Universidad de las Ciencias Informáticas

Facultad 7

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

Título: Implementación de una herramienta para la gestión del otorgamiento de licencias del sistema alas PACS

Autoras: Yarisley Gamboa Cobas
Caridad Naranjo Morales

Tutores: Ing. David Barreto Medina
Ing. Jorge Carlos Yero García

La Habana, junio de 2011

"Año 53 de la Revolución"

"(…) en el orden de las ideas, en el orden de las posiciones políticas, en el orden de las actitudes, los estudiantes de nuestras universidades ocupan, sin duda, una posición de vanguardia en el seno del proceso revolucionario".

Fidel Castro Ruz

Datos de Contacto

TUTOR: Ing. David Barreto Medina.

Profesor graduado de Ing. Industrial en el año 2004 en la Universidad de Holguín ¨Oscar Lucero Moya¨, posee la categoría docente de Asistente. Ha impartido asignaturas como Administración de Empresa, Contabilidad, Metodología de la investigación científica y Comercio Electrónico. Se desempeñó como asesor de la asignatura administración de empresa a nivel de universidad en el curso 2008-2009, jefe de colectivo de Ciencias empresariales en la facultad 7 en el curso 2008-2009, asesor estratégico para la producción en el curso 2008-2009, Jefe del departamento de Inteligencia Empresarial en el curso 2009-2010 y asesor de planificación y control en el curso 2009-2010 en el CESIM. Actualmente se desempeña como jefe del departamento de Despliegue, Soporte, Capacitación y Atención al Cliente en el centro de informática médica (CESIM).

Correo electrónico: dbarreto@uci.cu

TUTOR: Ing. Jorge Carlos Yero García.

Instructor recién graduado en el año 2007 de Ingeniero Industrial en la Universidad de Holguín “Oscar Lucero Moya”. Profesor de la Facultad # 7. Ha impartido las asignaturas de Comercio Electrónico, Matemática I y II. Se ha desempeñado como Planificador General del área temática de GPI, jefe de proyecto de Despliegue, Soporte y Capacitación y Comercial en el Centro de Informática Médica (CESIM).

Correo electrónico: jyero@uci.cu

RESUMEN

En la actualidad el licenciamiento ha constituido un eslabón fundamental en la protección del software que se realiza a raíz del creciente auge de la informatización en todas las esferas de la sociedad. Aunque los sistemas para la gestión del otorgamiento de licencias de software son comunes en muchos países, Cuba no tiene implantado ninguno de estos sistemas ya que la mayoría son propietarios y sería muy costosa su utilización.

Debido a la situación existente en el país, la Universidad de Ciencias Informáticas desarrolló un sistema para la gestión del otorgamiento de licencias del sistema alas PACSViewer por parte del Departamento de Software Médico Imagenológico. Durante el uso del mismo se ha comprobado que el proceso de gestión del otorgamiento de licencias no se lleva a cabo de la forma más eficiente.

El objetivo general de este trabajo es la implementación de dos módulos encargados del mejoramiento de la gestión del otorgamiento de licencias para el sistema alas PACS. Para lograr este objetivo con óptimos resultados se decide hacer uso del lenguaje C# y el marco de trabajo Framework .Net de Microsoft.

Como resultado de la investigación se obtuvo una aplicación de escritorio constituida por dos módulos para lograr que la asignación de licencias del sistema alas PACS se realice de forma segura y se pueda llevar el control sobre las mismas.

Palabras clave:
Licenciamiento, Licencia, Sistema, Otorgamiento.

CONTENIDO

INTRODUCCIÓN

En los últimos treinta años se han generado vertiginosos cambios en el desarrollo de la ciencia a nivel mundial. Como característica principal de esta etapa, se destaca la rapidez con que los nuevos conocimientos son llevados a la práctica, es decir, que el conocimiento científico modifica la sociedad al convertirse en un producto concreto. En la actualidad, la humanidad depende cada vez más de los progresos de la ciencia.

Es a partir del último decenio del siglo XX cuando comienza a marcarse una convergencia entre algunas ciencias como son la computación, las telecomunicaciones, la electrónica y los medios de comunicación masiva. Esto propició que desaparecieran poco a poco las fronteras entre dichas ciencias, debido a esto surgieron nuevos conceptos tales como: Tecnologías de Información, Sociedad del conocimiento, Telemática, Informática, etc.

Dichas ciencias en su conjunto son conocidas como Tecnologías de la Información y las Comunicaciones (TIC), las cuales no son más que el conjunto de recursos necesarios para manipular la información y entre los que se pueden encontrar por ejemplo, los ordenadores, los programas informáticos y las redes necesarias para convertir, almacenar, administrar, transmitir y encontrar dicha información. A medida que se desarrollan las TIC, la sociedad se ha podido beneficia de muchas maneras y en diferentes ramas, como la educación, el deporte, la salud, etc., facilitando así la vida de los habitantes del planeta.

Paralelo al empleo de las TIC en todos los procesos de desarrollo de la humanidad, han surgido peligros a la hora de proteger los productos que se obtienen de las mismas. Principalmente en la industria de software las últimas tendencias indican que la protección y el licenciamiento de las soluciones que se desarrollan tienen gran importancia, ya que el uso no autorizado de las mismas puede provocar grandes pérdidas.

El licenciamiento de un software es conocido como una especie de mecanismo que otorga el derecho legal de ejecutarlo y utilizarlo, para garantizar el cumplimiento de lo contratado. Usualmente, los contratos de licenciamiento admiten el derecho de realizar copias solo con propósitos de respaldo y que el software sea ejecutado en un número limitado de ordenadores.

El uso ilegal de los mismos puede propiciar una mayor exposición a virus de software, discos dañados o software defectuoso. Puede además generar la falta de documentación o documentación inadecuada, la ausencia de garantías, la falta de apoyo técnico del producto disponible para los

usuarios que cuentan con las licencias adecuadas, además de la imposibilidad de acceder a las actualizaciones de software que se ofrecen a los usuarios que cuentan con las mismas.

Se conoce que las licencias están basadas en los derechos del autor y son la base jurídica del software que se desarrolla y se utiliza. Mediante ellas se protegen los productos de los usos abusivos u oportunistas y son además una forma eficaz de proteger las inversiones que se realizan.

En Cuba, una de las instituciones encargadas del desarrollo de software es la Universidad de la Ciencias Informáticas (UCI) que fue fundada por el Comandante en Jefe Fidel Castro en el año 2002. Esta universidad tiene como misión formar profesionales, comprometidos con su Patria, calificados en la rama de la Informática, a partir de un modelo pedagógico flexible, que vincula dinámica y coherentemente el estudio con la producción y la investigación, acorde con las necesidades sociales del país y de otros pueblos hermanos.

En dicha universidad se encuentra ubicado el Centro de Informática Médica (CESIM) y dentro de éste el Departamento de Software Médico Imagenológico en el cual se han desarrollado productos como alas PACS y alas RIS que actualmente son utilizados en países como Cuba y Venezuela.

En estos momentos el CESIM, posee una herramienta que implementa un procedimiento basado en llaves de software que utiliza datos del hardware para la protección y licenciamiento del sistema alas PACS, para protegerse contra el uso no autorizado del mismo. Cada instalación de este producto debe ser activada usando un mecanismo que emplea datos del hardware. Este consiste en extraer un grupo de datos que son únicos para la PC[1] en la que se va instalar el sistema, debido a que están asociados a los componentes de hardware de dicha PC. Toda esta información es valiosa y confidencial. A partir de estos datos se aplica un procedimiento para obtener una clave única que permite usar el sistema en una PC exclusiva.

Esta herramienta es llevada por cada desarrollador encargado del despliegue en las instituciones hospitalarias, utilizando dispositivos de almacenamiento externo. Estos deben ir por cada PC para licenciar y al mismo tiempo activar cada una de las instalaciones del sistema.

Esto hace que haya una gran probabilidad de que ocurra alguna filtración o robo de información de los medios utilizados durante el despliegue del sistema para su licenciamiento o el software en general, lo que derivaría en pérdidas para la Universidad y al país.

[1] Computadora localizada en un área de la Institución Hospitalaria donde se va a instalar el sistema alas PACS.

Lo anterior unido a que dicha herramienta no ofrece la posibilidad de gestionar la información que en ella se maneja, así como obtener reportes de las licencias por institución, por PC, etc., y posee un diseño poco amigable para el usuario, hace que el proceso sea ineficiente y poco fiable.

La situación existente lleva a plantearse el siguiente **Problema a resolver:** ¿Cómo viabilizar el proceso de gestión de la información relacionada con el otorgamiento de licencias para el sistema alas PACS?, para resolver este problema se toma como **Objeto de Estudio:** el proceso de gestión de la información relacionada con el otorgamiento de licencias de software, delimitando como **Campo de Acción:** el proceso de gestión del otorgamiento de licencias de software para el sistema alas PACS.

Para dar solución al problema planteado se propone como **Objetivo General:** implementar un sistema para la gestión del otorgamiento de licencias de software del sistema alas PACS partiendo de la valoración crítica del diseño propuesto por el analista.

Para dar respuesta al objetivo propuesto se plantean las siguientes tareas de la investigación:

1. Valorar los sistemas informáticos de licenciamiento de software existentes a nivel nacional e internacional, estableciendo similitudes con la investigación en curso.
2. Valorar las herramientas y técnicas de programación definidas en el diseño dando cumplimiento al objetivo propuesto.
3. Realizar una valoración del diseño propuesto por el analista.
4. Implementar las funcionalidades establecidas para el módulo Cliente.
5. Implementar las funcionalidades establecidas para el módulo Gestor.
6. Describir las principales clases y algoritmos del sistema.
7. Realizar pruebas de caja blanca y caja negra a las funcionalidades del sistema.

Este trabajo consta de 4 capítulos estructurados de la siguiente forma:

Capítulo 1. Fundamentación Teórica: Contiene un estado del arte que brinda una panorámica general sobre el tema a nivel internacional, nacional y de la Universidad. Se exponen las características fundamentales de la metodología de desarrollo, tecnologías y herramientas, definidas por la arquitectura, que son utilizadas en la implementación de la herramienta.

Capítulo 2. Descripción de la Arquitectura: Describe la arquitectura del sistema que se propone, así como una breve descripción de las funcionalidades y una valoración crítica del diseño realizado por el analista.

Capítulo 3. Implementación del Sistema: Se describe la implementación de los módulos a partir de los requerimientos identificados. Se presenta el diagrama de componentes

correspondiente a cada uno de los módulos del sistema, representando sus elementos más significativos. Se definen los estándares de codificación a seguir durante la etapa de desarrollo.

Capítulo 4. Pruebas del Sistema: Este capítulo incluye una breve descripción de las pruebas de software, así como la validación de la solución a partir de pruebas de caja negra y caja blanca.

CAPÍTULO 1: Fundamentación Teórica

En este capítulo de forma general se mencionan los aspectos teóricos utilizados para el desarrollo de la investigación, que sirven de apoyo para la mejor comprensión del presente trabajo. Se realiza un estudio sobre el licenciamiento del software, las tendencias actuales del mismo y los diferentes sistemas automatizados que existen. 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 programación del sistema.

1.1 Situación actual del licenciamiento del software.

1.1.1 Licenciamiento del software.

El licenciamiento de software es el proceso de compra estatal de software legal, en tanto, dicha autorización o licencia se requiera. Es la autorización o permiso concedido por el titular del derecho de autor, en cualquier forma contractual, al usuario de un programa informático, para utilizar el mismo en una forma determinada y de conformidad con unas condiciones convenidas. Es un contrato que regula el seguimiento de las normas exactas que son descritas en la documentación que acompaña al producto, es decir, la licencia.

La licencia, que puede ser gratuita u onerosa, precisa los derechos (de uso, modificación y/o redistribución) concedidos a la persona autorizada y sus límites. Además, puede señalar el plazo de duración, el territorio de aplicación y todas las demás cláusulas que el titular del derecho de autor establezca.

En la actualidad, existen un gran número de Licencias de Software de diferentes tipos, sin embargo se pueden agrupar en 2 grandes grupos:

1.1.1.1 Licencias de Software Propietario.

Son similares a un acuerdo de renta en el que el usuario acepta pagar por el privilegio de usar el software, y se compromete con el autor del mismo a cumplir con todas las restricciones establecidas en dicho contrato.

La mayor parte de las aplicaciones de software empleadas en la actualidad se licencian bajo esta modalidad. No observar las condiciones del autor supone infringir la ley y puede exponerse a multas o sanciones, de ahí que sea muy importante que este tipo de productos sean adquiridos legalmente, a distribuidores reconocidos y que se sigan con detenimiento las reglas impuestas por los autores. (1)

1.1.1.2 Licencias de Software Libre o Abierto.

Son similares a un acuerdo de cesión de uso en el que el autor otorga al usuario una serie de privilegios o libertades, generalmente sin obligación económica. Se caracteriza además por garantizar a los usuarios las siguientes libertades:

- La libertad de utilizar el programa.
- La libertad de distribuir el programa.
- La libertad de estudiar el programa (El acceso al código fuente es un requisito).
- La libertad de modificar el programa y de distribuir las nuevas versiones del mismo.

Sin embargo en este tipo de licencia, existe la posibilidad de una generación más fácil de virus de software, dado que el código fuente también puede ser modificado con intenciones maliciosas. Este acceso libre al código permite que cualquier usuario pueda realizar con el software lo que estime conveniente ya que el distribuidor del mismo le otorga esa libertad.

A diferencia, el software propietario, debe solicitar el permiso al dueño o pagar para poder realizar algunas acciones como la copia, redistribución o modificación del software ya que la propiedad y decisión de uso del software es otorgado a la empresa desarrolladora con el objetivo de evitar que la competencia se pueda apropiar para sus propios fines, del producto una vez finalizado.

1.1.2 Tendencias actuales en el licenciamiento del software.

Además de proteger el software y la propiedad intelectual, se necesita proteger los ingresos de las ventas de los productos. Para el aseguramiento de que el software solo está disponible para los usuarios adecuados, de acuerdo con los términos que se definan, este proceso se controla a través de las licencias. Las licencias proporcionan la flexibilidad necesaria para implementar las estrategias del negocio para la distribución del software. (2)

Para obtener el máximo beneficio de la estrategia de licencias en una empresa, es necesario un sistema de licencias de software que le proporcione la flexibilidad para adaptar los términos de licencias, de modo que coincidan con la estrategia de negocio y para adaptarse rápidamente a los cambios del mercado y a las necesidades de la empresa. El sistema de licencias también debe ser capaz de imponer los términos de uso que definió mediante métodos de licencias seguros. (3)

1.2 Sistemas automatizados existentes.

Mediante la investigación realizada se ha podido observar que en los últimos años la utilización de diversos sistemas para la gestión del otorgamiento de licencias en el ámbito internacional, ha tenido un notable auge. Debido a que las empresas se han visto en la necesidad de proteger los programas que desarrollan y los datos con los que trabajaban, han creado herramientas que

contribuyen a combatir la piratería, además de permitir activar los software de forma segura; para evitar la copia de este por vía ilegal.

En el mundo existen gran cantidad de herramientas encargadas de la gestión del otorgamiento de licencias de software, entre las que se encuentran:

NetSupport DNA

NetSupport DNA es una completa solución que permite realizar inventarios de hardware y de software así como la gestión de licencias. Presenta avisos detallados y plenamente personalizables, medición y control del uso de aplicaciones y de Internet. Permite asignar información de licencias a los datos de activos de una PC. Tiene una interfaz muy amigable que permite que los operadores puedan asignar rápidamente licencias a cuantas PCs deseen.

Permite monitorear el uso de licencias dentro de una organización. Al introducir el número de licencias para cada aplicación es posible identificar el uso de ellas cada vez que se ejecute el inventario de software. Además de mostrarse en las listas de Inventario de software, la información de licencia también se puede mostrar en la sección de Medición de aplicaciones de DNA. NetSupport DNA soporta Windows 98 y posteriores, junto con las siguientes distribuciones Linux Red Hat 9, Red Hat Enterprise, SuSE 9 y FedoraCore 6. (4)

MSIA (Microsoft Software Inventory Analyzer)

Microsoft Software Inventory Analyzer (MSIA) explora las estaciones de trabajo individuales o redes completas de PC para los productos de software de Microsoft. MSIA genera los informes que indican los títulos de software, tipo y número de licencias para cada título, el número de instalaciones para ese título y los vínculos a una lista de PC exploradas. (5)

El MSIA podrá funcionar con redes que cuentan con hasta 250 o menos ordenadores. La última versión de MSIA 5.1 tiene características que le permite:

- Actualización de la información de licencias en formato HTML cuando Internet Explorer 8 está instalado.
- Soporte para Windows 7 y Office 2010.

ElecKey (Scientsoft Research) Versión 2.0

ElecKey es la solución completa para la protección de copia de software, licencias de software, distribución de software de forma segura. ElecKey permite el uso de software de seguridad que ayuda a proteger las aplicaciones de software contra la piratería. ElecKey proporciona la capacidad de concesión de licencias de software, permite crear una gran variedad de modelos de licencia, por

ejemplo, por un plazo limitado, versión de evaluación, el equipo específico de licencia, licencia flotante, la activación del software, etc. (6)

Una vez que se distribuye el software, ElecKey ofrece una amplia variedad de opciones que permiten gestionar la licencia y la concesión de licencias desde un servidor de activación completamente automatizado.

Por otra parte, ElecKey 2.0 también proporciona la solución para hacer frente a las difíciles cuestiones de soporte al usuario final de concesión de licencias. Puede crear las utilidades de usuario final, junto con la aplicación protegida que permitirá al usuario activar, desactivar, transferir y retirar la licencia de software fácil y cómoda. Además, estos servicios pueden ayudar a gestionar los certificados expedidos. Mediante ellos se puede verificar si el usuario ha destruido la licencia antes de la emisión de una restitución. (7)

Protector de Licencias (License Protector) 2.5

Protector de Licencias, administra licencias y módulos (administración y control de licencias), genera versiones de demostración y de tiempo limitado (versiones de prueba), proporciona un programa de protección contra copia y soporta pruebas de usuario concurrentes (licencia flotante). Posibilita generar licencias automáticamente para tiendas en línea (opción para comercio electrónico). El servidor de activación a través de Internet permite activar una licencia en línea. (8)

Ofrece archivos de licencia cifrados con claves hechas por el cliente para cada proyecto de informática y Claves de Activación Segura que pueden ser usadas solo una vez. Sin pagos por derechos de uso. Pague una vez, distribúyalo tanto como guste. Protector de Licencias está disponible en tres ediciones para satisfacer diferentes requerimientos: Principiante, Básico y Edición Profesional. (9)

HASP SRM (Software Rights Management)

Esta es una solución para la administración de derechos del mercado, que permite usar o bien llaves de protección por software o por hardware para proteger y expedir las licencias del software.

Con HASP SRM, se simplifica la capacidad de protección y licencias del software mediante la utilización de protección flexible y herramientas de licencias, junto con una llave de protección HASP SRM que subsecuentemente protege su software. Estas llaves pueden estar basadas en Hardware o en Software. (10)

Sentinel RMS

Sentinel RMS es la solución de protección por software que le permite incorporar a sus aplicaciones algoritmos propietarios de alta seguridad brindándole la comodidad de distribuir sus aplicaciones por todo el mundo con un solo clic del mouse. Con el uso de un algoritmo propietario de alta seguridad, Sentinel RMS protege el software frente al uso no autorizado de forma que puede colocarse en la Web o en millones de CDs para su distribución masiva. La potente función de Sentinel RMS para crear huellas dactilares de sistema asegura que solo los usuarios registrados pueden utilizar una aplicación al anclarla a las características únicas de la computadora en la que se encuentra instalada. Además, Sentinel RMS convierte fácilmente los productos demo en aplicaciones completas que pueden alquilarse o comprarse, para ofrecer mayores oportunidades de ingresos a los desarrolladores.

Provee de un modo costo efectividad para recibir el beneficio de un sistema de administración de licencias seguras, sin la necesidad de desarrollar y mantener el sistema de licencias por usted mismo.

Contiene un modelo de licencias built-in (incorporadas) que son apropiadas para licenciar independiente, conectado a una red, y una administración de licencias basadas en Internet. Para agregar seguridad, las licencias Sentinel RMS pueden ser ajustadas a un único dispositivo de hardware.

Adicionalmente, Sentinel RMS en conjunto con Sentinel Express le da a usted la habilidad de administrar seguramente las licencias a través de Internet. Si Usted desea implementar un simple modelo de Administración de licencias o crear un modelo más complejo de licencias con el nivel de seguridad más alto, existe una solución Sentinel que se adapta a sus necesidades. (11)

Debido a que algunos de estos sistemas son propietarios y por ende sus costos por uso son muy elevados, financiar los gastos de la implantación de un sistema de los existentes en el mundo, para la gestión del otorgamiento de licencias de software sería muy difícil para países subdesarrollados como Cuba.

Todo esto ligado a que otros sistemas no cubren en su totalidad con lo que se quiere diseñar, puesto que algunas monitorean el uso de licencias pero no logran asignarlas, se limitan además a chequear la cantidad de instalaciones de cada sistema informático, con el objetivo de verificar el cumplimiento de las licencias, sin embargo no implementan mecanismos para asegurar el acatamiento de éstas. Algunos de estos sistemas funcionan solo para productos de empresas específicas, por lo que el Departamento de Software Médico Imagenológico no tendría acceso a las

mismas. Es por ello que se ha decidido desarrollar soluciones propias para facilitar el proceso de gestión del otorgamiento de licencias para el sistema alas PACS.

Actualmente en Cuba a pesar de que se han hecho numerosos esfuerzos en el campo de la Industria de Software, esta es relativamente nueva y su desarrollo se ha visto frenado por las consecuencias de un bloqueo económico, cada vez más recrudecido. Pero a pesar de los contratiempos externos, muchas cosas faltan por hacer aún y una de ellas es llegar a alcanzar experiencia en la protección de software mediante licenciamiento. En la actualidad este proceso se ha visto retrasado, tanto así que no existe un sistema nacional mediante el cual se le otorguen las licencias al software a través de su activación y que lleve el control sobre las mismas, para de esta forma tener una mayor seguridad del uso del software que se distribuye.

1.3 Técnicas de Programación.

Una técnica de programación es una metodología que debe de seguirse y tomarse en cuenta al momento de programar. Deberá entenderse que para la programación deberán asumirse ciertas normas que permitan la estandarización de la programación, lo que implica una disminución de costos, independencia del programador y seguridad.

Hoy en día los tipos o técnicas de programación son bastante variados, aunque a veces muchas personas solo conocen una metodología para realizar programas. En la mayoría de los casos, las técnicas se centran en programación modular y programación estructurada, pero existen otros tipos de programación.

Programación modular

La programación modular consta de varias secciones dividas de forma que interactúan a través de llamadas a procedimientos que integran el programa en su totalidad. El programa principal coordina las llamadas a los módulos secundarios y pasa los datos necesarios en forma de parámetros. A su vez cada módulo puede contener sus propios datos y llamar a otros módulos o funciones.

Programación estructurada (PE)

La programación estructurada está compuesta por un conjunto de técnicas que han logrado evolucionar y aumentar considerablemente la productividad del programa y reducir el tiempo de depuración y mantenimiento del mismo. Esta programación utiliza un número limitado de estructuras de control, para reducir considerablemente los errores.

Las principales ventajas de la programación estructurada son:

- Los programas son más fáciles de entender.
- Se reduce la complejidad de las pruebas.

- Aumenta la productividad del programador.
- Los programas quedan mejor documentados internamente.

Un programa está estructurado si posee un único punto de entrada y solo uno de salida, existen de "1 a n" caminos desde el principio hasta el fin del programa y por último, que todas las instrucciones son ejecutables sin que aparezcan bucles infinitos. (12)

Programación orientada a objetos (POO)

Esta técnica, gracias a la reutilización de los objetos, aumenta considerablemente la velocidad de desarrollo de los programas. Su elemento principal es el objeto, el cual es un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización. Los datos contenidos en estos están bien estructurados y pueden ser visibles o no en dependencia del programador y las acciones del programa en ese momento. Entre sus principales características se encuentran la herencia, el polimorfismo y el encapsulamiento.

La programación orientada a objetos constituye una de las formas más populares de programar y ha tenido gran acogida en el desarrollo de software desde los últimos años. Esto se debe a sus grandes capacidades y ventajas frente a las antiguas formas de programar.

Entre las ventajas más importantes se pueden destacar:

- Favorece la comunicación entre analistas, diseñadores, programadores y usuarios finales al utilizar todos los mismos modelos conceptuales.
- Esto se traduce en un aumento de la productividad, ya que la comunicación es uno de los puntos críticos en las primeras fases del proyecto.
- Se facilita la representación de estructuras complejas sin necesidad de adaptarse a normas y modelos, ya que lo que se maneja son objetos del mundo real, lo que facilita la tarea del analista.
- La semántica de estas técnicas es más rica (al ser más natural); al usuario final le es más fácil comprender lo que el analista representa en sus modelos (ya que representa los objetos que lo rodean habitualmente).
- Favorece la modularidad, la reusabilidad y el mantenimiento del software.
- Estas técnicas son más resistentes al cambio que las tradicionales técnicas de análisis orientadas a flujos de datos. (13)

Lo interesante de la POO es que proporciona conceptos y herramientas con las cuales se modela y representa el mundo real tan fielmente como sea posible.

Programación concurrente

Este tipo de programación se utiliza para realizar varias acciones a la vez y para controlar los accesos de usuarios y programas a un recurso de forma simultánea. Se trata de una programación más lenta y laboriosa, por lo que se obtienen resultados lentos en las acciones.

Programación funcional

Esta trata a la computación como una evaluación de funciones matemáticas. Aquí se enfatiza en la definición de funciones que implementen estados de la máquina, en contraparte con la programación basada en procedimientos, la cual está basada en la ejecución de comandos de forma secuencial. Un programa puramente funcional no usa la mutación, cuando modifica el valor de un estado para producir valores, construye nuevos valores, sin sobre escribirlos, de los existentes.

Programación lógica

Se suele utilizar en la inteligencia artificial y pequeños programas infantiles. Se trata de una programación basada en el cálculo de predicados (una teoría matemática que permite lograr que un ordenador basándose en hecho y reglas lógicas, pueda dar soluciones inteligentes).

Programación orientada a eventos

No se programa de forma tradicional, donde se sigue un patrón que controla el propio flujo del programa, solamente cambia algunas veces cuando se llega a una posición donde existen dos caminos, el control del flujo de un programa orientado a eventos es extremadamente conducido por eventos externos. En vez de esperar por un comando el cual ordene que se proceda determinada información, el sistema está pre-programado para realizar un ciclo de forma repetida, para verla y realizar una función que desencadena en su procesamiento. Por lo tanto, cuando se desencadena la función previamente definida, corresponderá al comportamiento requerido.

Cada una de las diferentes técnicas de programación pueden utilizarse y obtener lo mejor de cada una de ellas en dependencia del lenguaje de programación, por lo que debe de aprovecharse cada uno de los aspectos que más convenga. Cada una de las técnicas ha sido eficientada y creada para resolver determinadas situaciones, se deben conocer, aprovechar y utilizar en el lenguaje de programación de preferido.

1.4 Herramienta de Programación.

Una herramienta de programación o herramienta del software, es aquella que permite realizar aplicativos, programas, rutinas, utilitarios y sistemas para que la parte física del computador u ordenador, funcione y pueda producir resultados. Estas empezaron a aparecer con las primeras

computadoras en los comienzos de los años 50, las cuales eran originalmente simples y ligeras. Las herramientas de programación más comunes del mercado, cuentan hoy día con programas de depuración o debugger, que son utilitarios que permiten detectar los posibles errores en tiempo de ejecución o corrida de rutinas y programas. (14)

1.4.1 Lenguaje de Programación.

Un lenguaje de programación es una construcción mental del ser humano en un idioma artificial para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras, en las que para que ésta sea operable, debe existir otro programa que controle la validez o no de lo escrito. Está constituido por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos con sentido único y una regla principal que resume las demás. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. (15)

En otras palabras se puede decir que es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo.

Los lenguajes pueden ser de alto o bajo nivel. En los de bajo nivel las instrucciones son simples y cercanas al funcionamiento de la máquina, como por ejemplo el código máquina y el ensamblador.

C#

Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. La creación del nombre del lenguaje, C, proviene de dibujar dos signos positivos encima de los dos signos positivos de “C++”, para dar una imagen de salto evolutivo.

Este lenguaje estrella de la plataforma de desarrollo.NET cuenta con una extensa librería base para el desarrollo de todo tipo de aplicaciones. C# es un lenguaje de programación simple pero eficaz, este toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en un solo lenguaje. (19)

Presenta entre otras características:

- Sencillez.
- Modernidad.
- Orientación a Objetos.
- Orientación a componentes.
- Gestión automática de memoria.

- Seguridad de tipos.
- Instrucciones seguras.
- Sistema de tipos unificado.
- Extensibilidad de tipos básicos.
- Extensibilidad de operadores.
- Extensibilidad de modificadores.
- Posibilidad de crear versiones.
- Eficiencia.
- Compatibilidad.

Este está tomando gran importancia en el desarrollo de aplicaciones debido a que ha superado la potencialidad de Java o el mismo C/C++, esto es debido a que este lenguaje está realmente orientado a objetos y fue avizorado además para soportar la programación orientada a componentes. (20)

1.4.2 IDE de Desarrollo.

Los ambientes integrados de desarrollo (IDEs) proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación ya que son un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto. Son generalmente más simples y hacen más fácil hacer tareas simples, tales como buscar para contenido solamente en archivos en un proyecto en particular.

Microsoft Visual Studio 2008

Es el nuevo Entorno Integrado de Desarrollo que Microsoft ha desarrollado para la creación de Software enfocado a su Sistema Operativo Microsoft Windows Vista y la realización de aplicaciones para trabajar con su paquetería Microsoft Office 2007. Ha sido desarrollado específicamente para la plataforma.NET y en especial para el lenguaje C#. (21)

Microsoft Visual Studio 2008 viene con muchas mejoras y funcionalidades, a continuación se hace referencia a las más importantes.

- Utilizar el Framework .NET 3.5 y poder programar para las versiones anteriores (2.0, 3.0).
- Conjunción con XAML (se pronuncia “zammel” y significa “eXtensible Application Markup Language” por sus siglas en inglés).

- Un diseñador para Windows Presentation Foundation y Workflow Foundation que son parte del Framework .NET 3.0.
- IntelliSense para JavaScript.

El nuevo Lenguaje LINQ (significa “Language Integrated Query”) es un agregado a los lenguajes Visual Basic y Visual C# para la realización de consultas SQL.

Microsoft Visual Studio 2008 es una nueva herramienta para los desarrolladores, ya que presenta nuevas funcionalidades como el IntelliSense para JavaScript, que ayudará a los programadores de Aplicaciones Web a realizar sus proyectos mucho más rápido; también el hecho de tener la posibilidad de cambiar entre versiones del Framework .NET ayuda a las empresas a elegir el tipo de estructura que ocuparán sus nuevos Sistemas. (22)

1.4.3 Metodología de desarrollo de software.

Un modelo de ciclo de desarrollo se define como un esquema que contiene procesos, actividades y tareas involucradas en el desarrollo, operación y mantenimiento de un producto de software, que abarcavdesde la definición de requerimientos de un sistema hasta la terminación de su uso. Actualmente se catalogan los procesos de desarrollo en: pesados, ligeros, ágiles y prescriptivos.

Rational Unified Proccess (RUP)

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

Las mejores prácticas del RUP, son un conjunto de procesos web-enabled 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 lo 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 iterativamente. Rational Unified Process organiza los proyectos en términos de disciplinas y fases, que consisten cada una en una o más iteraciones. Con esta aproximación iterativa, el énfasis de cada flujo de trabajo variará a través del ciclo de vida. La aproximación iterativa ayuda a mitigar los riesgos en forma temprana y continua, con un progreso demostrable y frecuentes releases ejecutables. (24)

Utiliza el Leguaje Unificado de Modelado (Unified Modeling Lenguaje, UML), el cual es utilizado para visualizar, especificar y construir los artefactos del sistema automatizado a definir.

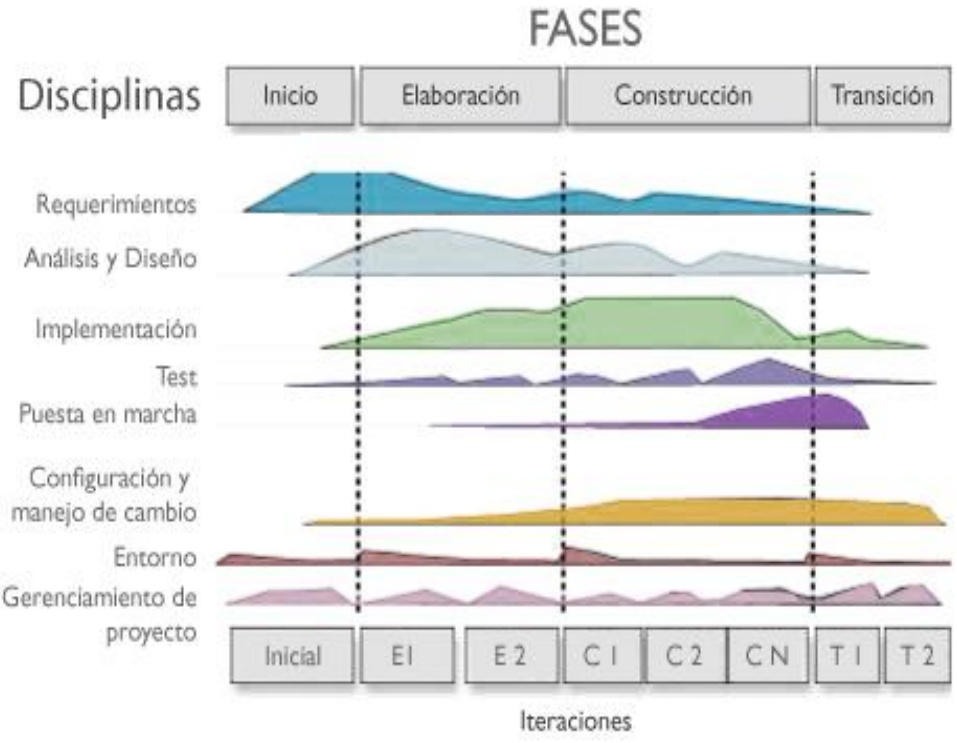


Fig. 1 Fases e Iteraciones de la Metodología RUP.

1.4.4 Sistemas de Gestión de Base de Datos (SGBD).

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos, para asegurar su integridad, confidencialidad y seguridad. Por tanto debe permitir:

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

Algunas de las características deseables en un Sistema Gestor de Base de Datos son:

- Control de la redundancia: La redundancia de datos tiene varios efectos negativos (duplicar el trabajo al actualizar, desperdiciar espacio en disco, puede provocar inconsistencia de datos) aunque a veces es deseable por cuestiones de rendimiento.
- Restricción de los accesos no autorizados: Cada usuario ha de tener permisos de acceso y autorización.
- Cumplimiento de las restricciones de integridad: El SGBD ha de ofrecer recursos para definir y garantizar el cumplimiento de las restricciones de integridad. (27)

PostgreSQL8.3

Es la primera base de datos de código abierto en implementar Recorrido Sincronizado, que reduce el uso de Entrada/Salida en aplicaciones de minería de datos. El grupo de Windows ha implementado un sistema de compilación con Visual C++, lo cual mejora la estabilidad y rendimiento en Windows, así como la accesibilidad para otros contribuyentes. Nuevas opciones de registro (logging) han sido agregadas y el sobrecosto del recolector de estadísticas ha sido disminuido para hacer más fácil el monitoreo de sus servidores. Entrega mayor consistencia en el rendimiento que versiones anteriores. (32)

Las mejoras de rendimiento más importantes incluyen:

- Heap Organized Tuples (HOT), que eliminan hasta un 75% de la sobrecarga de mantenimiento en tablas frecuentemente actualizadas.
- Checkpoints extendidos y autoafinamiento del escritor en segundo plano, que reducen el impacto de los checkpoints en los tiempos de respuesta.
- Opción de confirmación (commit) asincrónico de transacciones, que permite tiempos de respuesta más breves para algunas transacciones.

Estos cambios, aceleran significativamente la tasa de procesamiento de transacciones.

1.4.5 Plataforma de Programación.

Una plataforma refiere a la tecnología básica del hardware y software de un ordenador que define como funciona y que otro tipo de software se puede emplear con él. En casi todos los casos las plataformas son descritas como la combinación del sistema operativo y el hardware que en ocasiones se puede considerar sinónimo de sistema operativo (Windows).

.Net

Es la base de la nueva generación de software en la cual los Servicios Web son un medio que permitirá a distintas tecnologías interoperar entre sí, así como conectar diversos sistemas operativos, dispositivos, información y usuarios, da a los desarrolladores las herramientas y tecnologías necesarias para desarrollar soluciones de negocios de manera rápida sin importar que involucren diversos medios y tecnologías.

La plataforma .NET no es un producto, sino un conjunto de ellos que de manera conjunta proporcionan una solución al problema, sus componentes principales son:

- Clientes Inteligentes: Son dispositivos muy variados, Lo que los hace inteligentes es su capacidad para hacer uso de servicios Web. Pueden ser computadoras de bolsillo, personales, portátiles, teléfonos inteligentes, handhelds e incluso consolas de juegos como XBOX.
- Servidores: Proveen de la infraestructura para implementar el modelo de computación distribuida en Internet. Son sistemas operativos y de aplicación.
- Servicios Web basados en XML: Son los bloques de construcción de la tercera generación de Internet. Permiten a las aplicaciones compartir datos y están basados en XML.
- Herramientas de desarrollo: Visual Studio .NET y el .NET Framework. Ambos permiten al desarrollador hacer servicios Web basados en XML además de otro tipo de aplicaciones. (34)

1.4.6 Librería de Programación.

Las librerías son pequeños subprogramas que contienen un conjunto de funciones, códigos, datos, así como tipos relacionados y macros que proporcionan servicios a programas independientes que son utilizados en el desarrollo de software. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular. Algunos programas ejecutables pueden ser a la vez programas independientes y bibliotecas, pero la mayoría de éstas no son ejecutables.

Las librerías de los lenguajes de programación ahorran la tarea de escribir las funciones comunes que por lo general pueden necesitar los programadores. Un lenguaje de programación bien desarrollado tendrá una buena cantidad de ellas.

Todas las herramientas y tecnologías a utilizar fueron seleccionadas por los analistas en la tesis de análisis y diseño, además por formar parte del documento de arquitectura del Departamento de Software Médico Imagenológico de la Universidad de las Ciencias Informáticas.

Durante el desarrollo de este capítulo se valoraron los sistemas informáticos de licenciamientos de software existentes a nivel internacional, se estimó que las herramientas a utilizar durante la implementación del sistema son las indicadas para dar cumplimiento al objetivo propuesto. A partir del análisis de la documentación existente se logro comprender la necesidad de realizar un sistema que sea capaz de asignar y llevar el control de las licencias generadas para el sistema alas PACS.

CAPÍTULO 2: Descripción de la Arquitectura

El presente capítulo está centrado en la descripción de la arquitectura del sistema que se propone para dar solución al problema planteado, para ello, se realizará un análisis detallado del diseño proveniente del analista del sistema para la transición del diseño a la implementación, a partir de los requerimientos de la aplicación.

2.1 Propuesta del sistema.

La solución propuesta por el analista consta de dos módulos: el Cliente, el cual es una aplicación que se aloja en un dispositivo de almacenamiento externo, éste se encarga de obtener los datos de las PC donde se va a instalar el sistema o el producto alas PACS, genera una solicitud de licencia para cada una de estas, el otro módulo es el Gestor, que es el encargado de realizar las licencias para cada solicitud de licencia creada en el Cliente a partir de los datos de cada PC.

El Cliente es portado por los Especialistas en Despliegue encargados de desplegar el sistema en las Instituciones Hospitalarias, es usado para obtener los datos únicos de hardware de las PC, almacenándolos en una representación interna que luego son exportados a un fichero de solicitudes de licencias. Además carga el paquete de licencias generadas por el Gestor y permite activar cada una de las instalaciones del sistema en las PC, cuyos datos fueron recogidos.

El Gestor se usa de forma centralizada por el Licenciador que es la persona encargada de generar las licencias. Este a partir de las solicitudes de licencias exportadas por el Cliente, genera las licencias asociadas a cada una de las solicitudes exportándolas a un paquete de licencias o individualmente como le sea más conveniente. Usa una Base de Datos en la que almacena toda la información relacionada con las solicitudes de licencias, las licencias generadas para cada solicitud, así como la renovación de las licencias. Además permite generar informes asociados a toda esta información.

2.2 Descripción de las principales clases y funcionalidades

Durante el flujo de trabajo de análisis y diseño los analistas identificaron una serie de funcionalidades, las cuales se muestran a continuación:

Tabla: 1 Requisitos funcionales del módulo Cliente.

Requerimientos Funcionales	Descripción
1. Obtener Datos de la PC.	

1.1. Obtener datos de ubicación de la PC.	Se llenan los datos asociados a las PC que se van a licenciar, como son: nombre de la Institución a la que pertenece la PC, Número de serie de la PC, y Ubicación de la PC.
1.2. Obtener datos de hardware de la PC.	Se recogen los datos de configuración de hardware de cada PC en la que se va a instalar el sistema de forma automática.
2. Almacenar Datos Obtenidos.	Los datos recogidos se mantienen en una representación interna que solo puede ser accedida con el Cliente.
3. Generar Solicitud.	A partir de los datos recogidos se crea una solicitud de licencia.
4. Gestionar Solicitud.	
4.1. Visualizar Solicitud.	Incluye la solicitud de licencia en una lista que contiene cada una de las solicitudes de las PC cuyos datos han sido recogidos y muestra dicha lista.
4.2. Buscar Solicitud.	Permite encontrar una solicitud dada usando como criterio de búsqueda: Número de serie de la PC.
4.3. Eliminar solicitud.	Permite eliminar una solicitud correspondiente a una PC que no deba estar licenciada y se le hayan recogido los datos por equivocación.
5. Exportar Fichero de Solicitud de Licencias.	Se exportan los datos almacenados en el Cliente hacia un fichero con un formato conocido tanto por el Cliente como por el Gestor.

6. Importar Licencias.	Incorpora y almacena los datos de las licencias generadas correspondientes a las capturas realizadas.
7. Identificar una licencia que corresponda a una estación dada.	Busca que haya en el paquete de licencias una correspondiente a la PC a activar.
8. Verificar que la licencia se corresponda con la estación a activar.	Verifica con los datos de hardware de la PC que la licencia identificada sea la correcta.
9. Activar Licencia.	Genera la licencia en una ubicación donde el sistema necesita para su activación, a partir de aquí permite la utilización del mismo.
10. Renovar Licencia.	En caso de que una PC se haya roto y se le haya sustituido algún componente de hardware, la licencia que disponía ya no es válida entonces es necesario renovarla.
11. Gestionar Licencias.	
11.1. Visualizar Lista de Licencia.	Incluye la licencia a una lista que contiene todas las licencias generadas correspondientes a las capturas realizadas y muestra dicha lista.
11.2. Buscar Licencia.	Permite encontrar una licencia dada, usando como criterio de búsqueda: Número de serie de la PC.
12. Vaciar almacén de solicitudes de licencias y paquete de licencias.	Se encarga de borrar todas las solicitudes de licencia y las licencias asociadas a cada una de ellas una vez que se haya terminado de licenciar la Institución Hospitalaria.

Tabla: 2 Requisitos funcionales del módulo Gestor.

Requerimientos Funcionales	Descripción
1. Importar Fichero de Solicitudes de Licencias.	Incorpora y almacena el fichero donde están contenidas las capturas de información de Hardware de cada PC.
2. Visualizar Solicitud.	Muestra una lista con cada solicitud de licencia incluida en el

	fichero importado.
3. Eliminar Solicitud.	Permite eliminar una solicitud de licencia que por alguna razón no deba ser atendida.
4. Generar Licencias.	Genera una licencia para cada una de las PC que se muestran en la lista de solicitudes de licencias.
5. Almacenar Licencias.	Guarda de manera persistente las licencias generadas en una representación interna del mismo.
6. Gestionar Paquete Licencias.	
6.1. Visualizar Licencias.	Muestra una lista con todas las licencias generadas.
6.2. Buscar Licencias.	Permite encontrar una licencia dada usando como criterio de búsqueda: Número de serie de la PC.
7. Exportar Licencias.	Exporta el paquete de licencias o el fichero de licencia según se desee.
8. Generar reporte de licencias.	Muestra un informe con todas las solicitudes de licencias de una determinada Institución Hospitalaria, las licencias generadas para cada una de ellas, así como todas aquellas que han sido renovadas.

Además, determinaron las siguientes clases para llevar a cabo la implementación del sistema:

Tabla: 3 Clases para la implementación del sistema.

Clases	Descripción
1. ClientModel	Implementa la lógica del negocio del Cliente.
2. GestorModel	Implementa la lógica del negocio del Gestor.
3. PCDataForm	Formulario para llenar algunos datos asociados a la PC y recoger de forma automática los datos de Hardware de la PC.

4. SolicitudLicencia	Contiene los datos de la solicitud y permite realizar objetos de su tipo.
5. ClientMainForm	Interfaz principal del Cliente.
6. Licencia	Contiene los datos de la licencia y permite realizar objetos de su tipo.
7. GestorMainForm	Interfaz principal del Gestor.
8. DBManager	Permite almacenar los datos asociados a las solicitudes de licencias, las licencias generadas para cada solicitud así como todas aquellas licencias que hayan sido renovadas.

Para el modelo de datos se seleccionaron 6 tablas:

Tabla: 4 Tablas para el almacenamiento de datos.

Tabla	Descripción
tbl_institución	Guarda los datos de relacionados a la institución donde se encuentra la PC a licenciar o licenciada.
tbl_ubicación	Guarda la información referente a la ubicación de la PC que solicita la licencia o que ya ha sido licenciada.
tbl_pc	Almacena los datos de las PC que van hacer licenciadas o en las que ya se encuentra activa el sistema alas PACS.
tbl_licencia	Almacena la información referente a las licencias generadas.
tbl_solicitudlicencia	Guarda la información relacionada con las solicitudes realizadas.
tbl_autenticacion	Guarda la información referente a los usuarios que van a tener acceso a la base de datos.

2.3 Valoración crítica del diseño propuesto por el analista.

Como es conocido el objetivo de la implementación es tomar el diseño e implementarlo en archivos de código fuente, librerías de clases y ejecutables. Durante la implementación se incorporan cosas que no son posibles definirlas durante el diseño o son mal definidas durante éste, debido a la complejidad o simplemente por las particularidades de cada lenguaje de programación entre otros factores.

El sistema a desarrollar no es más que una aplicación de escritorio la cual necesita para su funcionamiento algunos datos únicos del hardware de la PC en la cual se va a instalar el sistema alas PAC, dichos datos en algunos casos no son visibles o no están disponibles, como lo es por ejemplo el número de serie de la placa base, por lo cual se realizaría el proceso de generación de licencia sin esta información poniendo a cambio el valor” None”.

Durante el análisis y diseño los analistas propusieron que los tipos de datos de los números de series tanto de la PC como de la placa base y el id del microprocesador fueran de tipo de dato int, lo que se decidió por los implementadores tomar estos con el tipo de dato string como son en realidad, ya que estos además de números incluyen caracteres, por esta razón se vio la necesidad de realizar los cambios necesarios.

En algunos casos existían métodos los cuales se les introducían parámetros en el formulario y en el momento de invocar este método en la clase controladora no llevaban los parámetros necesarios por lo que no había forma de obtener los datos introducidos, debido a esto los desarrolladores realizaron los cambios necesarios.

Adicionalmente, con el objetivo de obtener los reportes en algún formato para proteger el flujo de información y guardar informes con la información obtenida se le agregó al sistema la opción de exportar los datos en formato pdf y Excel.

En el modelo de datos se llevaron a cabo algunos cambios, ya que durante la implementación de algunas funcionalidades, los desarrolladores se dieron cuenta de que este no estaba estructurado de la manera más indicada para obtener los resultados esperados, es por esto que en la tbl_pc se le agregó un nuevo atributo “codigopc” el cual pasaría a ser él identificador de la misma ya que el numeroSerie que era el identificador anterior y por ende tenía que ser único, a la hora de querer entrar una nueva solicitud de licencia para alguna PC que por algún motivo se le hubiese tenido que cambiar los datos del hardware ya no sería posible. En la tbl_solicitud_licencia se cambió el atributo numeroSerie por codigopc ya que éste dejó de ser el identificador de la tbl_pc.

Una vez realizadas las licencias a partir de las solicitudes estas dejan de existir, luego para acceder a las licencias o para saber a qué PC correspondía la misma no era posible como estaba, por lo que se agregó una nueva relación entre las tablas tbl_pc y tbl_licencia donde el identificador de la tbl_pc pasó como un nuevo atributo para la tbl_licencia, se le agregó un nuevo atributo “importada” para que el licenciador pueda saber cuándo una licencia ya fue importada de la base de datos y se eliminó el atributo codigoActivacion ya que ese no aportaba ningún valor significativo.

Debido a que en la base de datos se guarda información de vital importancia para el licenciamiento del alas PACS y por ende el acceso a la misma debe de ser restringida y controlada para evitar robo, filtración o perdidas, se le agregó una nueva tabla llamada tbl_autenticacion donde se guardara toda la información referente a los usuarios a los cuales se le dará acceso a la base de datos.

Se adicionó la funcionalidad Búsqueda Avanzada al módulo Gestor el cual permite realizar la búsqueda de licencias y solicitudes a partir de algún criterio seleccionado por él usuario. A partir del punto de vista de los desarrolladores se cambió el método Eliminar Solicitud existente en el módulo Cliente del sistema por el método Editar, este permite cambiar los datos de ubicación de la PC en caso de que hayan sido introducidos incorrectamente, ya que el método eliminar no cumplía con ninguna funcionalidad debido a que una vez cerrada la aplicación ya se eliminaba la solicitud existente en ese momento.

Los requisitos funcionales no son solo los que juegan un papel importante en la implementación por lo que hay que tener en cuenta los requisitos no funcionales que son los que hacen el producto usable, atractivo, rápido y confiable. A continuación se describen los requisitos no funcionales con que debe contar el sistema:

Seguridad: El servidor de bases de datos y el servidor de aplicación estarán instalados en computadoras diferentes.

Apariencia o interfaz externa: El sistema debe poseer una interfaz amigable al usuario para brindar facilidades que permitan interactuar y navegar con el sistema de forma fácil y rápida.

Soporte: Se debe brindar capacitación a los usuarios de la herramienta para el adecuado uso y explotación de la misma.

Del diseño propuesto se pudieron extraer las clases fundamentales que deben ser definidas para que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, dando una idea clara de lo que se debe implementar. El diseño fue creado a partir del seguimiento de patrones, que de manera general constituyen soluciones simples y elegantes a

problemas específicos y comunes del diseño orientado a objetos, que permiten llevar a cabo la implementación clara y limpia del subsistema.

2.4 Arquitectura de la solución.

La arquitectura para darle solución al problema planteado es la definida en el marco de trabajo del centro, es decir, la arquitectura en capa o por niveles, la misma implementa el patrón modelo – vista – controlador.

La arquitectura 3 capas o programación 3 capas consiste literalmente en separar un proyecto en **Capa de Presentación**, **Capa de Negocio** y **Capa de Datos**. Esto permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API (Interfaz de Aplicación) que existe entre niveles.

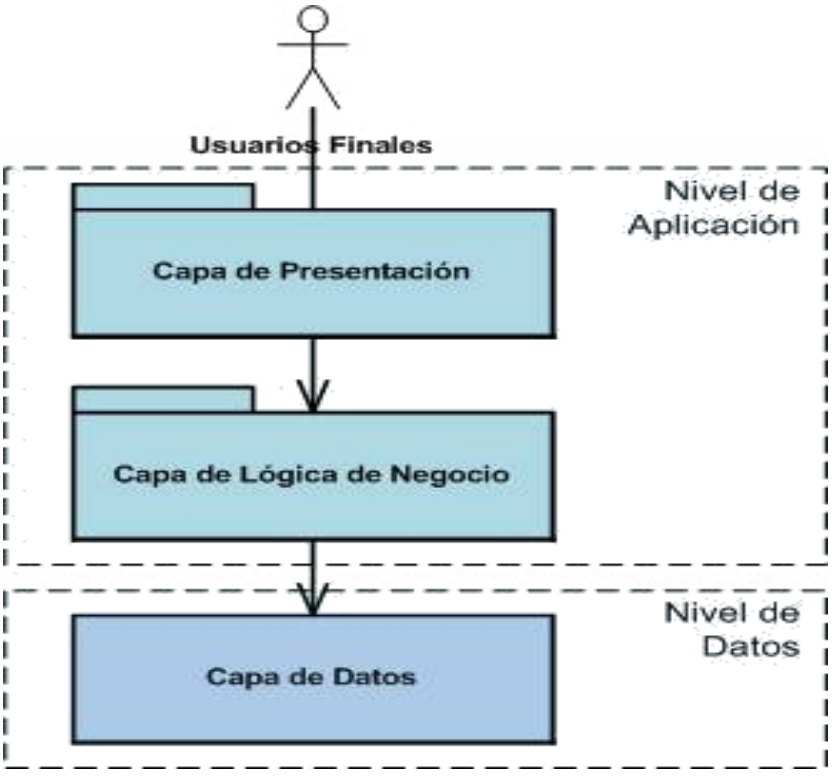


Fig. 2 Representación gráfica de la Arquitectura 3 Capas.

Ventajas de esta Arquitectura:

- El desarrollo se puede llevar a cabo en varios niveles.
- Desarrollos paralelos (en cada capa).
- Aplicaciones más robustas debido al encapsulamiento.
- En caso de que sobrevenga algún cambio, solo se ataca al nivel requerido sin tener que revisar entre código mezclado.

- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica).
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad).
- Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente con añadir más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

2.4.1 Capas y Niveles.

Capa de Presentación

Esta es la parte que el usuario ve, las pantallas que se le muestra para que él interaccione con el programa (para algunos “capa de usuario”), comunicándole la información y recolectando la información suministrada por el usuario en un mínimo de proceso (realiza validaciones para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio para llevar y traer los datos o registros necesarios, es la interfaz gráfica del programa y debe ser lo más amena posible para una mejor comunicación con el usuario.

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 todos los procesos que deben realizarse.

Capa de datos

Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. (35)

Esta solución de tres capas (presentación, lógica del negocio, datos) reside en dos ordenadores por lo que se dice que la arquitectura de la solución es de tres capas y dos niveles.

2.4.2 Patrones a utilizar.

Modelo-Vista-Controlador

El patrón de diseño Modelo-Vista-Controlador o MVC describe una forma muy utilizada de organizar el código de una aplicación que consiste en separar los datos, la interfaz de usuario, y la lógica de control en tres componentes distintos:

Modelo: Es la representación específica de la información con la cual el sistema opera. Encapsula

los datos y las funcionalidades del mismo, al ser independiente de cualquier representación de entrada y/o salida.

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. Ante estos eventos usualmente invoca peticiones al modelo y probablemente a la vista.

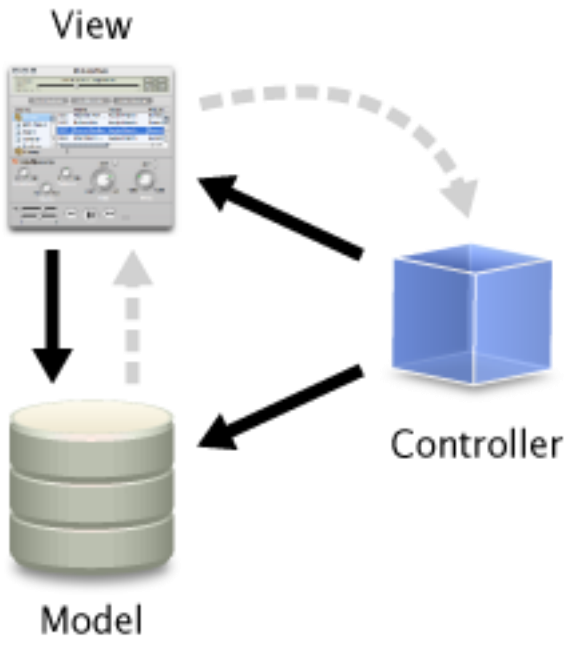


Fig. 3 Patrón Modelo-Vista-Controlador.

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema).
- Lleva un registro de las vistas y controladores del sistema.

El controlador es responsable de:

- Recibir los eventos de entrada
- Contiene reglas de gestión de eventos. Estas acciones pueden suponer peticiones al modelo o a las vistas.
- Carga el modelo, las librerías, helpers, plugins, y todos los demás recursos necesarios para satisfacer nuestra petición.

Las vistas son responsables de:

- Recibir datos del controlador y mostrarlos al usuario.

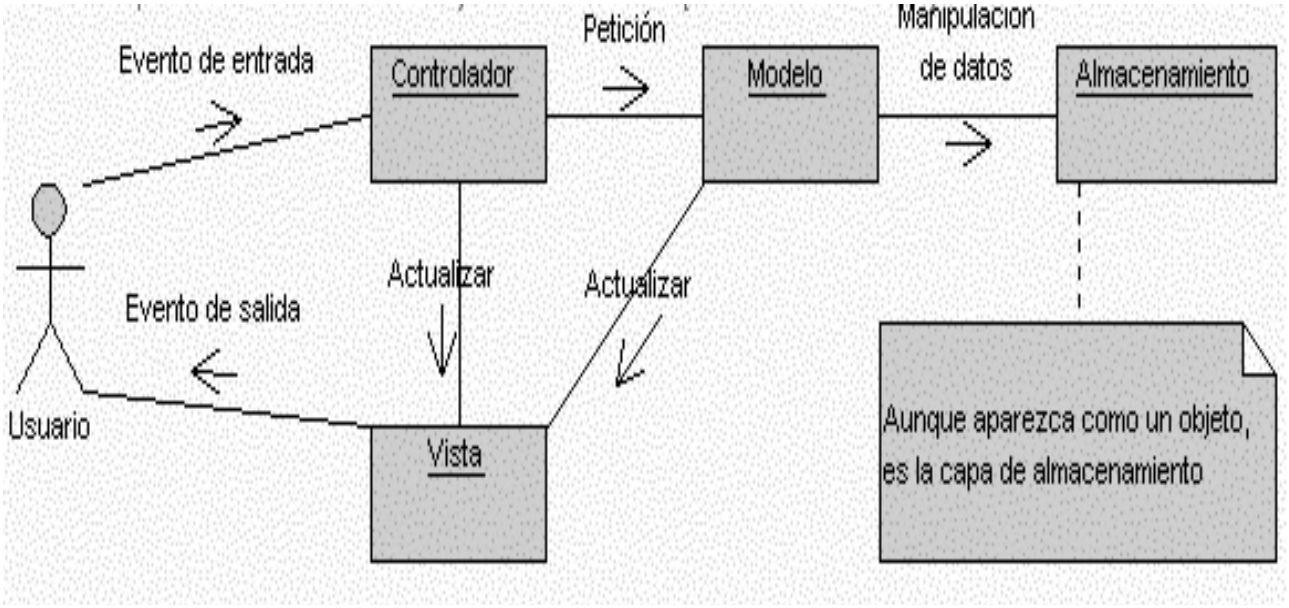


Fig. 4 Flujo del patrón Modelo-Vista-Controlador

En este capítulo se describe de forma general el flujo de la solución propuesta por el analista para el sistema que se desea desarrollar. Al mismo tiempo se realiza una descripción de las principales funcionalidades a implementar para un mejor entendimiento de los requisitos y se analiza de manera detallada el diseño propuesto por el analista, se realiza un análisis crítico y se señalan los cambios realizados, en caso de que así lo requiera a la hora de la transición del diseño a la codificación. Esta sección también incluye la descripción de la arquitectura que se va a seguir mientras se lleve a cabo el flujo de implementación así como los patrones que la misma implementa.

CAPÍTULO 3: Implementación del Sistema

El presente capítulo presenta la implementación del sistema el cual tiene como objetivo definir la organización del código al tener en cuenta los subsistemas de implementación, la implementación de los elementos de diseño en términos de ficheros fuentes y ejecutables, para poder integrar los diferentes componentes y generar un ejecutable entregable o producto final. Se definen los estándares de codificación a seguir durante la implementación y se presenta el diagrama de componentes, el cual muestra las dependencias entre los componentes más significativos del software.

3.1 Estándares de codificación.

Los estándares de codificación son pautas de programación, comprende todos los aspectos de la generación de código que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Es prudente establecer estándares de codificación para todos los programadores de cada equipo de desarrollo para asegurarse de que todos trabajen de forma coordinada.

Las técnicas de codificación incorporan muchos aspectos del desarrollo del software, aunque generalmente no afectan a la funcionalidad de la aplicación, sí contribuyen a una mejor compresión del código fuente. La legibilidad del código repercute directamente en lo bien que un programador comprende un sistema de software.

En general una técnica de codificación no pretende formar un conjunto inflexible de estándares de codificación. Más bien intenta servir de guía en el desarrollo de un estándar de codificación para un proyecto específico de software.

Los aspectos para los que generalmente se establecen estándares son los siguientes:

- Identificadores.
- Indentación.
- Líneas y espacios en blanco.
- Comentarios.
- Declaraciones.
- Algoritmización.

Cada equipo de desarrolladores define cuales de estos aspectos estandarizar y que estilo aplicar a cada aspecto. Existen innumerables combinaciones y muchos más aspectos que los especificados anteriormente.

El cumplimiento de los estándares de codificación debe reflejar un estilo armonioso, como si un

único programador hubiese escrito todo el código. De esta manera se logra mayor legibilidad y facilidad de mantenimiento. Los estándares deben responder además a acciones prácticas que acomoden al programador. La adopción de un estándar de codificación solo es viable si se sigue desde el principio hasta el final del proyecto de software. Los estándares se definen según el estilo de cada programador, las características propias del lenguaje de programación, los recursos del lenguaje que se utilizarán y el tipo de programa que se debe implementar.

3.1.1 Identificadores.

Existen estilos de estándares de codificación para los identificadores definidos mundialmente como el lowerCamelCase y el UperCamelCase. De la escritura del nombre de los mismos se puede inferir en qué consisten. Cada palabra interna en identificadores compuestos comienza con mayúsculas para ambos estilos. Para el primero, el identificador comienza con minúscula y para el segundo, el identificador comienza con mayúscula. Además no se colocan caracteres de separación entre las palabras que conforman un identificador compuesto.

Cada identificador debe ser lo suficientemente descriptivo sin importar del tipo que sea. Se debe evitar escribir abreviaturas que pueden confundir a otros programadores por desconocer su significado. No es conveniente utilizar palabras que den un significado ambiguo. A la vez, es necesario buscar los sinónimos más cortos para que el identificador sea sencillo y a la vez expresivo.

3.1.1.1 Identificadores de las clases.

El nombre de las clases comienzan con mayúscula y las demás letras con minúscula, en caso de que sea un nombre compuesto su utiliza la notación Camel Casing.

3.1.1.2 Identificadores de las funciones.

Para el nombre de las funciones se establece el UperCamelCase.

3.1.1.3 Identificadores de las variables.

Para el nombre de las variables se escribe la primera palabra con minúscula, en caso de que sea un nombre compuesto se utiliza la notación lowerCamelCase.

3.1.1.4 Identificadores de las tablas.

El nombre se escribe con minúscula antecedido de tbl_, será lo suficientemente descriptivo, y sin el uso de abreviaturas, en caso de que el nombre sea compuesto se recurre a la notación lowerCamelCase.

3.1.2 Indentación.

La indentación en una buena práctica de programación consiste en comenzar a escribir cada línea de código a diferentes distancias desde el borde izquierdo del área de texto del editor. Esta distancia

está determinada por la jerarquía que se forma al introducir sentencias dentro de bloques de estructuras. Esto persigue mayor legibilidad y entendimiento para el programador e igualmente depende del propio estilo de cada persona, del lenguaje y tipo de programa que se implementa.

Se definió que la indentación se hará al agregar un tab al inicio de la línea que se desee escribir por cada unidad del grado de nivel que tenga la línea al comenzar desde cero. La cantidad de espacios que incluya el tab dependerá del editor de texto y será, preferentemente, cuatro. Se escribirá solo una sentencia por línea de código y en el caso de cortar las líneas, se hará luego de una coma o antes de un operador. La sección de la derecha de la línea que se corte se ubicará en la línea siguiente indentada al nivel de la expresión correspondiente en la línea superior.

3.1.3 Llaves.

Algunos programadores difieren en criterios en cuanto a la ubicación de las llaves que delimitan el cuerpo de los bloques de código en los lenguajes que contienen este tipo de estructura. Muchos prefieren hacerlo ubicando la llave de apertura inmediatamente detrás de la línea cabecera del bloque mientras otros apuestan por ubicarlas de forma solitaria en la línea siguiente a la línea cabecera. Para éste último estilo existen además diferencias en cuanto al nivel de indentación de las mismas. Algunos lo hacen al nivel de la línea cabecera y otros al nivel de las líneas del cuerpo del bloque. De igual forma algunos prefieren usar siempre las llaves mientras otros prefieren aprovechar las ventajas del lenguaje obviándolas en los casos que el cuerpo del bloque contenga solo una sentencia.

En la implementación de este trabajo las llaves de apertura se colocarán solitarias en la línea siguiente e indentadas al nivel de la línea cabecera del bloque. Las llaves de cierre se colocarán solitarias en la línea que sigue a la última línea dentro del bloque e indentadas al nivel de la línea cabecera del bloque. En el caso de cuerpos de bloque con una sola sentencia se podrá o no usar las llaves a gusto del programador. Este estilo agrega más líneas de código al programa al ubicar las llaves solitarias en una línea pero a su vez se gana en legibilidad del código. Es por eso que no existe una definición exacta de cual sería una buena práctica de programación en el uso de llaves debido a que cada método tiene ventajas y desventajas.

3.1.4 Líneas y espacios en blanco.

Para mejorar la legibilidad del código muchas veces se utilizan líneas en blanco para separar segmentos de código que pueden corresponder a clases, funciones, declaraciones, implementaciones, comentarios, bloques o sencillamente secciones críticas que se deseen despejar. Así mismo sucede cuando se utilizan para separar elementos dentro de las sentencias de código. A veces se separan con espacios cada operador de su respectivo operando, paréntesis, identificadores, símbolos y algunos lenguajes exigen que se separen las palabras propias del

vocabulario de las adyacentes para ser comprendidas por los compiladores.

Durante el desarrollo del trabajo se colocaron:

Líneas en blanco:

- Entre funciones.
- Antes de estructuras de control.
- Entre declaraciones de variables e implementaciones dentro del cuerpo de las funciones.
- Entre las declaraciones de interfaces, espacios de nombre, estructuras y clases.

Espacios en blanco:

- Entre las palabras reservadas y los elementos adyacentes a las mismas.
- Después de las comas en la lista de argumentos de las funciones.
- Entre los operadores binarios y los elementos adyacentes a los mismos.
- Después de cada punto-y-coma (“;”) en las estructuras for.

Se puede observar que el uso de líneas en blanco incrementa la longitud en líneas de código del programa y el uso de espacios en blanco incrementa la longitud de cada línea de código del programa. Esto, sin embargo, mejora la legibilidad del código. Es por esto que se deben establecer los estándares de conjunta aprobación entre los miembros del equipo de programadores.

3.1.5 Comentarios.

Aunque parezca innecesario el uso de comentarios y que atrasa el proceso de codificación, se ha demostrado, que esta práctica es beneficiosa por varias razones:

- Ayuda al programador a entender cada elemento o sección de código.
- Ayuda a adaptar el código durante su reutilización.
- Sirve de guía en los casos en que varios programadores trabajen sobre las mismas secciones del código.
- Disminuye el esfuerzo de análisis ya que el lenguaje natural es más legible que cualquier lenguaje de programación.
- Su importancia se aprecia al trabajar con código entre grandes intervalos de tiempo donde generalmente se olvida lo que se pensó en un momento.

Los comentarios se pueden utilizar para varios fines:

- Se utilizan para explicar el propósito de las funciones.
- Para explicar las características fundamentales de las clases.
- Para sintetizar las acciones de los algoritmos complejos.
- Para aclarar los datos que representan las variables.
- Para dividir secciones de código en dependencia de los diferentes contextos.

- Para dejar constancia del autor y algunas condiciones en que se generó el código.
- A veces, se usan comentarios temporales para recordar cosas que faltan o que se deben modificar o analizar en otro momento.

Al escribir comentarios es necesario tener en cuenta algunos detalles:

- La capacidad de síntesis.
- El uso de lenguaje técnico.
- No repetir exactamente paso por paso lo que hace el algoritmo sino expresar un resumen de su propósito.
- Usar un estilo uniforme de comentario definido en estándares para todo el equipo.
- Escribir el comentario solo donde sea necesario.
- No usar expresiones obscenas e informales que dañen el ambiente de respeto entre los desarrolladores.

Los propósitos, la importancia y las buenas prácticas en el uso de comentarios varían en dependencia de los programadores, el lenguaje de programación y el tipo de programa que se realice. Solo se presentaron los aspectos más usuales.

Existen dos tipos de comentarios:

- Comentarios de bloque o de varias líneas cuya sintaxis para el C# es “/*. Éste es un comentario de bloque */.
- Comentarios lineales o de una sola línea cuya sintaxis para el C# es “//. Éste es un comentario de línea.

Identificada la necesidad de incluir comentarios en el código se definen estándares para homogeneizar los mismos.

En este trabajo se estableció la utilización de ambos tipos (comentarios de bloque y comentarios lineales). Se colocaron encima de la línea a la que se le quiso aplicar el comentario o encima de la línea cabecera del bloque al que se le deseo aplicar. La indentación se hizo al nivel de la línea en cuestión y se utilizaron en funciones y clases. Se puede utilizar en algoritmos no triviales y secciones de diferentes contextos dentro de los métodos. Se deben realizar revisiones para eliminar los comentarios temporales cuyos segmentos de código ya se hayan tratado.

3.1.6 Otros estándares.

Se deben asignar los tipos de datos adecuados a las variables para no hacer uso excesivo de recursos. Por ejemplo: Programación en C#; para una variable que almacena la cantidad de ocurrencias de un objeto en una colección cuya cantidad no supera las 100 unidades, es lógico

asignar un tipo int a esa variable en lugar de un tipo *long* para ahorrar memoria y aumentar la eficiencia en tiempo de ejecución.

Cuando se precise realizar recorridos a través de colecciones, se deben utilizar las estructuras de control adecuadas. Por ejemplo: Programación en C#; al intentar recorrer una colección y en cuyo recorrido solo se tomen valores de los objetos en cada posición, es oportuno hacerlo mediante un *foreach*. Sin embargo si se modificará la información en alguno de estos objetos, se puede utilizar un *for* en lugar del *foreach* porque este último no permite modificaciones. Si se produce un entramado complejo de expresiones *if*, es prudente emplear el *switch.*

Todos los atributos de las clases son privados[2] y deben tener sus correspondientes propiedades[3]. El identificador de éstas, como todo método, debe responder al estilo *UperCamelCase*. Para los atributos estrictamente internos de las clases, que no tengan intercambio de información con el exterior, no es necesario implementar sus propiedades. Para referirse a atributos dentro de los métodos de una clase se hará al invocar su propiedad.

En las clases, se declaran primeramente los atributos, luego las propiedades, seguidamente los constructores y por último el resto de los métodos. Dentro de las funciones se declaran primero todas las variables locales y luego el resto de las instrucciones. Debe tenerse en cuenta que el identificador de cada parámetro que reciba la función será leído por todo aquel que invoque a ese método dentro y fuera de la clase, por tanto debe tener una palabra razonable y descriptiva.

3.2 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.

[2] Privado: Condición establecida por el modificador de acceso “private” que mantiene al miembro accesible solo dentro de la clase.

[3] Propiedades: Métodos especiales llamados descriptores de acceso que controlan la asignación o lectura de información de los atributos de una clase.

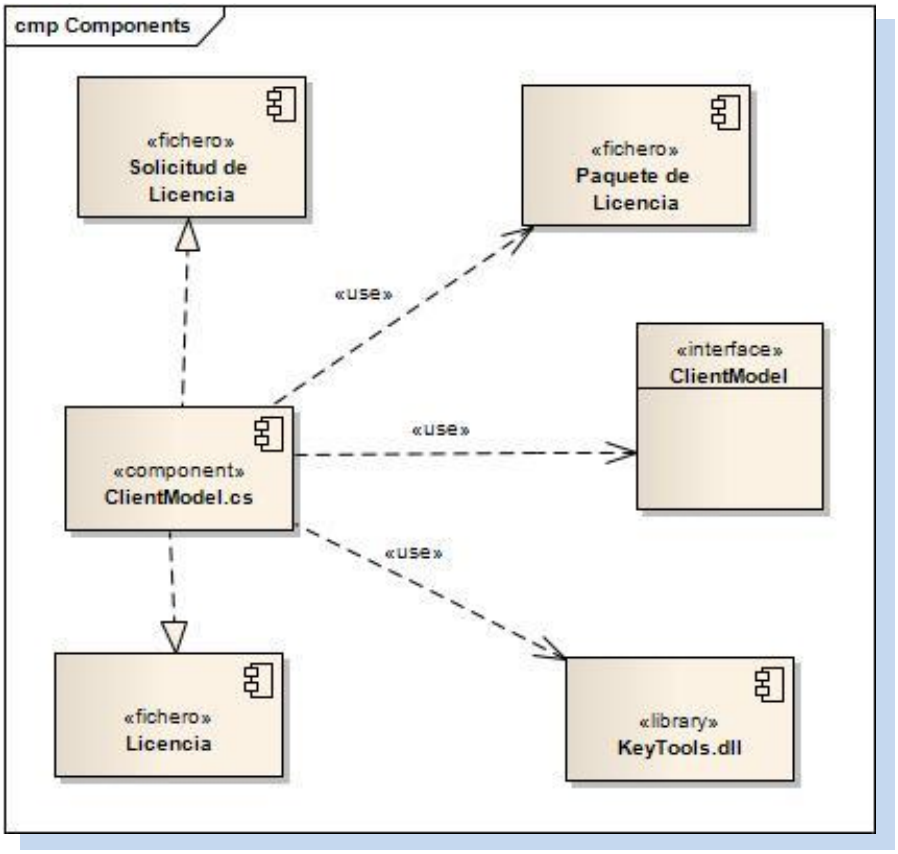


Fig. 5 Diagrama de Componentes del Cliente.

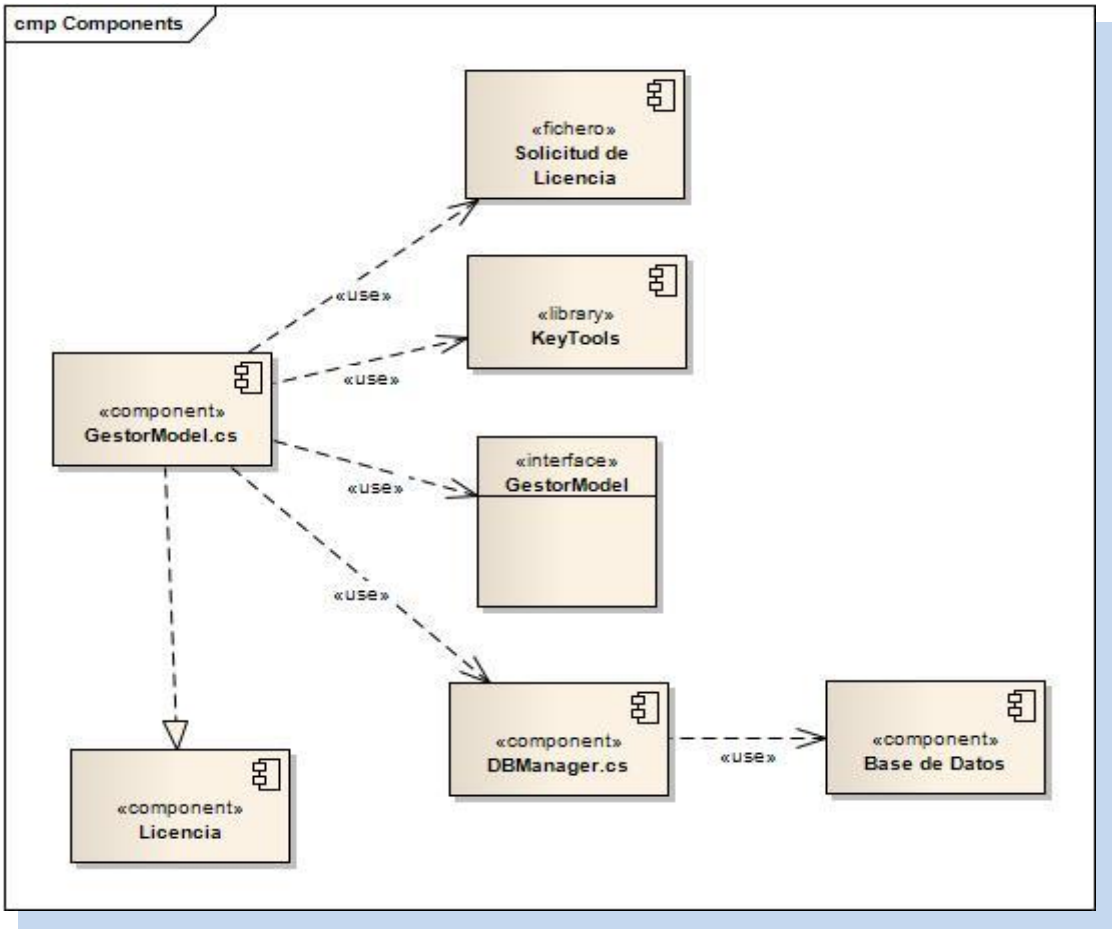


Fig. 6 Diagrama de Componentes del Gestor.

Con el desarrollo de este capítulo se realizó un análisis de los estándares de codificación que permiten la uniformidad y organización del código fuente. Se presentó el diagrama de componentes para mostrar los principales componentes del sistema y las relaciones entre ellos para lograr un mejor entendimiento del flujo del mismo y se consiguió la solución al problema planteado.

CAPÍTULO 4: Pruebas del Sistema

En el desarrollo de este capítulo se valida mediante pruebas si el software realizado ha cumplido el objetivo planteado con la calidad requerida. Se definen las pruebas que se le realizan al sistema como las pruebas de caja negra y de caja blanca, y se diseñan los casos de pruebas correspondientes.

4.1 Pruebas de software.

Durante el proceso de desarrollo de un software, los errores pueden comenzar a aparecer incluso desde el mismo momento en que fueron definidos los objetivos y estos a su vez especificados de forma incorrecta; de la misma forma en los posteriores pasos del diseño y desarrollo. A raíz de la incapacidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software debe ser acompañado de una actividad que garantice su calidad.

El único instrumento adecuado para determinar el status de la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos.

La prueba del software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación. Es una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. Estas involucran un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa.

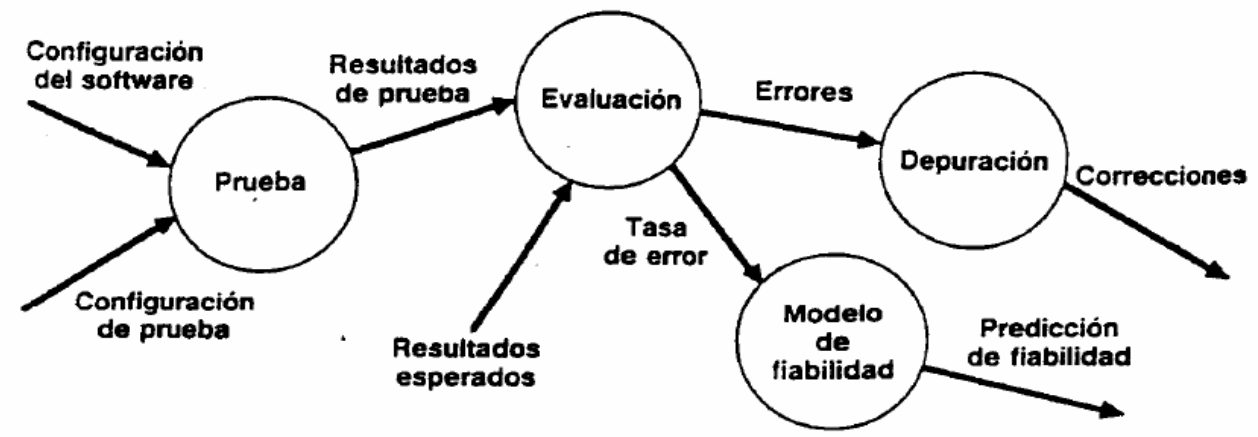


Fig. 7 Proceso de Pruebas.

Objetivos de la prueba:

- La prueba es un proceso de revisión de un programa a partir de su ejecución que consiste en descubrir errores.
- Un caso de prueba bien diseñado es aquel que tiene mayor posibilidad de descubrir un error que no ha sido hallado hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta el momento.

A la hora de realizarle una prueba a un producto de software se debe escoger los tipos de prueba que se adapte mejor al sistema que se va a probar. Por lo que se debe tener en cuenta el lenguaje de programación utilizado, así como el proceso de desarrollo, las características de los desarrolladores, la plataforma en que se ejecutan los procesos, los errores más importantes, el tipo de aplicación implementada y si realiza conexiones a bases de datos o no.

Antes de liberar el producto es necesario tener la certeza de que éste se encuentra libre de errores, aunque se considera que no se puede estar 100% seguro de esto, pero mientras más pruebas se le realicen al software más cerca se podrá estar de lograr un producto con la calidad esperada por los usuarios.

4.1.1 Diseño de casos de prueba.

Una vez generado el código fuente, es necesario probar el software para descubrir la mayor cantidad de errores posible y corregirlos antes de entregarlo al cliente. A partir de este momento, las técnicas de evaluación de software empiezan a jugar un papel significativo dentro del desarrollo de software .Su objetivo es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar el mayor número de errores con la menor cantidad posible de tiempo y esfuerzo.

Los casos de pruebas son condiciones o variables bajo las cuales debe funcionar un programa para proporcionar resultados satisfactorios al analista según los objetivos definidos.

Cualquier producto de ingeniería se puede probar de dos formas:

4.1.1.1 Pruebas de caja negra.

Este tipo de prueba no permite la comprensión del comportamiento interno del software. Actúan fundamentalmente sobre la interfaz, para probar la funcionalidad según los requisitos. Tiene como propósito verificar las relaciones de entrada y salida de datos de las operaciones concernientes a qué hace la unidad sin indagar en cómo lo hace.

Los casos de prueba de caja negra pretenden demostrar que:

- Los requisitos funcionales del software son operantes.
- La entrada se acepta de forma correcta.
- Se produce una salida correcta.

- La integridad de la información externa se mantiene.

4.1.1.1.1 Partición de equivalencia.

La partición de equivalencia es un tipo de pruebas de caja negra de las cuales se pueden obtener casos de prueba a partir de las clases de datos obtenidas de las divisiones de los campos de entrada de un programa. Cada una de las clases de equivalencia representan a un conjunto de estados válidos o inválidos para las condiciones de entrada de los datos de un programa.

4.1.1.2 Pruebas de caja blanca.

Para la realización de estas pruebas el probador tiene acceso a las estructuras de datos, al código, y a los algoritmos internos. Se basan en el análisis minucioso de los detalles procedimentales. Se comprueban los caminos lógicos del sistema y se generan los casos de prueba que ejerciten las estructuras condicionales y los bucles. Existen varios métodos que analizan diferentes partes del programa y se complementan entre sí para garantizar la calidad del sistema.

Las pruebas de caja blanca pretenden responder a que:

- Todos los caminos independientes de cada módulo se ejecuten al menos una vez.
- Se manejan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecuten todos los bucles en sus límites.
- Se utilizan todas las estructuras de datos internas.

4.1.1.2.1 Cobertura de caminos.

Este asegura que los casos de prueba diseñados permitan que todas las sentencias del programa sean ejecutadas el menos una vez y que las condiciones sean probadas tanto para su valor verdadero como falso.

Para aplicar este criterio se puede realizar la Prueba del Camino Básico. Ésta se utiliza para comprobar la complejidad lógica de un diseño procedimental, permite diseñar casos de prueba para cubrir todas las sentencias de un programa a partir de la obtención de un conjunto de caminos independientes.

Los pasos para realizar esta técnica son:

- Representar el programa en un grafo de flujo.
- Calcular la complejidad ciclomática.
- Determinar el conjunto básico de caminos independientes.
- Derivar los casos de prueba que fuerzan la ejecución de cada camino.

A continuación se detallan cada uno de estos pasos.

4.1.1.2.1.1 Representar el programa en un grafo de flujo.

El grafo de flujo se utiliza para representar el flujo de control lógico de un programa, donde:

- Los nodos representan cero, una o varias sentencias en secuencia. Cada nodo comprende como máximo una sentencia de decisión.
- Las aristas son líneas que unen dos nodos.
- Las regiones son áreas delimitadas por aristas y nodos. Cuando se contabilizan las regiones de un programa debe incluirse el área externa como una región.
- Los nodos predicados: cuando en una condición aparecen uno o más operadores lógicos (AND, OR, XOR,…) se crea un nodo distinto por cada una de las condiciones simples. Cada nodo generado de esta forma se denomina nodo predicado.

Así cada construcción lógica de un programa tiene una representación:

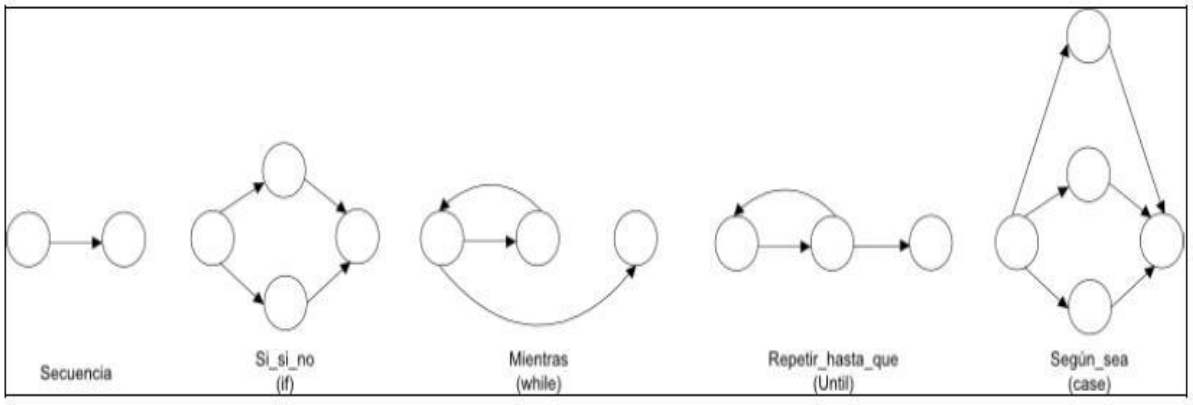


Fig. 8 Representación en grafo de flujo de las estructuras lógicas de un programa

4.1.1.2.1.2 Calcular la complejidad ciclomática.

La complejidad ciclomática es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. La métrica, propuesta por Thomas McCabe en 1976, se basa en la representación gráfica del flujo de control del programa. De dicho análisis se desprende una medida cuantitativa de la dificultad de prueba y una indicación de la fiabilidad final. Cuando se utiliza en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y proporciona el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

Para ayudar a los ingenieros de software a determinar la estabilidad y el riesgo inherente de un programa se ha medido un gran número de programas, con el fin de establecer rangos de complejidad. La medida resultante puede ser utilizada en el desarrollo, mantenimiento y reingeniería para estimar el riesgo, costo y estabilidad.

Algunos estudios experimentales indican la existencia de distintas relaciones entre la métrica de McCabe y el número de errores existentes en el código fuente, así como el tiempo requerido para encontrar y corregir esos errores.

Se suele comparar la complejidad ciclomática obtenida contra un conjunto de valores límite como se observa en la figura 9.

Complejidad Ciclomática	Evaluación de Riesgo
1-10	Programa simple, sin mucho riesgo
11-20	Más Complejo, riesgo moderado
21-50	Complejo, programa de alto riesgo
50	Programa no testeable, muy alto riesgo

Fig. 8 Complejidad ciclomática contra evaluación de riesgo.

Existen varias formas de calcular la complejidad ciclomática de un programa a partir de un grafo de flujo:

1. El número de regiones del grafo coincide con la complejidad ciclomática, V (G).
2. La complejidad ciclomática, V (G), de un grafo de flujo G se define como
   V (G) = Aristas – Nodos + 2.
3. La complejidad ciclomática, V(G), de un grafo de flujo G se define como V (G) = P + 1.(36)

4.1.1.2.1.3 Determinar el conjunto básico de caminos independientes.

Un camino independiente es cualquier camino del programa que introduce, por lo menos un nuevo conjunto de sentencias de proceso o una condición, respecto a los caminos existentes. En términos de diagrama de flujo, un camino independiente está constituido al menos por una arista que no haya sido recorrida anteriormente a la definición del camino. En la identificación de los distintos caminos de un programa para probar se debe tener en cuenta que cada nuevo camino debe tener el mínimo número de sentencias nuevas o condiciones nuevas respecto a los que ya existen, de esta manera se intenta que el proceso de depuración sea más sencillo.

Este método permite obtener V (G) caminos independientes cubriendo el criterio de cobertura de decisión y sentencia.

4.1.1.2.1.4 Derivar los casos de prueba que fuerzan la ejecución de cada camino.

El último paso es construir los casos de prueba que fuerzan la ejecución de cada camino. Se trata de diseñar pruebas que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y de tiempo.

El proceso de realización de pruebas a un software, está precedido por el diseño de los casos de prueba, que se definen según las funcionalidades descritas en los casos de uso. Se parte de las descripciones de los casos de uso del sistema, como apoyo para las revisiones.

Las desventajas de este tipo de prueba es que resulta imposible abarcar todo el código fuente de un sistema medianamente grande. El tiempo necesario para realizarlas sería considerable y se torna compleja su aplicación sobre algoritmos críticos.

4.2 Aplicación de pruebas de caja negra.

A continuación se le aplicará prueba de caja negra al caso de uso Obtener Datos de la PC, dentro de éste se encuentran Obtener datos de ubicación de la PC y Obtener datos de hardware de la PC. El objetivo del mismo es obtener estos datos para realizar la solicitud de licencia de una PC específica, para evitar que con una solicitud se puedan licenciar varias PC. En la tabla 5 se muestra el caso de uso a probar.

Tabla: 5 Secciones a probar en el Caso de Uso Obtener datos de la PC.

Nombre de la sección	Escenario de la sección	Descripción de la funcionalidad
SC1: Obtener datos de ubicación de la PC.	EC1.1: Obtener datos de ubicación de la PC correctamente.	Se introducen todos los datos de la ubicación de la PC ya que son obligatorios.
	EC1.2: Obtener datos de la PC dejando campos obligatorios vacíos.	Se dejan campos vacíos cuando se introducen los datos de ubicación de la PC.

A partir de esta descripción se detallan las variables que se encuentran en las interfaces asociadas al caso de uso Obtener Datos de la PC.

Tabla: 6 Descripción de variables.

No.	Nombre de campo	Clasificación	Puede ser nulo	Descripción

1	No. Serie.	Texto	No	Número de serie de la PC donde se va a instalar el sistema.
2	Institución	Lista desplegable	No	Nombre de la institución donde se va a desplegar el sistema.
3	Dirección de la Institución.	Lista desplegable	No	Dirección de la institución donde se va a desplegar el sistema.
4	Ubicación.	Lista desplegable	No	Ubicación de la PC donde va a ser instalado el sistema.

Esta descripción posibilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los valores introducidos en el sistema. Para ello se utilizó un juego de datos válidos e inválidos, a partir de la técnica de partición de equivalencia.

Tabla: 7 Matriz de datos.

Id del Escenario	**Escenario**	**Variables**				**Respuesta del sistema**	**Resultado de la prueba**	**Flujo central**
		1	2	3	4			
EC 1.1	Obtener datos de ubicación de la PC correctamente.	V	V	V	V	Registra los datos de la ubicación de la PC, muestra el mensaje “Datos de la PC registrados correctamente “y activa las opciones de Editar y	Satisfactorio	1. Acceder a la pestaña Solicitud. 2. El licenciador introduce los datos de la PC. 3. El sistema verifica que los campos requeridos hayan sido completados.

						Exportar.		
EC 1.2	Obtener datos de la PC dejando campos obligatorios vacíos.	I	V	V	V	Muestra un asterisco al lado de cada campo indicando que el campo es obligatorio.	Satisfactorio	1.1 Si quedan campos sin llenar el sistema muestra un asterisco al lado del campo indicando que es obligatorio introducir los datos.
		V	I	V	V			
		V	V	I	V			
		V	V	V	I			
		I	I	V	V			
		V	V	I	I			
		I	I	I	I			
		V	I	I	I			
		I	V	I	I			
		I	I	V	I			
		I	I	I	V			

Con la ejecución de los Casos de Prueba de Caja Negra se obtuvieron resultados satisfactorios, quedó demostrado que todas las funcionalidades con que cuenta la aplicación se corresponden en su totalidad con las descripciones. Los demás casos de pruebas se muestran en los anexos.

4.3 Aplicación de pruebas de caja blanca.

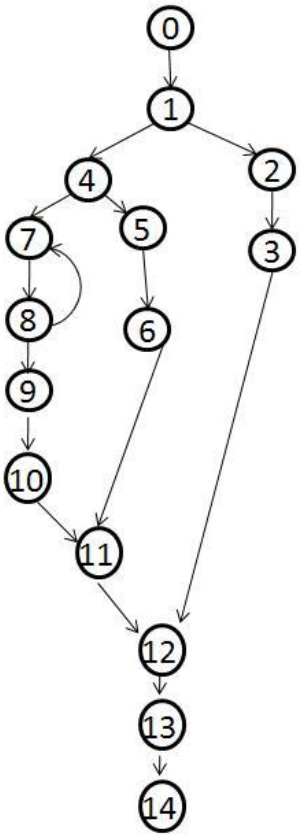
En esta sección se escogerán algunas funciones importantes para la realización de pruebas de caja blanca de forma manual. Primeramente se realizará el grafo correspondiente, luego se calculará la complejidad ciclomática mediante las tres variantes explicadas anteriormente y por último se obtendrá el conjunto de caminos independientes para el grafo obtenido.

Entre las funcionalidades escogidas para la realización de pruebas de caja blanca se encuentra Eliminar Licencia del módulo Gestor, este se encarga de eliminar una o varias licencia que por alguna causa específica ya no cumpla objetivo tenerla almacenada en base de datos, este se realiza cuando se selecciona la solicitud que se desea eliminar. En la figura 9 se representa el código correspondiente al método EliminarLicencia, mientras que la figura 10 muestra el grafo correspondiente y los resultados de las pruebas.

```
// ************ Eliminar Licencia **********************
0  private void belimlicencias_Click(object sender, EventArgs e)
   {
1    if (gestormodel.Licencias.Count == 0)
     {
2      button18.Image = Gestor.Properties.Resources.advertencia1;
       button18.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
       button18.Visible = true;
       button18.Text = "   No existen licencias en base de dato";
       timer1.Start();
3    }
     else
     {
4      if (listView2.CheckedItems.Count == 0)
       {
5        button18.Image = Gestor.Properties.Resources.advertencia1;
         button18.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
         button18.Visible = true;
         button18.Text = "   Debe de seleccionar la licencia a eliminar";
         timer1.Start();
6      }
       else
       {
7        for (int i = 0; i < listView2.CheckedItems.Count; i++)
         {
8          string num_serie = listView2.CheckedItems[i].SubItems[1].Text;
           gestormodel.EliminarLicencia(num_serie);
9        }
10       button18.Visible = true;
         button18.Text = "La licencia ha sido eliminada satisfactoriamente";
         timer1.Start();
11     }
12   }
13   MostrarLicencias();
14 }
```

Fig. 9 Método EliminarLicencia.

A continuación se muestra el grafo correspondiente, así como los resultados de la complejidad ciclomática y los caminos independientes.



Complejidad ciclomática

A=17(aristas)
N=15(nodos)
V(G)=17-15+2
V(G)=2+2
V(G)=4

P=3(nodos predicados)
V(G)=P+1
V(G)=3+1
V(G)=4

R=4(regiones)
V(G)=R
V(G)=4

Caminos independientes:
0-1-2-3-12-13-14
0-1-4-5-11-12-13-14
0-1-4-7-8-9-10-11-12-13-14
0-1-4-7-8-7-8-9-10-11-12-13-14

Fig. 10 Resultados de las pruebas de Caja Blanca del método EliminarLicencia.

Casos de pruebas que fuerzan la ejecución de cada camino para el método EliminarLicencia.

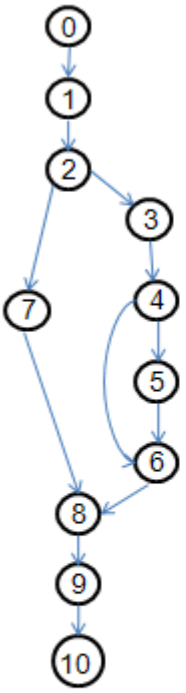
No del camino	Caso de prueba	Resultado esperado
1	No existen licencias.	Realiza el primer if y sale del método.
2	Contiene elementos la lista de licencias pero no se ha seleccionado las licencias a eliminar.	Realiza el else del primer if, realiza el segundo if y sale del método.
3	Existe un elemento seleccionados de la lista de licencias.	Realiza el else del primer if, el else del segundo if y ejecuta el bucle solo una vez.
4	Existen varios elementos seleccionados de la lista de	Realiza el else del primer if,

	licencias.	el else del segundo if, y ejecuta el bucle según la cantidad de elementos seleccionados.

Otra de las funcionalidades para la realización de pruebas de caja blanca es Comprobar Licencia del módulo Cliente, este es el encargado de comprobar que la licencia se corresponda con la PC a licenciar a partir de la comparación de los datos de hardware de la PC con los datos que trae la licencia. La figura 11 representa el método ComprobarLicencia, mientras la figura 12 muestra los resultados de las pruebas.

```
//*************** Comprobar Licencia ****************
0  public bool ComprobarLicencia(Licencia lic)
   {
1     string numeroSeriePlacaBase = "";
      string idMicro = "";
      bool devolver = false;
      CapturarDatosHardware(out numeroSeriePlacaBase, out idMicro);
      bool esCompatible = false;
2     if (lic != null && !esCompatible)
      {
3        esCompatible = lic.LicenciaAdecuadaParaDatosHardware
         (numeroSeriePlacaBase, idMicro);
4        if (esCompatible == true)
5           devolver = true;
6     }
      else
      {
7        devolver = false;
8     }
9     return devolver;
   }
10
```

Fig. 11 Método ComprobarLicencia.



Complejidad ciclomática

A=12(aristas)
N=11(nodos)
V(G)=A-N+2
V(G)=12-11+2
V(G)=3

P=2(nodos predicados)
V(G)=P+1
V(G)=2+1
V(G)=3

R=3(regiones)
V(G)=R
V(G)=3

Caminos independientes:
0-1-2-7-8-9-10
0-1-2-3-4-6-8-9-10
0-1-2-3-4-5-6-8-9-10

Fig. 12 Resultados correspondiente al método ComprobarLicencia

Casos de prueba que fuerzan la ejecución de cada camino para el método ComprobarLicencia.

No del camino	Caso de prueba	Respuesta del sistema
1	La licencia está vacía o no es compatible con la PC.	No ejecuta ninguno de los if, retorna falso.
2	La licencia no es vacía y la compatibilidad es falsa.	Realiza el if, comprueba la compatibilidad, retorna falso.
3	La licencia no es vacía y es compatible con la PC.	Realiza el primer if, el segundo y retorna true porque le la licencia es compatible.

La ejecución de los Casos de Prueba de Caja Blanca de las funcionalidades seleccionados para probar, fue satisfactoria en todos los casos. Se logró cubrir todos los caminos y con ellos todas las

aristas del grafo. La complejidad ciclomática es baja lo que permite un mayor entendimiento, facilidad de prueba, facilidad de modificación y de reutilización.

En el capítulo se ejercitaron los requisitos funcionales del sistema, demostrándose que las funciones del software son totalmente operativas. En efecto, fueron derivadas un conjunto de reglas en soporte a las técnicas aplicadas de acuerdo a los métodos establecidos, encaminadas a obtener los casos de pruebas asociados a los casos de usos del sistema. Se realizó el estudio y análisis de la complejidad del sistema lo que permitió arribar a resultados concretos y dar validez a la calidad del sistema.

BENEFICIOS

Con el desarrollo de la aplicación se obtuvieron los siguientes beneficios:

- Asigna licencias de software para el sistema alas PACS.
- Gestiona la información relacionada con el otorgamiento de licencias para el sistema alas PACS.
- Genera reportes de las licencias generadas y renovadas.
- Se pueden obtener los reportes en diversos formatos.
- Propicia mayor seguridad a la hora del despliegue del alas PACS, minimizando así las pérdidas económicas que podría ocasionar al país.

CONCLUSIONES

Con la realización de la presente investigación se lograron favorables resultados que pueden concluirse de la siguiente forma:

- Se analizó la documentación existente donde se logró comprender la necesidad de desarrollar una aplicación de gestión del otorgamiento de licencias para el sistema alas PACS.
- Se analizó el diseño propuesto por el analista lo que permitió la implementación de una aplicación de escritorio constituida por dos módulos.
- Las pruebas realizadas arrojaron resultados satisfactorios demostrando que los requisitos del sistema son completamente funcionales.

Como resultado de la investigación se logró solucionar el problema científico que da origen al objetivo general trazado, al desarrollarse el sistema para la gestión del otorgamiento de licencias del sistema alas PACS.

RECOMENDACIONES

A partir de la investigación efectuada y los conocimientos y experiencias acumuladas durante el trabajo realizado se proponen las siguientes recomendaciones:

- Realizar una variante web del sistema para humanizar el trabajo de los licenciadores.
- Continuar el desarrollo del sistema para que admita generar otros tipos de licencias.

REFERENCIAS BIBLIOGRÁFICAS

1. Jerez Pompa, Danayti Lorenmys. Estrategia de protección y licenciamiento para los softwares del Departamento de Software Médico Imagenológico del CESIM.
2. Averoff Cabrera, Danairy and Enriquez Pérez, María Luisa. Diseño de una herramienta para la gestión del otorgamiento de licencias para el sistema alas PACSViewer.
3. Idem al 2.
4. Ltda, Softart. NetSupport DNA Inventario. *Softart Ltda.* [Online] Octubre 18, 2010 http://ns-la.softart.cl/index.php?option=com_content&view=article&id=86&Itemid=100012.
5. Guía del usuario de Microsoft Software Inventory Analyzer 4.0. *Guía del usuario de Microsoft Software Inventory Analyzer 4.0.* [Online] Octubre 15, 2010. http://download.microsoft.com/download/6/c/9/6c9912cf-9bc9-43ed-b4ca-bf68a383b6d8 /Guia MSIA.pdf.
6. Sciensoft. *Products. Eleckey 2.0.* [Online] Octubre 18, 2010 http://www.sciensoft.com/products/eleckey/.
7. Repetir 6. [Online].
8. Free Download Manager. *Protector de Licencias (Licence Protector) 2.5.* [Online] Octubre 20, 2010 http://www.freedownloadmanager.org/es/downloads/Protector_de_Licencia_3512_p.
9. Repetir 8. [Online].
10. Repetir 2.
11. *Protección por* hardware - *Sentinel* LM. [Online] Octubre 21, 2010 http://www.sitepro-sa.com.ar/software_sentinel_introduccion.htm.
12. Álvarez Sara, Tipos de programación. [Online] http://www.desarrolloweb.com/articulos/2477.php.
13. Capítulo 1 INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS (POO). [Online] http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r51770.PDF.
14. Lenguajes de programación, *Herramientas de programación*. [Online] http://www.lenguajes-de-programacion.com/herramientas-de-programacion.shtml.
15. http://enciclopedia.us.es/index.php/Lenguaje_de_programaci%C3%B3n.
16. La tecnología Java: *Programación* Orientada a Objetos, Máquina http://java.ciberaula.com/articulo/tecnologia_java/.
17. Cuales son las principales caracteristicas del *C++* http://es.answers.yahoo.com/question/index?qid=20070811200816AAA6CvJ.
18. Taiwan excellence 2009. Canal visual basic .net. [Online] http://www.canalvisualbasic.net/.

19. Semanat Aldana, Edmis Deivis and Verdecia Four, Leonor. Sistema de Video Vigilancia. Ciudad de la Habana: UCI, 2009.
20. Conociendo el Lenguaje *C#* http://usuarios.multimania.es/micrologicsoftware/03-KnowCsharp/.
21. grupotressinternacional. Microsoft Visual Estudio 2008. [Online] febrero 2008 http://www.willydev.net/InsiteCreation/v1.0/WillyCrawler/2008.05.01.Articulo.Lo%20nuevo%20en%20Visual%20Studio%202008.pdf.
22. grupotressinternacional. Microsoft Visual Estudio 2008. [Online] febrero 2008 http://www.willydev.net/InsiteCreation/v1.0/WillyCrawler/2008.05.01.Articulo.Lo%20nuevo%20en%20Visual%20Studio%202008.pdf.
23. Repetir 19.
24. Grupo soluciones Innova http://www.rational.com.ar/Herramientas/rup.html.
25. Explorando Extremos en contexto Académico http:// www.coordinv.ciencs.ucv.ve/investigacion/genci/sitios/12/archivos/Explorando_Extremos_en_un_Contexto.
26. Programación Extrema-EcuRed http//:www.ecured.cu/index.php/Programacion_extrema.
27. Sistema Gestor de base de datos SGBD http://www.error500.net/garbagecollector/bases_de_datos/sistema_gestor_de_base_de_dato.html
28. Ibercom. [Online] 1996-2009. https://www.ibercom.com/soporte/index.php?_m...a...996
29. Oracle. [Online] http://www.uaem.mx/posgrado/mcruz/cursos/miic/oracle3.ppt.
30. PostGreSQL vs. MySQL. [Online] http://danielpecos.com/docs/mysql_postgres/x57.html.
31. Repetir 30. [Online].
32. Astra Nuevas Tecnologías de la Información. [Online] http://www.astra.es/noticias/postgresql-8-3-ya-disponible.
33. http://java.ciberaula.com/articulo/tecnologia_java/.
34. http://www.slideshare.net/PauloGuerraT/1-plataforma-net
35. Kernel Error | *Arquitectura 3 Capas* [Online].
36. Calahorrano Narváez, Amy Indira. Herramienta para evaluar la calidad del código fuente generado en C ANSI. Quito, septiembre 2007.Escuela Politécnica Nacional. Escuela de Ingenieria.

BIBLIOGRAFÍA

1. Jerez Pompa, Danayti Lorenmys. Estrategia de protección y licenciamiento para los softwares del Departamento de Software Médico Imagenológico del CESIM.
2. Averoff Cabrera, Danairy and Enriquez Pérez, María Luisa. Diseño de una herramienta para la gestión del otorgamiento de licencias para el sistema alas PACSViewer.
3. Idem al 2.
4. Ltda, Softart. NetSupport DNA Inventario. *Softart Ltda.* [Online] Octubre 18, 2010 http://ns-la.softart.cl/index.php?option=com_content&view=article&id=86&Itemid=100012.
5. Guía del usuario de Microsoft Software Inventory Analyzer 4.0. *Guía del usuario de Microsoft Software Inventory Analyzer 4.0.* [Online] Octubre 15, 2010 http://download.microsoft.com/download/6/c/9/6c9912cf-9bc9-43ed-b4ca-bf68a383b6d8__/Guia___MSIA.pdf.
6. Sciensoft. *Products. Eleckey 2.0.* [Online] Octubre 18, 2010 http://www.sciensoft.com/products/eleckey/.
7. Repetir 6. [Online].
8. Free Download Manager. *Protector de Licencias (Licence Protector) 2.5.* [Online] Octubre 20, 2010 http://www.freedownloadmanager.org/es/downloads/Protector_de_Licencia_3512_p.
9. Repetir 8. [Online].
10. Repetir 2.
11. *Protección por* hardware - *Sentinel* LM. [Online] Octubre 21, 2010 http://www.sitepro-sa.com.ar/software_sentinel_introduccion.htm.
12. Álvarez Sara, Tipos de programación. [Online] http://www.desarrolloweb.com/articulos/2477.php.
13. Capítulo 1 INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS (POO). [Online] http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r51770.PDF.
14. Lenguajes de programación, *Herramientas de programación*. [Online] http://www.lenguajes-de-programacion.com/herramientas-de-programacion.shtml.
15. http://enciclopedia.us.es/index.php/Lenguaje_de_programaci%C3%B3n.
16. La tecnología Java: *Programación* Orientada a Objetos, Máquina http://java.ciberaula.com/articulo/tecnologia_java/.
17. Cuales son las principales caracteristicas del *C++* http://es.answers.yahoo.com/question/index?qid=20070811200816AAA6CvJ.
18. Taiwan excellence 2009. Canal visual basic .net. [Online] http://www.canalvisualbasic.net/.

19. Semanat Aldana, Edmis Deivis and Verdecia Four, Leonor. Sistema de Video Vigilancia. Ciudad de la Habana: UCI, 2009.
20. Conociendo el Lenguaje *C#* http://usuarios.multimania.es/micrologicsoftware/03-KnowCsharp/.
21. grupotressinternacional. Microsoft Visual Estudio 2008. [Online] febrero 2008. http://www.willydev.net/InsiteCreation/v1.0/WillyCrawler/2008.05.01.Articulo.Lo%20nuevo%20en%20Visual%20Studio%202008.pdf.
22. grupotressinternacional. Microsoft Visual Estudio 2008. [Online] febrero 2008. http://www.willydev.net/InsiteCreation/v1.0/WillyCrawler/2008.05.01.Articulo.Lo%20nuevo%20en%20Visual%20Studio%202008.pdf.
23. Repetir 19.
24. Grupo soluciones Innova http://www.rational.com.ar/Herramientas/rup.html.
25. Explorando Extremos en contexto Académico http:// www.coordinv.ciencs.ucv.ve/investigacion/genci/sitios/12/archivos/Explorando_Extremos_en_un_Contexto.
26. Programación Extrema-EcuRed http//:www.ecured.cu/index.php/Programacion_extrema.
27. Sistema Gestor de base de datos SGBD http://www.error500.net/garbagecollector/bases_de_datos/sistema_gestor_de_base_de_dato.html
28. Ibercom. [Online] 1996-2009. https://www.ibercom.com/soporte/index.php?_m...a...996.
29. Oracle. [Online] http://www.uaem.mx/posgrado/mcruz/cursos/miic/oracle3.ppt.
30. PostGreSQL vs. MySQL. [Online] http://danielpecos.com/docs/mysql_postgres/x57.html.
31. Repetir 30. [Online.]
32. Astra Nuevas Tecnologías de la Información. [Online] http://www.astra.es/noticias/postgresql-8-3-ya-disponible.
33. http://java.ciberaula.com/articulo/tecnologia_java/..
34. http://www.slideshare.net/PauloGuerraT/1-plataforma-net
35. Kernel Error | *Arquitectura 3 Capas* [Online].
36. Gómez Labrador, Ramón M. Tipos de liencias de software.1ra versión-septiembre 2005 http://www.informatica.us.es/~ramon/articulos/LicenciasSoftware.pdf.

37. Calahorrano Narváez, Amy Indira. Herramienta para evaluar la calidad del código fuente generado en C ANSI. Quito, septiembre 2007.Escuela Politécnica Nacional. Escuela de Ingenieria.
38. Métricas del software. Teoría general del sistema. [Online] http://elmasterdelaweb.wikispaces.com/file/view/metricas-de-software+%5BModo+de+compatibilidad%5D.pdf.
39. La prueba del software http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/Pruebas.pdf.
40. Lopez Quesada, Juan Antonio. Ingeniería de software. Pruebas del software http://www.google.com/url?sa=t&source=web&cd=6&ved=0CD0QFjAF&url=http%3A%2F%2Fdis.um.es%2F~lopezquesada%2Fdocumentos%2FTema_6.ppt&rct=j&q=pruebas%20de%20particion%20de%20equivalencia%20&ei=hQkBTrO-JcLX0QHA3sTSBg&usg=AFQjCNFGwArbfJsKXxXt-v6Naydl4TvAvg&cad=rja.

ANEXOS

Anexo 1.

Descripción general

Exporta los datos de la PC y los datos de hardware en un fichero en una ubicación seleccionada.

Condiciones de ejecución

Obtener los datos de la PC.

Secciones

Exportar Fichero de solicitud.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1. Exportar fichero de solicitud.	EC 1.1 Exportar fichero de solicitud una vez obtenidos los datos de ubicación de la PC.	Permite generar y exportar un fichero con la solicitud que fue creada.	1. Pestaña Solicitud. 2. Botón Exportar. 3. Aceptar.
	EC1.2 Cancelar la opción de Exportar fichero de solicitud.	No se genera el fichero de solicitud de licencia.	1. Pestaña Solicitud. 2. Botón Exportar. 3. Cancelar.

Escenarios

ID del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EC 1.1	Exportar fichero de solicitud.	Muestra un cuadro de diálogo para exportar la solicitud en la dirección que se desee, crea un archivo en la dirección seleccionada, luego muestra el mensaje “Datos de la PC exportados correctamente”.	Satisfactorio.

EC1.2	Cancelar la opción de Exportar fichero de solicitud.	No genera el fichero de solicitud de licencia y se muestra el mensaje “Se ha cancelado la acción de exportar solicitud de licencia para la PC”.	Satisfactorio.

Anexo 2.

Descripción general

Carga un paquete o un fichero la licencia generado en el Gestor perteneciente a la PC a activar.

Condiciones de ejecución

Crear el fichero o el paquete de licencias en el Gestor.

Secciones

Importar Licencia

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1. Importar Licencia.	EC 1.1 Importar Licencias individual	Importa una licencia individual y la muestra.	1. Pestaña Licencia. 2. Seleccionar el radiobutton Individual. 3. Opción Importar.
	EC 1.2 Importar Licencias en paquete.	Importa un paquete de licencias y las muestra una vez importadas.	1. Pestaña Licencia. 2. Seleccionar el radiobutton Paquete. 3. Opción Importar.
	EC 1.3 Importar Licencias sin seleccionar la forma de importar.	El usuario no elige la forma en que desea importar la licencia si es individual o un paquete.	1. Pestaña Licencia 2. Opción Importar.
	EC1.4 Cancela la opción de Importar Licencia.	Se cancela la opción de importar licencia.	1. Menú Archivo. 2. Cancelar.

Escenarios

ID del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EC 1.1	Importar Licencias individual.	Filtra los archivos de tipo ".lic", muestra el mensaje "Licencia importada correctamente", muestra los datos de la licencia una vez que haya sido importada y activa la opción de Activar.	Satisfactorio.
EC 1.2	Importar Licencias en paquete.	Filtra los archivos de tipo ¨.pac¨, muestra el mensaje ¨Licencia importada correctamente¨, muestra los datos de la licencia una vez que haya sido importada y activa la opción Activar.	Satisfactorio.
EC 1.3	Importar Licencias sin seleccionar la forma de importar.	Muestra el mensaje "Debe de especificar cómo quiere importar la licencia".	Satisfactorio.
EC1.4.	Cancela la opción de Importar Licencia.	No importa el paquete o el fichero de licencia, ambos generados en el Gestor.	Satisfactorio.

Anexo 3.

Descripción general

Guarda el fichero de licencia en una dirección específica donde se activa el sistema alas PACS.

Condiciones de ejecución

Cargar el paquete de licencias o la licencia individual perteneciente a la PC a activar.

Secciones

Activar Licencia.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1. Activar Licencia.	EC 1.1 Activar licencia.	Crea fichero de licencia y lo guarda por defecto en una dirección específica.	1. Pestaña Licencia. 2. Opción Activar.

	EC1.3Activar licencia luego de que haya sido activado una vez.	El sistema fue activado anteriormente y se procede a activarlo nuevamente con la misma licencia generada.	1. Pestaña Licencia. 2. Opción Activar.
	EC1.3 Activa licencia si no existe ninguna licencia para la PC que se quiere activar.	Activa licencia sin que exista alguna para la PC que se quiere activar.	1. Pestaña Licencia. 2. Opción Activar.

Escenarios

ID del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EC 1.1	Activar Licencia.	Genera un fichero en la dirección C:\WINDOWS\system32, muestra el mensaje "El sistema ha sido activado satisfactoriamente" y activa la opción Renovar.	Satisfactorio.
EC 1.2	Activar Licencia luego de que haya sido activado una vez.	Muestra el mensaje "La licencia ya existe".	Satisfactorio.
EC 1.3	Activa licencia si no existe ninguna licencia para la PC que se quiere activar.	Muestra el mensaje "No existe licencia para esta PC".	Satisfactorio

Anexo 4.

Descripción general

Permite cambiar los datos de ubicación de la PC en caso de que se hayan introducido incorrectamente.

Condiciones de ejecución

Crear una solicitud y/o importa licencias.

Secciones

Editar solicitud

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1. Editar solicitud de licencia.	EC 1.1 Editar solicitud de licencia.	Permite cambiar los datos de ubicación de la PC si fueron introducidos incorrectamente.	1. Pestaña Solicitud. 2. Opción Editar.

Escenarios

ID del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EC 1.1	Editar solicitud de licencia.	Activa el formulario para obtener los datos de ubicación de la PC una vez que se hayan introducido incorrectamente, muestra el mensaje “Datos de la PC registrados correctamente”.	Satisfactorio.

Anexo 5.

Descripción general

Importa la solicitud de licencia y muestra la lista de solicitudes.

Condiciones de ejecución

Crear una solicitud de licencia en el Cliente.

Secciones

Importar fichero de solicitud de licencia.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central

SC1. Importar fichero de solicitud de licencia.	EC 1.1 Importar el fichero de solicitud de licencia.	Carga todos los ficheros de solicitudes de licencias creados en el Cliente y muestra la lista de solicitudes.	1. Pestaña Solicitudes. 2. Importar ficheros de solicitud.

Escenarios

ID del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EC 1.1	Importar el fichero de solicitud de licencia.	El sistema importa las solicitudes de licencias y muestra la lista de solicitudes, envía el mensaje "El fichero ha sido importado satisfactoriamente".	Satisfactorio.

Anexo 6.

Descripción general

Convierte las solicitudes creadas en licencias.

Condiciones de ejecución

Importar las solicitudes de licencias.

Secciones

Generar Licencia

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1. Generar Licencia.	EC 1.1 Generar licencia	Una vez seleccionadas las solicitudes se convierten en licencias.	1. Pestaña Solicitudes. 2. Opción Generar licencia.
	EC 1.2 Generar licencia sin seleccionar las solicitudes que se les realizarán las licencias.	Se debe seleccionar las solicitudes para luego generar las licencias correspondientes a las	1. Pestaña Solicitudes. 2. Opción Generar licencia.

		solicitudes seleccionadas.	

Escenarios

ID del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EC 1.1	Generar licencia.	Las solicitudes seleccionadas se eliminan de la lista de solicitudes y se convierten en licencias mostrándose en la lista de licencias, el sistema muestra el mensaje "Licencias generadas satisfactoriamente", se activan las opciones de Generar licencia y Eliminar solicitudes.	Satisfactorio.
EC 1.2	Generar licencia sin seleccionar las solicitudes a las cuales se les realizarán las licencias.	Si no se han seleccionado las solicitudes a las cuales se realizaran las licencias se muestra el mensaje "Debe de especificar la solicitud a la cual desea generar la licencia".	Satisfactorio.

Anexo 7.

Descripción general

Exporta los datos de la PC y los datos de hardware en un fichero en una ubicación seleccionada.

Condiciones de ejecución

Obtener los datos de la PC.

Crear la solicitud.

Secciones

Exportar Licencia.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central

SC1. Exportar licencia.	EC 1.1 Exportar paquete de licencias seleccionando las licencias a exportar.	Crea un paquete de licencia con las licencias seleccionadas y lo guarda donde desee el usuario.	1. Licencias. 2. Exportar paquete.
	EC 1.2 Exportar licencias individuales seleccionando las licencias a exportar.	Crea un fichero para cada una de las licencias seleccionadas y lo guarda donde desee el usuario.	1. Licencias. 2. Exportar individual.
	EC1.3 Exportar licencia sin seleccionar las licencias a exportar.	Muestra el mensaje ”Debe de seleccionar las licencias a exportar”.	1. Licencias. 2. Exportar paquete.

Escenarios

ID del escenario	**Escenario**	**Respuesta del sistema**	**Resultado de la prueba**
EC 1.1	Exportar paquete de licencias seleccionando las licencias a exportar.	Muestra un cuadro de diálogo para exportar las licencias en la dirección que se desee, crea un archivo en la dirección seleccionada, luego muestra el mensaje “Datos exportados correctamente”.	Satisfactorio.
EC 1.2	Exportar licencias individuales seleccionando las licencias a exportar.	Muestra un cuadro de diálogo para exportar las licencias en la dirección que se desee, crea un archivo en la dirección seleccionada, luego muestra el mensaje “Datos exportados correctamente”.	Satisfactorio.
EC 1.3	Exportar licencia sin seleccionar las licencias a exportar.	Muestra el mensaje ”Debe de seleccionar las licencias a exportar”.	Satisfactorio

Anexo 8.

Descripción general

Elimina la solicitud seleccionada.

Condiciones de ejecución

Importar las solicitudes.

Secciones

Eliminar Solicitud.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1. Eliminar Solicitud.	EC1.1 Eliminar solicitud seleccionando la solicitud que se desea eliminar.	Elimina las solicitudes seleccionadas.	1. Solicitudes. 2. Seleccionar las solicitudes a eliminar. 3. Eliminar Solicitud.
	EC1.2 Eliminar solicitud sin seleccionar la solicitud a eliminar.	No se selecciona la solicitud a eliminar.	1. Solicitudes. 2. Eliminar Solicitud.
	EC1.3 Eliminar solicitudes si no existen solicitudes en ese momento.	Se procede a eliminar solicitudes estando vacía la lista de solicitudes.	1. Solicitudes. 2. Eliminar Solicitud.

Escenarios

ID del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EC 1.1	Eliminar solicitud seleccionando la solicitud que se desea eliminar.	Elimina de la lista de solicitudes la solicitud seleccionada y muestra el mensaje “La solicitud ha sido eliminada satisfactoriamente”.	Satisfactorio.
EC 1.2	Eliminar solicitud sin seleccionar la solicitud a eliminar.	El sistema muestra el mensaje “Debe de seleccionar la solicitud a eliminar”.	Satisfactorio.

EC1.3	Eliminar solicitudes si no existen solicitudes en ese momento.	El sistema muestra el mensaje “No existen solicitudes de licencias en base de datos”.	Satisfactorio.

Anexo 9.

Descripción general

Genera un reporte con las licencias generadas y las licencias renovadas, permite exportar los reportes en formato PDF o Excel.

Condiciones de ejecución

Secciones

Generar Reporte

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1. Generar Reporte.	EC 1.1 Generar reporte.	Genera un reporte con la información de las licencias generadas y las licencias renovadas.	1. Reportes.
SC2. Exportar a PDF	EC2.1 Exportar a PDF	Genera un PDF con el reporte obtenido.	1. Reportes. 2. Exportar a PDF.
SC3. Exportar a Excel.	EC3.1 Exportar a Excel.	Genera un Excel con el reporte obtenido.	1. Reportes. 2. Exportar a Excel.

Escenarios

ID del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EC 1.1	Generar reporte.	Muestra una pantalla con las licencias generadas y renovadas.	Satisfactorio

EC 1.2	Exportar a PDF.	Crea un PDF con todos los datos de los reportes de licencias generadas y renovadas.	Satisfactorio
EC1.3	Exportar a Excel.	Crea un Excel con todos los datos de los reportes de licencias generadas y renovadas.	Satisfactorio

Anexo 10.

Descripción general

Muestra una lista con las solicitudes de licencias que se correspondan con el criterio de búsqueda especificado.

Condiciones de ejecución

Importar las solicitudes generadas.

Secciones

Buscar solicitudes.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1.1 Búsqueda simple.	EC 1.1 Buscar las solicitudes introduciendo el número de serie de la PC.	Se introduce el número de serie de la PC y se muestran las solicitudes que se corresponden con el criterio de búsqueda.	1. Introducir el número de serie de la PC. 2. Opción Buscar.
	EC 1.2 Buscar las solicitudes sin introducir el número de serie de la PC.	Muestra todas las solicitudes existentes en ese momento.	1. Opción Buscar.
	EC 1.3 Cancelar la opción de buscar solicitudes.	Muestra todas las solicitudes existentes en ese momento.	1. Introducir el número de serie de la PC. 2. Buscar.

			3. Cancelar.
SC2. Búsqueda avanzada.	EC2.1 Buscar solicitudes introduciendo los criterios de búsqueda deseados.	Muestra las solicitudes que se corresponden con el criterio de búsqueda especificado.	1. Búsqueda avanzada. 2. Introducir los criterios de búsqueda. 3. Buscar.
	EC2.2 Buscar solicitudes sin introducir ningún criterio de búsqueda.	Muestra todas las solicitudes existentes en ese momento.	1. Búsqueda avanzada. 2. Buscar.
	EC 2.3Cancelar la opción de buscar solicitudes.	Muestra las solicitudes existentes en ese momento.	1. Búsqueda avanzada. 2. Buscar. 3. Cancelar.

Escenarios

ID del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EC 1.1	Buscar las solicitudes introduciendo el número de serie de la PC.	Muestra una lista de solicitudes que se corresponden con el número de serie de la PC.	Satisfactorio.
EC 1.2	Buscar las solicitudes sin introducir el número de serie de la PC.	Muestra todas las solicitudes existentes en ese momento.	Satisfactorio.

EC1.3	Cancelar la opción de buscar solicitudes.	Muestra todas las solicitudes existentes en ese momento luego de realizada la búsqueda.	Satisfactorio.
EC2.1	Buscar solicitudes introduciendo los criterios de búsqueda deseados.	Muestra las solicitudes que se corresponden con los criterios de búsqueda especificados.	Satisfactorio.
EC2.2	Buscar solicitudes sin introducir ningún criterio de búsqueda.	Muestra todas las solicitudes existentes en ese momento.	Satisfactorio.
EC2.3	Cancelar la opción de buscar solicitudes.	Muestra todas las solicitudes existentes en ese momento luego de realizada alguna búsqueda.	Satisfactorio.