivirus propiamente) instalado en las computadoras de una red basada en un dominio de Microsoft Windows.

Este producto está formado por dos componentes: el Servidor, destinado a la comunicación con las aplicaciones antivirus y su gestión, y la Consola de Administración, interfaz del producto con las prestaciones típicas de una Consola de Administración de Microsoft. (7)

Resultados del estudio de los software

El estudio de los sistemas similares permitió una mejor comprensión de los distintos métodos de licenciamiento que utilizan los software en la actualidad, aunque no posibilitó el entendimiento de los procesos que realizan para generar la licencia, ya que es una información que se reserva la empresa. Por lo expuesto anteriormente, los autores de la presente investigación deciden diseñar e implementar una solución que resuelva el problema planteado.

1.4 El programa a licenciar

Propósito

El objetivo de licenciar el software GRHS, es con carácter comercial. Puesto que a la hora de su comercialización se quiere tener un control de la cantidad de ordenadores que se conectarán al servidor del sistema en una determinada entidad, quedando sin reconocer los demás ordenadores que sobrepasen la cantidad establecida en la licencia.

Funcionamiento y estructura

El Gestor de Recursos de Hardware y Software (GRHS) está compuesto de tres aplicaciones: Gclient, Gserver y Gadmin. La aplicación Gclient es la encargada de obtener las propiedades de las piezas y programas instalados en una computadora, así como, detectar los cambios, determinar las incidencias y tomar algunas acciones automáticas cuando ocurren las mismas. La aplicación Gserver es el centralizador de inventarios que recibe la información enviada por los recolectores, envía órdenes a las computadoras inventariadas y notifica las alertas a los interesados cuando ocurren incidencias. Por último y no menos

importante Gadmin, que es la interfaz de administración encargada de visualizar y gestionar toda la información recolectada por cada uno de los clientes. (8)

1.5 Metodologías de desarrollo del software

Un proceso de software detallado y completo suele denominarse “Metodología”. Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, espiral entre otros). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucradas, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc. Habitualmente se utiliza el término “método” para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo, por ejemplo, suele hablarse de métodos de análisis y diseño. (9)

Metodología seleccionada

La metodología de desarrollo seleccionada es Rational Unified Procces (RUP), la cual presenta una serie de beneficios citados a continuación:

- ✓ Permite desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, rendimiento, seguridad y mantenimiento del software, mediante la gestión sistemática de los riesgos.
- ✓ Permite la producción de software que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos y un costo predecible.
- ✓ Enriquece la productividad en equipo y proporciona prácticas óptimas de software a todos los miembros.
- ✓ Proporciona guías para las distintas áreas, tales como modelado de negocios, arquitectura web, pruebas y calidad. (10)

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. (11)

Se definió RUP como metodología de desarrollo de software debido a que es la utilizada por el equipo de desarrollo de GRHS, propone una fuerte estructura y organización a la hora de realizar el trabajo y genera una alta documentación favoreciendo el carácter cambiante del equipo de desarrollo de GRHS.

1.6 Lenguaje, tecnología y herramientas seleccionadas

Seguidamente se describe el lenguaje, la tecnología y las herramientas seleccionadas para dar solución a la problemática planteada.

Python v2.7

Python es un lenguaje de programación de alto nivel creado por Guido Van Rossum a principios de los años 90. Su uso permite que el código de la aplicación en la que se trabaje sea más corto que su equivalente en otros lenguajes de programación y con el tipado dinámico se puede manejar con mayor facilidad el cúmulo de información que se gestiona. Posee extensas bibliotecas estándar y módulos de terceros para prácticamente todas las tareas. Además la organización de los bloques de código hace más sencilla la lectura y comprensión de los mismos. Facilita la realización de pruebas, detecta muchos de los errores de programación que escapan al control de los compiladores y proporciona información muy rica para detectarlos y corregirlos.

Una ventaja fundamental que presenta dicho lenguaje es la gratuidad de su intérprete. Su intérprete tiene versiones para prácticamente cualquier plataforma en uso: ordenadores con sistema operativo Linux, ordenadores con sistema operativo Windows, entre otros. (12)

Se selecciona Python como lenguaje de programación para el desarrollo de la aplicación por las características y ventajas presentadas anteriormente, teniendo en cuenta principalmente que el sistema GRHS al cual se debe integrar la solución se encuentra desarrollada en el lenguaje de Python v2.7.

Django v1.4

Django es un framework de desarrollo web de código abierto, escrito en Python, basado en la arquitectura Model-View-Template. Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kansas, y fue liberada al público bajo una licencia BSD en julio de 2005; el framework fue nombrado en alusión al guitarrista de jazz gitano Django Reinhardt.

La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio “No te repitas (DRY, del inglés Don't Repeat Yourself)”. Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

Alguna de las características que posee Django son:

- ✓ Un mapeador objeto-relacional.
- ✓ Aplicaciones "enchufables" que pueden instalarse en cualquier página gestionada con Django.
- ✓ Una API de base de datos robusta.
- ✓ Un despachador de URLs (Localizador de Recursos Uniforme o Uniform Resource Locator por sus siglas en inglés) basado en expresiones regulares.
- ✓ Un sistema "middleware" para desarrollar características adicionales; por ejemplo, la distribución principal de Django incluye componentes middleware que proporcionan cacheo, compresión de la salida, normalización de URLs, protección CSRF (Falsificación de Petición en Sitios Cruzados o Cross-site Request Forgery por sus siglas en inglés) y soporte de sesiones.
- ✓ Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración. (13)

Se decide utilizar el framework de desarrollo Django v1.4 pues la versión actual del sistema GRHS se implementó haciendo uso del mismo.

Eclipse Ganymede v3.4.2

Eclipse es un IDE (Integrated Development Environment - por su nombre en inglés) de código abierto y multiplataforma que ha alcanzado un alto grado de madurez en el desarrollo de lo que se conoce como “Aplicaciones de cliente enriquecido”. Cuenta con herramientas para desarrollar aplicaciones de consola, web y presenta múltiples extensiones para programar en lenguajes como Java y Python para el cual existe el plugin Pydev. Fue desarrollado originalmente por IBM (International Business Machines - por su nombre en inglés) y su futuro está ahora en manos de la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

En cuanto a las aplicaciones clientes, Eclipse provee al programador de plataformas (frameworks) muy ricas para el desarrollo de aplicaciones gráficas, para la definición y manipulación de modelos de software, aplicaciones web, entre otros. (14)

Eclipse Ganymede v3.4.2 es un IDE que asegura robustez y rendimiento, es configurable de acuerdo a las necesidades del programador y de código abierto. Haciéndose una opción viable para el desarrollo del software.

QT Designer v4.8.6

QT es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario. La biblioteca la desarrolla la que fue su creadora, la compañía noruega Trolltech, actualmente renombrada a QT Software, y que desde junio de 2008 es propiedad de Nokia. Utiliza el lenguaje de programación C++ de forma nativa y además existen múltiples bindings[1] para otros lenguajes, como: Python (PyQt), Java (QtJambi), PHP (PHP-Qt), Ruby (QtRuby), entre otros. (15)

Para desarrollar la interfaz visual utilizaremos PyQt, ya que el lenguaje de programación que se usa en el proyecto es Python. PyQt se estructura en Python como un conjunto de módulos con más de 300 clases y 6000 métodos. (15)

Siendo estos módulos: QtGui, QtCore, QtOpenGL, entre otros.

- ✓ **QtGui:** Componentes gráficos y clases relacionadas.
- ✓ **QtCore:** Núcleo de Qt, no contiene funciones gráficas, trabaja con funciones de tiempo, directorios, tipos de datos, urls, entre otros.
- ✓ **QtOpenGL:** Conjunto de clases para renderizado 2D y 3D usando OpenGL.

Visual Paradigm for UML Enterprise Edition v5.0

Visual Paradigm for UML es una herramienta UML (Unified Modeling Language - por su nombre en inglés: lenguaje de modelado de sistemas de software) profesional que soporta el ciclo de vida completo del

[1] Binding: Adaptación de una biblioteca para ser usada en un lenguaje de programación distinto de aquél en el que ha sido escrita.

desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una construcción rápida de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación.

Sus principales características son:

- ✓ Brinda la posibilidad de crear los artefactos necesarios para la confección de un software cumpliendo además con el estándar UML v2.0.
- ✓ Brinda numerosos estereotipos a utilizar lo cual permite que los diagramas se entiendan mejor.
- ✓ Brinda la posibilidad de documentar todo el trabajo sin tener que utilizar herramientas externas. (16)

Visual Paradigm for UML Enterprise Edition v5.0 es la herramienta de modelado de software elegida para representar los artefactos de la aplicación debido a las facilidades que brinda para la modelación de sistemas ya que agiliza la creación de los diagramas definidos en la metodología de desarrollo RUP. Se puede utilizar en sistemas operativos GNU/Linux incluyendo además que el equipo de desarrollo está familiarizado con el uso de la herramienta.

PostgreSQL v9.2

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (17)

PostgreSQL puede ser extendido por el usuario añadiendo tipos de datos, operadores, funciones agregadas, funciones ventanas y funciones recursivas, métodos de indexado y lenguajes procedurales. También soporta conexiones que abarcan TCP/IP, sockets Unix y sockets NT, además de soportar completamente ODBC. Es bueno ante grandes informaciones pues soporta hasta 32 TB de capacidad en las tablas con tuplas de hasta 1,6 TB y campos de 1 GB. A pesar poseer las capacidades anteriormente descritas está bajo licencia libre sin ningún costo por su uso. A partir de su versión 9.1 tiene soporte para replicación de tablas siendo necesario en escenarios donde existan muchas estaciones inventariadas. (17)

Se selecciona PostgreSQL v9.2 como gestor de base de datos por las características antes mencionadas y principalmente porque la versión actual del software GRHS funciona haciendo uso de dicho gestor.

La criptografía es la vía esencial que se utiliza en los diferentes software para el ocultamiento de la información. Las licencias son productos informáticos que en su concepción tienen como elemento imprescindible los diferentes algoritmos criptográficos, de ahí su importancia y la necesidad de ser abordada en esta tesis.

1.7 Criptografía

La palabra Criptografía proviene etimológicamente del griego Kruiptoz (Kriptos-Oculto) y Grajein (Grafo-Escritura) y significa "arte de escribir con clave secreta o de un modo enigmático".
La Criptografía hace años que dejó de ser un arte para convertirse en una técnica (o conjunto de ellas) que tratan sobre la protección (ocultamiento ante personas no autorizadas) de la información. Entre las disciplinas que engloba la criptografía se destacan: la Teoría de la Información, la Matemática Discreta, la Teoría de los Grandes Números y la Complejidad Algorítmica. Constituye un elemento fundamental de la criptografía los algoritmos de cifrado y sus diferentes tipos. (18)

Algoritmos Criptográficos

Modifican los datos de un documento con el objetivo de alcanzar algunas características de seguridad como autenticación, integridad y confidencialidad. Basan su funcionamiento en funciones matemáticas usadas en los procesos de encriptación y desencriptación, estos procesos o métodos son llamados cifrado y descifrado. Todos los algoritmos modernos basan su seguridad en la utilización de llaves; y un mensaje solo puede ser desencriptado si la llave utilizada para desencriptar coincide con la utilizada para encriptar. (19)

Existen dos grupos de criptosistemas: los algoritmos que utilizan una única clave tanto en el proceso de cifrado como en el de descifrado y los que utilizan una clave para cifrar mensajes y una clave distinta para descifrarlos. Los primeros se denominan sistemas simétricos o de llave simétrica y son la base de los algoritmos de cifrado clásico. Los segundos se denominan sistemas asimétricos, de llave pública y forman el núcleo de las técnicas de cifrado modernas. (19)

Algoritmos simétricos

Los algoritmos de clave simétrica, también llamados de clave secreta o privada, son los algoritmos clásicos de encriptación en los cuales un mensaje es encriptado utilizando para ello una cierta clave sin la cual no puede recuperarse el mensaje original. (19)

Algoritmos asimétricos

Los algoritmos asimétricos poseen dos claves diferentes en lugar de una, denominadas clave privada y clave pública. Una de ellas se emplea para codiciar, mientras que la otra se usa para decodificar. Dependiendo de la aplicación que le demos al algoritmo, la clave pública será la de cifrado o viceversa. Para que estos criptosistemas sean seguros también ha de cumplirse que a partir de una de las claves resulte extremadamente difícil calcular la otra. (19)

Hash (o funciones de resumen)

Hash se refiere a una función o método para generar claves o llaves que representen de manera casi unívoca a un documento o archivo; garantizando su integridad. Son usados en múltiples aplicaciones, como en la criptografía, el procesamiento de datos y las firmas digitales, entre otras. Una buena función de hash es una que experimenta pocas colisiones en el conjunto esperado de entrada; es decir que se podrán identificar unívocamente las entradas. (20)

Entre los algoritmos simétricos podemos encontrar:

Algoritmos simétricos
Data Encryption Standard (DES)
Advanced Encryption Standard o Algoritmo de Rijndael (AES)
International Data Encryption Algorithm (IDEA)
Blowfish
Twofish
RC4 o ARC4

Tabla 1

Entre los algoritmos asimétricos podemos encontrar:

Algoritmos asimétricos
Rivest, Shamir y Adleman (RSA)
Diffie-Hellman
ElGamal
Rabin
Digital Signature Algorithm (DSA)

Tabla 2

Entre las funciones hash podemos encontrar:

Algoritmos hash
Message-Digest Algorithm 5 (MD5)
Secure Hash Algorithm (la familia SHA)
DSA

Tabla 3

1.8 Conclusiones del capítulo

- ✓ Mediante el estudio de los tipos de licencias de software, se determinó que existen cuatro tipos de licencias: Propietaria, Shareware, Freeware y Software libre.
- ✓ A partir del estudio de los software Kaspersky y Segurmática se pudo concluir que los productos informáticos para su comercialización están acompañado de una licencia, lo que permite mayor control de su utilización.
- ✓ Los tipos de software que tienen licencias por cantidad de ordenadores son utilizados por grandes y medianas empresas.
- ✓ La metodología seleccionada para el desarrollo del módulo de licenciamiento por cantidad de computadoras es RUP, la cual permite llevar una vasta documentación de los procesos que se realicen durante el desarrollo del producto.
- ✓ El estudio de la criptografía permitió conocer los diferentes algoritmos criptográficos y su importancia para controlar las acciones de modificación a las que son sometidas las licencias de software.

CAPÍTULO 2. ANÁLISIS Y DISEÑO DEL MÓDULO DE LICENCIAMIENTO

En este capítulo se abordan los aspectos relacionados con el dominio de la investigación y la propuesta del módulo de licenciamiento por cantidad de ordenadores. Se crea el diagrama de modelo de dominio. Se plantean los requerimientos funcionales y no funcionales para el funcionamiento del módulo. Se abordan los diagramas de casos de uso del sistema y los prototipos de interfaz. Se fundamenta el uso de los patrones de diseño y se modela el diagrama de clases del diseño.

2.1 Propuesta del módulo de licenciamiento por cantidad de ordenadores

Luego de un arduo estudio y de un profundo proceso investigativo acompañado del intercambio con especialistas del centro CISED y la empresa Segurmática (Anexo 1), se ha llegado a la maquetación del resultado que se quiere obtener con la problemática propuesta, la que queda estructurada de la forma siguiente: aplicación de escritorio y módulo licensure.

La aplicación de escritorio llamada **Generador de Licencia**, tiene como función principal generar un código de licencia mediante la introducción del nombre de la entidad y la cantidad de ordenadores de un cliente determinado, el código de licencia generado se exportará a un archivo (licencia_grhs) el cual se le entregará al cliente para que introduzca la licencia en el servidor de GRHS de su entidad. El módulo **licensure** estará integrado al servidor de GRHS, siendo el encargado de controlar la cantidad de ordenadores que se puedan conectar, inicialmente esa cantidad estará en cero ordenadores, actualizándose en el momento de la inserción del código de licencia, validando simultáneamente si la licencia es válida o no. Para los autores de esta tesis resultó, para una mejor referencia, el nombrar como **Módulo de Licenciamiento** a los productos: generador de licencia y módulo licensure.

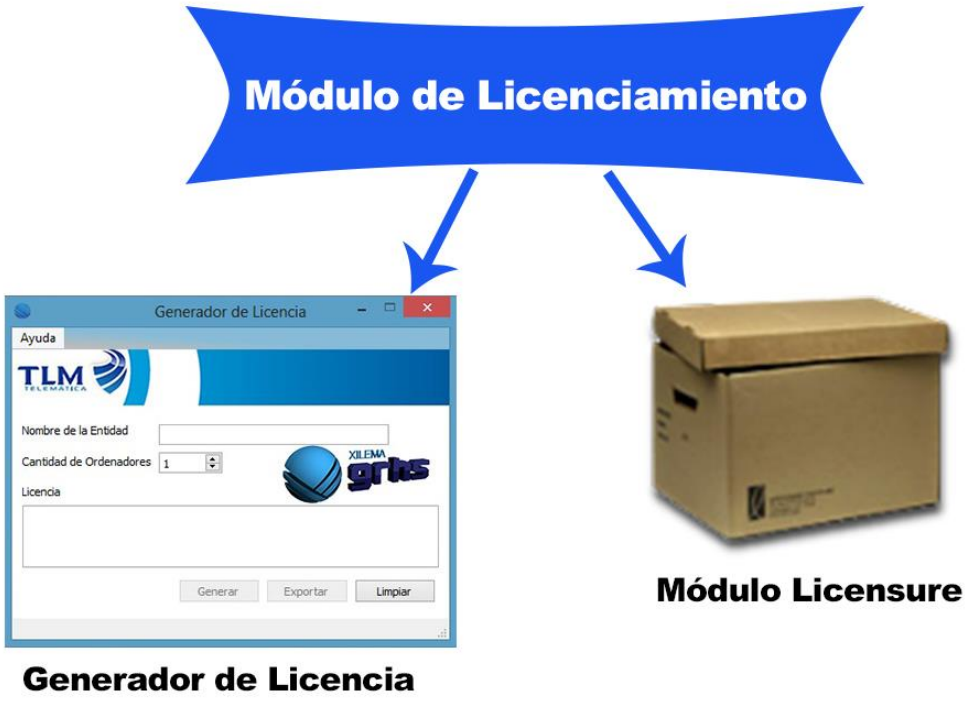


Figura 1. Componentes del módulo de licenciamiento

Son esenciales en estos procesos los algoritmos criptográficos y de codificación, ya que la información que se manejará es muy sensible, y se debe garantizar un buen funcionamiento del software GRHS.

Del estudio realizado sobre los algoritmos en función de esta investigación los autores de la tesis son del criterio de que los algoritmos AES, MD5 y la codificación Base64 deben ser los que prevalezcan en la encriptación utilizada en el Generador de Licencias. La librería PyCrypto v2.6.1, es una de las librerías criptográficas de Python más fiable, soportando funciones para cifrado por bloques, cifrado por flujo y cálculo de hash. Además incorpora sus propios generadores de números aleatorios, por ello los autores la utilizaron en la implementación.

Las causas que llevaron a los autores a tomar esta decisión son las siguientes:

AES

Es un esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos. Se espera que sea usado en el mundo entero y analizado exhaustivamente, como fue el caso de su predecesor, el DES.

Es un algoritmo simétrico de cifrado por bloques, donde las longitudes del bloque y de la clave son variables. Los bloques adoptados por el estándar son de 128 bits (no de 64 bits como el DES) y las llaves de 128, 192 o 256 bits (no 56 bits como el DES o 112 bits como el Triple DES). (19)

Ventajas:

- ✓ AES es rápido tanto en software como en hardware, es relativamente fácil de implementar, y requiere poca memoria.
- ✓ Posee longitud de clave variable.
- ✓ Combina la seguridad-eficiencia-velocidad, sencillez y flexibilidad.

MD5

Crea, a partir de un texto cuyo tamaño es elegido al azar, una huella digital de 128 bits procesándola en bloques de 512bits. Es común observar documentos descargados de Internet que vienen acompañados por archivos MD5, este es el hash del documento que hace posible verificar su integridad. (19)

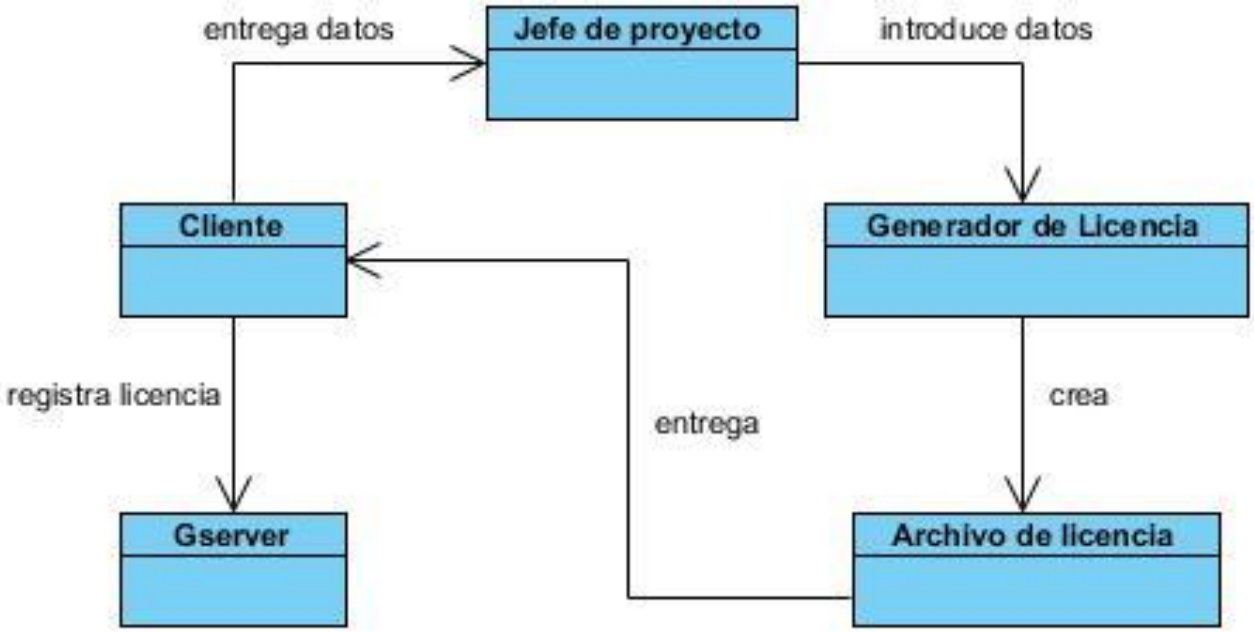
Base64

Es un sistema de numeración posicional que usa 64 como base. Es la mayor potencia de dos que puede ser representada usando únicamente los caracteres imprimibles de ASCII. Esto ha propiciado su uso para codificación de correos electrónicos, PGP y otras aplicaciones. Todas las variantes famosas que se conocen con el nombre de Base64 usan el rango de caracteres A-Z, a-z y 0-9 en este orden para los primeros 62 dígitos, pero los símbolos escogidos para los últimos dos dígitos varían considerablemente de unas a otras. (21)

2.2 Modelo de dominio

Siguiendo la metodología RUP, un proceso de vital de importancia al inicio del proyecto es la modelación del negocio, pero en este caso al no identificarse procesos de negocio con claridad los autores de esta tesis deciden realizar el modelo de dominio.

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. (22)



Esquema 1. Modelo de dominio del módulo de licenciamiento

Para brindar una mejor comprensión del Esquema 1: modelo de dominio del módulo de licenciamiento, a continuación se realiza una breve descripción de los conceptos tratados:

- ✓ **Cliente:** El concepto hace referencia a una entidad cualquiera, que esté interesada en adquirir el software GRHS para usarlo en su red de ordenadores.

- ✓ **Jefe de proyecto:** El concepto hace referencia a la persona que interactúa con el cliente en el momento de pedirle los datos y generar la licencia.
- ✓ **Generador de Licencia:** El concepto hace referencia a la aplicación de escritorio que será la encargada de generar el código de licencia que se guardará en un archivo (licencia_grhs).
- ✓ **Archivo de licencia:** El concepto hace referencia al archivo con la licencia resultante generado por la aplicación de escritorio.
- ✓ **Gserver:** El concepto hace referencia al servidor de GRHS, donde se registra el código de licencia que contiene el archivo (licencia_grhs), permitiendo limitar la cantidad de clientes que se puedan conectar a él.

2.3 Requisitos funcionales y no funcionales del software

Para la creación del módulo de licenciamiento se tomaron un conjunto de requisitos obtenidos como resultado del levantamiento de información. Los requisitos constituyen una característica que el sistema debe tener o una restricción que el sistema debe satisfacer en respuesta de las necesidades del cliente.

Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el software debe cumplir, más específicamente son acciones que el software debe ser capaz de realizar. En los requisitos funcionales se definen los servicios que el software debe proporcionar.

Requisitos funcionales		
RF	**Funcionalidad**	**Descripción**
1	Generar licencia.	Permite la creación de la licencia con los parámetros definidos.
2	Exportar licencia.	Permite guardar en el ordenador un archivo (licencia_grhs) con la licencia generada.
3	Limpiar campos.	Permite limpiar los campos del Generador de Licencia.
4	Insertar licencia.	Permite la inserción del código de licencia que contiene el archivo (licencia_grhs) en el servidor de GRHS, comprobando la validez de la licencia.

5	Limitar la cantidad de ordenadores conectados al servidor GRHS.	Permite controlar la cantidad de ordenadores que se conecten al servidor GRHS.

Tabla 4. Requisitos funcionales

Requisitos no funcionales

Describen aspectos del sistema que son visibles por el usuario que no incluyen una relación directa con el comportamiento funcional del software. Son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

Requisitos no funcionales para el "Generador de Licencia"		
RNF	**Categoría**	**Descripción**
1	Software	✓ Sistema operativo Linux, o Windows 7 o superior. ✓ Tener instalado WinRAR (en caso de utilizarlo en el sistema operativo Windows).
2	Hardware	✓ Se necesitan ordenadores con un mínimo de 1GB de memoria RAM. ✓ Procesador Dual-Core o superior. ✓ 500MB de espacio libre en disco duro.
3	Seguridad	✓ El acceso al **Generador de Licencia** debe estar controlado por la administración del centro, para impedir la distribución o copia no autorizada del mismo.

Tabla 5. Requisitos no funcionales para el "Generador de Licencia"

Requisitos no funcionales para el módulo "Licensure"		
RNF	**Categoría**	**Descripción**
1	Software	✓ Sistema operativo Linux, o Windows 7 o superior. ✓ Tener instalado Python v2.7. ✓ Tener instalado Django v1.4. ✓ Tener instalada la libreria pycrypto v2.6.1. ✓ Se deberá de disponer de un navegador web, este puede ser Mozilla Firefox v10.0 o versiones superiores. ✓ Servidor PostgreSQL v9.1 o superior instalado.

2	Hardware	✓ Al no realizarse algoritmos con complejidad computacional considerables en el módulo, los requisitos no funcionales de hardware están dados por los mismos que se definen para el correcto funcionamiento de Gserver en su versión actual.
3	Seguridad	✓ El servidor solo debe permitir la gestión a lo sumo de la cantidad de ordenadores definidos en la licencia.
4	Soporte	✓ Debe crearse un manual con la configuración y pasos a seguir para la correcta integración del módulo **Licensure** al Gserver.

Tabla 6. Requisitos no funcionales para el módulo "Licensure"

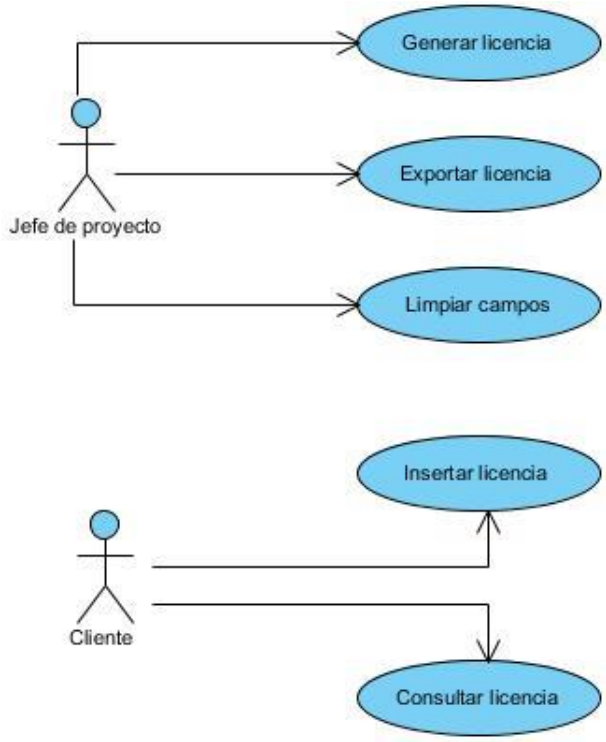
2.4 Definición de los casos de uso

Casos de uso (CU): Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema. (9)

Un diagrama de casos de uso del sistema representa de manera gráfica a los procesos y su interacción con los actores.

Actores: Un actor es una persona identificada por un rol en un sistema informatizado u organización y que realiza algún tipo de interacción con el sistema. (9)

Diagrama de casos de uso del sistema



Esquema 2.Caso de uso del sistema

Descripción de los casos de uso

CU_1 Generar licencia

Objetivo	Generar la licencia para la utilización del software GRHS.	
Actores	Jefe de proyecto	
Resumen	El Jefe de proyecto introduce los parámetros de la entidad permitiendo generar y guardar el archivo con la licencia en una ubicación del ordenador, finalmente limpia los campos de la ventana.	
Complejidad	Media	
Prioridad	Alta	
Flujo Normal de Eventos		
Flujo básico < Generar licencia >		
	Acción del Actor	**Respuesta del Sistema**

1	Establece los parámetros de la entidad: Nombre de la Entidad y Cantidad de Ordenadores, y presiona el botón Generar.	
2		Genera un código de licencia, mostrándolo en la interfaz.
3	Presiona el botón Exportar.	
4		Muestra una ventana permitiendo guardar el código de licencia dentro de un archivo en una ubicación del ordenador.
5	Selecciona la ubicación del ordenador donde desea guardar el archivo con la licencia y presiona el botón Guardar.	
6		Guarda el archivo con la licencia en la ubicación especificada.
7	Presiona el botón Limpiar.	
8		Limpia los campos para generar otras licencias.
9		Termina el caso de uso.

Tabla 7. CU_1 Generar licencia

CU_2 Insertar licencia

Objetivo	Insertar el código de la licencia en el servidor de GRHS, comenzando a ser uso de ella y observa la información de la licencia.	
Actores	Cliente	
Resumen	El cliente inserta el código de la licencia en el servidor de GRHS y observa la información que contiene. El servidor comprueba la validez de la licencia, si es válida el cliente configura los ordenadores que se conectaran al servidor, controlando este la cantidad a conectarse.	
Complejidad	Media	
Prioridad	Alta	
Flujo Normal de Eventos		
Flujo Básico < Insertar Licencia >		
	Acción del Actor	**Respuesta del Sistema**

1	Selecciona la opción Licencia en la barra de herramientas de la página principal.	
2		Se despliega una ventana con las opciones: • Insertar • Información
3	Si selecciona la opción Insertar, ver **Alternativa 1 "Insertar".** Si selecciona la opción Información, ver **Alternativa 2 "Información".**	
4		Termina el caso de uso.
Flujos Alternos		
Alternativa 1 "Insertar"		
	Acción del Actor	**Respuesta del Sistema**
1	Presiona Insertar.	
2		Muestra una ventana permitiendo la inserción del código de la licencia.
3		Verifica si la licencia es válida. Si la licencia no es válida, ver **Alternativa 3 "Licencia no válida".** Si la licencia es válida, ver **Alternativa 2 "Información"**, respuesta del sistema.
4		Controla la cantidad de ordenadores que deben conectarse al servidor de GRHS. Si la cantidad de ordenadores supera la definida en la licencia, no se podrán conectar al servidor de GRHS.
5		**Paso 4** del Flujo básico.
Alternativa 2 "Información"		
	Acción del Actor	**Respuesta del Sistema**
1	Presiona Información.	
2		Muestra una ventana con la información de la licencia: Nombre de la Entidad y Cantidad

		de Ordenadores que admitirá el servidor de GRHS.
3		**Paso 4** del Flujo básico.
Alternativa 3 "La licencia no es válida"		
	Acción del Actor	**Respuesta del Sistema**
1		Muestra una ventana con el mensaje: "Licencia no válida".
2		**Paso 2** de **Alternativa 1 "Insertar"**.

Tabla 8. CU_2 Insertar licencia

2.5 Prototipo de interfaz para la generación de licencias

Las características de un prototipo de interfaz para la generación de licencias debe cumplir los requisitos siguientes:

1. Agradable a la vista del usuario, por ello no se debe utilizar combinación de colores fuertes, que no permita leer la información que brinda.
2. Identificación mediante el logotipo de la entidad que lo desarrolla y del software, permitiendo una distinción inmediata en relación con otros tipos de software.
3. Una interacción precisa y explicativa para el usuario.
4. Los campos constitutivos de la interfaz deben contener la identificación de la entidad, la cantidad de ordenadores, entre otros elementos.

Un ejemplo que reune los requisitos anteriormente señalados es el siguiente:

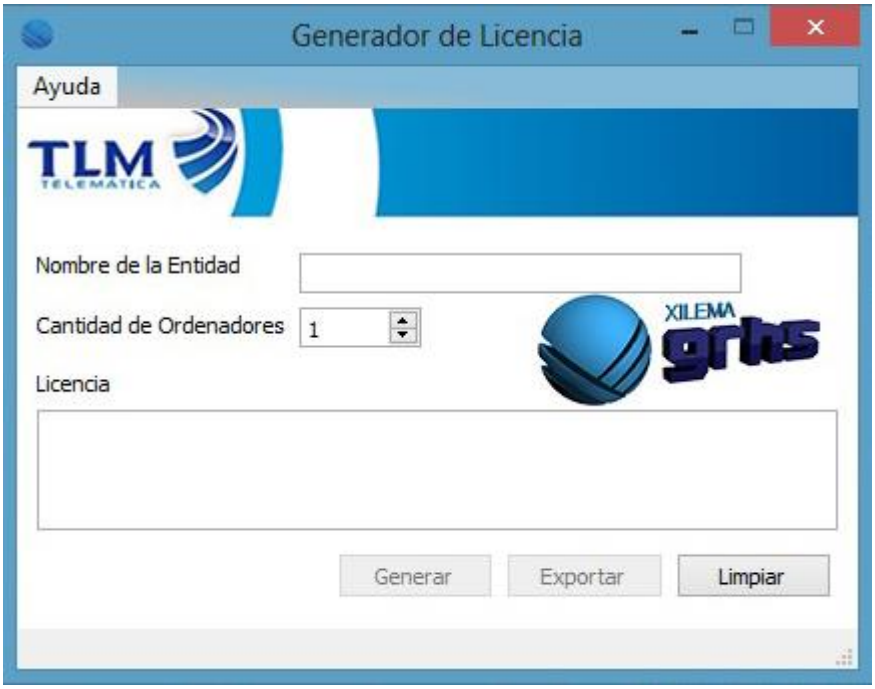


Figura 2. Generador de Licencia

2.6 Fundamentación del uso de patrones de diseño

De manera general un patrón es un modelo a seguir para la realización de una actividad. Existen diversos tipos de patrones en correspondencia al sistema que se desee implementar. En esta investigación serán abordados los Patrones Generales de Software para Asignación de Responsabilidades (GRASP): Experto, Creador, Alta Cohesión, Bajo Acoplamiento que fueron utilizados en el desarrollo del módulo de licenciamiento.

Los autores de este trabajo de diploma comparten el criterio del autor Christopher Alexander al expresar acerca de los patrones que. “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma.” [2]

[2] Christopher Alexander: Autor del libro “El lenguaje de Patrones”, ejemplar sobre arquitectura, escrito en 1977 junto a Sara Ishikawa y Murray Silverstein.

Los patrones GRASP agrupan diferentes tipos entre los que se encuentran los siguientes:

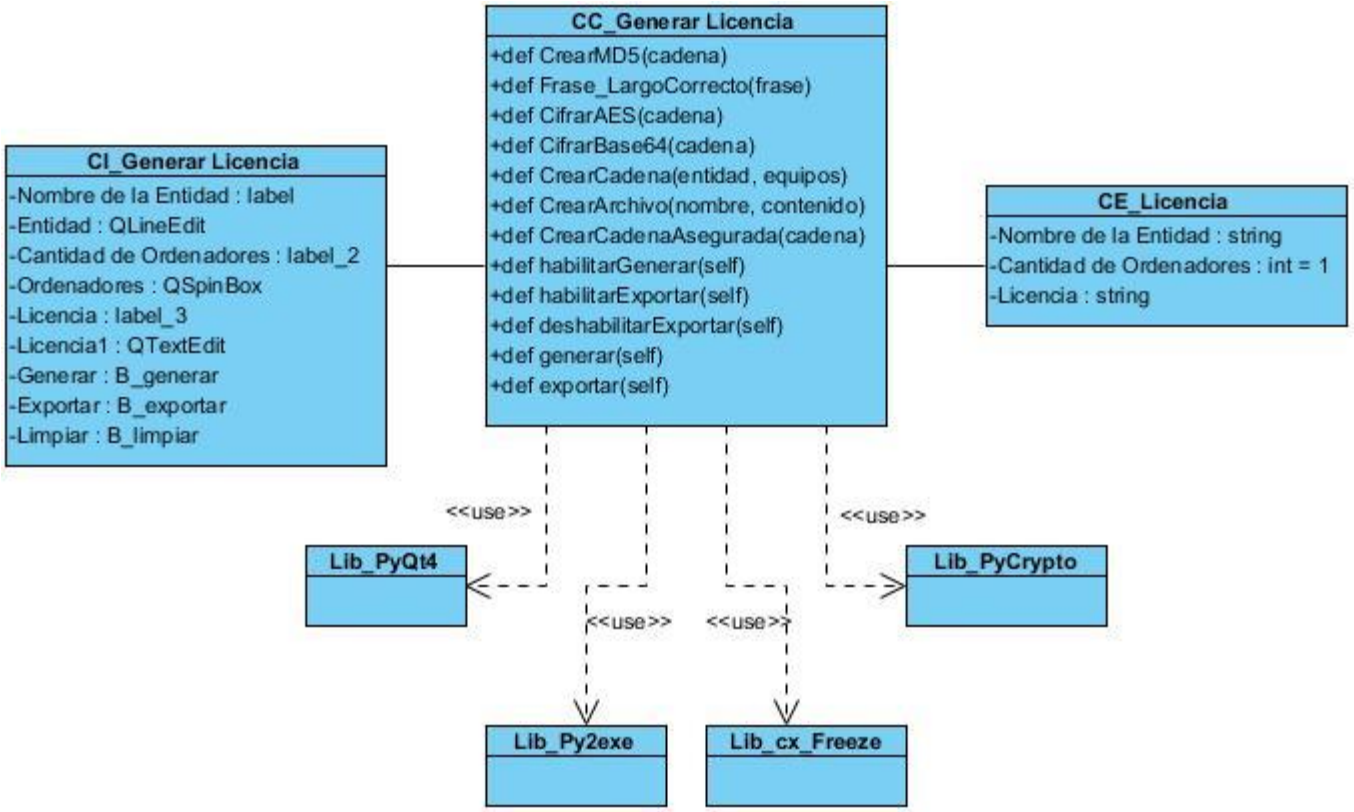
- ✓ **Experto:** Propone que la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados, es decir, cuando los objetos utilizan su propia información para llevar a cabo sus tareas. Se utiliza en las clases modelos de Django las cuales son expertas en la información de la base de datos.
- ✓ **Creador:** Ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase. Se utiliza en la solución, en la clase Register del módulo licensure donde se instancian las clases modelos para almacenar información en la base de datos.
- ✓ **Alta cohesión:** Expresa que la información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible relacionada con la clase. Este patrón se evidencia en la clase util del módulo licensure, la cual se encarga de realizar todos los procesos de descifrado.
- ✓ **Bajo acoplamiento:** Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. *(23)* Este patrón está presente en la solución debido a que las clases implementadas tienen un número pequeño de dependencias entre sí.

2.6 Diagrama de clases del diseño

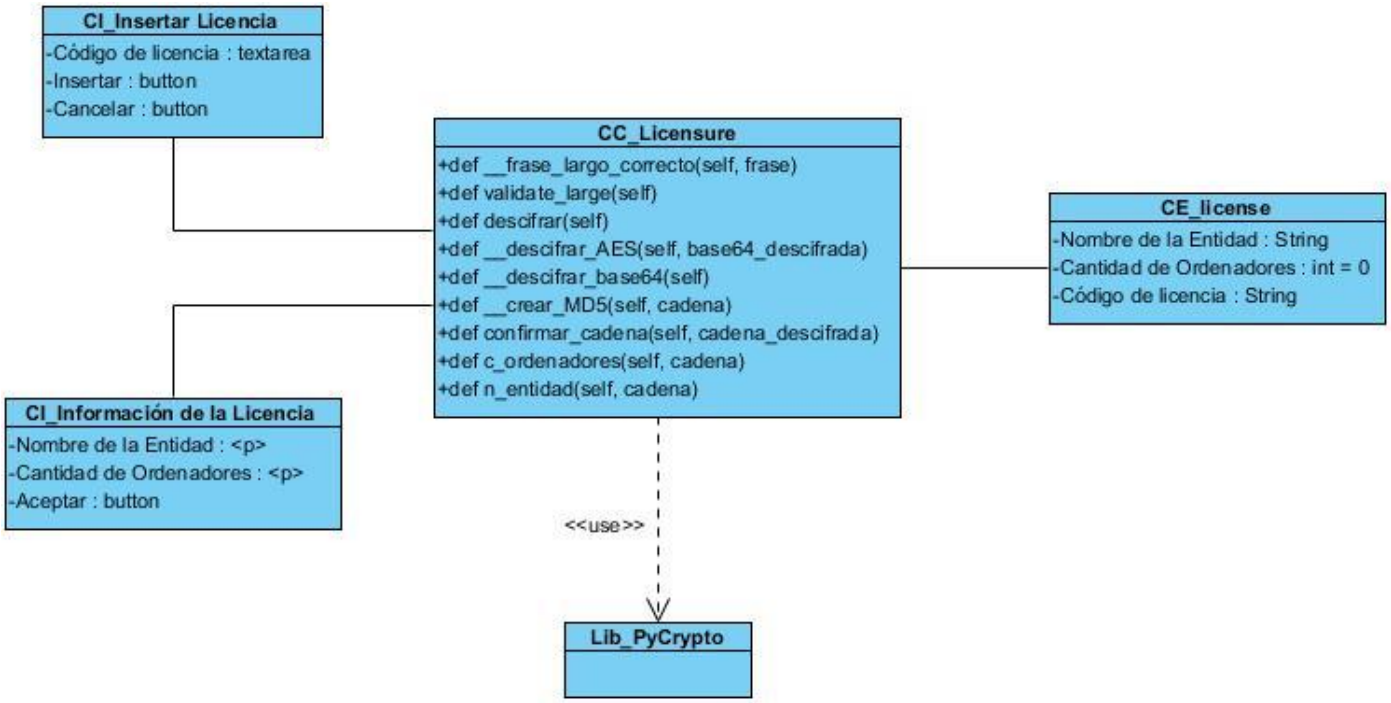
El diseño de software representa un papel importante en el desarrollo de aplicaciones informáticas y permite producir varios modelos del sistema o producto que se va a diseñar. En el diseño se modela el sistema de forma tal que sea capaz de soportar todos los requisitos, ya sean los no funcionales como los funcionales. Al unísono se va definiendo la arquitectura. En este modelo, las clases del diseño y sus objetos dan lugar a los casos de uso, mediante los que produce el diagrama de clases del diseño.

Una clase de diseño es aquella suficientemente detallada que sirve como base para generar código fuente o lo que es lo mismo, es aquella cuya especificación es completa hasta un nivel que se pueda implementar. (24)

A continuación se muestran los diagramas de clases del diseño (DCD) correspondientes los requisitos funcionales Generar Licencia e Insertar Licencia:



Esquema 3. DCD_Generar licencia



Esquema 4. DCD_Insertar licencia

2.7 Conclusiones del capítulo

- ✓ Una buena ingeniería de software permite la fácil organización y desarrollo en el proceso de creación de software.
- ✓ Los diseños con interfaces sencillas y colores claros son más agradables a la vista de los usuarios.
- ✓ Los patrones de diseño ayudan a estandarizar el código, haciendo que su estructura sea comprensible para otros programadores.
- ✓ Un detallado diagrama de clases del diseño permite la implementación fluida del software.

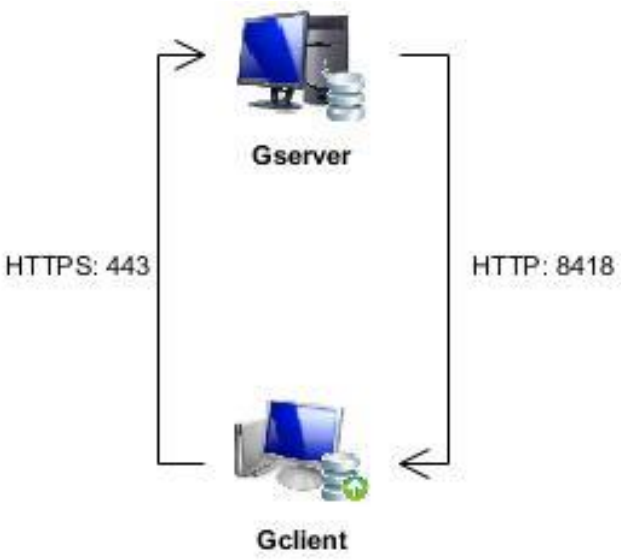
CAPÍTULO 3. IMPLEMENTACIÓN Y VALIDACIÓN DEL MÓDULO DE LICENCIAMIENTO

En este capítulo se abordan el diagrama de despliegue correspondiente al software GRHS y el diagrama de componentes correspondiente a la aplicación Gserver. Se realizan las pruebas de caja negra abordando su: objetivo, alcance y finalmente casos de prueba en los que se validará si el software funciona correctamente, a través de la obtención y solución de las no conformidades.

3.1 Diagrama de despliegue

Los diagramas de despliegue muestran la disposición física de los distintos nodos que entran en la composición de un sistema y el reparto de los programas ejecutables sobre estos nodos. Los nodos representan recursos de cómputo: procesadores o dispositivos de hardware. Los nodos se interconectan mediante soportes bidireccionales (en principio) que pueden a su vez estereotiparse. (25)

En otras palabras un diagrama de despliegue describe la arquitectura física del sistema durante la ejecución, en términos de procesadores, dispositivos y componentes de software.



Esquema 5. Diagrama de despliegue de GRHS

El esquema 5 muestra el diagrama de despliegue correspondiente al software GRHS, donde existirá un servidor (Gserver) que al introducirle el código de licencia podrá reconocer tantos clientes (Gclient) como se

haya definido. La comunicación que se establecerá entre servidor y clientes será haciendo uso del protocolo HTTP por el puerto 8418 y la que se establecerá entre los clientes y el servidor será haciendo uso del protocolo HTTPS por el puerto 443. La numeración de los puertos puede variar en dependencia de lo que determine la entidad donde se despliegue el software GRHS.

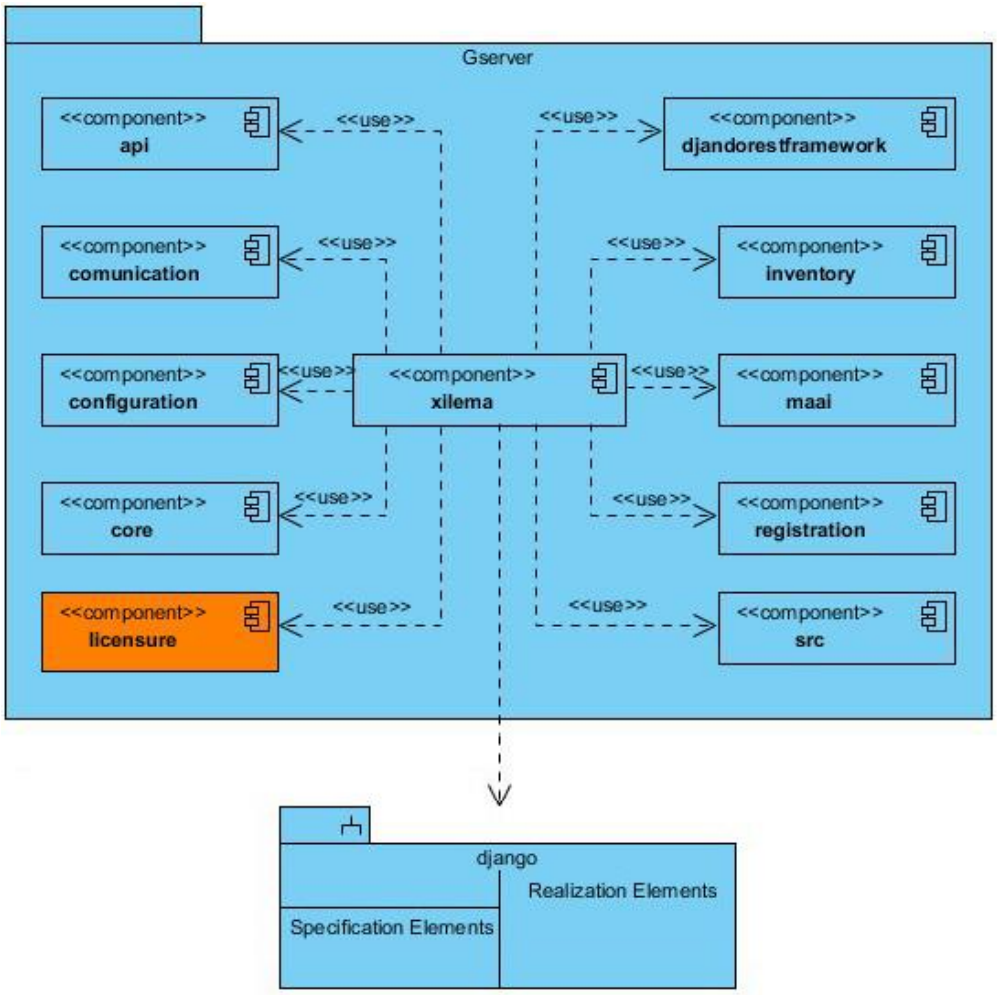
3.2 Diagrama de componentes

A partir de la búsqueda realizada sobre los diagramas de componentes los autores asumen los criterios de los Ing. Yaniel Alfredo Velázquez Bruceta y Sandy Fernando Pérez Matamoros cuando en su trabajo de diploma expresan que:

"El diagrama de componentes es utilizado para modelar los componentes de un sistema, incluyendo los artefactos que implementan dichos componentes. Un componente es la implementación física de un conjunto de elementos lógicos, como por ejemplo las clases. Un diagrama de componentes muestra la estructura de alto nivel del modelo de implementación, especificando los subsistemas de implementación y sus dependencias a la hora de importar código." [3]

Sobre esta base se realizó el diseño del diagrama de componentes correspondiente al módulo de licenciamiento, el que se expresa en el esquema siguiente:

[3] Ing. Yaniel Alfredo Velázquez Bruceta; Ing. Sandy Fernando Pérez Matamoros. Despliegue Jerárquico de Servidores de XILEMA GRHS. Trabajo de Diploma UCI 2014.



Esquema 6. Diagrama de componentes de Gserver

El esquema 6 muestra el diagrama con todos los componentes de Gserver, entre los cuales se ha resaltado **licensure**, que es el nombre del módulo donde se encuentra una parte de la implementación de la solución al problema de investigación.

El resto de los componentes realizan funciones específicas las que se describen a continuación:

- ✓ **api:** contiene implementaciones por medio de las cuales el administrador puede consultar información del servidor a través de la interfaz web Gserver.
- ✓ **comunication:** contiene las implementaciones que permiten la comunicación cliente - servidor.

- ✓ **configuration:** es la encargada de gestionar todas las configuraciones de las aplicaciones de GRHS.
- ✓ **core:** contiene las implementaciones sobre las cuales está implementado el núcleo del servidor.
- ✓ **djangorestframework:** contiene funcionalidades que se reutilizan en los demás componentes haciendo posible el uso de servicios web de tipo REST con Django.
- ✓ **inventory:** contiene las implementaciones encargadas de almacenar la información que llega desde los clientes en la base de datos.
- ✓ **maai:** es el componente de acciones ante incidencias, encargado de realizar las acciones configuradas por el administrador cuando los clientes reportan incidencias en las estaciones de trabajo. Entre las posibles acciones se encuentra la notificación vía correo al administrador.
- ✓ **registration:** contiene las implementaciones mediante las cuales Gserver registra la información de los agentes que están activos, su dirección IP, puerto de escucha, identificador del medio básico, etc.
- ✓ **src:** es el componente donde se encuentra implementada la interfaz web del sistema Gadmin.

3.3 Validación de los resultados obtenidos

Durante el desarrollo del software la etapa de pruebas es esencial para lograr la obtención de un producto final eficiente. Es aquí donde se descubren los errores latentes escondidos que pueden degradar el funcionamiento del sistema, por tal motivo es fundamental su realización ya que permite tener una seguridad de que las especificaciones del diseño y el código cumplan con lo establecido.

Prueba de caja negra

El objetivo de realizar este método de prueba al sistema es para detectar el incorrecto o incompleto funcionamiento de este, así como los errores de interfaces, rendimiento y errores de inicialización y terminación.

El proceso de prueba de caja negra se va a centrar principalmente en los requisitos funcionales del software para verificar el comportamiento de la interfaz gráfica, su interacción con el usuario y la calidad funcional. Esta prueba permite obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del programa. Dentro de las técnicas de Prueba de Caja Negra, se utilizó la **Partición**

equivalente. Esta técnica se basa en la división del campo de entrada en un conjunto de clases de datos denominadas clases de equivalencia. Mediante la aplicación de esta se examinó los valores válidos e inválidos de las entradas existentes en el software, descubriendo de forma inmediata clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar. (26)

Para definir las clases de equivalencia se tuvieron en cuenta un conjunto de reglas:

- ✓ Si una condición de entrada especifica un rango, entonces se confeccionan una clase de equivalencia válida y dos inválidas.
- ✓ Si una condición de entrada especifica la cantidad de valores, se identifica una clase de equivalencia válida y dos inválidas.
- ✓ Si una condición de entrada especifica un conjunto de valores de entrada y existen razones para creer que el programa trata en forma diferente a cada uno de ellos, se identifica una clase válida para cada uno de ellos y una clase inválida.
- ✓ Si una condición de entrada especifica una situación de tipo “debe ser”, se identifica una clase válida y una inválida. (9)

Luego de tener las clases válidas e inválidas definidas, se procedió a definir los casos de pruebas, para los requisitos más significativos. A continuación se muestra las pruebas realizadas a los requisitos Generar licencia e Insertar licencia. Cada uno de estos casos de pruebas se definió teniendo en cuenta lo siguiente:

a) Escribir un nuevo caso de prueba que cubra tantas clases de equivalencia válidas no cubiertas como sea posible hasta que todas las clases de equivalencia hayan sido cubiertas por casos de prueba.
b) Escribir un nuevo caso de prueba que cubra una y solo una clase de equivalencia inválida hasta que todas las clases de equivalencias inválidas hayan sido cubiertas por casos de pruebas.

Condiciones de ejecución:

Ejecutar el Generador de Licencia.

SC<Generar licencia**>**

Escenario	Descripción	Nombre de la Entidad	Cantidad de Ordenadores	Respuesta del sistema	Flujo central
EC1. Generar licencia sin haber introducido el nombre de la entidad.	Se muestra la interfaz del Generador de Licencia, permitiendo la inserción de los datos de la entidad.	I	V 1	El botón Generar se encuentra deshabilitado.	1-El Jefe de proyecto introduce los datos de la entidad y presiona el botón Generar. 2-Presiona el botón Exportar y selecciona la ubicación donde desee guardar la licencia. 3-Presiona el botón Limpiar si desea generar nuevas licencias.
EC2. Generar licencia con los campos llenos.	Se muestra la interfaz del Generador de Licencia, permitiendo la inserción de los datos de la entidad.	V Etecsa	V 50	El botón Generar se habilita.	1-El Jefe de proyecto introduce los datos de la entidad y presiona el botón Generar. 2-Presiona el botón Exportar y selecciona la ubicación donde desee guardar la licencia. 3-Presiona el botón Limpiar si desea generar nuevas licencias.
EC3. Exportar licencia.	Se muestra la interfaz del Generador de Licencia, permitiendo exportar la llicencia.	V Etecsa	V 50	Luego de haberse presionado el botón Generar, se habilita el botón Exportar.	1-El Jefe de proyecto introduce los datos de la entidad y presiona el botón Generar. 2-Presiona el botón Exportar y selecciona la ubicación donde desee guardar la licencia. 3-Presiona el botón Limpiar si desea generar nuevas licencias.

EC4. Limpiar los campos del Generador de Licencia.	Se muestra la interfaz del Generador de Licencia, permitiendo limpiar los campos.	NA	NA	Limpia los datos que se encuentren en el generador.	1-Presiona el botón Limpiar si desea generar nuevas licencias.
EC5. Dejar espacios en blanco al final del nombre de la entidad.	Se muestra la interfaz del Generador de Licencia, permitiendo la inserción de los datos de la entidad.	V Etecsa__	V 50	El botón Generar se habilita.	1-El Jefe de proyecto introduce los datos de la entidad y presiona el botón Generar. 2-Presiona el botón Exportar y selecciona la ubicación donde desee guardar la licencia. 3-Presiona el botón Limpiar si desea generar nuevas licencias.
EC5. Dejar espacios en blanco al comienzo del nombre de la entidad.	Se muestra la interfaz del Generador de Licencia, permitiendo la inserción de los datos de la entidad.	V __Etecsa	V 50	El botón Generar se habilita.	1-El Jefe de proyecto introduce los datos de la entidad y presiona el botón Generar. 2-Presiona el botón Exportar y selecciona la ubicación donde desee guardar la licencia. 3-Presiona el botón Limpiar si desea generar nuevas licencias.

Tabla 9. SC<*Generar licencia****>***

Condiciones de ejecución:

Poseer un código de licencia.

SC<Insertar licencia**>**

Escenario	Descripción	Código de licencia	Respuesta del sistema	Flujo central
EC1. Presiona el botón insertar sin haber introducido la licencia.	Se muestra la interfaz Insertar licencia.	I	Se muestra un mensaje de error.	1-El cliente selecciona en la interfaz principal la opción Licencia y selecciona Insertar.
EC2. Presiona el botón insertar luego de haber introducido una serie de caracteres.	Se muestra la interfaz Insertar licencia.	I C*/sdjfhk-54ieie-dsolk/sd	Se muestra un mensaje de error y se borran los datos entrados.	1-El cliente selecciona en la interfaz principal la opción Licencia y selecciona Insertar.
EC3. Presiona el botón insertar luego de haber introducido una licencia válida.	Se muestra la interfaz Insertar licencia.	V iBa7oP+Go0enTawWzHn3s8Q320YxQwSi/TGwJGArlYGiPFEVdTChDOOU5svhyeSQU0EYT6hwVxwTZk/pIZajYux9qPGPweLPGj4W1TvgECkBQHMhsq	Se muestra la interfaz Información de licencia, con el nombre de la entidad y la cantidad de ordenadores a la que pertenece la licencia.	1-El cliente selecciona en la interfaz principal la opción Licencia y selecciona Insertar.

		LVuCXBLY8Kub9C		
EC4. Seleccion a la opción Información de licencia sin haber introducido alguna.	Se muestra la interfaz Información de licencia.	NA	Se muestra la interfaz Información de licencia con los campos Nombre de la entidad y Cantidad de ordenadores vacíos.	1- El cliente selecciona en la interfaz principal la opción Licencia y selecciona Información.
EC5. Selecciona la opción Información de licencia luego de introducir una licencia valida.	Se muestra la interfaz Información de licencia.	V iBa7oP+Go0enTawWzHn3s8Q320YxQwSi/TGwJGArlYGiPFEVdTChDOOU5svhyeSQU0EYT6hwVxwTZk/pIZajYux9qPGPweLPGj4W1TvgECkBQHMhsqLVuCXBLY8Kub9C	Se muestra la interfaz Información de licencia con los campos Nombre de la entidad: ETECSA y Cantidad de ordenadores: 50.	1- El cliente selecciona en la interfaz principal la opción Licencia y selecciona Información.

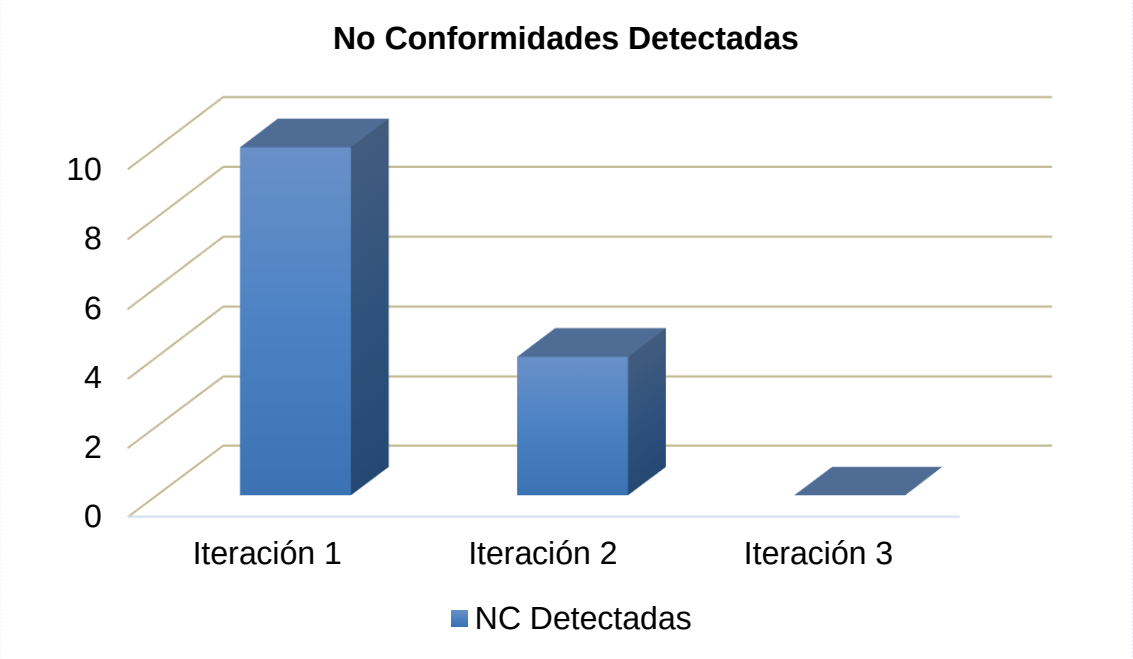
Tabla 10. SC<*Insertar licencia****>***

Leyenda:

V: Datos válidos. **I:** Datos inválidos. **NA:** No aplica. __: Espacios en blanco.

No conformidades detectadas

A continuación se muestra gráficamente un resumen de las no conformidades detectadas al sistema, utilizando los casos de prueba diseñados:



Gráfica 1. No conformidades detectadas

Entre las principales no conformidades encontradas se pueden mencionar:

- ✓ No se muestran las tildes en la interfaz “Acerca de” del Generador de Licencia.
- ✓ Incorrecta validación del formato de entrada de datos en el campo nombre de la entidad del Generador de Licencia.
- ✓ Problemas al exportar el archivo con la licencia.
- ✓ No se muestra el fondo ni el logotipo del Generador de Licencia.
- ✓ Problemas en la validación de la longitud del código de licencia en el módulo licensure.
- ✓ Al insertar el código de licencia, no se muestra la información que contiene la licencia en la ventana Información.

- ✓ En la ventana Insertar, al introducir el código de licencia, se mantiene la licencia al cerrarse y volver a abrir la ventana Insertar.

Las no conformidades encontradas fueron resueltas satisfactoriamente. Esto no evidencia que la aplicación se encuentre libre de posibles errores pero si se resolvieron todas las no conformidades detectadas durante las tres iteraciones realizadas.

El producto que consta del Generador de Licencia y el módulo Licensure, fueron entregados al proyecto GRHS donde se avaló su buen funcionamiento (Anexo 2).

3.4 Conclusiones del capítulo

- ✓ El diagrama de componentes ayuda a un mejor entendimiento de los conceptos relacionados con el software Gserver.
- ✓ Las pruebas de software son un mecanismo primordial para comprobar el buen funcionamiento del producto.
- ✓ Las no conformidades detectadas en cada iteración de las pruebas ayudaron a una mejor corrección del producto.

CONCLUSIONES

A partir del desarrollo de la presente investigación se arribaron a las siguientes conclusiones:

- ✓ Luego de realizar un estudio de los principales conceptos asociados a los procesos de licencias de software, se logró adquirir mayor entendimiento de las mismas.
- ✓ El análisis de las herramientas similares permitió identificar características que sirvieron de guía en la solución.
- ✓ Se realizó un estudio de las herramientas y tecnologías necesarias utilizadas en el desarrollo del módulo de licenciamiento.
- ✓ Permitió comprender que toda licencia debe tener métodos y estructuras de seguridad para que no sea fácilmente corrompida y modificada, para esto es fundamental la utilización de algoritmos criptográficos.
- ✓ La interfaz obtenida del Generador de Licencia es sencilla, posibilitando el fácil entendimiento y la interacción con el usuario sin que afecte su eficiencia con respecto al resultado que se quiere obtener.
- ✓ Luego de haber realizado la implementación del módulo de licenciamiento por cantidad de ordenadores, se obtuvo como resultado el producto deseado.
- ✓ Las pruebas de software permitieron comprobar el buen funcionamiento del producto, al aportar las no conformidades detectadas en cada iteración ayudando a una mejor corrección de su calidad.

Por todo lo anteriormente expuesto, se concluye que el objetivo propuesto para el presente trabajo de diploma se ha cumplido satisfactoriamente, poniendo en práctica todas y cada una de las tareas propuestas para el desarrollo del módulo de licenciamiento, compuesto por: el generador de licencia y el módulo licensure.

RECOMENDACIONES

- ✓ Elaborar el licenciamiento por tiempo para el software GRHS.
- ✓ Seguir perfeccionando los mecanismos utilizados para generar la licencia, con el fin de crear una cadena lo más robusta posible.
- ✓ Que la interfaz de Gserver permita la inserción de un archivo que contenga el código de la licencia.

REFERENCIAS BIBLIOGRÁFICAS

1. ***LINEAMIENTOS DE LA POLÍTICA ECONÓMICA Y SOCIAL DEL PARTIDO Y LA REVOLUCIÓN.*** **La Habana : s.n., 2011.**

2. Ecured Portable versión 1.5. ***Licencias Comerciales de los Software.*** **20 de Noviembre de 2014.**

3. NetSupport Manager. [En línea] [Citado el: 6 de Noviembre de 2014.] http://www.netsupportmanager.com/es/features.asp.

4. Everest. [En línea] [Citado el: 6 de Noviembre de 2014.] http://www.lavalys.com/products/everest-pc-diagnostics/.

5. Speccy. [En línea] [Citado el: 6 de Noviembre de 2014.] http://speccy.softonic.com/.

6. Kaspersky Small Office Security. [En línea] Kaspersky Lab, 2014. [Citado el: 17 de Noviembre de 2014.] http://www.kaspersky.es/software-antivirus-domestico/small-office-security.

7. Segurmática. Segurmática, Consultoría y Seguridad Informática. [En línea] 2014. [Citado el: 17 de Noviembre de 2014.] http://www.segurmatica.cu/laboratorio/lab9.jsp.

8. Wiki GRHS. [En línea] Centro de Telemática. [Citado el: 28 de Octubre de 2014.]

9. Pressman, Roger S. ***Ingeniería del Software: Un enfoque práctico. Quinta edición.*** **s.l. : McGraw-Hill.**

10. Metodologias de desarrollo. [En línea] [Citado el: 4 de Noviembre de 2014.] http://metodologiasdedesarrollodesoftware.blogspot.com/.

11. Sanchez, María A. Mendoza. Informatizate. [En línea] 7 de Junio de 2004. [Citado el: 4 de Noviembre de 2014.] http://informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

12. Duque, R.G. ***Python para todos.*** **España : s.n.**

13. Django. [En línea] Comunidad en español de Django. [Citado el: 3 de Noviembre de 2014.] http://www.django.es/.

14. Ecured Portable versión 1.5. ***Eclipse, entorno de desarrollo integrado.*** **20 de Noviembre de 2014.**

15. PADILLA AGUDELO , JESSE . ***Interfaces Graficas de Usuario en Python: Primeros paso en PyQT4.***

16. Visual Paradigm. [En línea] Visual Paradigm Fundation. [Citado el: 28 de Octubre de 2014.] http://www.visual-paradigm.com/support/.

17. Martínez, Rafael. PostgreSQL-es. [En línea] [Citado el: 4 de Noviembre de 2014.] http://www.postgresql.org.es/sobre_postgresql.

18. Seguridad Informática. [En línea] Cristian Borghello 2000 - 2009. [Citado el: 29 de Diciembre de 2014.] http://www.segu-info.com.ar/criptologia.

19. Lucena López, Manuel José. *Criptografía y Seguridad en Computadores. 4ª Edición v0.9.0.* Jaén : UNIVERSIDAD DE JAÉN, 2011.

20. P. Pfleeger, Charles y Lawrence Pfleeger, Shari . *Security in Computing, Fourth Edition.* s.l. : Pearson Education, Inc., 2006. pág. 800. ISBN 0-13-239077-9.

21. Kioskea. [En línea] Diciembre de 2014. [Citado el: 6 de Enero de 2015.] http://es.kioskea.net/contents/56-codificacion-base-64.

22. EcuRed. [En línea] 22 de Noviembre de 2014. [Citado el: 11 de Abril de 2015.] http://www.ecured.cu/index.php/Modelo_de_dominio.

23. Mora, Roberto Canales. AdictosAlTrabajo.com. [En línea] [Citado el: 30 de Noviembre de 2014.] http://www.adictosaltrabajo.com/tutoriales/grasp.php.

24. RUMBAUGH, James, JACOBSON, Ivar y BOOCH, Grady. *El lenguaje unificado de modelado. Manual de referencia.* s.l. : Addison Wesley, 2000.

25. Marca Huallpara, Michael Hugo y Quisbert Limachi , Susana Nancy. *ANALISIS Y DISEÑO DE SISTEMAS II - Diagrama de Despliegue.*

26. Departamento de Informática Universidad de Valladolid Campusde Segovia. *TEMA 7: VALIDACIÓN. TÉCNICAS DE PRUEBA DEL SOFTWARE.*

27. Sommerville, Ian. *Ingeniería del software. Séptima edición.* Madrid : Pearson Educación, S.A., 2005.

28. Procesos de Software. [En línea] 2014. [Citado el: 5 de Octubre de 2014.] http://procesosdesoftware.wikispaces.com/METODOLOGIAS+PARA+DESARROLLO+DE+SOFTWARE.

29. Hernández León, Rolando Alfredo y Coello González , Sayda . *EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA.* La Habana : Editorial Universitaria, 2002. ISBN: 959-16-0343-6.

30. Mateu, Carles. *Desarrollo de aplicaciones web.* Barcelona : s.n., 2004 . ISBN: 84-9788-118-4.

31. Rossum, Guido Van. *Guía de aprendizaje de Python Release 2.4.* s.l. : Python Software Foundation , 2005.

32. Pilgrim, Mark. *Inmersión en Python 3 .* 2009.

33. Fernández, David López. *Interfaces gráficas en Python .*

34. Nonius, Jorge. *Introducción a las licencias de software libre.* 2002.

35. Alsina, Roberto. ***Python no Muerde, Yo Sí.***

36. Jacobson, James Rumbaugh I. ***El Proceso Unificado de Desarrollo de Software.*** **Madrid : Pearson Educación, S.A, 2000. ISBN: 84-7829-036-2.**

37. Ecured Portable versión 1.5. ***Algoritmo criptográfico.*** **22 de Noviembre de 2014.**

38. Mejía, Julio César. Algoritmos Criptográficos. [En línea] 6 de Noviembre de 2012. [Citado el: 29 de Diciembre de 2014.] http://cripto-fi.blogspot.com.es/2012/11/diseno-de-un-algoritmo-de-criptografia.html.

39. raanglada@uci.cu. ***Desarrollo de aplicaciones de escritorio para gnu\linux Python + QT.*** **La Habana : s.n., 21 de Enero de 2009.**

40. Barrientos, Cristóbal , Rojas, Gonzalo y Sotomayor, Claudio . ***INTERFAZ CLIENTE/SERVIDOR PARA INTERCAMBIO DE INFORMACIÓN A DISTANCIA POR MEDIO DE PROTOCOLO TCP/IP.*** **2012.**

41. Departamento de Ciencias de la Computación e I.A. ***Interfaces gráficos en Qt con Qt-designer.*** **Granada : s.n., 2011.**

42. Finlay, John y Gil Sánchez, Lorenzo . ***Tutorial de PyGTK 2.0.***

43. Finlay, John , y otros. ***Tutorial de PyGTK 2.0 versión 2.3.***

44. QT Documentation. [En línea] Free Software Foundation, 2014. http://www.qt-project.org.

45. Pablog. Usar QtDesigner para actualizar nuestro QHolaMundo. [En línea] 2011. http://pablog-hg.blogspot.com.

46. Cáceres Tello , Jesús . ***Patrones de diseño: ejemplo de aplicación en los Generative Learning Object.*** **Alcalá : Dpto. Ciencias de la Computación.**

47. Tedeschi, Nicolás. ¿Qué es un Patrón de Diseño? [En línea] Microsoft, 2015. [Citado el: 10 de Febrero de 2015.] https://msdn.microsoft.com/es-es/library/bb972240.aspx.

48. rubenfa. Patrones de diseño: qué son y por qué debes usarlos. [En línea] 14 de Julio de 2014. [Citado el: 10 de Febrero de 2015.] http://www.genbetadev.com/metodologias-de-programacion/Patrones-de-diseño:-qué-son-y-por-qué-debes-usarlos.

49. Prácticas de Software . [En línea] 21 de Marzo de 2011. [Citado el: 10 de Febrero de 2015.] http://www.practicasdesoftware.com.ar/category/arquitectura/Patrones-GRASP.

50. ***Diseño Dirigido por Responsabilidades con los patrones GRASP.*** **s.l. : Pearson Educación, S.A.**

51. Matrínez Zurita, Alberto. ***Diagrama de Despliegue.***

52. Cruz Quispe, Víctor Fabio, Gutiérrez Mamani, Ever Dino y Mendivil Torrico, Luís Briam. ***Diagrama de Componentes.***

53. Python Software Foundation. Unicode HOWTO. [En línea] 2015. [Citado el: 12 de Febrero de 2015.] https://docs.python.org/2/howto/unicode.html.

54. Holovaty, Adrian y Kaplan-Moss , Jacob. ***The Definitive Guide to Django: Web Development Done Right.*** **s.l. : Apress, 2007.**

55. Hernández Sampieri , Roberto, Fernández Collado , Carlos y Baptista Lucio , María del Pilar. ***Metodología de la Investigación. Quinta edición.*** **México D.F : McGRAW-HILL / INTERAMERICANA EDITORES, S.A. DE C.V., 2010. ISBN: 978-607-15-0291-9.**

56. Diseño de casos de prueba - Procesos Prouctivos. [En línea] [Citado el: 2 de mayo de 2015.] http://www.angelfire.lycos.com/.

57. ***Impacto social de la ciencia y la tecnología en Cuba: una experiencia de medición a nivel macro.*** **Batista, Armando Rodríguez. 4, s.l. : Revista CTS (pág. 147-171), 2005, Vol. II.**

58. ***Constitución de la República de Cuba.*** **2003.**

BIBLIOGRAFÍA

1. ***LINEAMIENTOS DE LA POLÍTICA ECONÓMICA Y SOCIAL DEL PARTIDO Y LA REVOLUCIÓN.*** **La Habana : s.n., 2011.**

2. Ecured Portable versión 1.5. ***Licencias Comerciales de los Software.*** **20 de Noviembre de 2014.**

3. NetSupport Manager. [En línea] [Citado el: 6 de Noviembre de 2014.] http://www.netsupportmanager.com/es/features.asp.

4. Everest. [En línea] [Citado el: 6 de Noviembre de 2014.] http://www.lavalys.com/products/everest-pc-diagnostics/.

5. Speccy. [En línea] [Citado el: 6 de Noviembre de 2014.] http://speccy.softonic.com/.

6. Kaspersky Small Office Security. [En línea] Kaspersky Lab, 2014. [Citado el: 17 de Noviembre de 2014.] http://www.kaspersky.es/software-antivirus-domestico/small-office-security.

7. Segurmática. Segurmática, Consultoría y Seguridad Informática. [En línea] 2014. [Citado el: 17 de Noviembre de 2014.] http://www.segurmatica.cu/laboratorio/lab9.jsp.

8. Wiki GRHS. [En línea] Centro de Telemática. [Citado el: 28 de Octubre de 2014.]

9. Pressman, Roger S. ***Ingeniería del Software: Un enfoque práctico. Quinta edición.*** **s.l. : McGraw-Hill.**

10. Metodologias de desarrollo. [En línea] [Citado el: 4 de Noviembre de 2014.] http://metodologiasdedesarrollodesoftware.blogspot.com/.

11. Sanchez, María A. Mendoza. Informatizate. [En línea] 7 de Junio de 2004. [Citado el: 4 de Noviembre de 2014.] http://informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

12. Duque, R.G. ***Python para todos.*** **España : s.n.**

13. Django. [En línea] Comunidad en español de Django. [Citado el: 3 de Noviembre de 2014.] http://www.django.es/.

14. Ecured Portable versión 1.5. ***Eclipse, entorno de desarrollo integrado.*** **20 de Noviembre de 2014.**

15. PADILLA AGUDELO , JESSE . ***Interfaces Graficas de Usuario en Python: Primeros paso en PyQT4.***

16. Visual Paradigm. [En línea] Visual Paradigm Fundation. [Citado el: 28 de Octubre de 2014.] http://www.visual-paradigm.com/support/.

17. Martínez, Rafael. PostgreSQL-es. [En línea] [Citado el: 4 de Noviembre de 2014.] http://www.postgresql.org.es/sobre_postgresql.

18. Seguridad Informática. [En línea] Cristian Borghello 2000 - 2009. [Citado el: 29 de Diciembre de 2014.] http://www.segu-info.com.ar/criptologia.

19. Lucena López, Manuel José. *Criptografía y Seguridad en Computadores. 4ª Edición v0.9.0.* Jaén : UNIVERSIDAD DE JAÉN, 2011.

20. P. Pfleeger, Charles y Lawrence Pfleeger, Shari . *Security in Computing, Fourth Edition.* s.l. : Pearson Education, Inc., 2006. pág. 800. ISBN 0-13-239077-9.

21. Kioskea. [En línea] Diciembre de 2014. [Citado el: 6 de Enero de 2015.] http://es.kioskea.net/contents/56-codificacion-base-64.

22. EcuRed. [En línea] 22 de Noviembre de 2014. [Citado el: 11 de Abril de 2015.] http://www.ecured.cu/index.php/Modelo_de_dominio.

23. Mora, Roberto Canales. AdictosAlTrabajo.com. [En línea] [Citado el: 30 de Noviembre de 2014.] http://www.adictosaltrabajo.com/tutoriales/grasp.php.

24. RUMBAUGH, James, JACOBSON, Ivar y BOOCH, Grady. *El lenguaje unificado de modelado. Manual de referencia.* s.l. : Addison Wesley, 2000.

25. Marca Huallpara, Michael Hugo y Quisbert Limachi , Susana Nancy. *ANALISIS Y DISEÑO DE SISTEMAS II - Diagrama de Despliegue.*

26. Departamento de Informática Universidad de Valladolid Campusde Segovia. *TEMA 7: VALIDACIÓN. TÉCNICAS DE PRUEBA DEL SOFTWARE.*

27. Sommerville, Ian. *Ingeniería del software. Séptima edición.* Madrid : Pearson Educación, S.A., 2005.

28. Procesos de Software. [En línea] 2014. [Citado el: 5 de Octubre de 2014.] http://procesosdesoftware.wikispaces.com/METODOLOGIAS+PARA+DESARROLLO+DE+SOFTWARE.

29. Hernández León, Rolando Alfredo y Coello González , Sayda . *EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA.* La Habana : Editorial Universitaria, 2002. ISBN: 959-16-0343-6.

30. Mateu, Carles. *Desarrollo de aplicaciones web.* Barcelona : s.n., 2004 . ISBN: 84-9788-118-4.

31. Rossum, Guido Van. *Guía de aprendizaje de Python Release 2.4.* s.l. : Python Software Foundation , 2005.

32. Pilgrim, Mark. *Inmersión en Python 3 .* 2009.

33. Fernández, David López. *Interfaces gráficas en Python .*

34. Nonius, Jorge. *Introducción a las licencias de software libre.* 2002.

35. Alsina, Roberto. ***Python no Muerde, Yo Sí.***

36. Jacobson, James Rumbaugh I. ***El Proceso Unificado de Desarrollo de Software.*** **Madrid : Pearson Educación, S.A, 2000. ISBN: 84-7829-036-2.**

37. Ecured Portable versión 1.5. ***Algoritmo criptográfico.*** **22 de Noviembre de 2014.**

38. Mejía, Julio César. Algoritmos Criptográficos. [En línea] 6 de Noviembre de 2012. [Citado el: 29 de Diciembre de 2014.] http://cripto-fi.blogspot.com.es/2012/11/diseno-de-un-algoritmo-de-criptografia.html.

39. raanglada@uci.cu. ***Desarrollo de aplicaciones de escritorio para gnu\linux Python + QT.*** **La Habana : s.n., 21 de Enero de 2009.**

40. Barrientos, Cristóbal , Rojas, Gonzalo y Sotomayor, Claudio . ***INTERFAZ CLIENTE/SERVIDOR PARA INTERCAMBIO DE INFORMACIÓN A DISTANCIA POR MEDIO DE PROTOCOLO TCP/IP.*** **2012.**

41. Departamento de Ciencias de la Computación e I.A. ***Interfaces gráficos en Qt con Qt-designer.*** **Granada : s.n., 2011.**

42. Finlay, John y Gil Sánchez, Lorenzo . ***Tutorial de PyGTK 2.0.***

43. Finlay, John , y otros. ***Tutorial de PyGTK 2.0 versión 2.3.***

44. QT Documentation. [En línea] Free Software Foundation, 2014. http://www.qt-project.org.

45. Pablog. Usar QtDesigner para actualizar nuestro QHolaMundo. [En línea] 2011. http://pablog-hg.blogspot.com.

46. Cáceres Tello , Jesús . ***Patrones de diseño: ejemplo de aplicación en los Generative Learning Object.*** **Alcalá : Dpto. Ciencias de la Computación.**

47. Tedeschi, Nicolás. ¿Qué es un Patrón de Diseño? [En línea] Microsoft, 2015. [Citado el: 10 de Febrero de 2015.] https://msdn.microsoft.com/es-es/library/bb972240.aspx.

48. rubenfa. Patrones de diseño: qué son y por qué debes usarlos. [En línea] 14 de Julio de 2014. [Citado el: 10 de Febrero de 2015.] http://www.genbetadev.com/metodologias-de-programacion/Patrones-de-diseño:-qué-son-y-por-qué-debes-usarlos.

49. Prácticas de Software . [En línea] 21 de Marzo de 2011. [Citado el: 10 de Febrero de 2015.] http://www.practicasdesoftware.com.ar/category/arquitectura/Patrones-GRASP.

50. ***Diseño Dirigido por Responsabilidades con los patrones GRASP.*** **s.l. : Pearson Educación, S.A.**

51. Matrínez Zurita, Alberto. ***Diagrama de Despliegue.***

52. Cruz Quispe, Víctor Fabio, Gutiérrez Mamani, Ever Dino y Mendivil Torrico, Luís Briam. ***Diagrama de Componentes.***

53. Python Software Foundation. Unicode HOWTO. [En línea] 2015. [Citado el: 12 de Febrero de 2015.] https://docs.python.org/2/howto/unicode.html.

54. Holovaty, Adrian y Kaplan-Moss , Jacob. ***The Definitive Guide to Django: Web Development Done Right.*** **s.l. : Apress, 2007.**

55. Hernández Sampieri , Roberto, Fernández Collado , Carlos y Baptista Lucio , María del Pilar. ***Metodología de la Investigación. Quinta edición.*** **México D.F : McGRAW-HILL / INTERAMERICANA EDITORES, S.A. DE C.V., 2010. ISBN: 978-607-15-0291-9.**

56. Diseño de casos de prueba - Procesos Prouctivos. [En línea] [Citado el: 2 de mayo de 2015.] http://www.angelfire.lycos.com/.

57. ***Impacto social de la ciencia y la tecnología en Cuba: una experiencia de medición a nivel macro.*** **Batista, Armando Rodríguez. 4, s.l. : Revista CTS (pág. 147-171), 2005, Vol. II.**

58. ***Constitución de la República de Cuba.*** **2003.**

ANEXOS

Anexo 1. GUÍA PARA EL INTERCAMBIO CON ESPECIALISTAS.

Estimado (a) compañero (a).

Con el propósito de obtener su colaboración en la validación de la concepción teórico-metodológica con respecto a las licencias de software, en especial las referidas a cantidad de ordenadores y gestionadas desde el servidor del sistema, se solicita su valoración como especialista. De forma anticipada se agradece su preciada colaboración.

El intercambio tendrá como cuestiones fundamentales a intercambiar las siguientes:

- ✓ ¿Cuáles son los mecanismos que se utilizan para crear las licencias?
- ✓ ¿Con que tipo de licencias acompañan a los software que producen?
- ✓ ¿Qué criterios utilizan para intercambiar con el cliente y cuáles son los datos que le piden para la confección de la licencia?

Anexo 2. ACTAS DE ACEPTACIÓN.

Universidad de las Ciencias Informáticas

Acta de aceptación

ACTA DE ACEPTACIÓN

En cumplimiento con lo establecido con el Centro de Telemática y en función de la ejecución del Trabajo de diploma: "Licenciamiento del Gestor de Recursos de Hardware y Software" se hace entrega de los productos siguientes:

- Generador de Licencia.
- Módulo Licensure.

La parte Cliente, luego de haber revisado los productos determina que se aceptan los resultados obtenidos, lo cual permite la comercialización del software GRHS con un licenciamiento por cantidad de ordenadores.

Entrega	Recibe
Nombre y apellidos: Arian Simón Méndez.	**Nombre y apellidos:** Alexander Rodríguez Bonet.
Nombre y apellidos: Yosley Lugo Rojas.	
Cargo: Tesista.	**Cargo:** Jefe de departamento de aplicaciones. Centro de Telemática.
Firma:	**Firma:** UCI Centro Telemática CTLM
Comentarios:	

Fecha: 25/05/2015

Acta de aceptación

ACTA DE ACEPTACIÓN

En cumplimiento con lo establecido con el Centro de Telemática y en función de la ejecución del Trabajo de diploma: "Licenciamiento del Gestor de Recursos de Hardware y Software" se hace entrega de los productos siguientes:

- Generador de Licencia.
- Módulo Licensure.

La parte Cliente, luego de haber revisado los productos determina que se aceptan los resultados obtenidos, lo cual permite la comercialización del software GRHS con un licenciamiento por cantidad de ordenadores.

Entrega	Recibe
Nombre y apellidos: Arian Simón Méndez.	**Nombre y apellidos:** Jenny de la Rosa Pasteur.
Nombre y apellidos: Yosley Lugo Rojas.	
Cargo: Tesista.	**Cargo:** Líder del proyecto GRHS. Centro de Telemática.
Firma: 	**Firma:** 
Comentarios:	

Fecha: 25/05/2015

Acta de aceptación

ACTA DE ACEPTACIÓN

En cumplimiento con lo establecido con el Centro de Telemática y en función de la ejecución del Trabajo de diploma: "Licenciamiento del Gestor de Recursos de Hardware y Software" se hace entrega de los productos siguientes:

- Generador de Licencia.
- Módulo Licensure.

La parte Cliente, luego de haber revisado los productos determina que se aceptan los resultados obtenidos, lo cual permite la comercialización del software GRHS con un licenciamiento por cantidad de ordenadores.

Entrega	Recibe
Nombre y apellidos: Arian Simón Méndez. **Nombre y apellidos:** Yosley Lugo Rojas.	**Nombre y apellidos:** Antonio Hernández Domínguez.
Cargo: Tesista.	**Cargo:** Director del Centro de Telemática.
Firma:	**Firma:**
Comentarios:	

Fecha: 25/05/2015