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

Título: Infraestructura de licenciamiento para el Sistema de Información Hospitalaria del CESIM

Autores:

Leiny Amel Pons Flores

Mayelin Bruzón Ortega

Tutores:

Ing. Gerardo Morgade Donato

Ing. Leydis Hidalgo López

Co-Tutor:

Ing. Rosel Silva Pérez

La Habana, 18 de junio del 2014

"Año 56 de la Revolución"

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los __ días del mes de __ del año 2014.

Firma del autor
Mayelin Bruzón Ortega

Firma del autor
Leiny Amel Pons Flores

Firma del tutor
Leydis Hidalgo López

Firma del tutor
Gerardo Morgade Donato

Firma del co-tutor
Rosel Silva Pérez

Datos de contacto

Ing. Gerardo Morgade Donato

Graduado en el año 2009 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Ha impartido las asignaturas Inteligencia Artificial y Desarrollo de aplicaciones empresariales con Java y PostgreSQL. Actualmente se desempeña como arquitecto del Centro de Informática Médica (CESIM).

Correo electrónico: gmorgade@uci.cu

Ing. Leydis Hidalgo López

Graduada de ingeniera en Ciencias Informáticas en el año 2010. Trabajó un año de Especialista en el Centro de Desarrollo de Software Holguín, donde se desempeñó como líder, implementadora, analista y planificadora del proyecto Sistema de Gestión de Almacenes. Actualmente, Especialista General del Centro de Informática Médica(CESIM) en la UCI desempeñándose como analista de software en los proyectos de desarrollo alas BQO y el Sistema de Información Hospitalaria (HIS).

Correo electrónico: lhlopez@uci.cu

Ing. Rosel Silva Pérez

Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas en el año 2012. Actualmente labora en el Departamento de Sistemas de Gestión Hospitalaria del Centro de Informática Médica (CESIM). Se desempeña como desarrollador del Sistema de Información Hospitalaria del Centro.

Correo electrónico: rsperez@uci.cu

Resumen

En la actualidad los avances en las nuevas tecnologías demandan de una fuerte protección de la propiedad intelectual. Los productos informáticos están expuestos a ser víctimas de la piratería o uso ilegal de software, la cual consiste en la utilización de un determinado programa informático sin contar con la autorización requerida. Esto provoca el desempleo, enriquecimiento ilícito y causa pérdidas al estado ya que no se pagan los tributos que se deberían.

El objetivo principal de esta investigación es la implementación de un mecanismo de licenciamiento para el Sistema de Información Hospitalaria (HIS) del Centro de Informática Médica (CESIM), que permita disminuir los riesgos de violaciones de licencia. Para el proceso de la investigación científica se utilizaron los métodos teóricos: histórico-lógico y analítico-sintético, además métodos empíricos como la entrevista. Para el desarrollo de los componente se utilizó el Visual Paradigm v8.0, como herramienta de modelado de los artefactos ingenieriles, guiado por la metodología RUP, como Sistema de Bases de Datos PostgresSQL v8.4, pgAdmin v1.4 para su administración, Jboss Developer Studio v5.0 y Neat Beans IDE v7.3 como entornos de desarrollo integrado.

Como resultado se obtuvo una infraestructura de licenciamiento compuesta por tres componentes, generación, validación y activación de licencia, los cuales permitirán la protección de dicho software a partir de lo estipulado en el contrato de venta del mismo. La investigación permite concluir que con la implementación de estos tres componentes el riesgo de violación de licencias se reduce de una escala alta a baja.

Palabras claves: activación, generación, licencia de software, licenciamiento, riesgos, validación

Índice de contenido

INTRODUCCIÓN

Durante el siglo XX la actividad intelectual del hombre condujo al avance en gran escala de la ciencia y la tecnología. Una de las ramas que sintió este vertiginoso desarrollo fue la informática, la cual ha sido implementada en la mayoría de los sectores de la sociedad generando nuevas posibilidades de comunicación e impactando en la informatización de la salud, la educación y el comercio, haciendo a estos sectores grandes dependientes de las tecnologías de la información y las comunicaciones.

Unido al desarrollo mundial de la informática se han incrementado las actividades ilegales de individuos e instituciones que utilizan las tecnologías existentes para su propio beneficio sin contar con la autorización requerida por parte del creador. Este tipo de actividades trae consecuencias negativas que laceran y discriminan el trabajo y los resultados de muchas personas e instituciones.

La piratería o uso ilegal de software constituye una de las vías delictivas que atenta contra la actividad intelectual de los creadores de productos informáticos. Es común encontrar la copia e instalación de un software adquirido en más de una estación de trabajo, programas con fines de instalación y distribución, utilización de actualizaciones sin la licencia requerida así como adquirir el mismo con algún tipo de beneficio y destinarlo para uso comercial, descargarlos desde Internet sin la debida licencia o comprar copias no autorizadas.

Algunas de las consecuencias más importantes y conocidas de la piratería de software son las pérdidas económicas al país y a los productores de los programas, el desempleo formal, la venta y distribución ilegal de productos, pobreza y ausencia de pagos de impuestos a entidades gubernamentales. También se observan otras que afectan directamente a los usuarios y son el uso de productos de baja calidad, daños en equipos, por mal funcionamiento o virus, la falta de oportunidad de quejas o reclamos y la formación de malos hábitos de consumo a menores.

Para cerrar el paso a esta desfavorable actividad ilegal que es la piratería existen mecanismos y procesos que si bien no logran exterminar totalmente el problema actual, si brindan la posibilidad de disminuirlo en gran escala. Entre estos mecanismos se encuentran las patentes, el copyright y las licencias, los cuales protegen los productos de los usos abusivos u oportunistas y son además una forma eficaz de proteger las inversiones que se realizan.

La licencia de software es una especie de contrato, en donde se especifican todas las normas que rigen el uso de un determinado programa, principalmente se estipulan los alcances de uso, instalación, reproducción y copia de estos productos.

Estas no proveen un control completo sobre el uso del producto original. Por lo que es necesario reforzarlos por medio de mecanismos técnicos como, claves de activación e identificadores únicos de hardware y así evitar que dichos productos estén expuestos a diversas formas de violaciones entre las que se encuentran:

- La copia e instalación en cierta cantidad de ordenadores
- La distribución ilegal de software
- Comercialización de copias no autorizadas
- Descargas desde internet sin la debida licencia

En Cuba, en la Universidad de las Ciencias Informáticas existen actualmente varios centros de producción entre los que se encuentra el Centro de Informática Médica (CESIM) dividido en departamentos que se encargan de todo el proceso de implementación de software. Uno de estos es el Departamento de Sistemas de Gestión Hospitalaria del CESIM que se dedica a informatizar las actividades llevadas a cabo en cada una de las áreas de las instituciones de salud. Entre sus soluciones informáticas se encuentra el Sistema de Información Hospitalaria (SIH o HIS en sus siglas en inglés).

El HIS es un sistema informático que permite la recolección, almacenamiento, procesamiento, recuperación y comunicación de información relacionada con la atención al paciente y la labor administrativa para todas las actividades de una institución de salud.

Actualmente en el procedimiento de comercialización de dicho software, el comprador firma un documento o Licencia legal presentada por el vendedor, que avala legítimamente el uso de dicho producto para una o varias instituciones de salud. Dicha licencia legal tiene plasmada los términos y condiciones de uso del mismo.

Una vez instalado el software en la entidad hospitalaria donde se utilizará, se corre un alto riesgo de que cualquier usuario con privilegios en los servidores donde se encuentra desplegado el sistema, pueda utilizar, distribuir o reutilizar dicho producto, sin los permisos pertinentes. Además actualmente no se puede controlar que la cantidad de copias que solicitó el comprador, se corresponden con las que tiene en explotación, ni controlar la cantidad de usuarios, historias clínicas y entidades que están permitidas utilizar, así como el tiempo de uso por el cual se pagó.

Por lo anteriormente expuesto se identifica como **problema científico:** ¿Cómo disminuir el riesgo de violaciones de licencia del Sistema de Información Hospitalaria del CESIM?

El problema planteado se enmarca en el **objeto de estudio:** El proceso de licenciamiento de software.

El objeto de estudio delimita al **campo de acción:** El proceso de licenciamiento del Sistema de Información Hospitalaria del CESIM.

Por lo antes planteado se definió como **objetivo de la investigación**: Desarrollar una infraestructura de licenciamiento para el Sistema de Información Hospitalaria del CESIM para reducir el riesgo de violaciones de licencias.

Para alcanzar el objetivo planteado se proponen las siguientes tareas de la investigación:

- Analizar los mecanismos de licenciamiento para software existentes.
- Definir el mecanismo de generación, activación y validación de licencias de software.
- Evaluar las técnicas de análisis de riesgo.
- Definir la arquitectura y herramientas que se utilizarán para la implementación del componente de generación de licencia.
- Asimilar la arquitectura, herramientas y tecnologías de desarrollo definida por el departamento de Sistemas de Información Hospitalaria para los componentes de activación y validación de licencia.
- Obtener mediante el Proceso Unificado de Desarrollo, los artefactos asociados a los flujos de trabajos “Modelado de negocio”, “Gestión de Requerimientos”, “Diseño” e “Implementación”.
- Implementar los componentes de generación, activación y validación de licencia para el Sistema de Información Hospitalaria del CESIM.
- Validar la investigación mediante una técnica de análisis de riesgo.

Se espera como resultado que el HIS posea un mecanismo de licenciamiento que logre disminuir la posibilidad de distribuir, reutilizar, comercializar u obtener ilegalmente dicho producto informático, lo que permitirá lograr una mayor protección contra las violaciones de licencias.

Durante el desarrollo de la investigación se utilizaron los siguientes métodos científicos:

Métodos Teóricos

Histórico-Lógico: permitió conocer cómo ha evolucionado la utilización de los mecanismos de licenciamiento.

Analítico-Sintético: permitió realizar una investigación de los aspectos relacionados a los mecanismos de licenciamiento empleados en la actualidad, permitiendo analizar los métodos existentes y además facilitó extraer los elementos más importantes de estos sistemas.

Métodos Empíricos

Entrevista: permitió la realización de varios encuentros con personas en la universidad que ya habían realizado este tipo de sistema, para así obtener información de los principales componentes que se podrían implementar.

El presente trabajo consta de 4 capítulos, quedando estructurados de la siguiente manera:

Capítulo 1: Fundamentación Teórica de los componentes de la infraestructura: en este capítulo se abordan los aspectos teóricos utilizados para el desarrollo de la investigación. Se analiza el modelo DREAD para medir el riesgo de violación de licencia del HIS. Se realiza un estudio sobre los mecanismos de protección y el licenciamiento de software en diferentes productos informáticos. Además se describen las técnicas y herramientas que se van a utilizar para dar solución al problema planteado.

Capítulo 2: Características de los componentes de la infraestructura: en este capítulo se describe y esboza la propuesta de solución para el desarrollo de una infraestructura de licenciamiento. Se realiza el modelo de dominio, en el cual se definen los principales conceptos que se emplearán. Luego se puntualizan los requisitos funcionales y no funcionales que debe cumplir el sistema.

Capítulo 3: Análisis y diseño de los componentes de la infraestructura: en este capítulo se realiza una descripción del diseño del sistema dando a conocer la arquitectura que se utilizará para cada componente y patrones de diseño que se ponen en práctica. Se define la estructura de paquetes, para cada componente, que permite dividir dicho sistema en fragmentos manejables para su implementación. Además se muestran los diagramas de clases que forman parte del modelo de diseño y se describen las clases que los componen.

Capítulo 4: Implementación de los componentes de la infraestructura: en este capítulo se esboza y describe el Modelo de datos, para dar a conocer la estructura donde se almacenará toda la información requerida. Se definen las estrategias de codificación y estándares a utilizar para una mejor comprensión del código fuente, así como la forma en que se tratarán los errores. Además se muestra el modelo de implementación conformado por el diagrama de componentes, y el diagrama de despliegue para describir los componentes a construir, su organización y dependencias entre los

nodos físicos en los que funcionarán los componentes. También se calcula el riesgo de violaciones de licencia, luego de haber integrado la infraestructura al HIS.

Capítulo 1. Fundamentación Teórica de los componentes de la infraestructura

En el capítulo se abordan los aspectos teóricos utilizados para el desarrollo de la investigación, que sirven de apoyo para la mejor comprensión del presente trabajo. Se analizan técnicas de análisis de riesgo que sirven para medir el riesgo de violación de licencia que posee el HIS. Se realiza un estudio sobre los mecanismos de protección de software en diferentes productos informáticos. Se describen las técnicas y herramientas que se van a utilizar para dar solución al problema planteado, así como las plataformas que los soportan y las librerías usadas en la implementación de los componentes.

1.1 Conceptos básicos

Vulnerabilidad: es una debilidad en el sistema de seguridad que puede ser explotada para causar algún daño.

Licencia legal: es la facultad o permiso atribuido a una persona o empresa para ejercer una actividad, o gozar de ciertas libertades o concesiones fuera de las ordinarias mediante un documento en el que consta la aprobación o permiso (1).

Amenazas: es un grupo de circunstancias o acciones que pueden causar algún daño.

Riesgo: es la posibilidad de que ocurra algún daño.

Integridad: implica que la información solo puede ser modificada por las personas autorizadas y de la forma autorizada.

Confidencialidad: implica que debe protegerse la información de forma tal que sólo sea conocida por las personas autorizadas y se la resguarde del acceso de terceros (2).

No repudio: el no repudio es un servicio de seguridad que permite probar la participación de las partes en una comunicación (3). Existirán por tanto dos posibilidades:

- No repudio en origen: el emisor no puede negar que envío porque el destinatario tiene pruebas del envío.
- No repudio en destino: el receptor no puede negar que recibió el mensaje porque el emisor tiene pruebas de la recepción

Token: cadenas de texto compuestas por caracteres alfanuméricos, asociadas a material clave criptográfico. (4)

1.2 Técnicas de análisis de riesgo

El modelado de amenazas es una técnica de ingeniería cuyo objetivo es ayudar distinguir los riesgos de una aplicación o sistema informático. La realización de un modelado de amenazas contribuye a identificar y cumplir con los objetivos de seguridad específicos de cada entorno y facilita la priorización de tareas en base al nivel de riesgo. Una vez identificada la(s) amenaza(s), se procede a realizar una clasificación y puntuación del riesgo de ocurrencia de la misma. (5)

1.2.1 STRIDE

El término STRIDE es el acrónimo de "Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service, Elevation of privilege". Es decir, suplantación de identidad, manipulación de datos, repudio, revelación de información, denegación de servicio y elevación de privilegios. Este modelo es creado y utilizado por Microsoft[1], y se basa en el uso de árboles de ataques para luego extrapolar las amenazas y realizar una clasificación y un ranking de estas con el fin de priorizar las actuaciones necesarias para mitigar el riesgo. (5)

El método STRIDE propone la descomposición del sistema en componentes significativos, tras analizar cada componente para comprobar si es susceptible de sufrir amenazas, además propone acciones que traten de mitigarlas. (5)

Suplantación de identidad

Suplantar quiere decir adoptar la personalidad de otra persona en un equipo. Un ejemplo de suplantación de identidad es el acceso ilegal a la información de autenticación de otro usuario y el posterior uso de la misma, como el nombre de usuario y la contraseña. (5)

Manipular datos

La manipulación de datos implica la modificación malintencionada de estos. Algunos ejemplos son la modificación no autorizada de datos persistentes, como los que se mantienen en una base de datos, o

[1] Microsoft es una empresa multinacional de origen estadounidense, fundada el 4 de abril de 1975 por Bill Gates y Paul Allen, dedicada al sector del software.

la modificación de datos durante su transferencia entre equipos en una red abierta como, por ejemplo, Internet. (5)

Repudio

Las amenazas de repudio son aquellas en las que los usuarios niegan la autoría de una acción sin que otras partes puedan probar lo contrario. Por ejemplo, un usuario realiza una operación ilegal en un sistema sin que exista la posibilidad de realizar un seguimiento de la misma.

De igual forma, el no rechazo se refiere a la posibilidad de un sistema de contrarrestar las amenazas de repudio. Por ejemplo, un usuario que adquiere un artículo tiene que firmar un recibo al entregarse dicho artículo. De esta forma, el proveedor puede utilizar el recibo firmado como prueba de la entrega del paquete. (5)

Divulgación de información

Las amenazas de divulgación de información suponen la revelación de información a individuos que no deben tener acceso a la misma. Por ejemplo, la posibilidad de que los usuarios lean un archivo al que no se le ha proporcionado acceso, o la posibilidad de un intruso de leer datos que se están transfiriendo entre dos equipos. (5)

Denegación de servicio

Los ataques por denegación de servicio (DoS) ocasionan la pérdida de servicio a los usuarios válidos, por ejemplo, deshabilitando temporalmente un servidor web. Debe protegerse contra determinados tipos de amenazas DoS a fin de mejorar la disponibilidad y fiabilidad del sistema. (5)

Elevación de privilegios

En este tipo de amenaza, un usuario sin privilegios obtiene acceso con privilegios y, por tanto, la capacidad de poner en peligro o destruir todo el sistema. Un tipo de amenaza de elevación de privilegios es la situación en la que un agresor burla todas las defensas del sistema con éxito y se integra en la parte del sistema de confianza. (5)

Para almacenar los datos que se van recogiendo, se utiliza una plantilla con la información contenida en la siguiente tabla:

Descripción	
Objetivo de la amenaza	
Nivel de riesgo	
Técnicas de ataque	
Contramedidas	

Tabla 1.1 Modelo STRIDE

Luego se clasifican las amenazas por gravedad o impacto (el daño que ese tipo de ataque podría causar) y probabilidad. Para gravedad, se asigna un número entre 1 y 10, siendo 10 el nivel más grave. Para probabilidad, se elija un número, donde 1 es el nivel más probable y 10 es el menos probable y se calcula el riesgo general dividiendo gravedad por probabilidad. (6)

1.2.2 CORAS

CORAS (Consultative Objective Risk Analysis System), es un proyecto creado por la unión europea con el objetivo de proporcionar un framework orientado a sistemas donde la seguridad es crítica, facilitando el descubrimiento de vulnerabilidades de seguridad, inconsistencias y redundancias.

CORAS proporciona un método basado en modelos, para realizar análisis de riesgos, y se basa en el uso de tres componentes:

- Un lenguaje de modelado de riesgos basado en el UML.
- La metodología CORAS, una descripción paso a paso del proceso de análisis con una directriz para construir los diagramas CORAS.
- Una herramienta para documentar, mantener y crear los informes del análisis.

La metodología CORAS, hace un uso intensivo de los diagramas.

Existen 5 tipos diferentes:

- Diagrama superficial de activos: muestra una visión general de los activos y cómo el daño sobre un activo puede afectar al resto.

- Diagrama de amenazas: muestra una visión completa de la secuencia de eventos iniciados por las amenazas y las consecuencias que tienen éstas sobre los activos. Sus componentes básicos son: amenazas deliberadas, amenazas accidentales, amenazas no-humanas, vulnerabilidades, escenarios de amenazas, incidentes no deseados y activos.
- Diagrama superficial de riesgo: es un resumen del diagrama de amenazas, mostrando los riesgos. Tiene 5 componentes básicos: amenazas deliberadas, accidentales y no-humanas, riesgos y activos. A cada riesgo se le asigna un valor.
- Diagrama de tratamiento: ofrece una visión completa de las contramedidas propuestas. Se basa en el diagrama de amenazas, sustituyendo las consecuencias del impacto sobre los activos con los riesgos procedentes del diagrama superficial de riesgo, y añadiendo los escenarios de contramedidas propuestos.
- Diagrama superficial de tratamiento: es un resumen de las contramedidas, añadiendo los distintos escenarios posibles y mostrando las relaciones entre los distintos elementos propuestos para tratar el riesgo.

Los siete pasos necesarios para realizar un análisis de riesgos utilizando CORAS podrían resumirse de la siguiente forma (7):

- Paso 1: se realiza una entrevista inicial entre los representantes del cliente y los analistas para conocer cuáles son los objetivos principales del análisis. Se recopila la información necesaria en función de los requisitos del cliente.
- Paso 2: una segunda reunión con el cliente para comprobar que la información suministrada al analista ha sido suficiente. Se realiza un análisis superficial, procediendo a identificar las primeras amenazas, vulnerabilidades, los diferentes escenarios, así como los posibles incidentes no deseados.
- Paso 3: se realiza una descripción más detallada del sistema a analizar, facilitando al cliente la documentación necesaria para su aprobación.
- Paso 4: el analista, junto con las personas que mejor conocen el sistema, identifican todos los posibles incidentes no deseados, amenazas, vulnerabilidades y escenarios.
- Paso 5: se estiman las consecuencias y los valores de ocurrencia para cada uno de los posibles incidentes no deseados que se han identificado en los pasos anteriores.

- Paso 6: se proporciona al cliente un borrador del análisis para una primera revisión y corrección.
- Paso 7: se establece el tratamiento del riesgo, es decir, las contramedidas en función del coste/beneficio.

1.2.3 DREAD

El modelo DREAD es usado para ayudar a calcular el riesgo. Este modelo también fue creado por Microsoft, y permite priorizar las actuaciones a efectuar para mitigar el riesgo, ya que se puede cuantificar. Al utilizar el modelo DREAD, se llega a la calificación de riesgo de una amenaza determinada haciendo las siguientes preguntas (8):

- **D**año potencial: ¿Cuán grande es el daño si la vulnerabilidad se explota?
- **R**eproductividad: ¿Es fácil de reproducir el ataque?
- **E**xplotabilidad: ¿Es fácil de lanzar un ataque?
- **A**fectación a usuarios: ¿Cuáles usuarios se ven afectados?
- **D**etectabilidad: ¿Es fácil de encontrar la vulnerabilidad?

Al definir claramente lo que cada valor representa para el sistema de calificación, se evita la confusión. La tabla siguiente muestra el ejemplo típico de una tabla, utilizando un esquema simple, como Alto (1), Medio (2), y Baja (3).

	Clasificación	**Alto (3)**	**Medio (2)**	**Baja (1)**
D	Daño potencial	El atacante puede subvertir el sistema de seguridad, obtener la autorización plena; ejecutar como administrador, subir contenido.	La fuga de información sensible	Fuga de información trivial

R	Reproductividad	El ataque puede ser reproducido cada vez y no requiere una ventana de tiempo.	El ataque puede ser reproducido, pero sólo con una ventana de tiempo y una situación en particular.	El ataque es muy difícil de reproducir, incluso con conocimiento de un agujero de seguridad.
E	Explotabilidad	Un programador novato podría hacer el ataque en un corto tiempo.	Un programador experto podría hacer que el ataque, repetir los pasos.	El ataque requiere una persona muy hábil y un conocimiento en profundidad cada vez que se explote.
A	Afectación a usuarios	Todos los usuarios, la configuración por defecto, los clientes clave	Algunos usuarios, la configuración predeterminada	Muy pequeño porcentaje de usuarios, característica oculta; afecta a los usuarios anónimos
D	Detectabilidad	Las informaciones publicadas explican el ataque. La vulnerabilidad se encuentra en la característica más común y es muy notable.	La vulnerabilidad está en una zona de poco uso del producto, y sólo unos pocos usuarios deben venir a través de ella. Haría falta pensar un poco para ver el uso malicioso.	El error es oscuro, y es poco probable que los usuarios concluirán en los posibles daños.

Tabla 1.2 Modelo DREAD

Después de hacer las preguntas anteriores, se cuentan los valores (1-3) para una amenaza. El resultado se puede caer en el rango de 5-15. Entonces se puede tratar a las amenazas con

puntuaciones totales de 12-15 como de alto riesgo, 8-11 de riesgo mediano, y 5-7 como de bajo riesgo. Una vez que se ha obtenido la calificación de riesgo, se clasifica atendiendo al valor obtenido. (8)

A diferencia de SRIDE Y DREAD, el método proporcionado por CORAS, no define ningún paso donde se pueda obtener un valor cuantitativo para las amenazas que se identifiquen; este más bien contribuye a la adopción de contramedidas para la reducción de riesgos.

STRIDE es un sistema de clasificación, que utiliza un modelo de información basado en plantillas, para identificar los patrones de soluciones y problemas repetibles y organizarlos en categorías, además el método de clasificación que usa este modelo puede provocar dificultad de valorar de igual forma el riesgo cuando esta valoración se efectúa entre varias personas. Por su parte DREAD facilita el uso de criterios comunes respondiendo a las cinco cuestiones que plantea; es un modelo que facilitará la puntuación y clasificación de un riesgo teniendo en cuenta características propias del sistema a analizar. Una forma efectiva para alcanzar mejores resultados con este método sería variar y adaptar algunos campos a la situación específica.

Luego de analizar el modelo DREAD se adaptaron algunos campos de la tabla en función de dar solución al problema que se tiene, definiendo calcular el riesgo de la manera que a continuación se describe:

	Clasificación	**Alto (3)**	**Medio (2)**	**Baja (1)**
D	Daño potencial	El atacante puede redistribuir el HIS con permisos superiores a la licencia permitida por el contrato.	El atacante puede redistribuir el HIS con los mismos permisos a la licencia permitida por el contrato.	El atacante no puede redistribuir el HIS.
R	Reproductividad	El ataque puede ser reproducido cada vez y no requiere una ventana de tiempo.	El ataque puede ser reproducido, pero sólo con una ventana de tiempo y una situación de carrera en particular.	El ataque es muy difícil de reproducir, incluso con conocimiento de dicho agujero de seguridad.

E	Explotabilidad	Un programador novato podría hacer el ataque en un corto tiempo.	Un programador experto podría hacer el ataque, y luego repetir los paso	El ataque requiere una persona muy hábil y un conocimiento en profundidad cada vez que se explote.
A	Afectación a las entidades hospitalarias	El atacante puede utilizar, distribuir o reutilizar todos los módulos del sistema HIS.	El atacante puede utilizar, distribuir o reutilizar alguno de los módulos del sistema HIS.	El atacante puede utilizar, distribuir o reutilizar alguna funcionalidad del sistema HIS.
D	Detectabilidad	Cualquier usuario puede violar la seguridad del sistema. La vulnerabilidad es muy notable.	Solo usuarios con acceso al sistema pueden detectar la vulnerabilidad.	Solo usuarios con acceso a los servidores podrían violar la vulnerabilidad.

Tabla 1.3 Modelo DREAD de la solución propuesta

Con este modelo de calificación y clasificación se pudo definir como se muestra en la siguiente tabla, que el riesgo de violación de licencia que actualmente posee el HIS tiene una calificación de 13 puntos y se clasifica como Alto.

Amenaza	**D**	**R**	**E**	**A**	**D**	**Total**	**Clasificación**
Violación de licencia por copia del compilado	3	3	3	3	1	13	Alto

Tabla 1.4 Calificación de amenaza del HIS en la actualidad

1.3 Métodos de protección de software

Para la industria del software, la protección de sus productos es una característica importante, no sólo en cuanto a las copias ilegales del software, sino también a la protección de los derechos de propiedad intelectual del código. La transformación del software en un producto de comercialización y de alta vulnerabilidad (dada la posibilidad de su copia a bajo costo), son las causas más importantes que

condujeron a la implementación de mecanismos para proteger la actividad intelectual humana en la creación de dichos productos, puesto que estos representan el resultado de un esfuerzo creativo y de inversiones de tiempo y de dinero muy elevadas. Es por eso que para garantizar la seguridad del software se deben cubrir una variedad de técnicas que van desde protección legal por derechos de copia (copyright), licencias y patentes, a métodos técnicos.

1.3.1 Métodos de protección legal

Copyright

El copyright es un mecanismo legal para obtener protección del estado para producciones artísticas originales, música, trabajos literarios, e incluso programas informáticos, constituye un derecho exclusivo de un autor para controlar la distribución y reproducción de su trabajo original, en términos de software, tanto el código fuente (legible por personas) como el código objeto (ejecutado por la máquina), y los manuales relacionados, son los que se eligen para proteger por copyright. La duración de protección de este mecanismo varía en dependencia del país donde se conceda el derecho, pero por lo general el tiempo mínimo de protección es de 10 años.

La protección de derecho de autor protege como tal la obra o producto, pero no su uso, este mecanismo no fue concebido en sus inicios para proteger programas de computación, por lo cual sus efectos para este fin están desajustados y producen fallas en lo que concierne a la forma de protección. No es suficiente para los desarrolladores prohibir la duplicidad o reproducción de su trabajo, sino que también le es necesario impedir el uso del mismo sin autorización. (9)

Patentes

Las patentes son un mecanismo legal para obtener protección del estado para un invento o mejora a un invento. Una patente es un derecho legal certificado por una entidad del gobierno que permite a un inventor prevenir que otros fabriquen, vendan o utilicen su invención, esta impide totalmente la implementación de cierta idea, sin importar quien escriba el código ni el lenguaje de programación que se utilice.

La protección por patentes puede convertirse en una herramienta competitiva valiosa, comparada a la protección tradicional por copyright, ya que una patente protege ideas y algoritmos en un producto de software, mientras el copyright el código en sí mismo. Pero si bien las patentes son una solución a la

propiedad intelectual, también traen consigo controversias puesto que un software está compuesto por procedimientos, algoritmos, funcionalidades que ya pudiesen estar patentadas y su utilización conllevaría a problemas legales. (9)

Las personas implicadas en el movimiento de software libre advierten que el uso de patentes impediría el desarrollo de muchos proyectos que no pueden pagar licencia a costa de dejar de ser libres (libre uso y distribución del programa, acceso al código fuente, posibilidad de modificación) ... las patentes de software (y sobre medicamentos, métodos de negocio, procedimientos legales e ingeniería genética, estas últimas las denominadas "patentes de la vida") privatizan el conocimiento acentuando las desigualdades sociales y geográficas mediante la exclusión de la mayoría de la población como productores e incluso como consumidores de los objetos de dichas patentes. (10)

Licencias

Las licencias de software son 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. (11)

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

- Autor: el o conjunto de desarrolladores que crea el software son por antonomasia quienes en una primera instancia poseen el rol de licenciante al ser los titulares originales del software.
- Titular de los derechos de explotación: es la persona natural o jurídica que recibe una cesión de los derechos de explotación de forma exclusiva del software desde un tercero, transformándolo en titular derivado y licenciante del software.
- Distribuidor: es la persona jurídica a la cual se le otorga el derecho de distribución y la posibilidad de generar sublicencias del software mediante la firma de un contrato de distribución con el titular de los derechos de explotación.

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

- Usuario consumidor: persona natural que recibe una licencia de software otorgada por el licenciante, la cual, se encuentra en una posición desventajosa ante los términos y condiciones establecidas en ella.
- Usuario profesional o empresa: persona natural o jurídica que recibe una licencia de software otorgada por el licenciante, la cual, se encuentra en igualdad de condiciones ante el licenciante para ejercer sus derechos y deberes ante los términos y condiciones establecidos en la licencia.

Dichas licencias implican el cumplimiento de un conjunto de cláusulas introducidas en el contrato y alusivas a la seguridad y protección del programa, consignando el eventual acceso a los mismos por personas no autorizadas, uso inadecuado, modificaciones no permitidas, y destrucción de información.

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. (12)

1.3.2 Métodos técnicos de protección

Primeramente se debe entender algo, la garantía legal de los derechos exclusivos de protección de software por medio de patentes, derechos de copia (copyright) y licencias no proveen un control completo sobre el uso del producto original. Por ende surge la necesidad de tener que reforzarlos por medio de mecanismos técnicos. Estas técnicas incluyen esquemas y mecanismos capaces de prevenir el uso no autorizado de software.

Protección por número de serie o palabra clave

La protección por número de serie es un método para restringir la distribución de copias ilegales de software. Consiste en ingresar, cuando se instala el software, una palabra clave conformada por

números y caracteres. Adicionalmente, el software puede requerir el ingreso de otros números de serie para acceso a diferentes funciones del programa.

Los dos métodos más comunes para la utilización de números de serie son:

- En la instalación del software: antes de comenzar cualquier tarea relacionada con la instalación (copiado de archivos, petición de datos, etc.) se solicita al usuario el número de serie para verificar que él posee una copia original del software.
- Cada vez que se inicia el software: este método es el más usado por juegos interactivos, que requieren el ingreso de un código de acceso en base a datos que se proveen por pantalla.

Protección por tiempo

Pueden operar de distintas formas:

- El software comprueba si han transcurrido X días desde su instalación, y si es así procede a su salida inmediata o en el peor de los casos a su desinstalación automática. Durante la salida/desinstalación del software, éste puede mostrar algún mensaje informando al usuario del hecho en cuestión.
- El software comprueba si ha llegado a una fecha límite; si es así, procede de la misma manera que en el caso anterior. La diferencia está en que el software dejará de funcionar a partir de una fecha determinada.

Existen algunas variantes específicas para este tipo de protección, entre ellas se pueden encontrar:

Por fecha (obtención): la forma más difundida de protección es la que le permite al futuro comprador evaluar el programa un número de días preestablecido. La fecha de inicio se obtiene el día que se instala el programa, y se guarda el día que expira.

Por fecha (verificación simple): se verifica que la fecha actual sea menor que la de expiración. De este modo, cuando se instala el programa o cada vez que transcurre un día, se comprueba que la fecha sea válida.

Por fecha (verificación del límite superior e inferior): se guarda la fecha de instalación. Primero se verifica que la fecha actual sea menor que la de instalación, en caso afirmativo se da por finalizado el

período de uso. Después se compara la fecha de expiración con la fecha actual: si es menor se da por terminado el período de uso.

Por veces de uso: una manera de evitar el quiebre de la protección al cambiar la fecha es tener un contador de veces de funcionamiento; este se incrementa cada vez que se corre el programa. El inconveniente de este método es que si se corre la aplicación y no se cierra, los días transcurren y el contador no se incrementa.

Por veces de uso (contador de 24 horas): para evitar que al dejar funcionando la PC indefinidamente se anule la protección, los fabricantes de software decidieron agregarle un contador de días transcurridos mientras el programa está en funcionamiento; por cada día que transcurre se incrementa el contador de veces de ejecución. Cuando llega a la cantidad de veces establecida la licencia expira.

Identificadores únicos de Hardware

Para crear una licencia de software, es necesario obtener identificadores partiendo de las características físicas de las computadoras donde será desplegado el sistema. Esto se logra obteniendo propiedades que no se modifiquen cuando se reinstale el sistema operativo porque luego de aplicada la licencia, debe ser funcional durante el tiempo convenido por los proveedores y clientes. La única forma de lograr que estas características persistan es tomando identificadores que representen de forma unívoca los componentes del hardware de la máquina. Dentro de ellos se encuentran:

- Las direcciones de hardware de un equipo también llamadas direcciones MAC por su acrónimo en inglés como (Media Access Control) tiene una dirección única y particular a nivel mundial ya que estas son las que gestionan las direcciones de IP en el protocolo TCP/IP.
- El disco duro es el dispositivo de almacenamiento de datos no volátil que emplea el sistema operativo para almacenar datos. Este posee un número físico impreso conocido como PNPDeviceID que es muy útil para la identificación.

1.3.2.1 Encriptación de datos

En la comunicación de datos, es de vital importancia asegurar que la información viaje segura, manteniendo su autenticidad, integridad, confidencialidad y el no repudio de la misma, entre otros aspectos.

Métodos de encriptación

Para encriptar datos, se pueden utilizar procesos matemáticos diferentes, la criptografía, los algoritmos simétricos, los asimétricos y los híbridos.

Criptografía: la criptografía es la técnica de convertir un texto en claro en un texto ilegible, llamado criptograma, cuyo contenido de información es igual al anterior pero sólo lo pueden entender las personas autorizadas. En esta técnica se utilizan algoritmos criptográficos, que no es más que una función matemática usada en los procesos de cifrado y descifrado. (13)

Algoritmos simétricos: utilizan una clave con la cual se encripta y desencripta el documento. Todo documento encriptado con una clave, deberá desencriptarse, en el proceso inverso, con la misma clave. Es importante destacar que la clave debería viajar con los datos, lo que hace arriesgada la operación, imposible de utilizar en ambientes donde interactúan varios interlocutores.

Algoritmos asimétricos: los algoritmos asimétricos conocidos como algoritmos de llave pública, son mucho más seguros que los simétricos ya que generan dos llaves simultáneamente (llave pública y llave privada), y está ligada la una a la otra. También emplean longitudes de clave mucho mayores que los simétrico, esto garantiza que sean mucho más difíciles de descifrar, para terceras personas que quieran apoderarse de la información.

Las parejas de llaves tienen diversas funciones:

- Cifrar la información.
- Asegurar la integridad de los datos transmitidos.
- Garantizar la autenticidad del emisor.

Los algoritmos asimétricos proporcionan autenticidad, integridad y no repudio, mientras que los algoritmos simétricos sólo proporcionan confidencialidad. La ventaja del cifrado asimétrico sobre el simétrico radica en que la clave pública puede ser conocida por todo el mundo, no así la privada, sin

embargo en el cifrado simétrico deben conocer la misma clave los dos usuarios y esta debe hacerse llegar a cada uno de los distintos usuarios por el canal de comunicación.

Los algoritmos asimétricos tiene dos aplicaciones fundamentales: el cifrado de los archivos y la firma digital. Este último permite al receptor de un mensaje verificar que el origen es auténtico, por lo que se puede comprobar si el mensaje ha sido modificado. Falsificar una firma digital es casi imposible a no ser que se conozca la clave privada del que firma. Entre los algoritmos asimétricos se encuentran: RSA y DSA.

El RSA puede ser utilizado tanto para cifrar como para firmar digitalmente. En el cifrado asimétrico la clave que se hace pública es aquella que permite codificar los mensajes, mientras que la clave privada es la que permite descifrarlos. El algoritmo DSA se utiliza únicamente para realizar firma digital. (14)

Encriptación híbrida (asimétrica + simétrica): debido a que la encriptación asimétrica es más lenta que la simétrica, cuando la información a encriptar es mucha, se utiliza una combinación de algoritmos. El algoritmo simétrico se utiliza para encriptar la información y el asimétrico para encriptar la llave del algoritmo simétrico con que se encriptó la información.

Entonces, el proceso es mucho más rápido. En cada ida y vuelta al servidor se generan nuevas llaves y se realiza todo el proceso.

Un motivo para utilizar esta encriptación combinada es la necesidad de encriptar textos largos. La encriptación asimétrica además de ser ineficiente en tiempo, tiene limitaciones de tamaño. El tamaño máximo depende del largo de la llave.

Si se genera una llave, la cual se almacenará en una clase o un servicio, y luego se escoge el método asimétrico que se va a combinar, el proceso puede suceder de la siguiente forma:

1. Se encripta la información con la llave definida mediante un método de encriptación simétrico.

2. Luego se encriptará esa llave mediante el uso de un método de encriptación asimétrica garantizando así que la misma , aunque sea del cocimiento de la clase o servicio antes mencionado, no tenga ninguna utilidad a la hora de desencriptar la información que se desea proteger.

3. La encriptación se produce en un lado y la desencriptación en otro. (14)

Firma digital

La firma digital es una herramienta tecnológica que permite garantizar la autoría e integridad de los documentos digitales, permitiendo que estos gocen de una característica que únicamente era propia de los documentos en papel. (15)

Una firma digital es un conjunto de datos asociados, la cual facilitará con su uso la validación de los datos pertenecientes al comprador del software, en un documento digital permitiendo garantizar la identidad del mismo.

En general la firma digital está compuesta por una clave pública y otra privada. El esquema de funcionamiento sería el siguiente (16):

1. El emisor genera un resumen del documento ya que la firma no se realiza sobre el documento completo sino en un resumen del mismo o hash.
2. El emisor firma su resumen encriptándolo con la clave privada propia.
3. El emisor envía el documento y su resumen firmado al receptor.
4. El receptor genera también un resumen del documento que ha recibido usando la misma función que el emisor. Al mismo tiempo descifra el resumen recibido con la clave pública que el emisor ha publicado. Si los resúmenes coinciden la firma será validada.

Funciones hash

Debido a que los algoritmos de clave pública son complejos y lentos, para firmar un mensaje solo se emplea una porción de este. Esa porción es a lo que se denomina resumen del mensaje o hash (antes mencionado), normalmente con un tamaño fijo, que también es útil para comprobar la integridad del mensaje y se obtiene a partir de una función irreversible.

Algunos de los algoritmos de HASH más utilizados:

- MD5 (Message-Digest Algoritmo 5 o Algoritmo de Firma de Mensajes 5): desarrollado por Ron Rivest, ha sido hasta los últimos años el algoritmo hash más usado. Procesa mensajes de una longitud arbitraria en bloques de 512 bits generando un compendio de 128 bits. Debido a la capacidad de procesamiento actual esos 128 bits son insuficientes, además de que una serie de ataques criptoanalíticos han puesto de manifiesto algunas vulnerabilidades del algoritmo.

Puede ser útil para comprobar la integridad de un fichero tras una descarga, por ejemplo, pero ya no es aceptable desde el punto de vista del criptoanálisis. (17)

- SHA-1 (Secure Hash Algoritmo 1 o Algoritmo de Hash Seguro 1): SHA-1 toma como entrada un mensaje de longitud máxima 264 bits (más de dos mil millones de Gigabytes) y produce como salida un resumen de 160 bits. (18)

El hash SHA1 es un algoritmo de generación de firmas. Un hash es una cadena de letras y números que resulta del cálculo sobre una cadena de origen. Este algoritmo permitirá asignar distintas firmas (o hashes SHA) para distintos usuarios. (18)

La principal importancia en este tipo de algoritmos es el parámetro longitud de resumen, pues a mayor longitud, mayor seguridad en la clave que se genere.

Una vez se tiene la función hash productora de resúmenes de una longitud adecuada y óptima, se puede combinar con cifrados de clave secreta o/y pública (firma digital), aplicándose éstos sobre mensaje o/y resumen, logrando con ello un esquema de firma digital a la vez práctico y seguro.

1.4 Mecanismos de licenciamiento utilizados en la actualidad

1.4.1 Kaspersky Anti-Virus 6.0

Kaspersky Anti-Virus 6.0 está diseñado pensando en las posibles fuentes de amenazas. Dicho de otro modo, cada amenaza está controlada por un componente distinto de la aplicación, que supervisa y toma las acciones necesarias para evitar los efectos maliciosos en los datos del usuario, desde su mismo origen. Esta organización flexible permite configurar con facilidad cualquiera de los componentes y ajustarlos a las necesidades específicas de un usuario en particular, o de toda la organización.

Kaspersky Anti-Virus 6.0 requiere una licencia para poder funcionar. Su compra se realiza en línea desde la tienda electrónica Kaspersky Lab, después de realizar el pago, el usuario recibirá un archivo llave o un código de activación en la dirección indicada en el formulario de pedido. Con dicho código de activación se procede al proceso de activación.

El proceso de activación consiste en instalar una llave para registrar una licencia. Dependiendo de la licencia, la aplicación determina los privilegios existentes y evalúa las condiciones de uso.

El archivo llave contiene información de servicios necesarios para el funcionamiento completo de Kaspersky Anti-Virus 6.0, así como datos adicionales:

- Información de soporte (quién ofrece asistencia y dónde obtenerla)
- Nombre de archivo, número y fecha de caducidad de la licencia

Dependiendo de si ya dispone de un archivo llave o necesita obtenerlo desde un servidor de Kaspersky Lab, dispone de varias opciones para activar Kaspersky Anti-Virus 6.0 (19):

- Activación en línea: se selecciona esta opción de activación si se ha adquirido la versión comercial de la aplicación y recibido un código de activación. Este código de activación le permite obtener un archivo llave que le da acceso a todas las características de la aplicación durante el plazo de validez de la licencia.
- Activación de la versión de evaluación. Se selecciona esta opción de activación si se desea instalar la versión de evaluación de la aplicación antes de tomar la decisión de adquirir la versión comercial. Se recibirá una llave gratuita válida por el tiempo especificado en el contrato de licencia de la versión de evaluación.
- Activación con un archivo llave anterior. Activar la aplicación con el archivo llave de Kaspersky Anti-Virus 6.0 obtenido previamente.
- Activar más tarde. Al elegir esta opción, se pasa por alto la etapa de activación. La aplicación quedará instalada en su equipo y tendrá acceso a todas las características de la aplicación, salvo las actualizaciones (sólo se podrá actualizar una vez la aplicación, inmediatamente después de instalarla).

Al seleccionar cualquiera de las dos primeras opciones, la aplicación se activa mediante el servidor Web de Kaspersky Lab, lo cual requiere una conexión a Internet. Si en el momento de la instalación, la conexión Internet no está disponible, es posible realizar la activación más tarde, bien desde la interfaz de la aplicación. También puede conectarse a Internet desde otro equipo y obtener una llave mediante el código de activación atribuido al registrarse en el sitio Web del soporte técnico de Kaspersky Lab.

Si activó una versión de evaluación de la aplicación, al finalizar el plazo de prueba gratuita, Kaspersky Anti-Virus dejará de funcionar.

Cuando la llave comercial caduca, el programa sigue funcionando, pero no permite actualizar las bases de aplicación. Como antes, Kaspersky Anti-Virus 6.0 podrá analizar en busca de virus y utilizar los componentes de protección, pero sólo con las bases de aplicación disponibles cuando caducó la licencia.

1.4.2 Adobe Photoshop CS4

Adobe Photoshop CS4 es un editor de gráficos desarrollado por Adobe Systems. Su espacio de trabajo está organizado para ayudar a centrarse en la creación y edición de imágenes. Este espacio de trabajo incluye menús y una serie de herramientas y paneles para visualizar, editar y añadir elementos a las imágenes. (20)

Como la mayoría de los programas, Adobe Photoshop necesita de una licencia para poder funcionar correctamente. Una vez ejecutado el archivo de instalación, Adobe Photoshop CS4 solicita un número de serie para validar la licencia asociada, dicho número de serie es enviado por correo electrónico al usuario una vez comprado el software en el sitio web Adobe. También brinda la opción de instalar una versión de prueba la cual dejara usar el software sin licencia por un tiempo máximo de 30 días.

El software se conecta con los servidores de Adobe a través de internet para completar el proceso de activación de licencia y validar el derecho de usar el software en dicho equipo.Al activar una licencia de un solo usuario, ésta sirve para dos equipos. Si se desea instalar el software en un tercer equipo, se debe desactivar primero en uno de esos dos equipos.

Luego de activado y validado, la oferta de este producto está sujeta a la aceptación del acuerdo de licencia incluido en el producto, y con unos términos de garantía limitados. Posteriormente se da la opción de realizar una instalación fácil (recomendada) o una instalación personalizada, y así termina el proceso de instalación.

1.4.3 DalasQ

El producto DalasQ del Centro de Consultoría y Desarrollo de Arquitecturas Empresariales de la Universidad de las Ciencias Informáticas, viabiliza la gestión de las colas en cualquier institución que

brinde servicios de atención al público. Para evitar que el producto sea instalado en máquinas distintas a la que especifica el contrato, establecen un mecanismo de protección de software que evita su copia ilícita.

La solución cuenta con tres componentes fundamentales: la aplicación cliente que va a estar en manos de los encargados de desplegar el sistema, la aplicación gestora que se encuentra del lado de los administradores de la empresa y la biblioteca dalaslicensemanager.jar que va a estar ubicada dentro del producto DalasQ.

La aplicación cliente es la encargada de generar las solicitudes de licencia, que se traduce a un archivo XML que va a contener todos los datos del solicitante, así como datos específicos que se requieran para cada tipo de licencia:

- Licencia bloqueada
- Licencia flotante
- Licencia prueba

En caso de que el tipo de licencia sea una licencia bloqueada es necesario que la aplicación obtenga un identificador único que garantiza que la licencia se utilice sólo en una estación.

Esto se logra a través de la obtención del identificador de la MAC. Este identificador cumple con los estándares para los identificadores universales. Si la licencia es de prueba, se tendría en cuenta el tiempo de duración, para cuando se cumpla ese plazo se desactive el uso del producto DalasQ. En caso de que la licencia sea flotante, además de obtener el identificador del hardware, se tendría en cuenta la cantidad de máquinas que se van a conectar al servidor.

Una vez generada la solicitud, si existe en la institución un servidor de correo electrónico, es enviada automáticamente al administrador del sistema sino el proceso se realiza manualmente, utilizando un dispositivo de almacenamiento externo. Los administradores cuentan en su poder con la aplicación gestor, la cual se encarga de atender la solicitud y generar las licencias según el tipo especificado. Esta licencia no es más que otro archivo con extensión .key que es encriptado utilizando algoritmos asimétricos, la aplicación brinda la posibilidad de generar las llaves utilizando el tipo de algoritmo RSA. También brinda la posibilidad de generar reportes que ayuda a la toma de decisiones, los cuales pueden ser exportados a diferentes formatos PDF, XML, HTML, etc.

Luego de ser generada la licencia es almacenada en el dispositivo externo o es enviada automáticamente por correo electrónico al especialista encargado en el despliegue, el mismo la importa a la aplicación cliente y esta se encargará de ubicarla en la ruta que haya especificado. A la hora de instalar la aplicación DalasQ, esta utiliza la biblioteca dalaslicensemanager.jar para descubrir la licencia en la ruta donde se especificó y luego validarla. Si es válida se instala satisfactoriamente sino muestra un mensaje que no es válida para esa máquina. (21)

El mecanismo de licenciamiento que utilizan en el producto DalasQ no es adaptable al problema presente pues no se contemplan características del producto HIS para la creación de las licencias, ya que estos poseen distintos tipos de licencias con diversas peculiaridades. Además en su solución no separan los componentes de activación y validación.

Por otra parte los productos Kaspersky Anti-Virus 6.0 y Adobe Photoshop CS4, son software privativos y se desconoce cómo se implementan las funcionalidades que permiten realizar todo el proceso de licenciamiento.

Basados en los mecanismos de licenciamiento utilizados por dichos sistemas para la solución se determina realizar una infraestructura, lo cual consiste en un conjunto de elementos o componentes necesarios para el funcionamiento del mecanismo de licenciamiento del HIS, dirigidos para servir de soporte a su posterior uso.

Para dicha infraestructura se tendrán en cuenta los elementos esenciales estudiados que componen el proceso de licenciamiento, percibido desde la generación de la licencia, a partir de la solicitud del usuario o comprador, hasta la activación y validación de la misma, por parte del distribuidor, garantizando en este orden el cumplimiento exitoso de dicho proceso.

1.5 Metodología de Desarrollo

1.5.1 Rational Unified Process (RUP)

RUP es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable.

Las mejores prácticas del RUP, son un conjunto de procesos, habilitados para la web, de ingeniería de software que dan guía para conducir las actividades de desarrollo del equipo. Como una plataforma de

procesos que abarca todas las prácticas de la industria, el RUP permite seleccionar fácilmente el conjunto de componentes de proceso que se ajustan a las necesidades específicas del proyecto. Se podrán alcanzar resultados predecibles a partir de la unión del equipo con procesos comunes que optimicen la comunicación y creen un entendimiento común para todas las tareas, responsabilidades y artefactos. Desde un único sitio web centralizado de intercambio, el Software Rational, las plataformas, herramientas y expertos de dominios proveen los componentes de proceso necesarios para el éxito.

No existen dos proyectos de desarrollo de software que sean iguales. Cada uno tiene prioridades, requerimientos, y tecnologías muy diferentes. Sin embargo, en todos los proyectos, se debe minimizar el riesgo, garantizar la predictibilidad de los resultados y entregar software de calidad superior a tiempo. RUP, es una plataforma flexible de procesos de desarrollo de software que ayuda proveyendo guías consistentes y personalizadas de procesos para todo el equipo de proyecto. Las características del proceso unificado de modelado son:

- Centrado en los Modelos: los diagramas son un vehículo de comunicación más expresivo que las descripciones en lenguaje natural. Se trata de minimizar el uso de descripciones y especificaciones textuales del sistema.
- Guiado por los casos de uso: los casos de uso son el instrumento para validar la arquitectura del software y extraer los casos de prueba.
- Centrado en la arquitectura: los modelos son proyecciones del análisis y el diseño constituye la arquitectura del producto a desarrollar.
- Iterativo e incremental: durante todo el proceso de desarrollo se producen versiones incrementales (que se acercan al producto terminado) del producto en desarrollo.
- Una de las mejores prácticas centrales de RUP es la noción de desarrollar.

1.6 Lenguajes

1.6.1 Lenguaje Unificado de Modelado (UML)

UML es una especificación de notación orientada a objetos, el cual se compone de diferentes diagramas, los cuales representan las diferentes etapas del desarrollo del proyecto. Es el lenguaje de modelado de sistemas de software utilizado por RUP; permite la especificación, visualización, construcción y documentación de elementos de la Ingeniería del Software. Los varios tipos de diagramas de UML muestran diferentes aspectos de lo que se quiere representar (22) .

1.6.2 Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems de manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales. (23)

Dentro de las numerosas ventajas que proporciona el uso de esta tecnología se pueden citar algunas (24):

- Es una fuente abierta, así que los usuarios no tienen que luchar con los impuestos sobre patentes cada año
- Independiente de la plataforma
- Usando java podemos desarrollar aplicaciones web dinámicas
- Permite que se pueda crear programas modulares y códigos reutilizables

1.7 Tecnologías actuales a considerar

1.7.1 Java EE

Java Enterprise Edition o Java EE, es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles, distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process. (25)

1.7.2 Seam Framework 2.1

Es un framework desarrollado por JBoss, una división de Red Hat. Combina a dos frameworks: Enterprise JavaBeans (EJB) y JavaServerFaces (JSF). Gracias a él se puede acceder a cualquier componente EJB desde la capa de presentación refiriéndote a él mediante su nombre de componente seam. Seam introduce el concepto de contextos. Cada componente de Seam existe dentro de un contexto. El contexto conversacional por ejemplo captura todas las acciones del usuario hasta que éste sale del sistema o cierra el navegador, inclusive puede llevar un control de múltiples pestañas y mantiene un comportamiento consistente cuando se usa el botón de regresar del navegador. Puede automáticamente generarse una aplicación web de altas, bajas, cambio y modificaciones a partir de

una base de datos existente utilizando una herramienta de línea de comandos llamada seam-gen incluida con el framework. Seam puede ser integrado con las bibliotecas de componentes JBoss RichFaces (26).

1.7.3 EJB 3

EJB 3 es una especificación que simplifica el desarrollo sobre Java 5. Mediante EJB 3 se especifica la lógica del lado del servidor en las aplicaciones empresariales sobre Java con una notable simplificación del modelo de programación pues se reemplazan los descriptores de despliegue XML por anotaciones, se elimina el uso innecesario de las interfaces y se incluye la inyección de dependencias, mecanismo mediante el cual el contenedor web provee de las componentes a los controladores que lo necesiten. (27)

1.7.4 JavaServer Faces

Es un framework para aplicaciones Java basadas en web que simplifica el desarrollo interfaces de usuario en aplicaciones Java EE (23). JSF incluye:

- Un conjunto de API para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Dos librerías de etiquetas personalizadas para Java Server Pages que permiten expresar una interfaz Java Server Faces dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Beans administrados.

1.7.5 Rich Faces 3.2 como librería de componentes JSF

RichFaces es una amplia biblioteca de componentes para JSF que posee un avanzado marco para integrar fácilmente capacidades AJAX en el desarrollo de aplicaciones de negocios. Permite a los desarrolladores ahorrar tiempo y aprovechar las características de los componentes para crear aplicaciones Web, ricas en interfaz. Proporciona componentes fáciles de utilizar con etiquetas predefinidas, y brinda capacidades AJAX (Ajax4jsf). (28)

1.7.6 Ajax4jsf

Ajax4jsf es una librería open source que se integra totalmente en la implementación de JSF usada, y extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código Javascript. Mediante este framework se puede variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargar por completo, realizar peticiones al servidor automáticas, control de cualquier evento de usuario, etc. Estas características de la librería mencionada dotan a la aplicación JSF de contenido mucho más profesional con muy poco esfuerzo. (29)

1.7.7 JPA (Java Persistence API)

Java Persistence API, más conocida por su sigla JPA, es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos, y permitir usar objetos regulares (conocidos como POJOs). (30)

1.8 Herramientas

1.8.1 Visual Paradigm 8.0

Visual Paradigm es una herramienta CASE. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. (31)

1.8.2 NetBeans IDE 7.3

NetBeans IDE es un entorno de desarrollo visual de código abierto para aplicaciones programadas mediante Java, uno de los lenguajes de programación más poderosos del momento. Con NetBeans IDE es posible elaborar potentes aplicaciones para el Escritorio, sin que cambie la forma de programar. La programación mediante NetBeans se realiza a través de componentes de software modulares,

también llamados módulos. NetBeans pone a disposición de los usuarios decenas de módulos a través de su página web, que podrás integrar en él para conseguir mejores aplicaciones. (32).

1.8.3 Jboss Developer Studio 5.0

Es el servidor de aplicaciones de código abierto más ampliamente desarrollado del mercado. Por ser una plataforma certificada Java EE, soporta todas las funcionalidades de J2EE 1.4, incluyendo servicios adicionales como clustering, caching y persistencia. JBoss es ideal para aplicaciones Java y aplicaciones basadas en la web. También soporta Enterprise Java Beans (EJB) 3.0, esto hace que el desarrollo de las aplicaciones empresariales sean mucho más simples. (33)

1.8.4 PostgreSQL Server 8.4

PostgreSQL es un sistema de gestión de bases de datos objeto relacional. Considerado estable y robusto PostgreSQL encabeza a los sistemas de gestión de bases de datos objetos relacionales desarrollados bajo licencia de código abierto. El mismo implementa a una gran parte del lenguaje de consultas SQL y permite realizar consultas complejas, soporta el uso de llaves foráneas, eventos disparadores, vistas e integridad transaccional. También puede ser extendido añadiendo nuevos tipos de datos, funciones, operadores y lenguajes procedimentales. (34)

1.8.5 PgAdmin III 1.41

PgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usan en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres.

PgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I y mucho más. La

conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas Unix), y puede encriptarse mediante SSL para mayor seguridad. (35)

En este capítulo se realizó un análisis de la variable riesgo, para conocer la situación actual del HIS y posteriormente demostrar que con la implantación de un mecanismo de licenciamiento se disminuirán las violaciones de licencia. Además se analizaron los mecanismos existentes en distintos productos informáticos, y basados en los mecanismos de licenciamiento utilizados por dichos sistemas, y contemplando los elementos de generación, validación y activación de dicho proceso se determina diseñar e implementar una infraestructura que resuelva los problemas presentes.

Se analizó la herramienta que formará parte de la propuesta tecnológica para la implementación del componente de generación de licencia. Y se decidió para el desarrollo del mismo, debido al manejo eficiente y racional de los recursos de memoria, por estar bajo licencias de código abierto y facilitar el trabajo de los desarrolladores, utilizar NetBeans IDE 7.3 como entorno de desarrollo visual.

Capítulo 2. Características de los componentes de la infraestructura

En el presente capítulo se describe y esboza la propuesta de solución para el desarrollo de una infraestructura de licenciamiento. Para ello se comienza con la realización del modelo de dominio correspondiente, en el cual se definen los principales conceptos que se emplearán. Luego se define el mecanismo de licenciamiento que se implementará y se puntualizan los requisitos funcionales y no funcionales que deben cumplir los componentes.

2.1 Modelo de dominio

El Modelo de Dominio o Modelo Conceptual es una representación visual de los conceptos u objetos que se manejan en el dominio del sistema. Los objetos o conceptos incluidos en el modelo de dominio no describen clases u objetos del software; sino entidades o conceptos del mundo real que están asociadas al problema en cuestión. Dicho modelo podrá ser utilizado como una base de las abstracciones relevantes en el proceso de construcción del sistema. (36)

2.3.1 Conceptos relacionados con el dominio

Con la finalidad de una mejor comprensión del Diagrama del Modelo de Dominio a continuación se dará una breve descripción de los conceptos encontrados en el ámbito del problema. Estos son:

- **Entidad hospitalaria:** institución de salud que solicita la instalación del sistema.
- **Identidad digital:** contiene los datos identificativos de la licencia que permiten comprobar su integridad y procedencia.
- **Distribuidor:** persona encargada de la generación del fichero de licencia, a partir de los parámetros de configuración.
- **Licencia de software:** licencia digital que contiene las cláusulas de uso del software.
- **Administrador:** personal de la entidad hospitalaria encargada de la instalación del HIS.
- **Componente de activación:** componente que genera la clave del código de activación.

2.3.2 Diagrama de Modelo de dominio

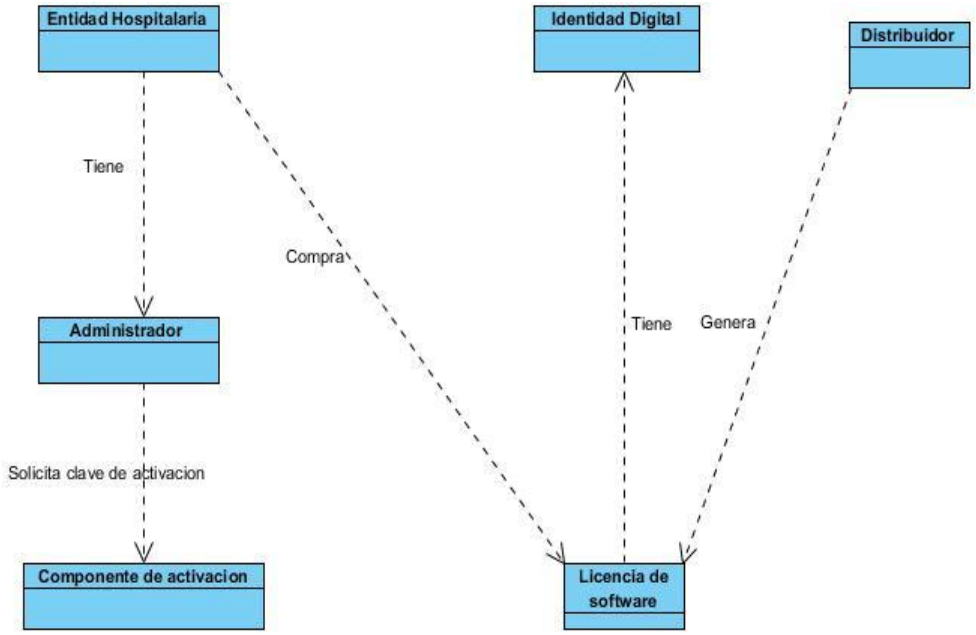


Figura 2.1 Diagrama de modelo de dominio

2.2 Definición del mecanismo de licenciamiento a implementar

Atendiendo a los métodos utilizados por los productos informáticos consultados y basados en las características esenciales y los datos a manejar en el Sistema de Información Hospitalaria del CESIM, y teniendo en cuenta que el mismo no posee un mecanismo para licenciar dicho sistema, se define realizar el proceso de la manera que a continuación se describe:

Una vez que el usuario posee el instalador del HIS, contacta a los distribuidores para solicitar una licencia. El funcionamiento del componente de generación comenzará cuando el distribuidor ingrese en el formulario los datos necesarios para crear la licencia como el número de la misma, usuario que la genera, módulos que se utilizarán, fecha de vencimiento, cantidad de Historias Clínicas Electrónicas (HCE), entidades y algunas propiedades físicas del servidor de despliegue. El componente generará un archivo XML con los datos de la licencia, utilizando el algoritmo de encriptación SHA1 calcula la función resumen del documento antes mencionado y a continuación genera un par de claves privada y pública, mediante el algoritmo RSA y con la clave privada se cifra el resumen y de esta forma se obtiene la firma digital. El componente de generación de licencia exportará un fichero compactado que contiene tres archivos, los datos de la licencia en un XML, la firma digital y la clave pública que se

utilizará posteriormente en el componente de validación. Ver Figura 2.2 Esquema de funcionamiento del componente generar licencia.

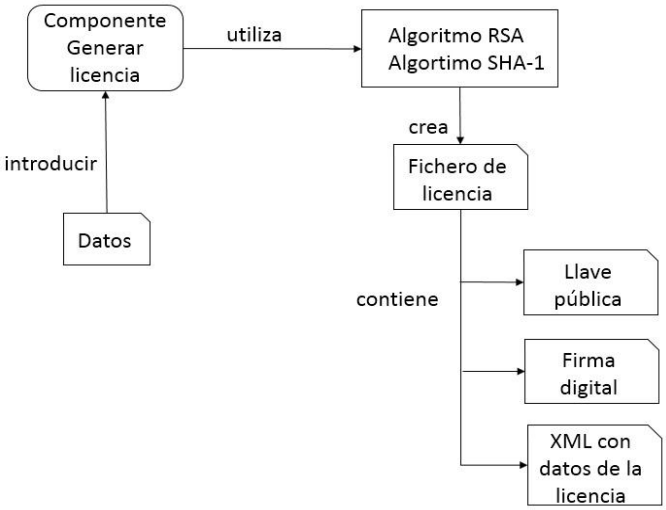


Figura 2.2 Esquema de funcionamiento del componente generar licencia.

A continuación el usuario cargará el fichero de licencia en el componente de validación, este calculará la función resumen del documento XML y descifrará la firma digital con la clave pública para verificar que ambos resultados sean iguales, siendo esta la prueba que permite asegurar la integridad del fichero de licencia. Inmediatamente el componente obtiene los datos de hardware del servidor de despliegue validando que correspondan a la licencia asociada y comprueba además que la fecha actual no supere la fecha de vencimiento. Por último genera y muestra un token de validación que se utilizará en el componente de activación. Este componente luego de crear el token de seguridad dará la opción de validar la clave de activación que tendrá que ser generada en el componente de activación. Ver figura 2.3 Esquema de funcionamiento del componente validar licencia (para validar el fichero de licencia).

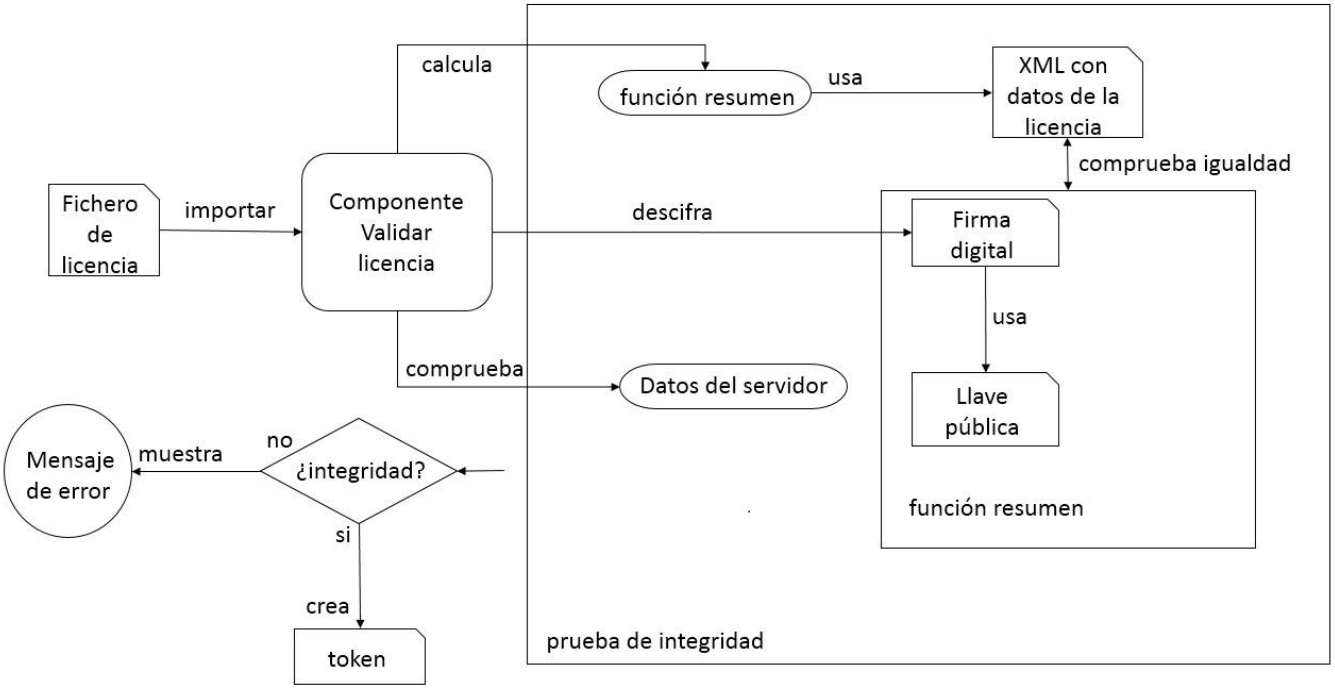


Figura 2.3 Esquema de funcionamiento del componente validar licencia (para validar el fichero de licencia).

En el componente de activación el usuario deberá introducir el token de seguridad emitido en el paso anterior. Este descifrará la información y la persistirá en la base de datos comprobando primeramente a través del número de licencia que la misma ya no exista. Seguidamente se genera y muestra la clave de activación del producto que será introducida en el componente de validación para terminar el proceso. Ver figura 2.4 Esquema de funcionamiento del componente activar licencia.

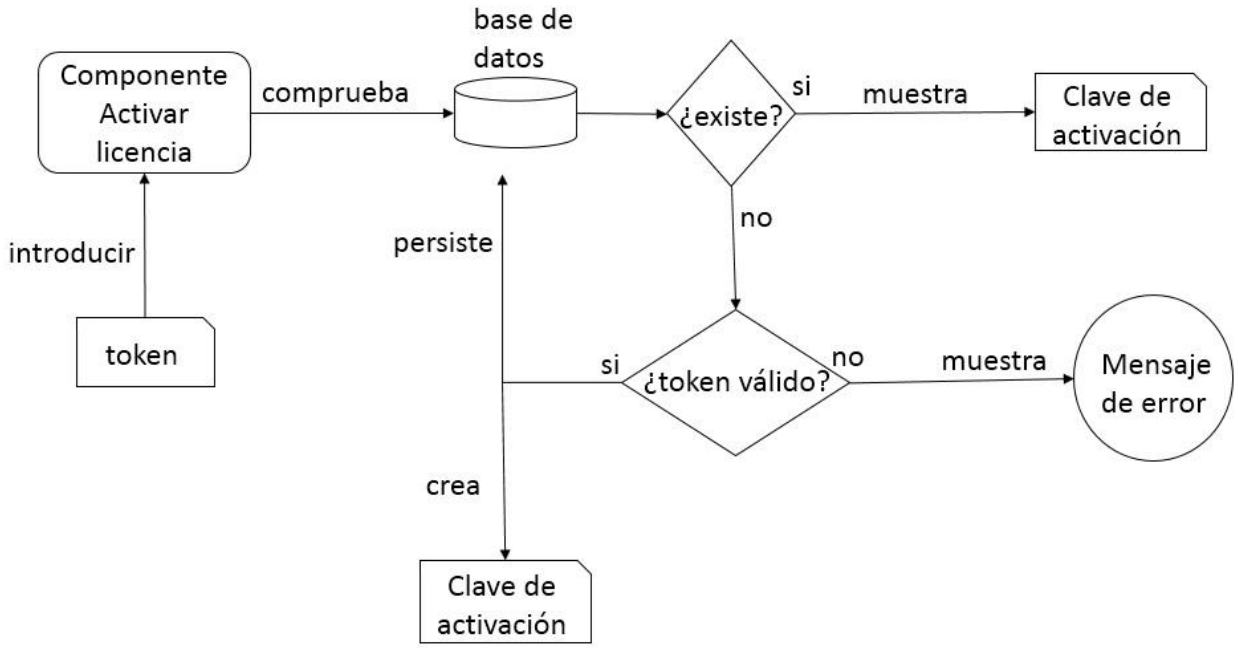


Figura 2.4 Esquema de funcionamiento del componente activar licencia.

El componente de validación recibirá la clave de activación generada en el paso anterior y comprobará la validez de la misma, probando que esta sea igual a la creada en el propio componente, pues este componente tendrá implementado el mismo método que se utiliza para crear la clave de activación en el componente de activación, y si son iguales finalmente se podrá activar el producto. Ver figura 2.5 Esquema de funcionamiento del componente validar licencia (para validar la clave de activación).

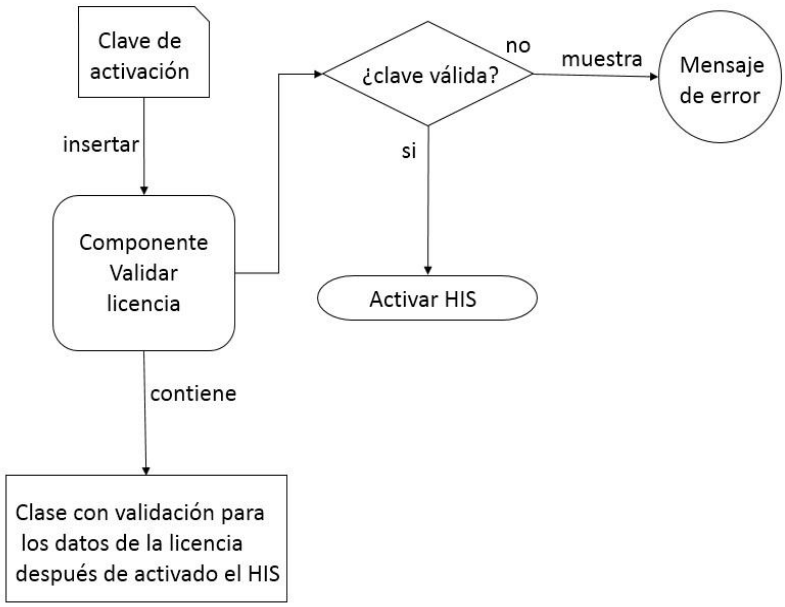


Figura 2.5 Esquema de funcionamiento del componente validar licencia para validar la clave de activación.

Este componente además posee una clase que contendrá los datos de la licencia y las validaciones necesarias para que se cumplan las restricciones pactadas, una vez que esté activo el sistema.

2.3 Especificación de los requerimientos del software

Un requerimiento de software podría definirse como una condición o capacidad que debe encontrarse o estar en un sistema o componente para satisfacer un contrato, norma, especificación u otro documento impuesto formalmente. El conjunto de todas las necesidades es el fundamento para el consiguiente desarrollo del sistema o componente. (37)

2.3.1 Requerimientos Funcionales (RF)

Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. (38).

El componente de Generar licencia será el encargado de:

- **RF1:** Generar licencia: crea una licencia en un XML a partir de los datos introducidos.

El componente de Validar licencia será el encargado de:

- **RF2:** Validar licencia: valida que el fichero de licencia proporcionado sea válido, y crea un token validador.
- **RF3:** Validar clave: valida la clave de activación proporcionada, permitiendo o no la activación del uso del HIS.

El componente de Activar Licencia será el encargado de:

- **RF4:** Activar licencia: dado un token validador crea una clave de activación.

2.3.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente están vinculados a requisitos funcionales y son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. (39)

Componente de generación

Requisitos de usabilidad:

- El componente debe trabajar con mensajes que permita al usuario conocer la situación de la ejecución de los procesos.

- Se debe trabajar en la armonía de la interfaz para asegurar que el usuario comprenda intuitivamente las funciones a realizar.

Requisitos de portabilidad:

- El componente deberá ser multiplataforma para que pueda ser soportado en varios sistemas operativos (Windows 7 o superior, Ubuntu, Debian y Nova).

- El componente podrán ser utilizado bajo cualquier arquitectura o sistema operativo donde exista instalada la máquina virtual de java 6.0 o superior.

Requisitos de rendimiento:

• El componente debe permitir la realización de varias acciones por parte del sistema, garantizando que la rapidez de realización de dichas acciones sea máxima.

Requisitos de funcionamiento:

• Se recomienda utilizar como sistema operativo Windows 7 o superior, o preferiblemente una distribución de software libre.

Componentes de activación y validación

Restricciones de diseño

Los componentes estarán dividido en las siguientes capas:

Capas físicas

- **Cliente:** computadora con cualquier tecnología o sistema operativo Firefox 4.x o superior.
- **Servidor de Aplicaciones:** servidor con cualquier tecnología o sistema operativo que soporte el Java Runtime Environment (JRE) 1.6.0_24 o superior y al JBoss AS 4.2.2.GA.
- **Servidor de Base de Datos:** servidor con cualquier tecnología o sistema operativo que soporte a PostgreSQL Server 8.4 o superior.

Capas lógicas

- **Presentación:** contiene todas las vistas y la lógica de la presentación. El flujo web se maneja de forma declarativa y basándose en definiciones de procesos del negocio.
- **Negocio:** se garantiza la seguridad a nivel de objeto y métodos.
- **Acceso a Datos:** contiene las entidades y los objetos de acceso a datos correspondientes a las mismas. El acceso a datos está basado en el estándar JPA y particularmente en la implementación del motor de persistencia Hibernate.

Interfaces de usuario

- Las ventanas del sistema contendrán claro y bien estructurados los datos, además de permitir la interpretación correcta de la información.

- La entrada de datos incorrecta será detectada claramente e informada al usuario.
- El diseño de la interfaz del sistema responderá a la ejecución de acciones de una manera rápida, minimizando los pasos a dar en cada proceso.

2.4 Definición de los Casos de Usos

2.4.1 Diagrama de Casos de Usos

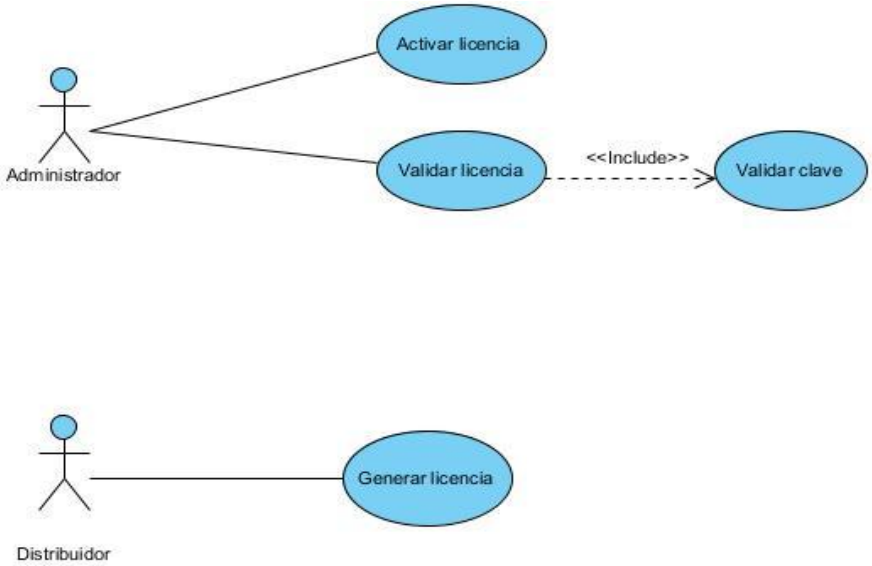


Figura 2.6 Diagrama de Casos de Usos

2.4.2 Descripción de Casos de Usos

Caso de Uso Generar licencia

Objetivo	Generar el fichero de licencia
Actores	Distribuidor
Resumen	El caso de uso inicia cuando el actor accede a la aplicación, el sistema brinda la posibilidad de introducir los criterios para generar la licencia. El actor selecciona la opción de generar, el sistema exporta un fichero.

<table>
<tr><td></td><td colspan="2">El caso de uso termina.</td></tr>
<tr><td>Complejidad</td><td colspan="2">Alta</td></tr>
<tr><td>Prioridad</td><td colspan="2">Crítico</td></tr>
<tr><td>Precondiciones</td><td colspan="2">No existen</td></tr>
<tr><td>Postcondiciones</td><td colspan="2">Se generó un fichero con los datos de la licencia</td></tr>
<tr><td colspan="3">Flujo de eventos</td></tr>
<tr><td colspan="3">Flujo básico Generar licencia</td></tr>
<tr><td></td><td>Actor</td><td>Sistema</td></tr>
<tr><td>1.</td><td>El caso de uso inicia cuando el actor accede a la aplicación.</td><td></td></tr>
<tr><td>2.</td><td></td><td>Brinda la posibilidad de introducir los criterios para generar la licencia:
Número de licencia
Cantidad máxima de usuario
Cantidad máxima de HCE
Cantidad máxima de entidades
Cantidad de días
Cantidad máxima de documentos
Lista de módulos a utilizar
Fecha de vencimiento
y permite:
Generar</td></tr>
</table>

		• Cancelar. Ver **Alternativa 1**: "Cancelar."
3.	Introduce los datos	
4.	Selecciona la opción Generar	
5.		Valida los datos. Si hay datos incompletos, ver **Alternativa 2:** "Existen datos incompletos.". Si hay datos incorrectos, ver **Alternativa 3:** "Existen datos incorrectos.". Si en el rango de fecha seleccionado, la fecha de vencimiento tiene un valor menor a la fecha actual, ver **Alternativa 4:** "Fecha incorrecta".
6.		*Exporta un fichero*
7.		Termina el caso de uso
Flujos alternos		
Alternativa 1. Cancelar		
	Actor	**Sistema**
1.	Selecciona la opción de Cancelar.	
2.		Termina el caso de uso

Flujos alternos		
Alternativa 2. “Existen datos incompletos.”		
	Actor	**Sistema**
1.		Muestra un indicador sobre los campos incompletos.
2.		Regresa al paso 3 del **Flujo Normal de Eventos.**
Flujos alternos		
Alternativa 3. “Existen datos incorrectos.”		
	Actor	**Sistema**
1.		Muestra un indicador sobre los campos incorrectos.
2.		Regresa al paso 3 del **Flujo Normal de Eventos.**
Flujos alternos		
Alternativa 4: “Fecha incorrecta”.		
	Actor	**Sistema**
1.		Muestra el mensaje de error “Fecha de vencimiento incorrecta”.
2.		Regresa al paso 3 del **Flujo Normal de Eventos.**

Caso de Uso Validar licencia

Objetivo	Crear token de validación
Actores	Administrador
Resumen	El caso de uso inicia cuando el usuario accede a la interfaz del componente, el sistema brinda la posibilidad de introducir los datos para Validar licencia, el actor introduce los datos, el sistema valida la licencia y muestra un token validador, el caso de uso termina.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Se ha generado un fichero de licencia
Postcondiciones	Se creó un token de validación.
Flujo de eventos	
Flujo básico Validar licencia	

	Actor	**Sistema**
1.	El caso de uso inicia cuando el actor accede a la interfaz del componente.	
2.		Permite: • Cargar fichero • Aceptar. Ver CU_Validar clave.

3.	Selecciona la opción Cargar fichero	
4.	Carga el archivo a la aplicación	
5.		Valida los datos. Si fichero no válido, ver **Alternativa 1:** “Fichero no válido.”
6.		*Crea un token de validación*
7.		Termina el caso de uso
Flujos alternos		
Alternativa 1. “Fichero no válido.”		
	Actor	**Sistema**
1.		Muestra el mensaje de error “Fichero no válido.”
2.		Regresa al paso 3 del **Flujo Normal de Eventos.**

Relaciones	**CU Incluidos**	*CU_Validar clave*
	CU Extendidos	No existen
Requisitos no funcionales		
Asuntos		

pendientes	

Caso de Uso Validar clave

Objetivo	Validar clave
Actores	Administrador
Resumen	El caso de uso inicia cuando el usuario accede a la interfaz del componente, el sistema brinda la posibilidad de introducir la clave de activación, el actor introduce la clave, el sistema Valida la clave, el caso de uso termina.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Se ha generado la clave de activación
Postcondiciones	Se activó el uso del HIS
Flujo de eventos	
Flujo básico Validar clave	

	Actor	**Sistema**
1.	El caso de uso inicia cuando el actor accede a la interfaz del componente.	

2.		Brinda la posibilidad de introducir los criterios para validar la clave de activación: • Clave de activación. y permite: • Aceptar • Cancelar operación. Ver **Alternativa 1:** “Cancelar.”
3.	Introduce clave de activación	
4.		Valida los datos. Si clave de activación no válida ver **Alternativa 2:** “Clave no válida.”
5.		*Se activa el uso del HIS.*
6.		Termina el caso de uso
Flujos alternos		
Alternativa 1. Cancelar		
	Actor	**Sistema**
1.	Selecciona la opción de Cancelar.	
2.		Termina el caso de uso

<table>
<tr><td colspan="4">Flujos alternos</td></tr>
<tr><td colspan="4">Alternativa 2: “Clave no válida.”</td></tr>
<tr><td></td><td colspan="2">Actor</td><td>Sistema</td></tr>
<tr><td>1.</td><td colspan="2"></td><td>Muestra el mensaje de error “Clave no válida.”</td></tr>
<tr><td></td><td colspan="2"></td><td></td></tr>
<tr><td>2.</td><td colspan="2"></td><td>Regresa al paso 3 del Flujo Normal de Eventos.</td></tr>
<tr><td rowspan="2">Relaciones</td><td>CU Incluidos</td><td colspan="2">No existen</td></tr>
<tr><td>CU Extendidos</td><td colspan="2">No existen</td></tr>
<tr><td>Requisitos no funcionales</td><td colspan="3"></td></tr>
<tr><td>Asuntos pendientes</td><td colspan="3"></td></tr>
</table>

Caso de Uso Activar licencia

Objetivo	Activar licencia
Actores	Administrador
Resumen	El caso de uso inicia cuando el actor accede a la interfaz del componente, el sistema brinda la posibilidad, de insertar el token validador, el sistema crea y muestra la clave de activación, el caso de uso termina.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Se ha generado el token de validación.
Postcondiciones	Se generó una clave de activación.
Flujo de eventos	
Flujo básico Generar licencia	

	Actor	Sistema
1.	El caso de uso inicia cuando el actor accede a la interfaz del componente.	
2.		Brinda la posibilidad de introducir los criterios para generar la clave de activación: • Token de validación y permite: • Aceptar
3.	Introduce el token de validación	
4.	Selecciona la opción Aceptar	
5.		Valida los datos. Si el token de validación no es válido, ver **Alternativa 1:** “Token de validación no válido”.
6.		*Crea la clave de activación de licencia*
7.		Persiste en la base de datos
8.		Termina el caso de uso
Flujos alternos		

Alternativa 1. "Token de validación no válido."		
	Actor	**Sistema**
1.		Muestra el mensaje de error "Token de validación no válido."
2.		Regresa al paso 3 del **Flujo Normal de Eventos.**

En este capítulo se describió la propuesta del mecanismo de licenciamiento a implantar en el HIS y el modelo de dominio del mismo, lo cual constituye una base fundamental para la comprensión de los términos que se trataron en la solución propuesta. Se definieron los requisitos funcionales para guiar el desarrollo de los componentes. Además se especificaron y describieron los casos de uso con el objetivo de conocer el comportamiento de estos mediante su interacción con los usuarios.

Capítulo 3. Análisis y diseño de los componentes de la infraestructura

En este capítulo se realiza una descripción del diseño del sistema dando a conocer la arquitectura que se utilizará para cada componente y patrones de diseño que se ponen en práctica, además se muestran los diagramas de clases del diseño que forman parte del modelo de diseño. Se ofrece una breve descripción de las clases u operaciones necesarias para el funcionamiento de los componentes.

3.1 Descripción de la arquitectura

La Arquitectura de Software se refiere a “las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos. La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad. (40)

3.1.1 Componente de generación de licencia

Este componente es una aplicación de escritorio que permite la creación de un fichero que contiene los archivos asociados a la licencia, la firma digital, y la clave pública que se utilizará en el componente de validación.

Arquitectura de n-capas

La programación por capas es una arquitectura en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño. Las capas dentro de una arquitectura son nada más que un conjunto de servicios especializados que pueden ser accesibles por múltiples clientes y fácilmente reutilizables. La necesidad de contar con porciones de la aplicación que se puedan "intercambiar" sin tener que modificar el resto de la aplicación es lo que impulsa el desarrollo en capas.

Para este componente se propone utilizar una arquitectura de 2 capas, no se utilizará una tercera capa de acceso a datos pues no es necesario persistir en una base de datos, la salida que emite dicha aplicación.

Capa de presentación: es la que ve el usuario (también se la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de acciones (realiza una validación previa para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica con la capa de negocio, en este caso está compuesta por la clase *FormularioDeLicencia*.

Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, está conformada por las clases *AplicaciónFormulario, GenerarLicencia, FirmaDigital y ElementosXML.*

3.1.2 Componentes de validación y activación de licencia

Los componentes de validación y activación de licencia son componentes web que permiten validar la integridad de la licencia a través de la firma digital, y obtener la clave de activación del producto, respectivamente.

Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es la más conocida y ampliamente adoptada en la actualidad. Hay un conjunto de procesos servidores, cada uno actuando como un gestor de recursos para una colección de recursos de un tipo, y una colección de procesos clientes, cada uno llevando a cabo una tarea que requiere acceso a algunos recursos hardware y software compartidos. Los gestores de recursos a su vez podrían necesitar acceder a recursos compartidos manejados por otros procesos, así que algunos procesos son ambos clientes y servidores. En el modelo, cliente-servidor, todos los recursos compartidos son mantenidos y manejados por los procesos servidores. Los procesos clientes realizan peticiones a los servidores cuando necesitan acceder a algún recurso. Si la petición es válida, entonces el servidor lleva a cabo la acción requerida y envía una respuesta al proceso cliente. (41)

Modelo Vista Controlador (MVC)

Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El estilo de

llamada y retorno MVC, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista. (42)

El patrón MVC divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto utiliza las siguientes abstracciones:

- Modelo: encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- Vista: muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- Controlador: reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio para el modelo o la vista.

En la arquitectura de la capa de presentación basada en MVC, JSF ofrece una clara separación entre el comportamiento y la presentación. Une los familiares componentes UI con los conceptos de la capa-Web sin limitarnos a una tecnología de script o lenguaje de marcas particular.

Los beans a los que se vinculan los formularios JSF son el modelo. También contienen acciones, que son una extensión de la capa del controlador y delegan las peticiones del usuario a la capa de la lógica de negocio. Las páginas Facelets con etiquetas JSF personalizadas son la capa de la vista. El Servlet Faces proporciona la funcionalidad del controlador.

Para el caso del modelo se utiliza Hibernate como herramienta de Mapeo objeto-relacional (ORM), el cual es la implementación para EJB y el API para la persistencia de java o JPA.

En el caso de la vista está desarrollada con JSF porque permite la creación de interfaces de usuarios para aplicaciones web, se usan componentes Seam de interfaz de usuario y las bibliotecas de componentes Richfaces y Ajax4jsf.

Para el controlador se utiliza JBossSeam como Framework de integración entre los componentes visuales y los de acceso a datos.

3.2 Modelo de diseño

El Modelo de Diseño es un modelo de objetos que describe la realización de los casos de uso, es abarcador y está compuesto por artefactos que engloban todas las clases del diseño, subsistemas, paquetes, y las relaciones entre ellos. Se centra principalmente en cómo los requisitos funcionales y no funcionales, junto a otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar.

3.3 Patrones de diseño

Por lo general, durante la elaboración del diseño se utilizan patrones. Estos expresan esquemas para definir relaciones entre clases y objetos, con las que construir software.

La etapa de implementación requiere de un buen diseño de clases basado en patrones que permitan aumentar la reutilización de código y contribuir al óptimo manejo de sus instancias. Seam asegura la persistencia de datos mediante la abstracción que brinda Hibernate del gestor de bases de datos a utilizar y al mapeo de las entidades del sistema.

EntityManagerFactory que es uno de los componentes principales de la Arquitectura JPA, es el encargado de crear objetos de EntityManager cuando el mismo se inyecte en algún contexto de la aplicación. Dicho componente implementa el patrón de diseño **Abstract Factory** permitiendo el acceso a la base de datos. EntityManager es la interfaz principal de JPA utilizada para la persistencia de las aplicaciones, cada instancia puede realizar operaciones como inserción, lectura, modificación y eliminación (CRUD por sus siglas en inglés) sobre un conjunto de objetos persistentes.

Hibernate implementa también el patrón **Active record** donde una fila en la tabla de la base de datos se envuelve en una clase, de manera que se asocian filas únicas de la base de datos con objetos del lenguaje de programación usado. También implementa los patrones de mapeo: **Identity field, Foreign Key Mapping** y **Association Table Mapping**. El primero centra su atención en la forma en que los objetos se relacionan en memoria, consiste básicamente en la generación de un único ID para definir la identidad de una entidad. El segundo es utilizado para mapear relaciones de uno a muchos y el tercero para mapear relaciones de muchos a muchos.

Hibernate implementa además los patrones de comportamiento **Identity map** y **Lazy Load**. El primero es utilizado para evitar tener en memoria dos representaciones distintas del mismo objeto en una

transacción de negocio; funciona como una caché de objetos de negocio. El segundo resuelve problemas de carga desmesurada y dependencias circulares o sea optimiza la carga de datos de la base de datos manteniendo en memoria sólo los que se invoquen en cada momento.

Uno de los patrones más importantes que implementa Hibernate es el patrón **Query object** que se encarga de interpretar la estructura de objetos y traducirlos a consultas (querys) SQL. Mediante este patrón se pueden crear estas consultas, haciendo referencia a las clases y sus campos, en lugar de tablas y columnas independientemente del esquema en que puedan estar.

Los patrones **GRASP** (Patrones para asignar responsabilidades) tienen una importante utilidad en el diseño. Se utilizaron con el objetivo de asignarle a las clases las tareas que podían realizar según la información que poseían poniéndose de manifiesto el patrón **Experto,** en las clases *Token, FirmaDigital y ElementoXML,* entre otras con las mismas características. Además los patrones, **Creador** para crear las instancias de otras clases en correspondencia con la responsabilidad dada y el **Controlador** que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado, evidenciándose en las clases *ActivadorLicencia, GeneralLicencia y ComponenteValidación.*

Otro de los utilizados es la **Alta cohesión**, que garantiza que la información contenida en las clases sea coherente y que, en la medida de lo posible, esté relacionada con las mismas. Además el **Bajo acoplamiento** con el fin de tener las clases lo menos ligadas posibles, de tal forma que, en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre estas.

Estos dos patrones se evidencian con el uso del MVC y la arquitectura n-capas, el cual permite dividir las clases según las funciones que realizan, lo que se puede apreciar en los diagramas de paquetes (ver documento Modelo de diseño (43)) y en los diagramas de clases.

3.4 Diagramas de clases

A continuación se representarán y describirán las clases más importantes en la solución que se brinda para cada componente.

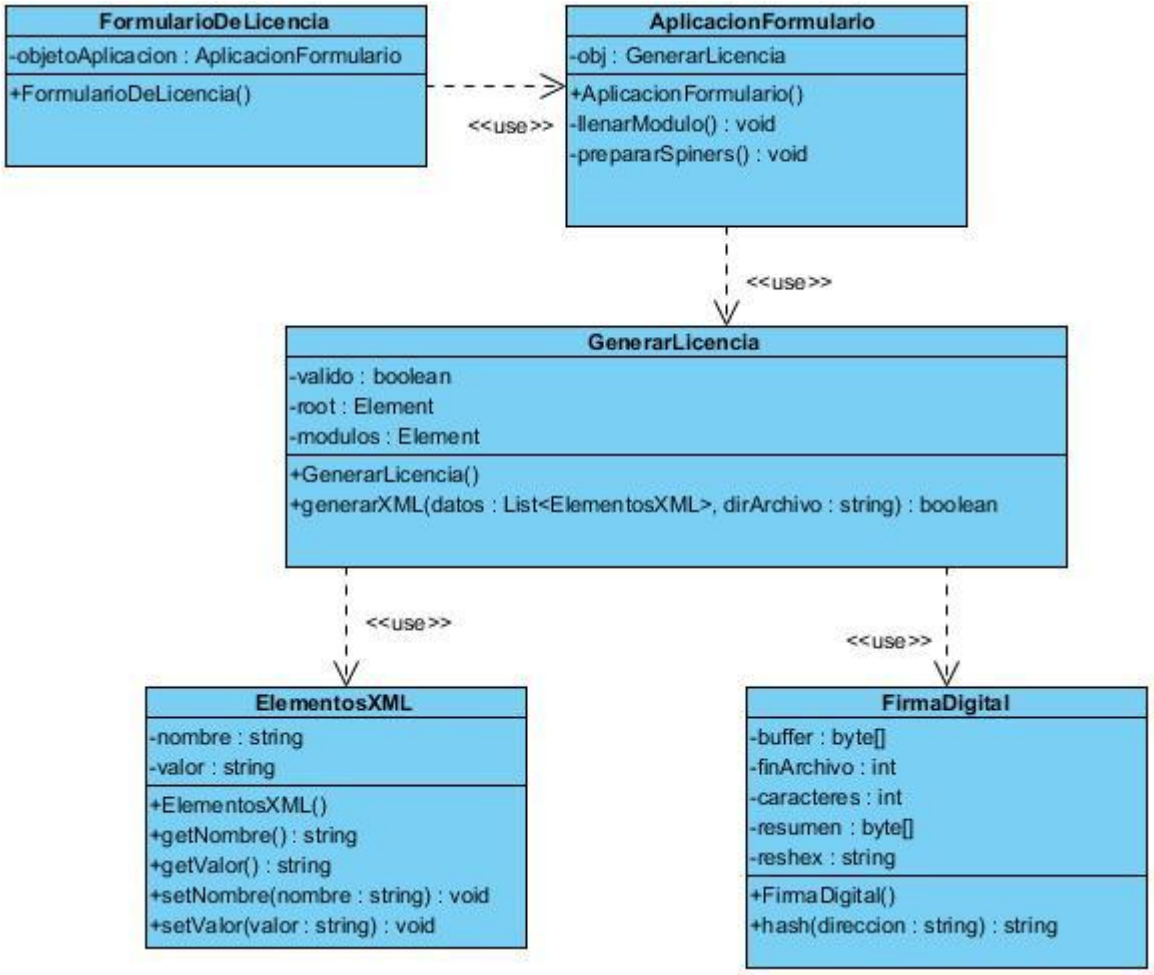


Figura 3.4 Diagrama de clase CU_Generar licencia

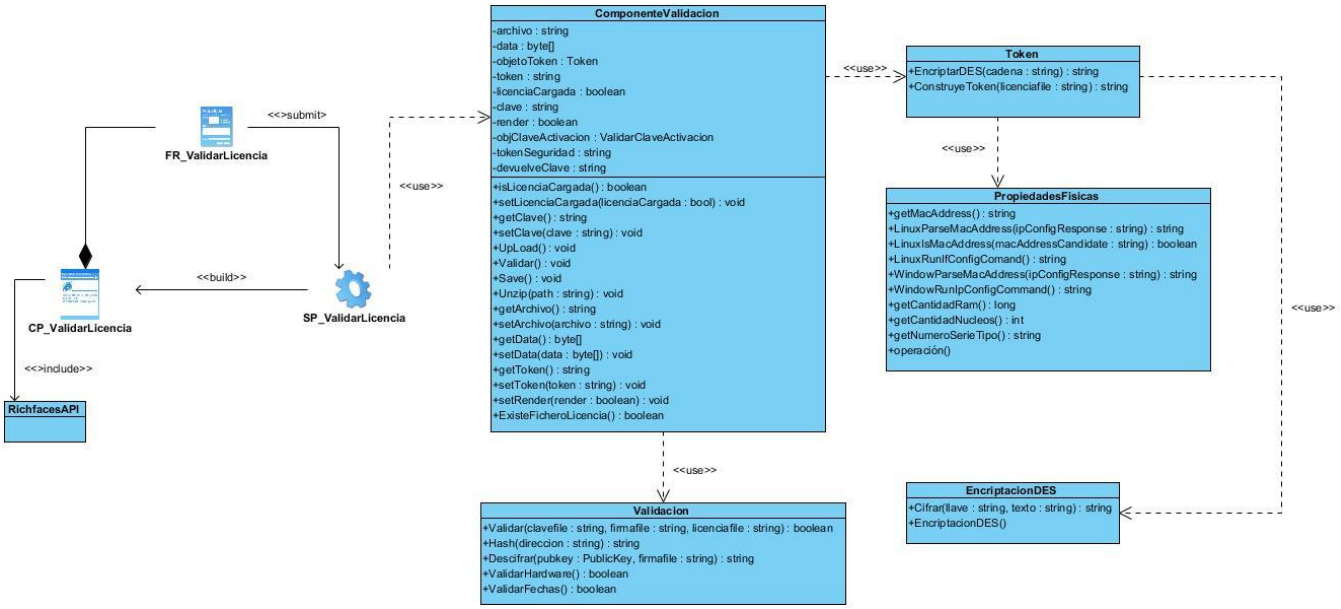


Figura 3.5 Diagrama de clase CU_Validar licencia

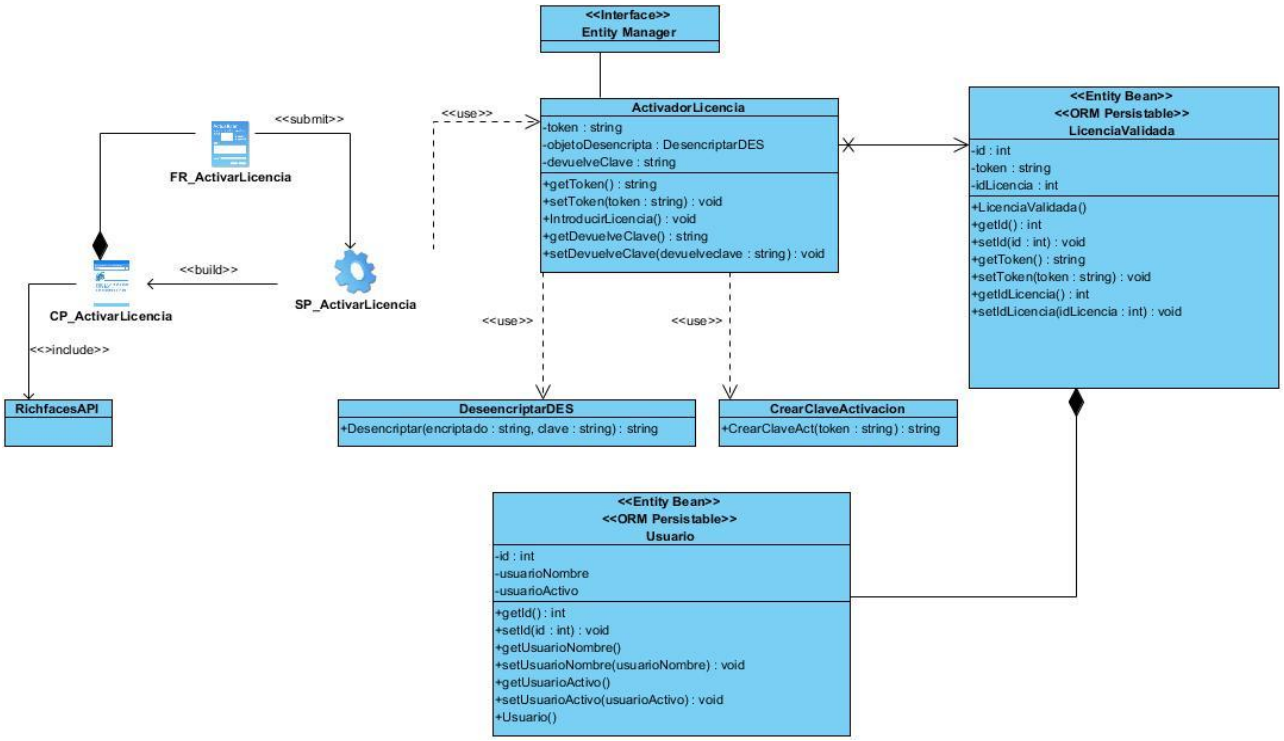


Figura 3.6 Diagrama de clase CU_Activar licencia

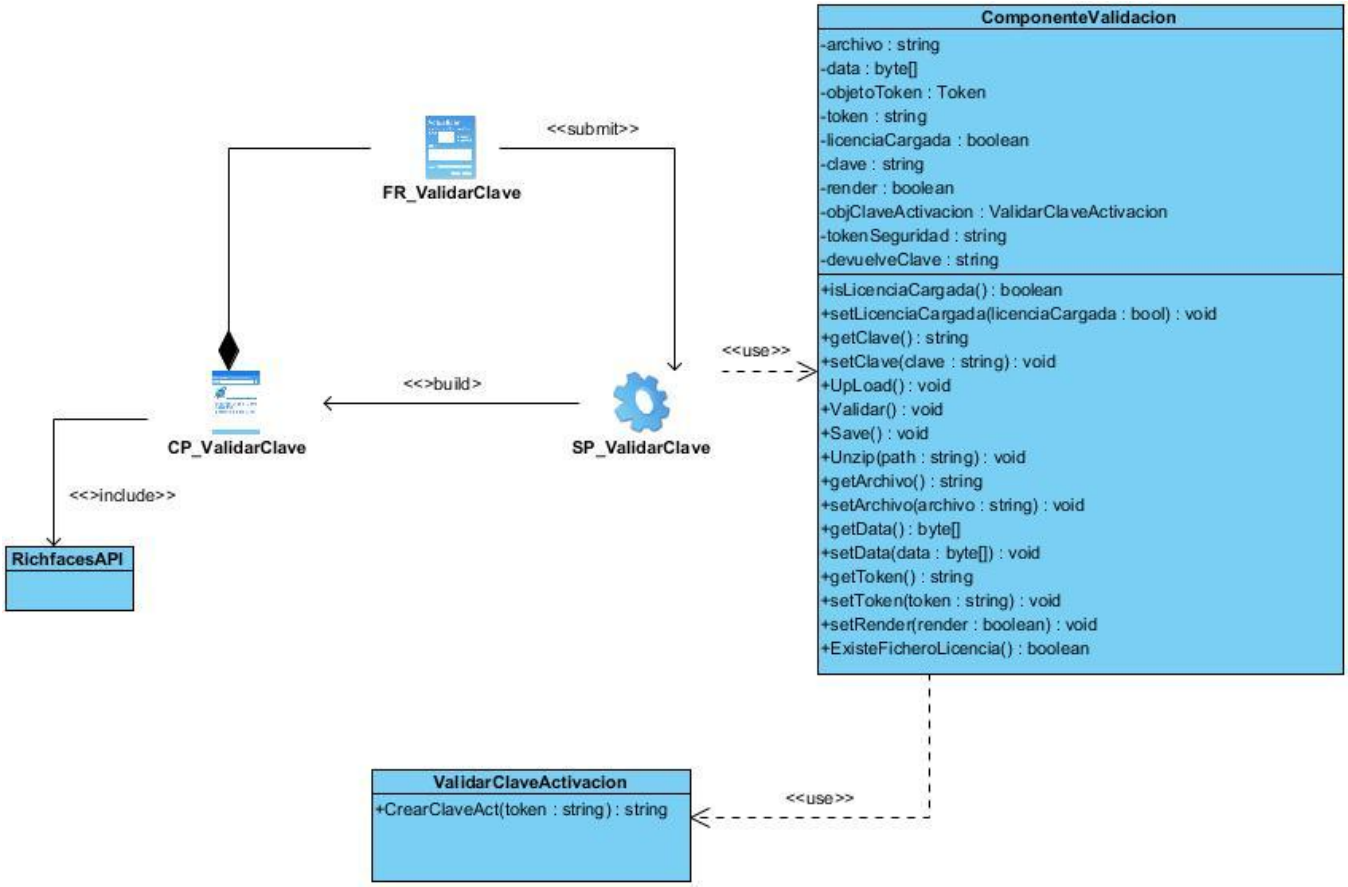


Figura 3.7 Diagrama de clase CU_Validar clave

3.4.1 Descripción de las clases

Componente Generar licencia

- **Clase GenerarLicencia:** clase encargada de generar el archivo XML donde se guardan todos los datos que conforman la licencia del sistema.
- **Clase ElementosXML:** clase encargada de crear las propiedades de los elementos que conforman en su conjunto el documento XML con los datos de la licencia del sistema.
- **Clase FirmaDigital:** clase encargada de generar la firma digital de los datos obtenidos en el formulario.

- **Clase FormularioDeLicencia:** clase encargada de ejecutar la aplicación de escritorio.
- **Clase AplicacionFormulario:** clase encargada de validar los datos de entrada y devolver un fichero con los archivos creados durante el proceso de firma digital.

Componente Validar licencia

- **Clase ComponenteValidacion:** es la encargada de manejar los datos que provienen del fichero de licencia para generar el token de seguridad que se utilizará en el componente de activación.
- **Clase Validar: :** es la encargada de realizar todas las validaciones pertinentes en cuanto a la fecha de vencimiento, los datos de hardware, calcular el resumen del documento de licencia y validar que el mismo no haya sido modificado**.**
- **Clase EncriptacionDES:** es la clase que implementa el algoritmo de encriptación DES.
- **Token:** Es la clase encargada de crear el token de seguridad a partir de los datos de la licencia y las propiedades físicas, además de encriptar el token.
- **PropiedadesFisicas:** es la encargada de obtener los datos de hardware de la estación de trabajo.
- **ValidarClaveActivacion:** es la encargada de validar que la clave de activación coincida con la que genera el componente de activación.

Componente Activar licencia

- **ActivadorLicencia:** es la encargada de manejar los datos que contiene el token encriptado, persistir en la base de datos las licencias generadas y crear la clave de activación.
- **DesencriptarDES:** es la encargada de descifrar el contenido del token para que pueda ser utilizada su información por el componente de activación.
- **CrearClaveActivacion:** es la encargada de crear la clave de activación del producto.

En este capítulo se describió la arquitectura de los componentes, la cual constituye una base fundamental para la construcción del software. Además se definieron los patrones de diseño que se aplicarán, para expresar las dependencias entre clases y objetos. Se presentaron los diagramas de clases del diseño y además se brindó una breve descripción de las clases que los componen, dando a

conocer las relaciones entre ellas para lograr un mejor entendimiento en función de brindar una solución al problema planteado.

Capítulo 4. Implementación de los componentes de la infraestructura

En este capítulo se introduce el flujo de trabajo de implementación, partiendo de los resultados obtenidos en el diseño. Primeramente, se esboza y describe el Modelo de datos, para dar a conocer la estructura donde se almacenará toda la información requerida. Luego se definen las estrategias de codificación y estándares a utilizar, así como la forma en que se tratarán los errores. También se muestra el modelo de implementación, que está compuesto por el diagrama de componentes, y el diagrama de despliegue. Además se calcula el riesgo de violaciones de licencias después de integrar la infraestructura al HIS, obteniendo una nueva calificación.

4.1 Modelo de datos

El Modelo de datos es la traducción del análisis de requisitos al esquema conceptual, mediante una representación gráfica de las entidades y sus relaciones. Es usado para definir el mapeo entre las clases del diseño y las estructuras de datos. Está compuesto por entidades, atributos y sus relaciones.

- Las entidades son objetos de los que el sistema necesita guardar información.
- Los atributos son las características asociadas a una entidad. Estos pueden ser clasificados en obligatorios, opcionales, claves foráneas y claves primarias (estas se dividen en simples y compuestas).
- Las relaciones, por su parte, muestran la forma en que dos entidades se asocian

En general un modelo de datos es la estructura o representación física de las tablas en una base de datos.

Teniendo en cuenta los datos de la licencia que se necesitan recoger para el funcionamiento del mecanismo de licenciamiento del HIS se representa el siguiente modelo de datos a utilizar:

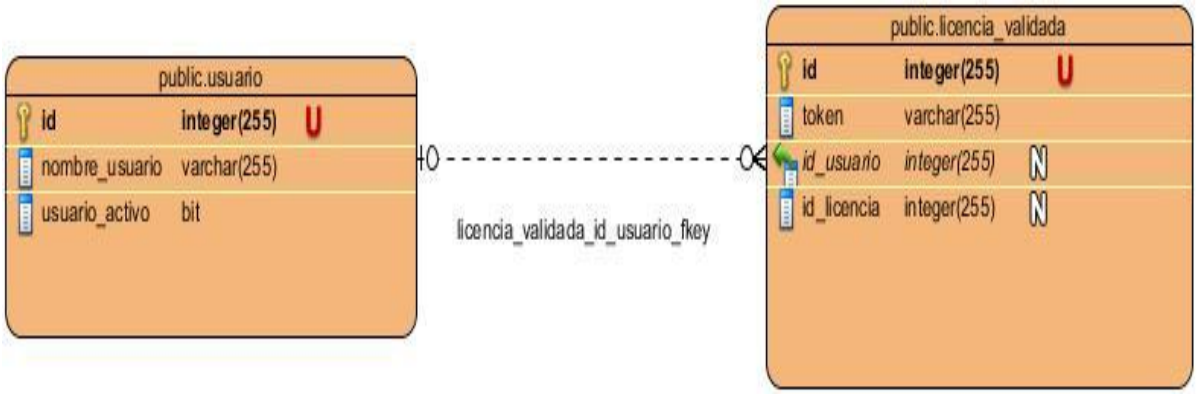


Figura 4.1 Modelo de datos.

Descripción de las tablas de la base de datos

A continuación se describen las entidades pertenecientes al modelo de datos anterior.

Nombre: licencia_validada		
Descripción: la tabla almacena los datos, id de licencia y token de validación, para poder verificar con los mismos la existencia de dicha licencia en algún servidor.		
Atributo	**Tipo**	**Descripción**
Id	integer	Identificador necesario en cada entidad para las referencias en las relaciones entre tablas.
id_licencia	integer	Es una cadena de números que identifica a la licencia que está asociada al código de activación, representando a la misma de forma unívoca.
Token	varchar	Es una cadena de caracteres, encriptada, con los datos de asociados a la licencia en cuestión.

Tabla 4.1 Descripción de la tabla: licencia_validada

Nombre: usuario		
Descripción: la tabla almacena el usuario del personal autorizado a crear licencias, e indica si está activo o no su permiso.		
Atributo	**Tipo**	**Descripción**
Id	integer	Identificador necesario en cada entidad para las referencias en las relaciones entre tablas.
nombre_usuario	varchar	Nombre de usuario del distribuidor que genera la licencia.
usuario_activo	boolean	Indica si el usuario tiene o no activo los permisos de crear licencias.

Tabla 4.2 Descripción de la tabla: usuario

4.2 Estrategias de codificación. Estándares y estilos a utilizar.

La legibilidad en el código fuente de un sistema es un atributo de calidad de primer orden. La programación es mucho más una tarea de comunicación con otras personas que una comunicación con una computadora, por eso es importante establecer y respetar estándares de codificación y lograr que la estructura del código fuente refleje la estructura lógica del programa. Para la implementación de los componentes propuestos se han seguido las pautas de desarrollo del Sistema de Información Hospitalaria (HIS). Ver documento Estrategia, estructura, métodos y pautas para la implementación (43).

4.3 Tratamiento de errores

Componente de generación

Para asegurar la integridad de la aplicación así como el funcionamiento correcto y eficiente del componente de generación de licencia, con vistas a controlar posibles situaciones inesperadas se trabaja en el control de las excepciones con énfasis en el manejo de los datos de entrada del

formulario de licencia. Se utiliza la clase IOException y la instrucción try-catch las cuales especifican controladores para diferentes excepciones.

Componentes de validación y activación

En los componentes de validación y activación se llevó a cabo el control de las excepciones en las clases y métodos vulnerables a situaciones inesperadas. Se utiliza la clase Validator y el componente FacesMessages del framework Seam para tratar errores y mostrar los mensajes de información al usuario respectivamente. Los mensajes se manejan a través del objeto FacesMessages el cual se incluye en las clases controladoras.

4.4 Modelo de implementación

Luego de haber descrito las clases, atributos y sus relaciones se puede comenzar la implementación del sistema. La construcción del sistema debe ser en todo momento consistente con la documentación creada por el Analista y con la base de datos creada por el diseñador de Base de Datos, es ahora donde se construye el sistema en términos de componentes: ejecutables, ficheros de código fuente y scripts. El objetivo principal es desarrollar la arquitectura y definir la organización del código. Durante esta etapa se obtiene el Modelo de Implementación que relaciona componentes y subsistemas. Para representar este tipo de modelo se emplea el Diagrama de Componentes (DC).

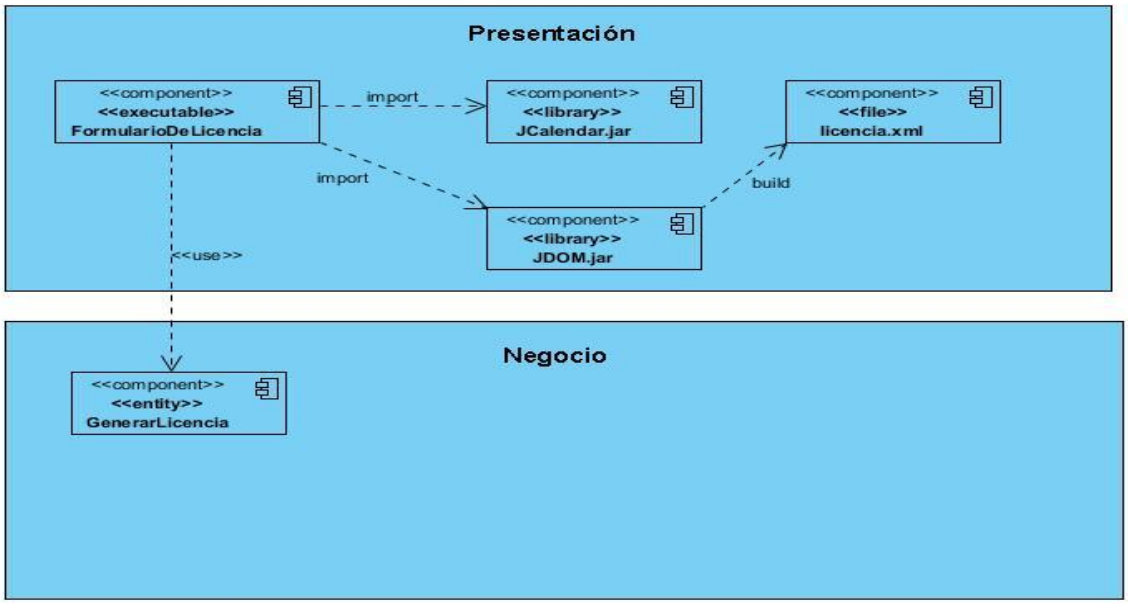


Figura 4.2 DC Componente de Generación

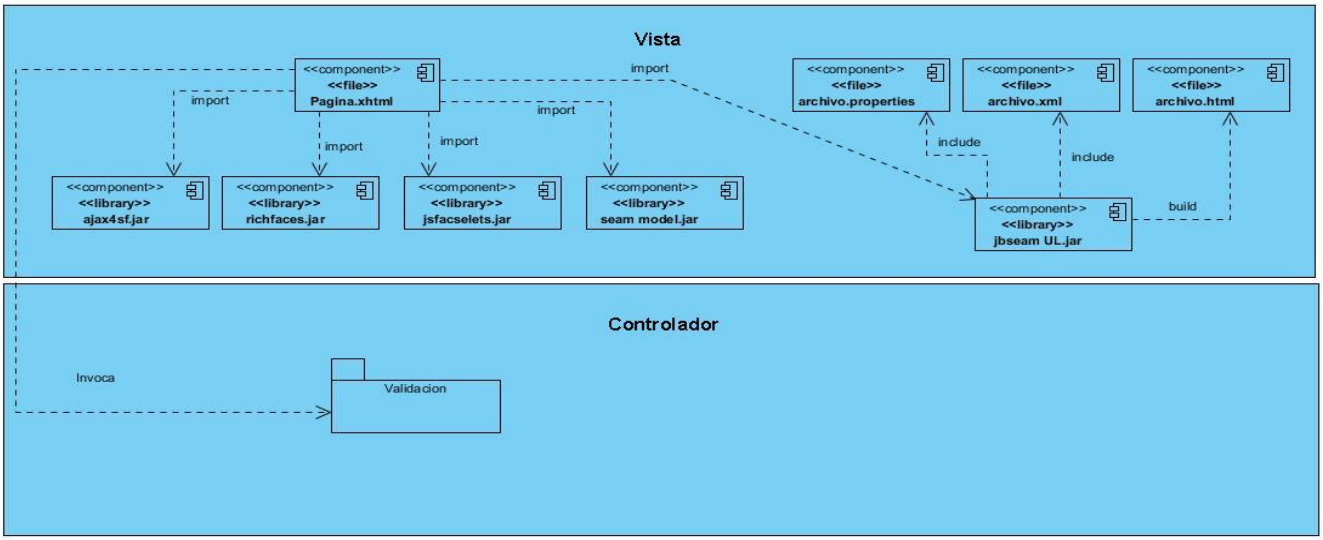
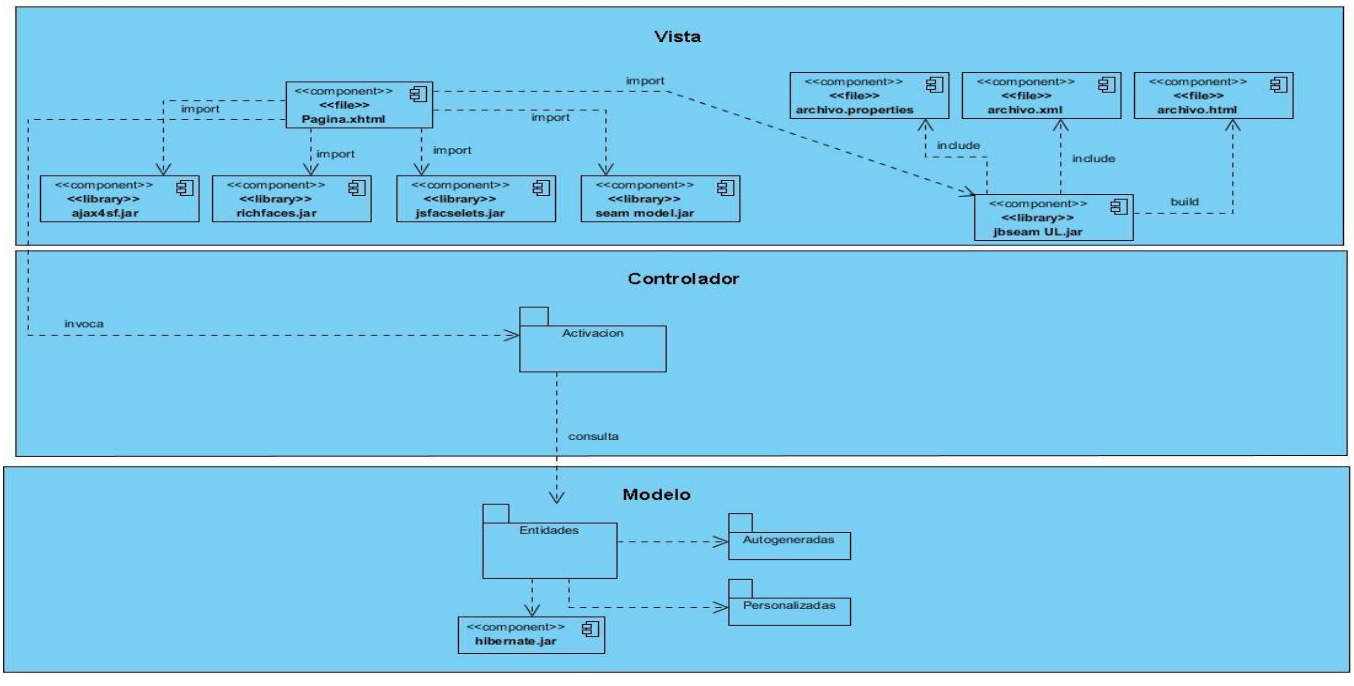


Figura 4.3 DC Componente de Validación

Figura 4.4 DC Componente de Activación

4.5 Modelo de despliegue

El diagrama de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes.



Teniendo en cuenta las características de los componentes implementados, el diagrama de despliegue quedó como se muestra a continuación:

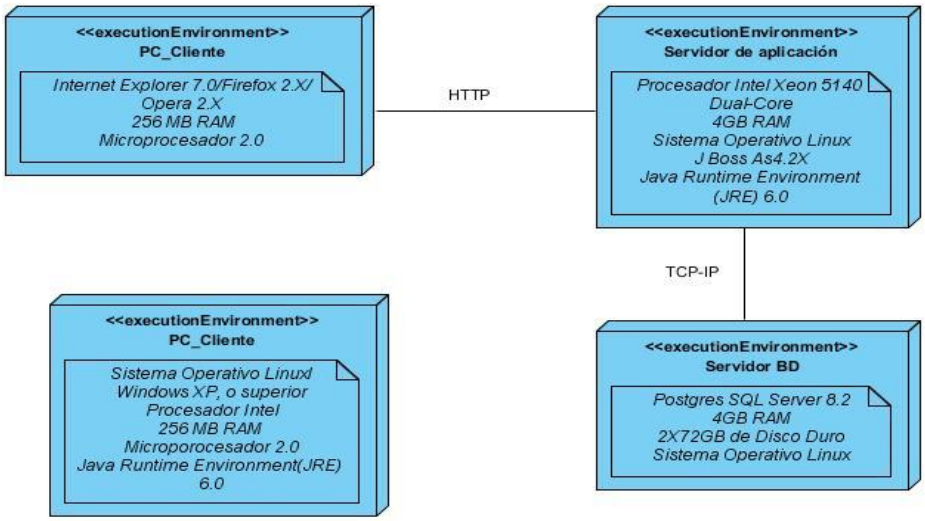


Figura 4.5 Diagramada de despliegue

Para la utilización del componente de generación de licencia se requiere de una pc cliente. Mientras que en la utilización del componente de validación de licencia que está integrado al sistema de información hospitalaria el usuario debe conectarse mediante una PC cliente utilizando un navegador web, realizando las peticiones por el protocolo HTTP, que serán procesadas por el servidor de aplicaciones que enviará la respuesta al cliente. En el caso del componente de activación de licencia el servidor de aplicaciones también hará peticiones mediante el protocolo TCP-IP al servidor de base de datos.

4.6 Estructura de las salidas de los componentes

Estructura del fichero del componente de Generar licencia

El fichero de licencia es generado por la aplicación de escritorio a través de un formulario y consiste en un archivo compactado que contiene 3 elementos. El primero de ellos es un documento XML que contiene todos los datos que se necesitan para generar una licencia del sistema. El segundo es un archivo que contiene la llave pública generada por la aplicación de escritorio como parte del proceso de firma digital y que será utilizado por el componente de validación para comprobar la integridad de la

licencia. El tercero es un archivo que contiene el resultado de aplicar el proceso de firma digital a los datos de la licencia que contiene el documento XML.

Estructura del token de validación del componente de Validar licencia

El token de validación consiste en una cadena de caracteres que contiene algunos datos de la licencia. Este token está formado por el id de la misma y los datos de hardware más distintivos del servidor donde será desplegado el sistema luego de habérseles aplicado un algoritmo de encriptación.

Estructura del token de activación (clave de activación) del componente de Activar licencia

El token de activación es el que se utilizará para finalmente activar el producto. Este se construye aplicándole un algoritmo de encriptación al token generado por el componente de validación para asegurar que no pueda reproducirse posteriormente en otro servidor.

4.7 Riesgo de violación de licencia después de integrada la infraestructura al HIS

Luego de integrado el componente de validación de licencia al HIS, se realizó una evaluación de la amenaza en cuestión, utilizando el modelo DREAD con las modificaciones realizadas, y se determinaron los siguientes valores para cada una de las variables implicadas:

Daño potencial: el atacante no podrá redistribuir el HIS debido a que las licencias están identificadas con características únicas del servidor de despliegue, tales como el número de serie del disco duro, dirección MAC, lo que evita que este producto sea usado en alguna PC para la cual no se ha comprado con anterioridad una licencia. Esta además posee un token de validación el cual está asociado al código de activación, que es único, y este a su vez es persistido en una base de datos en la cual se verifica que se correspondan los valores asociados de la misma, y llevando un control estricto de las licencias existentes, por lo que se obtiene una clasificación de Baja (1).

Reproductividad: con el objetivo de asegurar la integridad de las licencias a generar se utilizó la Firma Digital. Este proceso permite calcular la función resumen de un texto claro utilizando el algoritmo SHA1 y a través del algoritmo de cifrado RSA genera un par de claves que utiliza para firmar digitalmente el resumen calculado. Este mecanismo es efectivo para evitar que se modifique o se reproduzca una determinada información de carácter confidencial porque al cifrar con la clave privada que genera RSA la información que se obtiene es única y sólo puede ser descifrada con la clave

pública del emisor de la firma. Debido a estas razones incurrir en la violación de licencia sería muy difícil y casi imposible porque sería necesario romper estas normas de seguridad, por lo que se obtiene una clasificación de Baja (1).

Explotabilidad: para el intercambio de información entre componentes se utilizan mecanismos de encriptación como la firma digital y el algoritmo DES que hacen que la información sea ilegible para individuos comunes, solo aquellos no poseen un profundo conocimiento sobre el tema de la piratería (hacker) pudieran ser capaces de descubrir alguna vulnerabilidad, por lo que se obtiene una clasificación de Baja (1).

Afectación a las entidades hospitalarias: el componente de validación contiene una clase que se encarga de validar que los cambios que se realicen en el sistema estén pactados dentro de los datos de la licencia en cuestión, dentro de estos se contemplan los módulos que se permiten utilizar, así, cuando el usuario requiera añadir alguno de los módulos existente el servidor comprobará que se tiene permisos para hacerlo, y además se utiliza protección por fecha para los mismos, en dependencia de lo establecido en el contrato de licencia (por días o por fecha de vencimiento), por lo que se obtiene una clasificación de Baja (1).

Detectabilidad: esta amenaza está sujeta al compromiso ético de los distribuidores del producto que tienen acceso a los servidores donde se encuentre desplegado el componente de activación del HIS, pues estos son los únicos que tienen permisos de generar licencias, y estos pudieran incurrir en la creación de licencias no autorizadas legalmente, por lo que se obtiene una clasificación de Baja (1),

Como se muestra en la siguiente tabla, la amenaza analizada obtuvo un total de 5 puntos, clasificándose como Baja.

Amenaza	**D**	**R**	**E**	**A**	**D**	**Total**	**Clasificación**
Violación de licencia por copia del compilado	1	1	1	1	1	5	Bajo

Tabla 4.1 Calificación de amenaza del HIS.

Concluido este capítulo se pudo dar solución a los requisitos funcionales y no funcionales especificados con anterioridad. Para ello se estableció el Modelo de Datos del sistema, el Diagrama de Despliegue, el Diagrama de Componentes y se realizó una descripción de las tablas de la base de

datos. Durante el proceso de implementación se cumplió con los estándares y estilos definidos, lo que permitió obtener tres componentes con código fuente entendible para todos los programadores, y fácil de mantener en el transcurso del tiempo.

Conclusiones

Con la realización de la presente investigación se puede concluir que:

- Con la modificación de los parámetros, daño potencial, afectación a usuarios y Detectabilidad de la metodología DREAD, se logró cuantificar de forma adecuada el valor para la amenaza violación de licencia por copia del compilado.
- Con la generación de un token de seguridad en el componente de validación se logró representar unívocamente a la licencia y al servidor de despliegue donde se utilizará el HIS.
- El riesgo de violaciones de licencias para el HIS que anteriormente era de 13 puntos (Alta), se redujo a 5 puntos (Baja), con la integración de estos 3 componentes.

Recomendaciones

Con el objetivo de mejorar y seguir perfeccionando la infraestructura de licenciamiento se recomienda:

- Generar las claves de la firma digital a través de una entidad certificadora que pertenezca al CESIM o a la Universidad.

Referencias bibliográficas

1. **Medina, David Angel.** UNAM. [En línea] [Citado el: 21 de febrero de 2014.] http://biblio.juridicas.unam.mx/libros/1/322/16.pdf.

2. **Universidad Nacional de Lujan. [En línea] [Citado el: 28 de abril de 2014.] http://www.seguridadinformatica.unlu.edu.ar/?q=glossary/term/3.**

3. Universidad de Navarra. [En línea] [Citado el: 28 de abril de 2014.] https://www.unav.es/SI/servicios/seguridad/faq.html#13.

4. Microsoft Developer Network. [En línea] [Citado el: 31 de marzo de 2014.] http://msdn.microsoft.com/es-es/library/ms733083%28v=vs.110%29.aspx.

5. Danie P.F. [En línea] [Citado el: 1 de abril de 2014.] http://fortinux.com/wp-content/uploads/2010/12/Analisis-y-Modelado-de-Amenazas.pdf.

6. Microsoft DeveloperNetwork. [En línea] [Citado el: 1 de abril de 2014.] http://msdn.microsoft.com/es-es/library/ms172104%28v=vs.80%29.aspx.

7. Coras. [En línea] 1 de abril de 2014. http://coras.sourceforge.net.

8. Microsoft. [En línea] [Citado el: 20 de marzo de 2014.] http://msdn.microsoft.com/en-us/library/ff648644.aspx#c03618429_009.

9. Marin, Diego Fernandez. slideshar7. [En línea] [Citado el: 12 de febrero de 2014.] www.slideshare.net/dfmarin/patentes-copyright-y-licencias-en-el-software.

10. Mazzini, Juan José Blossiers. *DESCUBRIENDO LOS DELITOS INFORMÁTICOS – LA NUEVA DIMENSIÓN DE LOS ILÍCITOS PENALES.* Peru : Normas Legales S. A, 1999.

11. Navarro, Adolfo Castillo. Universidad de Sonora. [En línea] [Citado el: 13 de febrero de 2014.] http://ntic.uson.mx/plataforma/fotosntic/documentos/conceptos_y_definiciones_inmersos_en_las_NTIC.pdf .

12. Jair, Jona. [En línea] [Citado el: 21 de marzo de 2014.] http://es.scribd.com/doc/99662942/Aspectos-Legales-Del-Software.

13. Bravo, Msc Jorge Lopez. Scribd. [En línea] [Citado el: 28 de abril de 2014.] http://es.scribd.com/doc/59622988/Criptografia.

14. lic Adrian Pousa. [En línea] [Citado el: 28 de abril de 2014.] http://postgrado.info.unlp.edu.ar/Carreras/Especializaciones/Redes_y_Seguridad/Trabajos_Finales/Pousa_Adrian.pdf.

15. Universidad de Mendoza. [En línea] [Citado el: 21 de marzo de 2014.] www.um.edu.ar/catedras/PPT03/document/Modulo_III/Papers/pki/pki-conceptos.

16. Osma, Anibal. [En línea] [Citado el: 21 de marzo de 2014.] http://es.scribd.com/doc/186596524/Trabajo-de-Seguridad-de-Redes-Criptografia.

17. [En línea] [Citado el: 19 de febrero de 2014.] http://www.md5database.net/.

18. gaussianos. [En línea] [Citado el: 21 de marzo de 2014.] http://gaussianos.com/algoritmos-hash-ii-atacando-md5-y-sha-1/.

19. ZAO, Kaspersky Lab. Russian Federation, 2010.

20. Photoshop, Adobe. *Manual de ayuda.* Estados Unidos : Adobe Systems Incorporated, 2008.

21. *Sistema para la administración de licencias del Sistema producto DalasQ (DalasLicenseManager).* Alfonso, Delmis Velázquez. La Habana : s.n., 2013.

22. Microsoft. Microsoft Developer Network. [En línea] [Citado el: 26 de febrero de 2014.] http://msdn.microsoft.com/es-es/library/bb972214.aspx.

23. Ajay Vohra, Deepak Vohra. *Pro XML Development with Java TM Technology.*

24. Rodriguez, jose Maria Alvarez. Universidad de Oviedo. [En línea] [Citado el: 26 de febrero de 2014.] http://dspace.sheol.uniovi.es/dspace/1/TDJoseMariaAlvarezRodriguez.pdf.

25. Cali. [En línea] [Citado el: 26 de febrero de 2014.] www.cali.gov.co.

26. Allen, Dan. *Seam in Action.* s.l. : Manning Publications, 2008.

27. Roman, Ed, Amble, r Scott y ., Tyler Jewell. *Mastering enterprise JavaBeans.* 2002.

28. Red Hat. *RichFaces developer Guide.* 2007.

29. —. *Ajax4jsf Developer Guide.* 2007.

30. Java Persistence API. [En línea] [Citado el: 21 de marzo de 2014.] http://luchopancho.wordpress.com/2010/05/15/java-persistence-api/.

31. González, Maidelyn Piñero. Serie Cientifica UCI. [En línea] [Citado el: 14 de febrero de 2014.] http://publicaciones.uci.cu/index.php/SC/article/viewFile/1344/706.

32. NeatBeans. [En línea] [Citado el: 18 de febrero de 2014.] netbeans.org.

33. Ka lok Tong, Kent. *Begining JSF 2 API and Jboss Seam.* 2009.

34. PostgreSQL Global Development Group. [En línea] [Citado el: 20 de febrero de 2014.] http://www.postgresql.org.

35. [En línea] [Citado el: 20 de febrero de 2014.] http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

36. ISW. [En línea] [Citado el: 21 de marzo de 2014.] http://issuu.com/wdalguerreo/docs/isw-s21-analisis_orientado_a_objetos.

37. [En línea] [Citado el: 21 de febrero de 2014.] revistas.ucr.ac.cr/index.php/intersedes/article/download/790/851.

38. Fernández, Maria de los Ángeles. Universidad Nacional de la Plata. [En línea] [Citado el: 21 de febrero de 2014.] http://sedici.unlp.edu.ar/bitstream/handle/10915/21299/Documento_completo.pdf?sequence=1.

39. *Introducción a la Disciplina de Requisitos de RUP.* 2010.

40. L. Bass, P. Clements, R. Kazman. *Software Architecture in Practice.* s.l. : Addison Wesley, 2003.

41. Asociación de Usuarios de GNU/Linux de Castilla y León. AUGCyL. [En línea] [Citado el: 26 de febrero de 2014.] http://augcyl.org/?page_id=231.

42. Gómez, Carlos. *Diseño de sistemas software en UML. .* s.l. : Ediciones UPC, 2003. ISBN 8483017245..

43. Grupo soluciones Innova. [En línea] [Citado el: 12 de febrero de 2014.] http://www.rational.com.ar/Herramientas/rup.html..

44. Ajay Vohra, Deepak Vohra. *Pro XML Development with Java TM Technology .* 2006.

45. UCA. [En línea] [Citado el: 21 de marzo de 2014.] http://www.uca.edu.sv/investigacion/bdweb/tecnolog.html#Interfaz%20de%20Programaci%C3%B3n%20de%20Aplicaciones%20%28API%29.

46. Universidad Nacional de Lujan. [En línea] [Citado el: 28 de abril de 2014.] http://www.seguridadinformatica.unlu.edu.ar/?q=glossary/term/15.

47. Universidad Nacional de Lujan. [En línea] [Citado el: 28 de abril de 2014.] http://www.seguridadinformatica.unlu.edu.ar/?q=glossary/term/14.

48. Universidad Nacional de Lujan. Universidad Nacional de Lujan. [En línea] [Citado el: 28 de abril de 2014.] http://www.seguridadinformatica.unlu.edu.ar/?q=glossary/term/21.

49. Universidad nacional de Lujan. [En línea] [Citado el: 28 de abril de 2104.] http://www.seguridadinformatica.unlu.edu.ar/?q=glossary/term/4.

Bibliografía

Universidad Nacional de Lujan. [En línea] [Citado el: 28 de abril de 2014.] http://www.seguridadinformatica.unlu.edu.ar/?q=glossary/term/14.

Universidad de Navarra. [En línea] [Citado el: 28 de abril de 2014.] https://www.unav.es/SI/servicios/seguridad/faq.html#13.

Danie P.F. [En línea] [Citado el: 1 de abril de 2014.] http://fortinux.com/wp-content/uploads/2010/12/Analisis-y-Modelado-de-Amenazas.pdf.

Microsoft DeveloperNetwork. [En línea] [Citado el: 1 de abril de 2014.] http://msdn.microsoft.com/es-es/library/ms172104%28v=vs.80%29.aspx.

Microsoft. [En línea] [Citado el: 20 de marzo de 2014.] http://msdn.microsoft.com/en-us/library/ff648644.aspx#c03618429_009.

Coras. [En línea] 1 de abril de 2014. http://coras.sourceforge.net.

Marin, Diego Fernandez. slideshar7. [En línea] [Citado el: 12 de febrero de 2014.] www.slideshare.net/dfmarin/patentes-copyright-y-licencias-en-el-software.

Mazzini, Juan José Blossiers. *DESCUBRIENDO LOS DELITOS INFORMÁTICOS – LA NUEVA DIMENSIÓN DE LOS ILÍCITOS PENALES.* Peru : Normas Legales S. A, 1999.

Photoshop, Adobe. *Manual de ayuda.* Estados Unidos : Adobe Systems Incorporated, 2008.

***Sistema para la administración de licencias del Sistema producto DalasQ (DalasLicenseManager).* Alfonso, Delmis Velázquez. La Habana : s.n., 2013.**

Microsoft. Microsoft Developer Network. [En línea] [Citado el: 26 de febrero de 2014.] http://msdn.microsoft.com/es-es/library/bb972214.aspx.

Ajay Vohra, Deepak Vohra. *Pro XML Development with Java TM Technology.*

Cali. [En línea] [Citado el: 26 de febrero de 2014.] www.cali.gov.co.

Allen, Dan. *Seam in Action.* s.l. : Manning Publications, 2008.

Roman, Ed, Amble, r Scott y ., Tyler Jewell. *Mastering enterprise JavaBeans.* 2002.

Red Hat. *RichFaces developer Guide.* 2007.

***Ajax4jsf Developer Guide.* 2007.**

NeatBeans. [En línea] [Citado el: 18 de febrero de 2014.] netbeans.org.

Ka lok Tong, Kent. ***Begining JSF 2 API and Jboss Seam.*** **2009.**

PostgreSQL Global Development Group. [En línea] [Citado el: 20 de febrero de 2014.] http://www.postgresql.org.

[En línea] [Citado el: 20 de febrero de 2014.] http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

ISW. [En línea] [Citado el: 21 de marzo de 2014.] http://issuu.com/wdalguerreo/docs/isw-s21-analisis_orientado_a_objetos.

[En línea] [Citado el: 21 de febrero de 2014.] revistas.ucr.ac.cr/index.php/intersedes/article/download/790/851.

Introducción a la Disciplina de Requisitos de RUP. **2010.**

L. Bass, P. Clements, R. Kazman. ***Software Architecture in Practice.*** **s.l. : Addison Wesley, 2003.**

Asociación de Usuarios de GNU/Linux de Castilla y León. AUGCyL. [En línea] [Citado el: 26 de febrero de 2014.] http://augcyl.org/?page_id=231.

Ajay Vohra, Deepak Vohra. ***Pro XML Development with Java TM Technology .*** **2006.**

UCA. [En línea] [Citado el: 21 de marzo de 2014.] http://www.uca.edu.sv/investigacion/bdweb/tecnolog.html#Interfaz%20de%20Programaci%C3%B3n%20de%20Aplicaciones%20%28API%29.