Universidad de las Ciencias Informáticas

Facultad 7

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

“Implementación del módulo de Mantenimiento para el Sistema de Gestión para Ingeniería Clínica y Electromedicina”

Autor:

Nelson Martinez Romero

Tutores:

Ing. Ranniel Rivero Sevila

Ing. Idayana Bastarreche Calistre

La Habana, junio de 2013

“Año 55 de la Revolución”.

Datos de Contacto

Tutores:

- **Ing. Ranniel Rivero Sevila** (rrivero@uci.cu):

Graduado de Ingeniero en Ciencias Informáticas del curso 2007-2008. Líder de desarrollo del proyecto SACCEM durante un año. Ha impartido la asignatura de Práctica Profesional e Inteligencia Artificial, Historia de la Informática y Segundo Perfil. En el curso 2008-2009 formó parte de un tribunal en seis tesis de grado y fue tutor de tres tesis y en el curso 2009-2010 de una. En el curso 2009-2010 fue tutor de dos tesis de grado, las cuales alcanzaron la máxima calificación de cinco puntos, de la misma forma en el curso 2010-2011 tutoró un trabajo de diploma con el mismo resultado. También se ha desempeñado como oponente de trabajos de diplomas en dos ocasiones. Ha participado en numerosos eventos tanto nacionales como internacionales en los que ha tenido importantes publicaciones. Actualmente, se desempeña como especialista del Centro de Informática Médica, es profesor adjunto a la Facultad 7, presenta el rol de jefe del proyecto Sistema de Gestión para Ingeniería Clínica y Electromedicina.

- **Ing. Idayana Bastarreche Calistre** (ibastarreche@uci.cu):

Graduado de Ingeniero en Ciencias Informáticas, en el año 2011, en la Universidad de las Ciencias Informáticas. Actualmente se desempeña como analista en el proyecto productivo SIGICEM.

Resumen

El Centro de Ingeniería Clínica y Electromedicina tiene como objeto social la gestión de la información referente a los equipos médicos existentes en Cuba y los recursos asociados a estos. Para la informatización de sus procesos cuenta con el Sistema de Gestión para Ingeniería Clínica y Electromedicina; el cual no tiene incorporado funcionalidades relacionadas con el proceso de mantenimiento, realizándose de forma manual o mediante la utilización de medios obsoletos.

El presente trabajo se propone implementar un módulo de Mantenimiento para el Sistema de Gestión para Ingeniería Clínica y Electromedicina. Para el desarrollo del sistema fue necesario el estudio, profundización y utilización de tecnologías y herramientas entre las que se encuentran fundamentalmente: como metodología el Proceso Unificado de Desarrollo, apoyado en el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) 2.1 y sustentado por el Visual Paradigm 6.4 y DB Designer Fork 4.0 como herramientas CASE. Además, de la utilización de los *frameworks* Symfony 1.4 y Ext JS 2.2, soportado sobre una plataforma LAMP, integrado por las tecnologías Ubuntu 10.4, Apache 2.2, MySQL 5.3 y PHP 5.3.

La puesta en práctica de este sistema en el Centro de Ingeniería Clínica y Electromedicina permitirá una correcta planificación de los mantenimientos, y en consecuencia, prolongar la vida útil de los equipos médicos.

Palabras clave: equipos médicos, mantenimiento, planificación.

Índice

Introducción

Los avances en el sector de la informática y las comunicaciones han sacudido al mundo durante las últimas décadas, abarcando un gran conjunto de funciones, que van desde las más simples cuestiones domésticas hasta los cálculos científicos más complejos. Estos adelantos se aplican a numerosas y variadas áreas del conocimiento y de la actividad humana. La rapidez con que cambian las tecnologías, establece una dinámica diferente en las sociedades, imponiendo nuevos retos a alcanzar, para no quedar rezagados en el camino dominante de la informatización. (1)

A partir de las tendencias actuales, la expansión de las Tecnologías de la Información y la Comunicación (TIC) en todos los ámbitos y estratos de la sociedad se ha producido a gran velocidad, propiciando un impulso hacia lo novedoso. El mundo está viviendo una época de innovaciones profundas que implican cambios en el desarrollo de la ciencia y la tecnología. En tal sentido, puede señalarse que las TIC fomentan el avance del desarrollo tecnológico.

En la actualidad, el aporte de las TIC resulta fundamental en la medicina. Debe existir una interrelación entre medicina y tecnología, pues el manejo de los equipos médicos de alta complejidad es parte de los avances científico-técnicos, que se han producido a través del tiempo. El desarrollo tecnológico ha propiciado un cambio asombroso en la medicina; su evolución ha permitido conocer infinidad de procesos que explican el por qué de muchas enfermedades que afectan el cuerpo humano.

Cuba, a pesar de no ser un país desarrollado y de estar sometido a un injusto bloqueo, no está exento de estos avances, ya que ha dedicado una parte importante de sus recursos a la informatización de diferentes esferas de la sociedad, siendo una de ellas el sector de la salud. Este proceso es controlado por el Ministerio de Salud Pública (MINSAP), encargado del desarrollo de las ciencias médicas y la industria farmacéutica. A raíz de esto, comienza la creación de nuevas instituciones de salud con su correspondiente equipamiento, lo cual hace necesaria la creación de talleres para ofrecer mantenimiento y atención especializada al mismo.

La historia del mantenimiento acompaña el desarrollo Técnico-Industrial de la humanidad. Al final del siglo XIX, con la mecanización de las industrias, surgió la necesidad de las primeras reparaciones. En términos generales el mantenimiento consiste en el conjunto de acciones oportunas, continúas y permanentes dirigidas a prever y asegurar el funcionamiento normal, la eficiencia y la buena apariencia. En los equipos

médicos se establecen procedimientos con los cuales examinar periódicamente las condiciones materiales de los mismos, a fin de asegurar la eliminación o minimización y control de riesgos, así como su conservación en condiciones óptimas de funcionamiento, reduciendo las posibles averías y fallos provocados por el mal estado. (2)

El Centro de Ingeniería Clínica y Electromedicina (CICEM) es la institución que se encarga de brindar servicios técnicos y gestionar la tecnología médica existente en el Sistema Nacional de Salud (SNS). Debido al desarrollo del centro y ante la misión de brindar soluciones de alta calidad, con el objetivo de mejorar la atención médica, se tomó la decisión de crear un sistema capaz de cubrir todo el proceso de gestión tecnológica llevada a cabo por los especialistas de dicho centro, garantizando la fiabilidad y actualización de la información referente al equipamiento médico instalado en el país. El Sistema de Gestión para la Ingeniería Clínica y Electromedicina (SIGICEM) se encuentra en desarrollo por parte de un equipo de trabajo del Departamento Sistemas Especializados en Salud (SES), el cual conjuntamente con otros departamentos integran el Centro de Informática Médica (CESIM), subordinándose este, a la facultad 7 de la Universidad de las Ciencias Informáticas (UCI).

Es de vital importancia mantener en condiciones óptimas y adecuadas de operación y funcionamiento los equipos médicos, de manera que estos cumplan con su cometido de ser parte del proceso de atención a la salud. Aprovechando los recursos invertidos de una forma más eficiente y racional, para garantizar su utilidad en la mejoría de la salud y de la calidad de vida de los pacientes. (3)

Actualmente en el CICEM el proceso de gestión de los mantenimientos se realiza de forma manual, provocando desorganización en la asignación de especialistas que llevan a cabo las reparaciones porque no se tienen en cuenta las afectaciones de dichos especialistas en ese mes y pueden planificárseles varios mantenimientos el mismo día, falta de elementos en los procedimientos y pérdida en algunos casos de registros de planificación.

El módulo de Mantenimiento solo se realizó hasta la fase de Análisis y Diseño, careciendo el sistema de un mecanismo que permita:

- Planificar los mantenimientos mensuales identificando cada uno de sus elementos.
- Gestionar las herramientas que intervienen en el proceso de mantenimiento, lo cual provoca pérdida de información durante el proceso.

- Gestionar las guías técnicas, por lo que en ocasiones no se tiene evidencia de los pasos a seguir para la reparación de los equipos médicos.
- Crear procedimientos para organizar todos los elementos relacionados con: piezas, herramientas y guías técnicas utilizadas en el mantenimiento de los equipos médicos.

Por lo anteriormente expuesto se plantea el siguiente **problema a resolver:** ¿Cómo contribuir a la gestión de la información en el proceso de mantenimiento de los equipos médicos llevado a cabo por el Centro de Ingeniería Clínica y Electromedicina, que permita elevar los niveles de disponibilidad técnica de la tecnología médica?

Centrándose en el **objeto de estudio:** Proceso de gestión de información de equipos médicos en el Centro de Ingeniería Clínica y Electromedicina; enmarcado en el **campo de acción:** Gestión de la información del mantenimiento de equipos médicos en el Sistema de Gestión para la Ingeniería Clínica y Electromedicina.

Para dar solución al problema planteado se trazó como **objetivo general:** Implementar el módulo de Mantenimiento del Sistema de Gestión para Ingeniería Clínica y Electromedicina.

Para dar cumplimiento al objetivo planteado para esta investigación se han definido las siguientes **tareas de investigación:**

- Establecimiento de los fundamentos teórico-metodológicos para el desarrollo de sistemas informáticos de gestión de información, que sirvan de guía en la investigación en curso.

- Análisis de la gestión de la información del mantenimiento de los equipos médicos en el Centro de Ingeniería Clínica y Electromedicina, para una mejor comprensión del proceso.

- Diseño de los artefactos correspondientes a la fase de implementación para la obtención de la documentación necesaria.

- Implementación de las funcionalidades asociadas al módulo de Mantenimiento para el Sistema de Gestión para Ingeniería Clínica y Electromedicina, siguiendo lo establecido en la Especificación de Requisitos de Software.

Una vez finalizada la investigación, la solución desarrollada traerá consigo los siguientes resultados:

- Contribuirá a elevar los niveles de disponibilidad técnica de la tecnología médica debido a una correcta planificación de su mantenimiento.
- Logrará una mejor organización y asignación de los especialistas que realizan las labores de mantenimiento a los equipos médicos.
- Facilitará a los especialistas la realización de los mantenimientos proporcionándoles una guía técnica con los pasos que deben seguir.
- Prevención de fallas en los equipos, con los que se evita paro y gastos imprevistos al país.

Los principales **métodos y técnicas de investigación** empleadas en función de dar solución a estas tareas fueron:

Métodos teóricos:

- ✓ *Análisis Histórico-Lógico:*

Su objetivo en una investigación es constatar teóricamente cómo ha evolucionado un determinado fenómeno en un período de tiempo, en toda su trayectoria o en un fragmento temporal de la lógica de su desarrollo (4). Se utiliza durante todo el desarrollo de la investigación, incluye el estudio de las diferentes técnicas de programación existentes, para conocer su evolución y desarrollo.

- ✓ *Analítico-Sintético*:

El análisis es una operación intelectual que posibilita descomponer mentalmente un todo complejo en sus partes y cualidades. El análisis permite la división mental del todo en sus múltiples relaciones y componentes. La síntesis es la operación inversa, que establece mentalmente la unión entre las partes, previamente analizadas y posibilita descubrir relaciones y características generales entre los elementos de la realidad (4). Este método se utilizó para lograr extraer los elementos esenciales de la bibliografía consultada durante la descripción de los conceptos asociados al dominio del problema.

- ✓ *Modelación:*

Se trata de explicar la realidad con la creación de diagramas; los cuales pueden ser presentados en sustitución de la realidad. Mediante su utilización se elaborarán diferentes tipos de diagramas que

brindarán información clara sobre el tema de estudio, mediante el descubrimiento de nuevas relaciones y cualidades del objeto de estudio.

Métodos empíricos:

- ✓ *Entrevista:*

  La entrevista es no estructurada porque no se lleva un cuestionario rígido y puede variar de una persona a otra, fue aplicada a los especialistas del proyecto SIGICEM durante la etapa de familiarización con el flujo de los procesos que allí se desarrollan y que serán informatizados posteriormente, proporcionando los datos necesarios para las siguientes etapas de la investigación.

El presente documento está estructurado en tres capítulos:

Capítulo 1 "Fundamentación teórica":

Se explican y justifican, los principales conceptos que se abordan, e incluye una explicación de las técnicas, tecnologías, metodologías, herramientas y software empleados en la investigación para darle solución al problema.

Capítulo 2 "Descripción de la arquitectura":

Se describe la arquitectura, definiéndose además los requerimientos no funcionales, el análisis de posibles implementaciones o componentes a ser rehusados así como la estrategia de codificación.

Capítulo 3 "Descripción y análisis de la solución propuesta":

Se centra en el análisis del diseño propuesto por el analista, descripción de las tablas de la base de datos, modelo de datos, se valoran las técnicas de validación y se expone una descripción del desarrollo de la solución inicialmente propuesta mediante la representación del Modelo de Implementación.

Capítulo 1: *Fundamentación Teórica de la Investigación*

La investigación científica es un proceso sistemático, organizado y objetivo, destinado a dar respuesta a una interrogante. Toda investigación implica un conjunto de pasos, o etapas secuenciadas, enlazadas de manera lógica, unas con otras. En la presente investigación ya se ha definido la interrogante a la que se debe dar respuesta, por lo que el siguiente paso lógico sería plantear los fundamentos teóricos que sostienen y darán continuidad al proceso investigativo.

En este capítulo se mostrará una panorámica sobre las distintas técnicas de programación existentes. Se abordarán conceptos que facilitarán la comprensión del presente trabajo y las características de las herramientas, tecnologías y metodología, previamente definidas en el proyecto SIGICEM, utilizadas en el diseño e implementación del módulo de Mantenimiento.

1.1 Conceptos básicos asociados al problema

Mantenimiento

En términos generales, por mantenimiento, se designa al conjunto de acciones que tienen como objetivo mantener un artículo o restaurarlo a un estado, en el cual, el mismo pueda cumplir el propósito para el que fue construido (5). Durante la investigación se hará referencia al mantenimiento de equipos médicos.

Guía Técnica

Una guía es algo que tutela, rige u orienta. Es el documento que incluye los principios o procedimientos para encauzar el listado con informaciones que se refieren a un asunto específico (6). En este caso, una guía técnica, incluye los pasos que se deben seguir para realizar el mantenimiento de los equipos médicos.

Procedimiento

Un procedimiento consiste en seguir ciertos pasos predefinidos para desarrollar una labor de manera eficaz. Su objetivo debería ser único y de fácil identificación, aunque es posible que existan diversos procedimientos que persigan el mismo fin, cada uno con estructuras y etapas diferentes, y que ofrezcan más o menos eficiencia (7). Los procedimientos relacionados con el proceso de mantenimiento incluyen los aseguramientos, las herramientas y las guías técnicas que se necesitan para ejecutarlo.

Gestión de la información

Gestión de la información es el proceso que incluye operaciones como extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida por una organización a través de diferentes fuentes y que gestiona el acceso y los derechos de los usuarios sobre la misma, y se establece como un recurso básico para cualquier organización. (8)

La gestión de la información se puede definir como el conjunto de actividades realizadas con el fin de controlar, almacenar y, posteriormente, recuperar adecuadamente la información producida, recibida o retenida por cualquier organización en el desarrollo de sus actividades y tiene como objetivos garantizar la integridad, disponibilidad y confidencialidad de la información. (9)

1.2 Técnica de Programación

El diseño de un programa que se realiza sin seguir una metodología puede funcionar, pero se debe tener en cuenta que con el tiempo se convertirá en un conjunto de instrucciones, es decir, que las consecuencias de no utilizar un método determinado llevarán a cometer errores que pueden costar el buen funcionamiento del mismo. La creación de programas debe tener la flexibilidad suficiente para ser modificables en el momento en que se requiera. Estos deben ser claros, simples, con el fin de poder ser leídos e interpretados de forma fácil. (10)

Una técnica de programación es una metodología que debe 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 de la misma, implicando una disminución de costos, independencia del programador y seguridad. Los tipos o técnicas de programación son bastante variados, aunque puede que muchos de los lectores sólo conozcan una metodología para realizar programas.

Programación orientada a objetos

La programación orientada a objetos (POO) promete mejoras de amplio alcance en la forma de diseño, desarrollo y mantenimiento del software ofreciendo una solución a largo plazo a los problemas y preocupaciones que han existido desde el comienzo en el desarrollo de software: la falta de portabilidad del código y reusabilidad, código que es difícil de modificar, ciclos de desarrollo largos y técnicas de codificación no intuitivas. (11)

Un lenguaje orientado a objetos tiene tres características básicas: debe estar basado en objetos, basado en clases y ser capaz de tener herencia de clases. Muchos lenguajes cumplen uno o dos de estos puntos. La barrera más difícil de sortear es usualmente la herencia. (12)

Un objeto puede considerarse como una especie de cápsula dividida en tres partes: (12)

a. Relaciones

Las relaciones permiten que el objeto se inserte en la organización y estas están formadas esencialmente por punteros a otros objetos.

b. Propiedades

Las propiedades distinguen un objeto determinado de los restantes que forman parte de la misma organización y tiene valores que dependen de la propiedad de que se trate. Las propiedades de un objeto pueden ser heredadas a sus descendientes en la organización.

c. Métodos

Los métodos son las operaciones que pueden realizarse sobre el objeto, que normalmente estarán incorporados en forma de programas (código) que el objeto es capaz de ejecutar y que también pone a disposición de sus descendientes a través de la herencia.

Los conceptos generales más utilizados en el modelo orientado a objetos, son los siguientes: abstracción, encapsulación, modularidad, herencia y polimorfismo**.**

Abstracción: es una descripción o especificación simplificada de un sistema que hace énfasis en algunos detalles significativos y suprime los irrelevantes. (13)

Modularidad: la modularidad consiste en dividir un programa en partes llamadas módulos, los cuales pueden trabajarse por separado. (14)

Encapsulación: típicamente, la información acerca de un objeto está encapsulada por su comportamiento. Esto significa que un objeto mantiene datos acerca de cosas del mundo real a las que representa en su sentido verdadero. (15)

Herencia: la herencia define relaciones entre clases, donde una clase comparte la estructura o comportamiento definidos en una o más clases. (16)

Polimorfismo: otro de los mecanismos aportados por la POO es el de polimorfismo, el cual es la capacidad de tener métodos con el mismo nombre pero que su implementación sea diferente. (14)

Programación estructurada

La programación estructurada es un estilo con el cual se busca que el programador elabore programas sencillos y fáciles de entender. Para ello, la programación estructurada hace uso de tres estructuras básicas de control.
Éstas son:

a. **Secuencia: sucesión simple de dos o más operaciones**

Indica que las instrucciones de un programa se ejecutan una después de la otra, en el mismo orden en el cual aparecen en el programa. Se representa gráficamente como una caja después de otra, ambas con una sola entrada y una única salida.

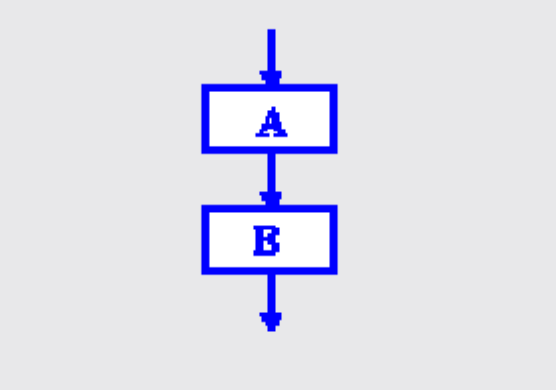

Figura 1. Estructura Secuencial

Las cajas A y B pueden ser definidas para ejecutar desde una simple instrucción hasta un módulo o programa completo, siempre y cuando éstos también sean programas apropiados.

b. **Selección: bifurcación condicional de una o más operaciones**

También conocida como la estructura SI-VERDADERO-FALSO, plantea la selección entre dos alternativas con base en el resultado de la evaluación de una condición; equivale a la instrucción IF de todos los lenguajes de programación y se representa gráficamente de la siguiente manera:

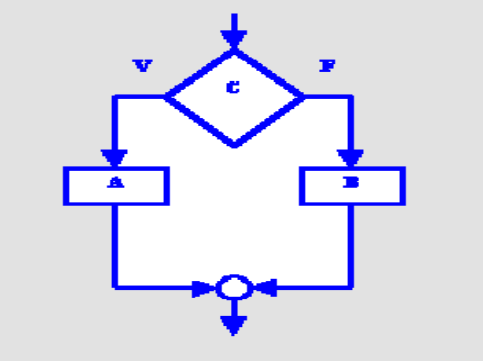

Figura 2. Estructura Selectiva

En el diagrama de flujo anterior, C es una condición que se evalúa; A es la acción que se ejecuta cuando la evaluación de esta condición resulta verdadera y B es la acción ejecutada cuando el resultado de la evaluación indica falso. La estructura también tiene una sola entrada y una sola salida; y las funciones A y B también pueden ser cualquier estructura básica o conjunto de estructuras.

c. Iteración: repetición de una operación mientras se cumple una condición

También llamada la estructura HACER-MIENTRAS-QUE, corresponde a la ejecución repetida de una instrucción mientras que se cumple una determinada condición. El diagrama de flujo para esta estructura es el siguiente:

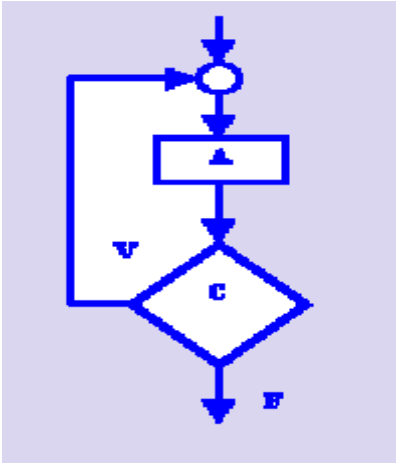

Figura 3. Estructura Repetitiva (Iterativa)

Aquí el bloque A se ejecuta repetidamente mientras que la condición C se cumpla o sea cierta. También tiene una sola entrada y una sola salida; igualmente A puede ser cualquier estructura básica o conjunto de estructuras.

La programación Estructurada está basada en el Teorema de la Estructura, el cual establece que cualquier programa propio (un programa con una entrada y una salida exclusivamente) es equivalente a un programa que contiene solamente las estructuras lógicas mencionadas anteriormente. (17)

Con la programación estructurada, elaborar programas de computadora sigue siendo una labor que demanda esfuerzo, creatividad, habilidad y cuidado. Sin embargo, con este nuevo estilo se obtienen las siguientes ventajas:

1. Los programas son más fáciles de entender. Un programa estructurado puede ser leído en secuencia, de arriba hacia abajo, sin necesidad de estar saltando de un sitio a otro en la lógica, lo cual es típico de otros estilos de programación.
2. Se logra una reducción del esfuerzo en las pruebas. El seguimiento de las fallas o depuración se facilita debido a la lógica más visible, de tal forma que los errores se pueden detectar y corregir más fácilmente.
3. Se crean programas más sencillos y más rápidos.

Programación modular

La programación modular es uno de los métodos de diseño más flexible y potentes para mejorar la productividad de un programa. Se presenta históricamente como una evolución de la programación estructurada para solucionar problemas de programación más grandes y complejos de lo que ésta puede resolver.

En la programación modular el programa se divide en módulos (partes independientes), cada una de las cuales ejecuta una única actividad o tarea y se codifica independientemente de otros módulos. Cada uno de estos módulos se analiza, codifica y se pone a punto por separado. Cada programa contiene un módulo llamado programa principal, que controla todo lo que sucede; se transfiere el control a submódulos (posteriormente se denominarán subprogramas), de modo que ellos pueden ejecutar sus

funciones; sin embargo, cada submódulo devuelve el control al módulo principal cuando se haya completado su tarea. (18)

Los módulos son independientes en el sentido en que ningún módulo puede tener acceso directo a cualquier otro módulo, excepto el módulo al que llama y a sus propios submódulos. Sin embargo, los resultados producidos por un módulo pueden ser utilizados por cualquier otro módulo cuando se transfiera a ellos el control. Dado que los módulos son independientes, diferentes programadores pueden trabajar simultáneamente en diferentes partes del mismo programa. Esto reducirá el tiempo de diseño del algoritmo y posterior codificación del programa. Además, un módulo se puede modificar radicalmente sin afectar a los otros módulos, incluso sin alterar su función principal. La descomposición de un programa en módulos independientes más simples se conoce también como el método «divide y vencerás».

Se diseña cada módulo con independencia de los demás, y siguiendo un método ascendente o descendente se llegará hasta la descomposición final del problema en módulos de forma jerárquica.

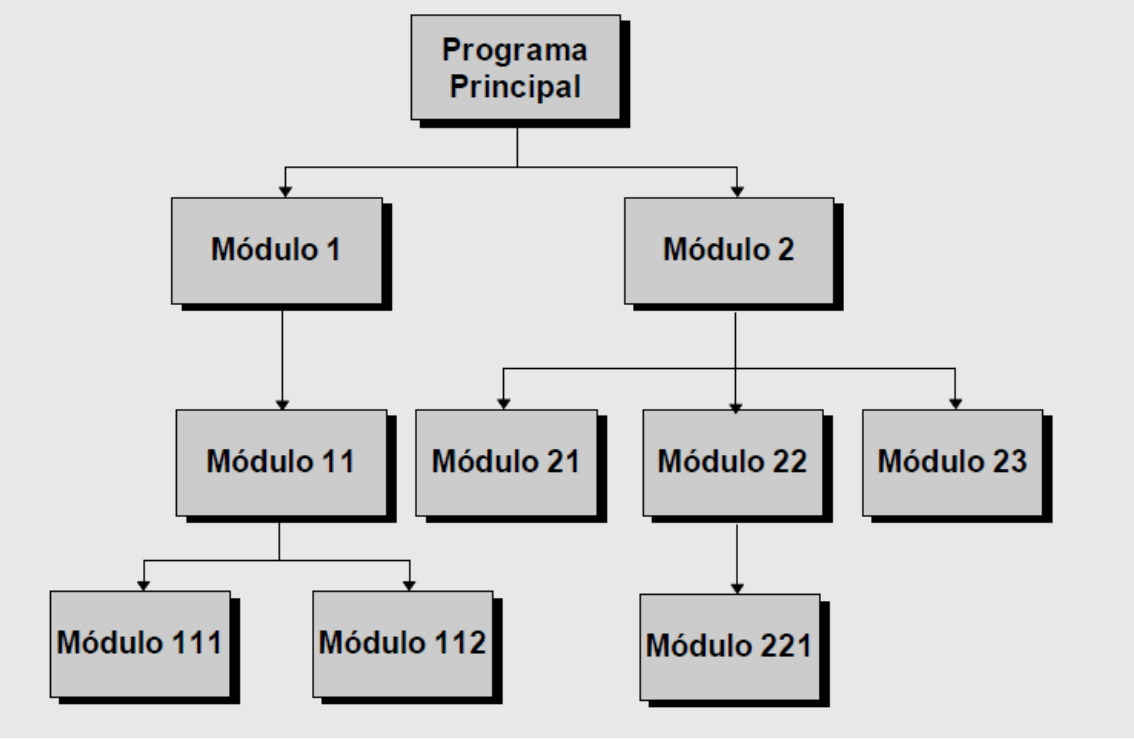

Figura 4. Programación Modular

1.3 Valoración de la técnica de programación usada para el desarrollo del módulo

Luego de haber analizado las diferentes técnicas de programación se llega a la conclusión de utilizar la POO para dar solución al problema planteado, teniendo en cuenta los siguientes aspectos:

- Reusabilidad: permite utilizar una clase definida previamente en las diferentes aplicaciones que sea conveniente.
- Mantenibilidad: las clases que conforman una aplicación, son fáciles de mantener sin afectar a los demás componentes de la aplicación.
- Extensibilidad: gracias a la modularidad y a la herencia una aplicación diseñada bajo el paradigma de la orientación a objetos puede ser fácilmente extensible para cubrir necesidades de crecimiento de la aplicación.
- Modificabilidad: la facilidad de añadir, suprimir o modificar nuevos objetos permite hacer modificaciones de una forma muy sencilla.
- Fiabilidad: al dividir el problema en partes más pequeñas permite probarlas de manera independiente y aislar mucho más fácilmente los posibles errores que puedan surgir.

En la implementación de la solución actual la POO se evidencia en los siguientes aspectos:

- En el Mapeo Relacional de Objetos (ORM, por sus siglas en inglés) a la base de datos que realiza Propel como ORM que utiliza symfony, que permite que las tablas de la base de datos estén disponibles como objetos en el código.
- En la generación de código orientado a objetos para las funcionalidades más comunes del manejo de bases de datos.
- En la utilización clases definidas en PHP para la creación de componentes de ExtJS.

1.4 Metodologías, tecnologías y herramientas

En el desarrollo de soluciones informáticas la selección de las herramientas correctas tiene una relación directa con el tiempo y la calidad de los artefactos asociados al ciclo de vida del software. En el presente epígrafe se identificarán las herramientas, tecnologías y metodología utilizada en el desarrollo del trabajo.

Proceso Unificado de Software

El Proceso Unificado de Software (RUP, por sus siglas en inglés) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

En esta metodología se divide el proceso en cuatro fases: Concepción o Inicio, Elaboración, Construcción y Transición, cada una de ellas representa un ciclo de desarrollo en la vida de un producto de software. Además agrupa las actividades en grupos lógicos, definiéndose nueve flujos de trabajo principales, los seis primeros son conocidos como flujos de ingeniería y los tres últimos como flujos de apoyo, ellos son: Modelado del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, Instalación o Despliegue, Administración del proyecto, Administración de configuración y cambios y Ambiente.

El ciclo de vida de RUP se caracteriza por (19):

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.

- **Iterativo e Incremental:** Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

Lenguaje Unificado de Modelado

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables; está integrado por un conjunto de diagramas, los cuales se encuentran agrupados en dos categorías fundamentales: diagramas de estructura y diagramas de comportamiento.
Este lenguaje de modelado permite crear un nivel de comprensión y entendimiento entre los analistas, desarrolladores o cualquier personal involucrado y hace más fácil la comunicación existente entre ellos. (20)

Herramientas CASE

Las herramientas de Ingeniería de Software Asistida por Ordenador (CASE, por sus siglas en inglés) son aplicaciones informáticas destinadas a facilitar el desarrollo de software optimizando el costo de las mismas en términos de tiempo y de dinero.

Visual Paradigm for UML 3.6

Es una herramienta CASE que utiliza Lenguaje de Modelado Unificado: como lenguaje de modelado. Esta herramienta está desarrollada por Visual Paradigm Internacional una de las principales compañías de herramientas CASE donde su mayor éxito consiste en el uso libre del producto mencionado.

Dicha herramienta ofrece soporte al ciclo de vida completo del software; permite la captura de requisitos, la creación de diagramas UML, la realización de ingeniería inversa, la generación de código Hypertext PreProcessor (PHP, por sus siglas en inglés) y otros lenguajes de programación.

Entorno Integrado de Desarrollo

Un Entorno Integrado de Desarrollo (IDE, por sus siglas en inglés) es una herramienta utilizada por los programadores para la implementación del producto deseado. Con su utilización es posible crear la interfaz gráfica de usuario y escribir el código de lo que hará la aplicación de una manera más amena y en una interfaz amigable. Además hace que el código sea más entendible y organizado, pues

implementa en su interior ciertos estándares y niveles de identación, hace más fácil la integración de un equipo de trabajo.

NetBeans 6.9

NetBeans v6.9 es una herramienta para escribir, compilar, depurar y ejecutar programas. Consta de una gran comunidad de usuarios en constante crecimiento, lo que le ha permitido, al igual que a muchos otros sistemas libres, el progreso paulatino de sus prestaciones y la eliminación de errores que pudiesen existir. El soporte para PHP ha sido extendido para incluir el Framework Symfony y PHP 5.3.

Es gratuito para desarrolladores de software y ofrece todas las herramientas necesarias para crear aplicaciones Web y de escritorio con el lenguaje Java, C/C ++ y lenguajes dinámicos como PHP y Java Script. Es fácil de instalar y se puede ejecutar tanto en Windows como en Linux; también tiene detección de errores de sintaxis en tiempo real. Permite que las aplicaciones se desarrollen a partir de un conjunto de módulos o componentes de software. Brinda una barra de navegación para el acceso rápido a funciones en una clase muy extensa, además de un completamiento de código fuente eficiente y seguro.

La integración con Symfony permite desarrollar aplicaciones de forma más sencilla y productiva. En primer lugar, es posible crear nuevos proyectos y aplicaciones directamente desde el IDE. También se pueden ejecutar todas las tareas de Symfony, incluso pasándole argumentos y opciones, visualizando el resultado sin necesidad de utilizar una consola de comandos externa. (21)

Framework

Ayudan en el desarrollo de software, proporcionan una estructura definida la cual ayuda a crear aplicaciones con mayor rapidez y a la hora de realizar el mantenimiento del sitio gracias a la organización durante el desarrollo de la aplicación. Son desarrollados con el objetivo de brindarles a los programadores y diseñadores una mejor organización y estructura a sus proyectos. Utilizan la POO, permitiendo la reutilización de código. (22)

Symfony 1.4

Su primera versión fue publicada en octubre de 2005. Para mantener su enfoque orientado a objetos, hace uso de Propel, un mapeo de objetos a bases de datos. Este hace que el programador se evite

realizar grandes consultas SQL, simplemente a través de llamadas a los objetos, este las traduce a optimizadas consultas SQL sin tener en cuenta el gestor de base de datos utilizados.

Diseñado con el objetivo de optimizar la creación de las aplicaciones web, con el uso de sus características. Posee una librería de clases que permiten reducir el tiempo de desarrollo (22). Está desarrollado en PHP5, se puede utilizar en plataformas Unix, Linux y Windows. Requiere de una instalación, configuración y líneas de comando, incorpora el patrón Modelo-Vista-Controlador, soporta AJAX, plantillas y un gran número de bases de datos.

Symfony es un enorme conjunto de herramientas y utilidades que minimizará el desarrollo de la aplicación a desarrollar, ya que es una de las mejores copias para PHP del famoso framework Ruby on Rails (22). También ha tomado las mejores ideas de Rails y de muchos otros framework, incorporando ideas propias y el resultado es un framework elegante, estable, productivo y muy bien documentado.

ExtJS 2.2

Framework de presentación JavaScript del lado del cliente para el desarrollo de aplicaciones web. Ofrece múltiples opciones para el trabajo con las validaciones y manejo de errores en el cliente. La personalización de temas de estilos es posible en su utilización, además que provee el trabajo con una amplia configuración e intenso trabajo con las Hojas de Estilo en Cascada (CSS, por sus siglas en inglés).

Toda su funcionalidad en JavaScript se logra mediante librerías Yahoo User Interface (YUI, por sus siglas en inglés), jQuery, o con la utilización de la librería nativa, así en tiempo de ejecución carga y crea todos los objetos de Lenguaje de Marcado de Hipertexto (HTML, por sus siglas en inglés) a través del uso intenso del Modelo de Objetos de Documento (DOM, por sus siglas en inglés). Cuenta con dos licencias, una comercial y otra de código abierto.

Ventajas de ExtJS:

- Código reutilizable.
- Independiente o adaptable a framework diferentes (prototype, jquery, YUI).
- Orientada a la programación de interfaces en la web.
- Soporte comercial.

- Extensa comunidad de usuarios.

En el desarrollo del capítulo se realizó un análisis de las diferentes técnicas de programación existentes que permitió llegar a la conclusión de utilizar la programación orientada a objeto, debido a las ventajas que esta ofrece. El análisis de las tecnologías y herramientas permitió seleccionar el ambiente de desarrollo para dar solución al problema planteado. Para ello se utilizará la metodología de desarrollo RUP, apoyado en el lenguaje de modelado UML 2.1, las herramientas: NetBeans 6.9 como IDE, así como Visual Paradigm 3.6 y DB Designer 4 Fork 1.0 como herramientas CASE, además de la utilización de los frameworks Symfony 1.4 y ExtJS 2.2, soportado sobre una plataforma LAMP.

Capítulo 2: *Descripción de la Arquitectura.*

En el presente capítulo se realiza una descripción de la arquitectura, donde se exponen los requerimientos no funcionales, se analizan los posibles componentes o módulos ya existentes y que pueden ser reutilizados, se representa el Modelo de Despliegue del sistema y además se definen los estándares de codificación a utilizar.

2.1 Requerimientos no funcionales

Los requisitos no funcionales (RnF) son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable (23). Los requerimientos no funcionales son fundamentales en el éxito del producto y normalmente están vinculados a los requisitos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse, qué cualidades debe tener y cuán rápido o grande debe ser. Existen múltiples categorías para clasificar a los requisitos no funcionales, a continuación se muestran los más destacados en este sistema.

Usabilidad

- Los usuarios del sistema deben tener un tiempo de entrenamiento no mayor de un mes para que sean productivos operando en el mismo.
- La aplicación web deberá facilitar la interacción usuario – sistema con el objetivo de evitar rechazo en el uso de la misma, y guiará mediante mensajes al usuario en las diferentes acciones que realice.

Confiabilidad

- El sistema solo podrá estar inactivo durante cinco minutos, al cabo de ese tiempo se cerrará la sesión del usuario que estaba interactuando con él.
- Al ocurrir un error el sistema muestra un mensaje indicando dónde se ha producido el mismo para que pueda ser corregido por el usuario.

Eficiencia

- La capacidad máxima que podrá alojar el sistema para su funcionamiento en cada nodo es de 40 PC_Clientes.

- La aplicación tendrá un tiempo promedio de respuesta por transacciones de 156.7 ms aproximadamente.
- La utilización de los recursos de la aplicación será de 115.208 KB de memoria aproximadamente para casos de uso de complejidad alta.

Soporte

- La implementación del sistema se regirá por las normas de codificación definidas en el documento de arquitectura de software del Centro de Informática Médica para obtener un producto legible y homogéneo.
- Se utilizarán para el nombrado la notación CamellCasing para identificar las variables y la PascalCasing para los métodos, lo cual hace el código más legible y facilita el soporte y mantenimiento de la aplicación.

Restricciones de diseño

- El sistema estará desarrollado en base a las políticas del software libre, que fueron ajustadas al sistema nacional de salud como: uso de servidores GNU/Linux.
- Se utilizará como gestor de base de datos MySQL y Symfony como framework de desarrollo de aplicaciones web, el cual utiliza PHP como lenguaje de programación del lado del servidor.
- Se utilizará NetBeans 6.9 como IDE de desarrollo.
- Para el diseño de las interfaces se utilizará ExtJS, librería javascript para el desarrollo de aplicaciones RIA (Rich Internet Application).
- Se utilizará Visual Paradigm como herramienta para el modelado de los artefactos generados por la metodología RUP.

Requisitos para la documentación de usuarios en línea y ayuda del sistema

- La aplicación web contará con una ayuda donde el usuario podrá suplir las dudas que se le puedan presentar durante la utilización de la misma. El usuario del módulo deberá recibir un adiestramiento con el fin de que pueda explotar las funcionalidades del sistema sin contratiempos ocasionados por la falta de preparación técnica.

Interfaz

- La aplicación será sencilla, amigable e intuitiva, de fácil navegación por parte del usuario. Estará diseñada para una óptima visualización siendo adaptable a cualquier resolución.
- La interfaz de la aplicación será similar a la de los sistemas operativos de la plataforma Windows, para garantizar que el personal que trabaje con el sistema esté lo más familiarizado posible, de manera que agilice y facilite el trabajo con el software.
- La entrada de datos incorrecta será detectada claramente e informada al usuario.
- Todos los textos y mensajes en pantalla aparecerán tanto en idioma español como en inglés, en dependencia de la opción que seleccione el usuario.

Estándares aplicables

- El desarrollo de la aplicación estará basado en Integración de Modelos de Madurez de Capacidades (CMMI, por sus siglas en inglés), un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

2.2 Descripción de la arquitectura

Patrón arquitectónico

El patrón arquitectónico definido por la dirección del proyecto para el desarrollo de sus productos es el Modelo Vista Controlador (MVC), el mismo separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El estilo de llamada y retorno MVC, se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista. La descripción de cada componente es la siguiente: (24)

- **Modelo:** contiene la lógica de negocio, la base de datos se encuentra almacenada en esta capa, en la que se engloban los datos y las funcionalidades. Se comporta de manera independiente a cualquier representación de salida y comportamiento de entrada, asegurando la integridad de los datos y permitiendo derivar nuevos.

- **Vista:** es la parte que utilizan los usuarios para interactuar con la aplicación, simplemente la página HTML que se muestra al usuario. Esta capa se centra en presentar al modelo en un

formato adecuado para el intercambio con el usuario final. Los gestores de plantillas se incluyen dentro de esta.

- **Controlador:** se enfoca en realizar las llamadas necesarias al modelo para capturar los datos y devolverlos a la vista para que en esta sean mostrados al usuario. Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo. En resumen, se encarga de asimilar las peticiones del usuario y realizar los cambios apropiados en el modelo o en la vista.

En la figura 5 se observa la interacción que se efectúa entre las distintas capas, específicamente en el framework Symfony que es el que se emplea en la presente investigación.

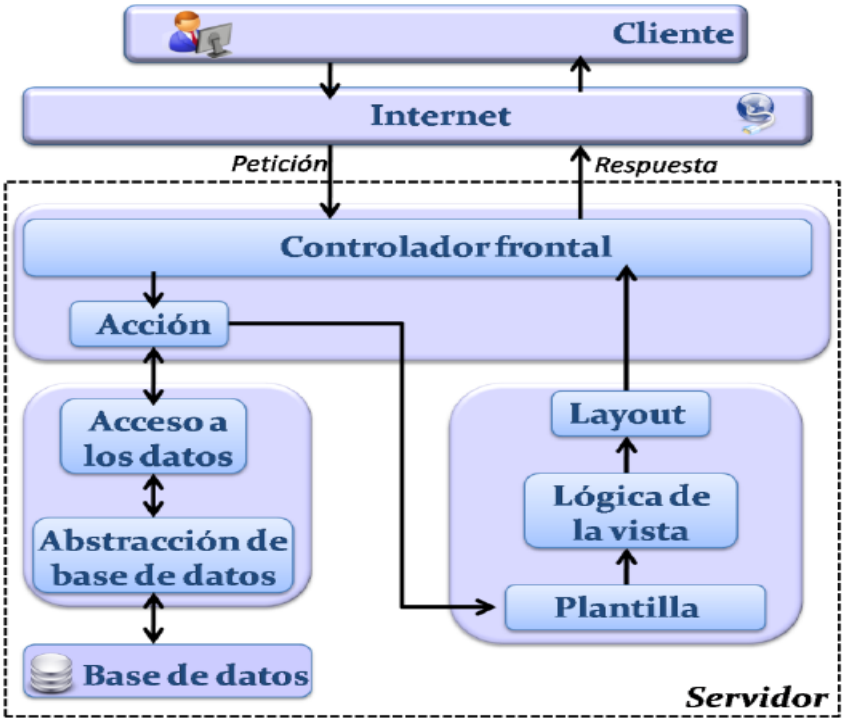

Figura 5. Comportamiento del MVC en Symfony

Las capas del MVC que implementa Symfony, se encuentran compuestas por siete scripts fundamentalmente, los cuales se distribuyen de la siguiente forma: en el modelo se encuentra la abstracción de la base de datos y el acceso a los datos; en la vista se podrá encontrar la vista, las

plantillas y los layouts correspondientes; mientras que en el controlador se cuenta con el controlador frontal y las acciones.

Propel, se encarga de generar de forma automática las clases de la capa del modelo en dependencia de la estructura de datos de la aplicación, creando métodos especiales para acceder y modificar los datos. La abstracción de la base de datos se gestiona de manera transparente para el programador, con la utilización Objetos de Datos de PHP (PDO, PHP Data Objects), razón por la cual es posible migrar de sistema gestor de bases de datos. Por su parte la lógica de la vista se puede transformar en un archivo de configuración sencillo, sin necesidad de que esto implique grandes cambios, debido a que generalmente, el layout es global en toda la aplicación o en un grupo de páginas. La plantilla sólo se encarga de dar salida a las variables definidas en el controlador.

El controlador en Symfony normalmente se divide en un controlador frontal, que es único para toda la aplicación y las acciones, que incluyen el código específico del controlador de cada página. Una de las principales ventajas de utilizar un controlador frontal es que ofrece un punto de entrada único para toda la aplicación. Así, en caso de que sea necesario impedir el acceso a la aplicación, solamente es necesario editar el script correspondiente al controlador frontal. Si la aplicación no dispone de un controlador frontal, se debería modificar cada uno de los controladores. (22)

Patrones de diseño empleados

Los patrones de diseño no son más que la descripción de un problema y la solución del mismo, de forma que se pueda utilizar en diferentes contextos dando respuesta a interrogantes comunes. Los patrones Asignación General de Responsabilidades (GRASP, por sus siglas en inglés) y Banda de los cuatros (Gof, por sus siglas en inglés) tienen una importante utilidad en el diseño realizado. Los utilizados en la realización del presente trabajo son los siguientes:

- **Bajo Acoplamiento:** es el encargado de disminuir la dependencia de una clase con las demás. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño, estimulando asignar una responsabilidad de modo que su colocación no incremente el acoplamiento. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. En symfony se evidencia en la capa del Modelo donde se encuentran las clases que implementan la lógica de negocio y de acceso a datos, estas clases tienen pocas asociaciones con otras de la Vista o el Controlador por

lo que la dependencia en este caso es baja, poniéndose de manifiesto el patrón Bajo Acoplamiento. (25)

- **Alta Cohesión**: Su utilización mejora la claridad y facilidad para comprender el diseño, simplifica el mantenimiento y las mejoras de funcionalidad, genera un bajo acoplamiento, soporta mayor capacidad de reutilización. Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase Actions, la cual está formada por varias funcionalidades que están estrechamente relacionadas, siendo la misma responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las properties.

- **Experto:** es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos, con este se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil manteniendo. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clase "sencillas" y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión. (26)

- **Controlador:** este se basa en la existencia de un intermediario entre las páginas clientes y el algoritmo que responde a las peticiones realizadas por estas. La existencia del controlador frontal, es el ejemplo básico que evidencia de forma clara su utilización en Symfony, este maneja las peticiones del usuario, la seguridad, carga la configuración de la aplicación y otras tareas, siendo único para cada aplicación. En busca de aminorar un poco la carga que este posee se cuentan con las acciones que contienen las especificaciones de cada página.

- **Factory:** consiste en la definición de una clase que realiza una determinada tarea. Symfony utiliza las factorías en su funcionamiento interno, como por ejemplo, para los controladores y las sesiones. Cuando el framework necesita crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea. La principal ventaja de utilizar las definiciones de las factorías es que se hace muy sencillo modificar las características internas. (27)

- **Decorador:** es utilizado cuando deseamos modificar el comportamiento básico de una instancia específica sin la necesidad de crear una nueva subclase. Symfony presenta el denominado archivo layout.php o también conocido como plantilla global, en la que convergen todos los elementos comunes a cada una de las páginas del sistema en construcción. Este fichero se complementa con las plantillas, decorándolas y obteniéndose la interfaz final que será mostrada al usuario. (27)

2.3 Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados

En la implementación del módulo en cuestión, se utilizaron componentes del framework de presentación ExtJS, como por ejemplo: combobox y grid, los cuales son declarados (o definidos) como se muestra a continuación:

```
new Ext.form.ComboBox({
        id:'".$this->id."',
        url: '".$this->url."',
        ids: '".$this->ids."',
        store: getStoreCb('".$this->url."', '".$this->ids."', '".$this->sendValues."'),
        displayField: 'value',
        valueField: 'key',
        typeAhead: true,
        fieldLabel: '".$this->label."',
        mode: 'local',
        forceSelection: true,
        invalidText : 'Seleccione...',
        emptyText:'Seleccione...',
        allowBlank: false,
        selectOnFocus:true,
        listeners: {
            select: function(a,b,id){
            eval('".$this->functionSelect."');
            clearCbs('".$this->idsClear."');},
            focus: function(cb){
            eval('".$this->functionFocusAntes."');
            this.reset();
            varObj= Ext.getCmp('".$this->id."');
            varObj.clearValue();
            this.store.proxy= getProxyCb('".$this->url."', '".$this->ids."', '".$this->sendValues."');
            this.store.load({
            callback: function(){
            eval('".$this->functionFocusDespues."');}});}}
            ".$propiedades." })
```

Figura 6. Código del comboBox en ExtJS

```
new Ext.grid.GridPanel({
        width:800,
        autoHeight:true,
        id: '" . $this->id . "',
        title:'" . $this->title . "',
        urlPrinter: '" . $this->url . "',
        cmPrinter: new Array(" . $this->cm . "),
        store: Ext.getCmp('myStore" . $this->idClass . "').getValue(),
        cm: cmGrid({" . $this->titles . "}, {" . $this->titlesProp . "}),
        trackMouseOver: true,
        loadMask: true,
        stripeRows: true,
        viewConfig: viewConfigGrid(),
        plugins: getFilterGrid(new Array(" . $this->filter . ")),
        bbar: bbarGrid( Ext.getCmp('myStore" . $this->idClass . "').getValue(), '" . $this->id . "', " . $this->toolBar . " )
        " . $this->initPropiedades() . "})
```

Figura 7. Código del Grid en ExtJS

En aras de facilitar el proceso de desarrollo en el SIGICEM, agilizar el tiempo de implementación y lograr una mayor reutilización de código, dichos componentes fueron integrados en clases PHP, permitiendo la creación de objetos a la hora de utilizarlos como se muestra a continuación:

```
<?php
    $comboMeses= new ComboBox('identificador', 'nombre', 'metodo donde cargar los datos');
    echo $comboMeses->renderJs();
?>,
```

Figura 8. Creación de la clase comboBox hecha en PHP

```php
<?php
    $grid= new Grid("identificador", 'metodo donde cargar los datos', 'parametros');
    $columnas= array();
    $grid->setColunModel($columnas);
    echo $grid->renderJs();
?>,
```

Figura 9. Creación de la clase Grid hecha en PHP

2.4 Seguridad

Actualmente en el mundo, la información constituye un eslabón de vital importancia, por lo que es necesario que existan mecanismos que garanticen la protección de los datos que se manipulan en la realización del subsistema, así como los permisos a los usuarios en dependencia del nivel jerárquico que cumple su rol dentro del mismo.

SIGICEM será responsable de la seguridad porque implementará una jerarquía de accesos para los diferentes usuarios del sistema donde todos no tendrán los mismos privilegios sobre las posibles acciones a realizar.

Para añadir funciones de autorización y autenticación a las funciones de seguridad estándar de Symfony se utiliza el plugin sfGuardPlugin. Este brinda el modelo (usuario, grupo y objetos de permisos) y los módulos (backend y frontend) para asegurar la aplicación en un minuto de configuración. (28)

En Symfony el acceso a las acciones puede ser restringido en el archivo security.yml de tal manera que si un usuario pide realizar una acción que está asegurada, este tiene que estar autenticado y tener las credenciales correctas. El manejo de sesiones se hace a través del objeto sfUser, este es un objeto singlenton y puede ser accedido desde cualquier parte de la aplicación.

Una vez instalado el plugin se agregan siete tablas a la base de datos:

- **sf_guard_user:** tabla con todos los usuarios.
- **sf_guard_permission:** tabla con todos los permisos.
- **sf_guard_user_permission:** tabla que relaciona los usuarios y permisos de los mismos.

- **sf_guard_group:** tabla que define grupos de usuarios.
- **sf_guard_user_group:** tabla que lista los usuarios de un grupo.
- **sf_guard_group_permission:** tabla que lista los permisos de un grupo.
- **sf_guard_remember:** almacena la dirección ip y las llaves de las sesiones de los usuarios.

2.5 Vista de Despliegue

Un diagrama de despliegue muestra la disposición física de los distintos nodos que componen un sistema y se indican cuáles son los enlaces de comunicación existentes entre los distintos componentes en tiempo de ejecución. Para la estructura de la distribución física del subsistema se utilizó un nodo PC_Cliente que contiene las distintas funcionalidades ofrecidas al usuario. Este se conecta mediante el protocolo de Transferencia de Hipertexto (http, por sus siglas en inglés) al nodo Servidor Web en el que se encontrará el Sistema de Gestión para la Ingeniería Clínica y Electromedicina. Desde el servidor se puede acceder al nodo Servidor de Base de Datos mediante el protocolo diseñado para facilitar la reutilización de código de bases de datos (TCP/IP). Además, en la estación PC_Cliente podrá estar conectada una impresora mediante el puerto USB.

En la Figura 10 se muestra la propuesta de despliegue antes descrita.

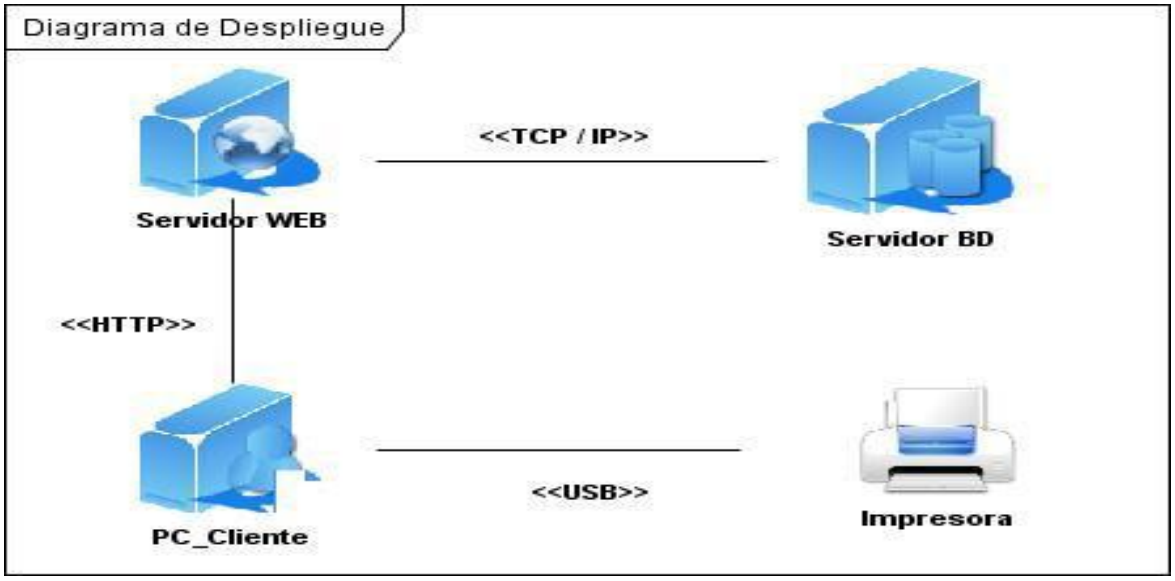

Figura 10. Diagrama de despliegue del módulo de Mantenimiento

Requerimientos de Hardware

Nodo PC Cliente

- Computador Pentium a 333 MHz o superior, 128 MB RAM o superior, MODEM o red con TCP-IP para conexión al servidor.

Nodo Servidor Web

- Computador Pentium a 2.8 GHz o superior, 1 GB RAM o superior, 1 GB de espacio libre en Disco Duro como mínimo.

Nodo Servidor de Base de Datos

- Computador Pentium a 2.8 GHz o superior, 1 GB RAM o superior, 20 GB de espacio libre en Disco Duro como mínimo.

2.6 Estrategias de codificación. Estándares y estilos a utilizar

Los estándares y estilos de codificación permiten que el código fuente de la solución que esté en desarrollo posea mayor calidad. Contar con un estilo de programación único permite que todos los involucrados puedan entender en un menor tiempo y que sea más fácil de ofrecer mantenimiento al código. Luego de analizar lo anteriormente planteado, y de efectuar un estudio de los estilos y estándares definidos por el CESIM y el departamento de SES se han identificado aquellos que serán utilizados para la implementación del módulo de Mantenimiento propuesto.

Identación ***Objetivo:*** **Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento.**	
Aspectos Generales	El identado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la PC o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deber estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción. Nunca colocar { en la línea de un código cualquiera, esto requiere una línea propia.

Comentarios, separadores, líneas, espacios en blanco y márgenes. ***Objetivo:*** **Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez.**		
Ubicación de comentarios	Al inicio de cada clase o función y al final de cada bloque de código.	Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas.
Aspectos Generales	Sobre el comentario	Se debe evitar comentar cada línea de código. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción
	Sobre los espacios en blanco	No se debe usar espacio en blanco: Después del corchete abierto y antes del cerrado de un arreglo. Después del paréntesis abierto y antes del cerrado. Antes de un punto y coma.
Variables y constantes		
Apariencia de variables	Las variables tendrán un prefijo para el tipo de datos en minúscula.	El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, la cual identificara el tipo de datos al que se refiere (ver tabla 1.1), en caso de que sea un nombre compuesto se empleará notación CamellCasing**. Ejemplo: sNombrePaciente
Apariencia de constantes	Todas sus letras en mayúscula.	Se deben declarar las constantes con todas sus letras en mayúscula.
Aspectos Generales	Nombres de las variables y constantes.	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.
Clases y Objeto ***Objetivo:*** **Nombrar las clases e instancias de forma estándar para todas las aplicaciones.**		
Apariencia de clases y objetos	Primera letra en mayúscula.	Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*. Ejemplo: MiClase(). Para el caso de las instancias se comenzara con un prefijo que identificara el tipo de dato, este se escribirá en minúscula.
	Sobre las clases, los	El nombre empleado para las clases, objetos,

Aspectos Generales	objetos, los atributos y las funciones.	atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.
Bases de Datos, Tablas, esquemas y Campos		
Apariencia de la Base de Datos.	Las dos primeras letras representan el tipo.	Los nombres de las Bases de Datos deben comenzar con el prefijo bd a continuación underscoard y luego el nombre completamente en minúscula, en caso de que sea un nombre compuesto se empleará notación. Los nombres serán cortos y descriptivos. Ejemplo: bd_balancematerial.
Apariencia de las tablas	Las dos primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para las tablas debe comenzar con el prefijo tb seguido de underscoard y luego debe escribirse todas las letras en minúscula, en caso de que sea un nombre compuesto se utilizara underscoard para separarlo. Ejemplo: “tb_producto‟;
Tablas que representen Relaciones	Las dos primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tr seguido de underscoard y el nombre será la concatenación del nombre de las dos tablas que la generaron separados por uderscoard todo en minúscula. Ejemplo: “tr_paciente_enfermedad‟
Apariencia de los campos	Todas las letras en minúscula.	El nombre a emplear para los campos debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor. Ejemplo: “id_producto‟;
Nombre de los campos	En caso de identificadores.	Todos los campos identificadores van a comenzar con el identificador id seguido de underscoard y posteriormente el nombre del campo Ejemplo: “id_municipio.”
Aspectos Generales	Sobre las BD, vistas, tablas, atributos y procedimientos.	El nombre empleado para las Bases de Datos, las vistas, las tablas, los campos y los procedimientos almacenados, deben permitir que con sólo leerlos se conozca el propósito de los mismos.
Controles		
Apariencia de los controles.	Los controles tendrán un prefijo para el tipo de datos en minúscula.	El nombre que se le da a los controles deben comenzar con las primeras letras en minúscula, las cuales identificaran el tipo de datos al que se refiere (ver tabla 1.2), en caso de que sea un nombre compuesto se empleará notación CamellCasing**.

		Ejemplo: btnAceptar

Tabla 1. Estándares de Codificación

Tipo Datos	Prefijo	Ejemplos
Int	i	iCantPacientes
float	f	fPesoPaciente
double	d	dPesoCarro
bool	b	bPacienteActivo
string	s	sNombrePaciente
char	c	cLetra
De tipo enum	ev	evSexo
byte	b	bCantDiaspaciente
sbyte	sb	sbEdadPaciente
short	sh	shVariableUshort
uint	ui	uiVariableUint
long	l	lVariableLong
ulong	ul	ulValiableUlong
decimal	dc	dcVariableDecimal
Objetos	o	oPacienteHistorico
Objetos de tipo Struct	st	stUnaStruct

Tabla 2. Prefijos para Tipos de Datos de las Variables

Control	Prefijo	Ejemplo
Boton	btn	btnAceptar
Etiqueta	lbl	lblNombre
Lista/Menu	mn	mnPrincipal
Campo de Texto	txt	txtFecha
Botón de Opción	bpt	bptSexo
Casilla de Verificación	chx	chxBorrar
Casilla de Selección	cbx	cbxSexo

Tabla 3. Prefijos para Tipos de Datos de los Controles

Durante la confección del actual capítulo se analizaron los aspectos relacionados con la seguridad del sistema, los cuales, sin lugar a dudas, contribuyen a que el mismo sea robusto y confiable. Se describió la arquitectura, aspecto de vital importancia para entender la aplicación, organizar su desarrollo, plantear la

reutilización del software y hacerla evolucionar. Además se obtuvo la distribución física que se debe seguir para el despliegue del módulo de Mantenimiento, especificando cada uno de sus elementos. Con la utilización de los estándares y estilos de codificación para la implementación de los componentes definidos, se logró establecer las pautas por las que se regirá el desarrollo del sistema.

Capítulo 3: *Descripción y Análisis de la solución propuesta*

En el presente capítulo se valora de forma crítica, el diseño propuesto por el analista, permitiendo refinar elementos del diseño tales como los diagramas de clases del diseño y los diagramas de interacción. Unido a esto se describen las nuevas clases u operaciones empleadas. Se realiza el modelo de datos, así como una breve valoración de las técnicas de validación. Finalmente se presentará una vista de la implementación mediante el diagrama de componentes.

3.1 Valoración crítica del diseño propuesto por el analista

El diseño, es el primer paso en la fase de desarrollo de cualquier producto o sistema de ingeniería, en él se desarrollan, revisan y documentan los refinamientos progresivos de la estructura de datos, arquitectura, interfaces y datos procedimentales de los componentes del software. Toma en cuenta los requisitos funcionales y no funcionales para crear un plano del modelo de implementación; para la consolidación de una arquitectura sólida y estable, suficiente para lograr que el sistema sea implementado sin errores. (29)

"El modelo de diseño es una abstracción de la implementación del sistema. Se utiliza para concebir y para documentar el diseño del sistema de software. Es un producto de trabajo integral y compuesto que abarca todas las clases de diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos". (30)

El levantamiento de requisitos realizado por el analista, arrojó que el módulo de Mantenimiento debía cumplir con 15 requisitos funcionales, los cuales se listan a continuación:

- Planificar Mantenimiento anual(frecuencia, meses)
- Planificar Mantenimiento mensual(mes, equipo a planificar mantenimiento, fecha, especialista responsable, procedimiento a seguir)
- Re – planificar Mantenimientos(fecha, especialista responsable, procedimiento a seguir)
- Registrar un nuevo procedimiento de mantenimiento(nombre, frecuencia, nomenclador del equipo(especialidad, denominación, marca, modelo), piezas, herramientas y guías técnicas)
- Modificar datos de un procedimiento de mantenimiento(nombre, frecuencia, nomenclador del equipo(especialidad, denominación, marca, modelo), piezas, herramientas y guías técnicas)
- Visualizar datos de procedimiento de mantenimiento

- Eliminar procedimiento de mantenimiento
- Registrar herramientas(nombre, tipo de herramienta, marca y descripción de la herramienta)
- Modificar datos de herramientas(nombre, tipo de herramienta, marca y descripción de la herramienta)
- Visualizar datos de herramientas
- Eliminar herramientas
- Registrar nueva guía técnica(nombre, descripción de la guía, nomenclador del equipo(especialidad, denominación, marca, modelo), datos de la guía, especialista responsable)
- Modificar datos de guía técnica(nombre, descripción de la guía, nomenclador del equipo(especialidad, denominación, marca, modelo), datos de la guía, especialista responsable)
- Visualizar datos de guía técnica
- Eliminar guía técnica

Una vez analizados los mismos se pudo llegar conclusión de que no satisfacían todas las necesidades del cliente, por lo que se adicionaron siete nuevos requerimientos funcionales:

- Registrar tipo de herramientas(nombre del tipo de herramienta)
- Modificar datos de tipo de herramientas(del tipo de herramienta)
- Visualizar datos de tipo de herramientas
- Eliminar tipo de herramientas
- Adicionar valoración de procedimiento(valor de la valoración, especialista)
- Modificar valoración de procedimiento(valor de la valoración)
- Visualizar valoración de procedimiento

El modelo de datos del módulo de Mantenimiento se encontraba incompleto por lo que se agregaron nuevas tablas con sus correspondientes atributos. En algunos casos, se le eliminaron atributos que no tenía sentido que existieran, las relaciones entre algunas de las tablas eran innecesarias y en otras, necesarias para poder llevar a cabo la implementación. Todas estas razones, dieron origen al refinamiento del diseño propuesto por el analista.

Diagrama de clases del diseño

A continuación se presentan los diagramas de clases de diseño, en ellos se describen gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación.

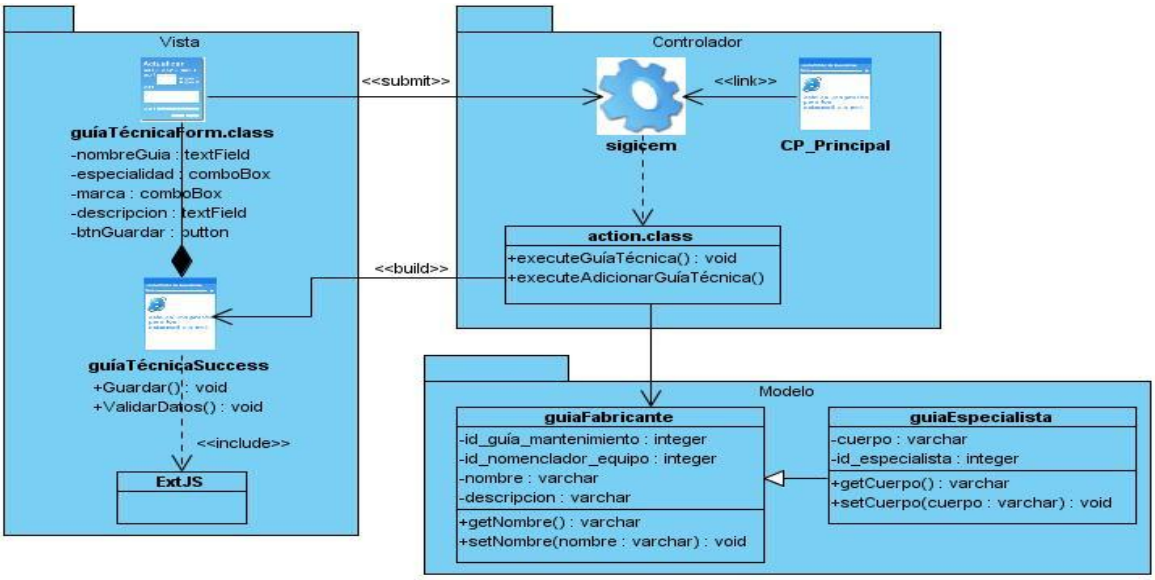

Figura 11. Diagrama de Clases de Diseño: Adicionar Guía Técnica

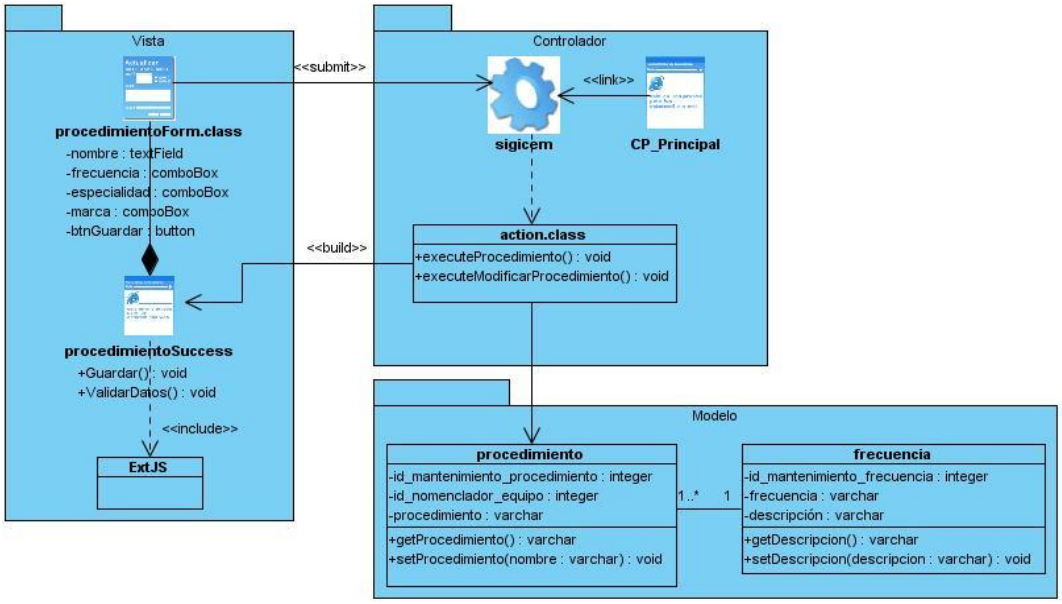

Figura 12. Diagrama de Clases de Diseño: Modificar Procedimiento

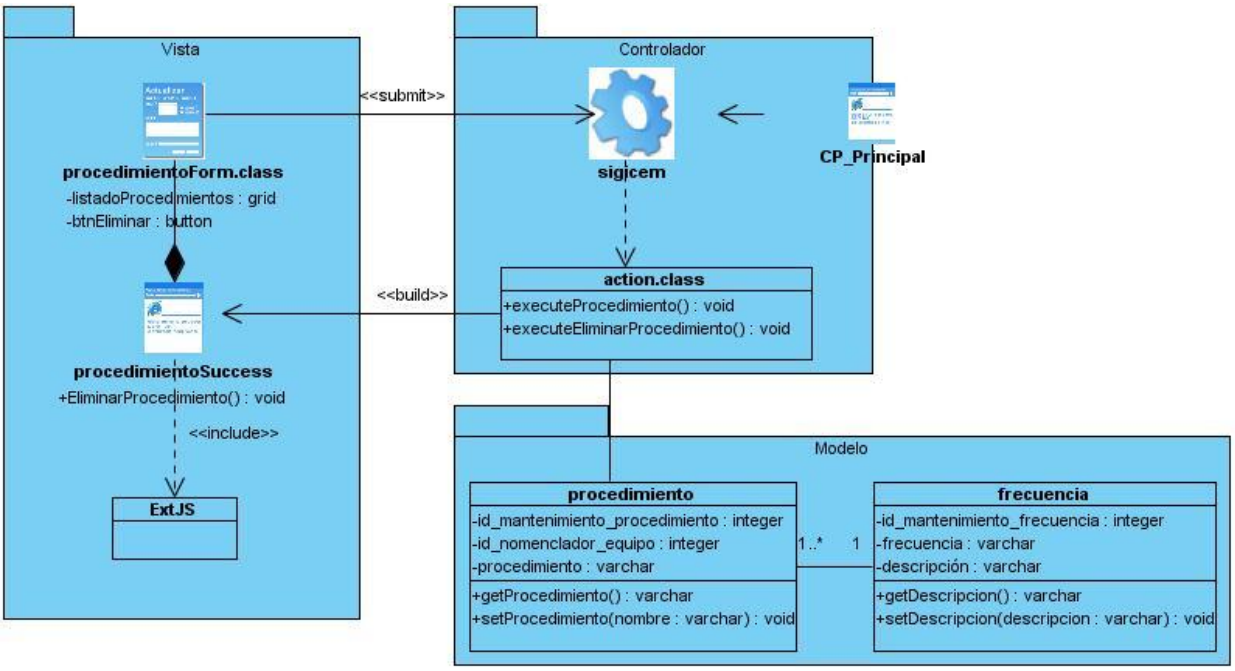

Figura 13. Diagrama de Clases de Diseño: Eliminar Procedimiento

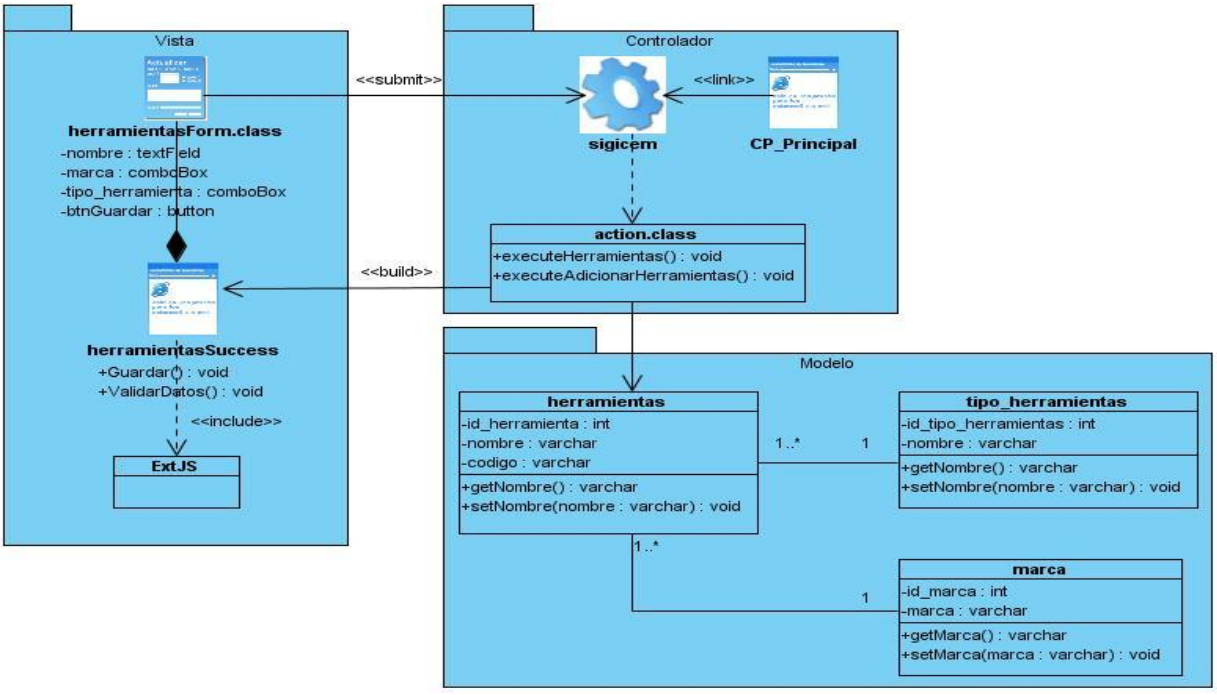

Figura 14. Diagrama de Clases de Diseño: Adicionar Herramienta

Diagramas de secuencia

Los diagramas de secuencia que a continuación se presentan muestran la interacción de un conjunto de objetos en una aplicación a través del tiempo.

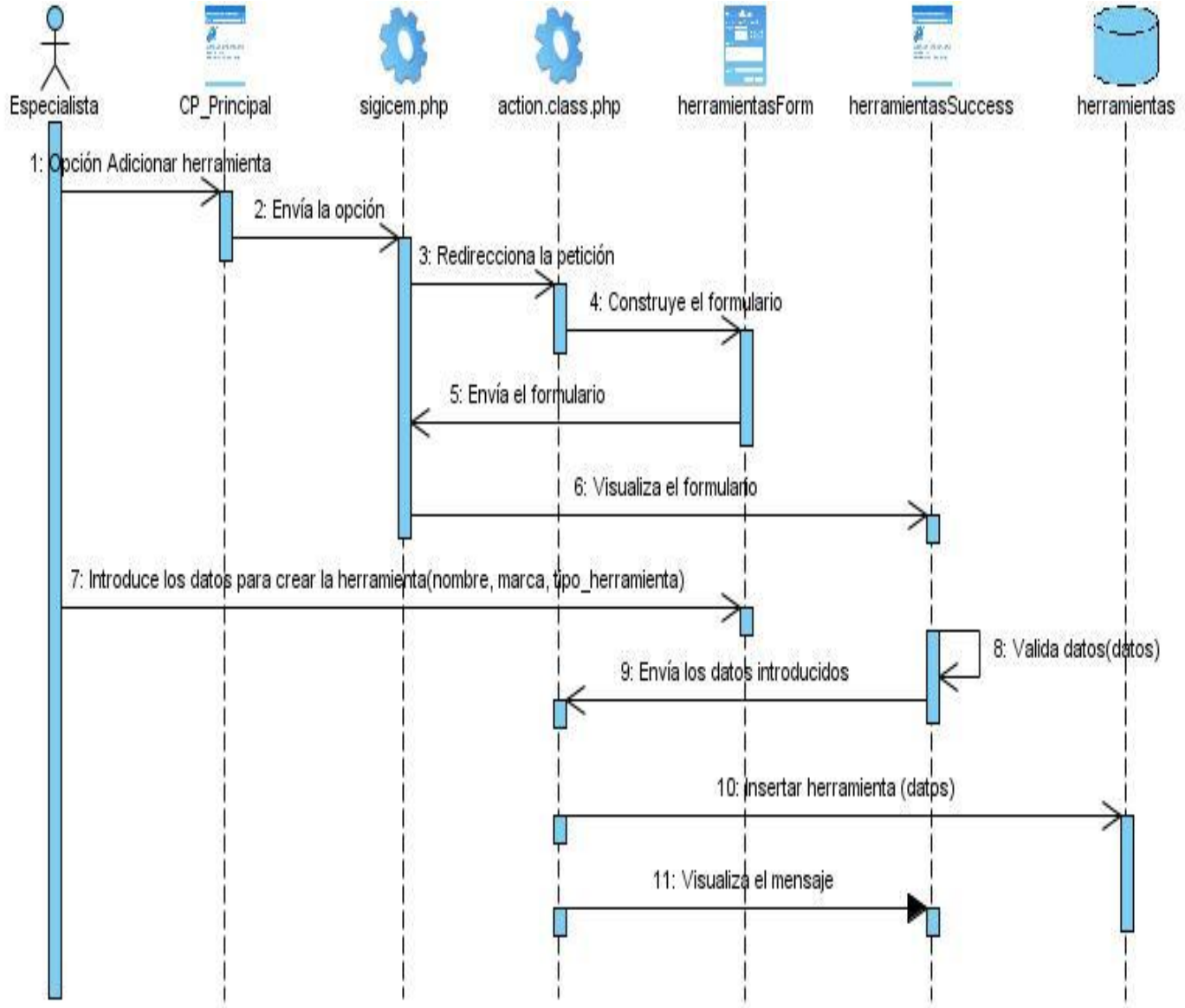

Figura 15. Diagrama de Secuencia: Adicionar Herramienta

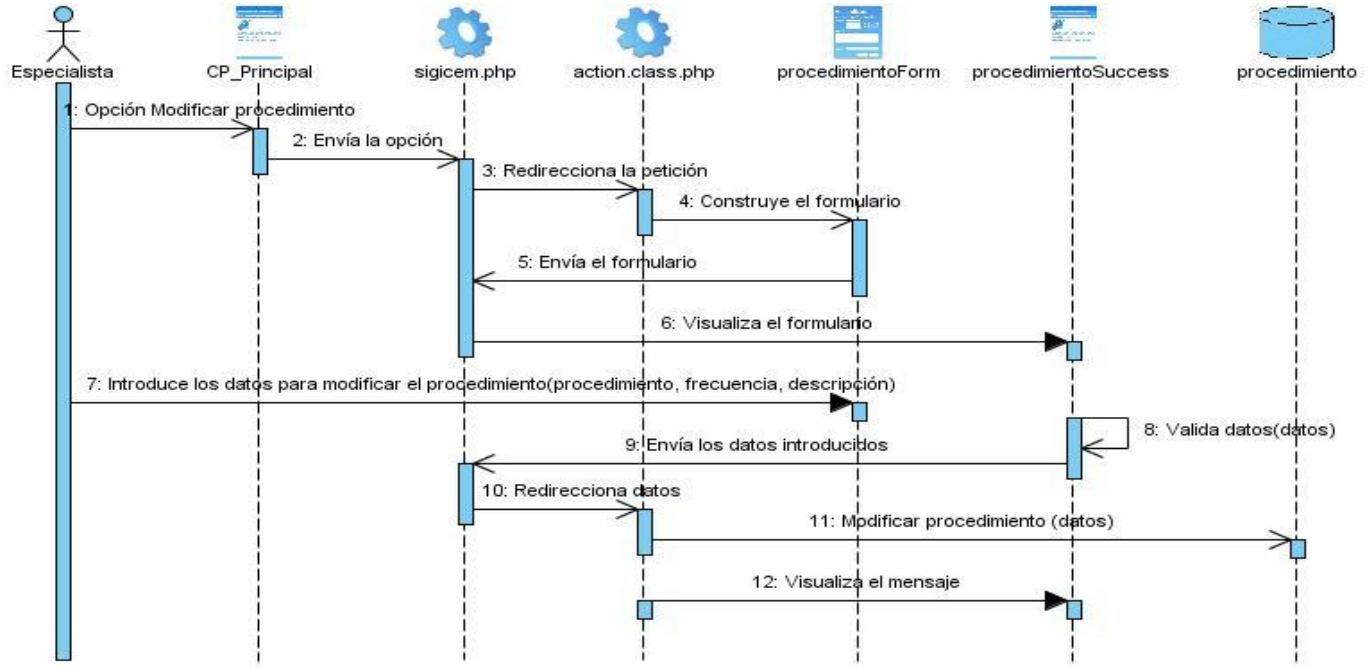

Figura 16. Diagrama de Secuencia: Modificar Procedimiento

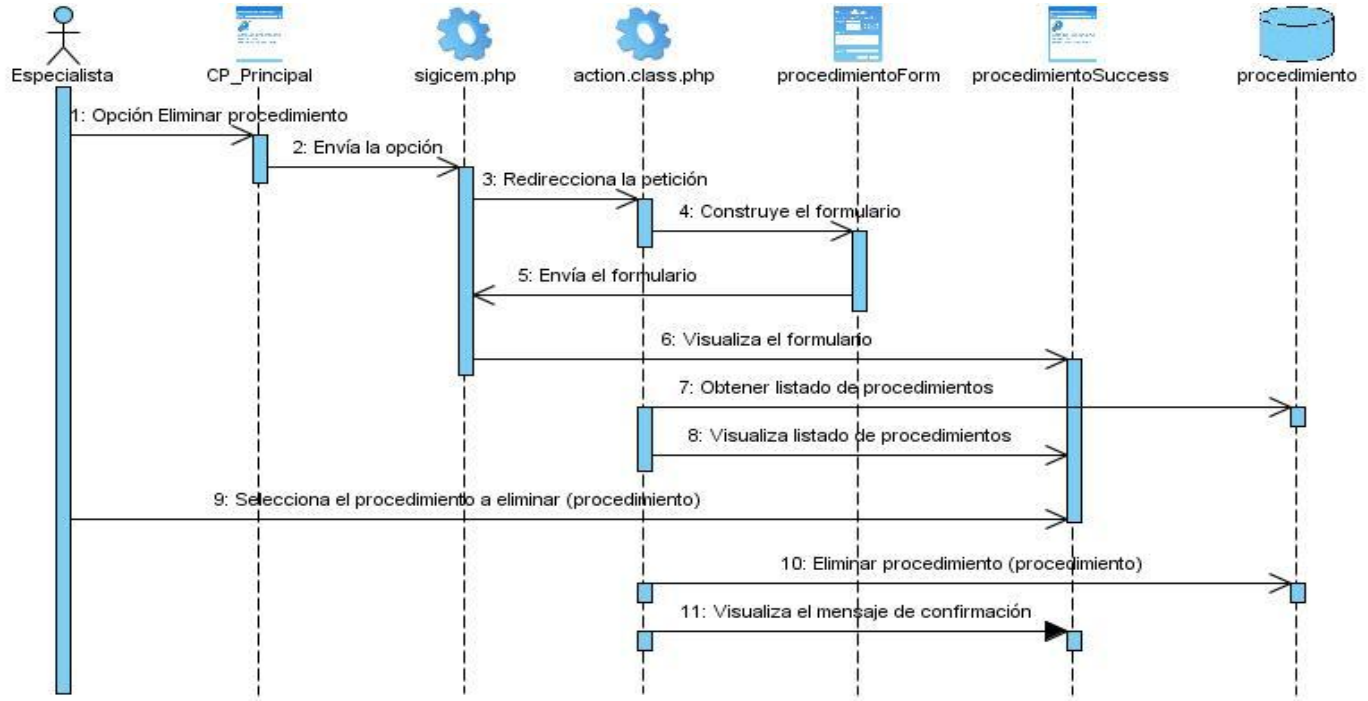

Figura 17. Diagrama de Comunicación: Eliminar Procedimiento

3.2 Descripción de las nuevas clases u operaciones necesarias

A continuación serán expuestas algunas de las clases, con sus atributos y operaciones más importantes, del sistema para lograr un mejor entendimiento sobre el funcionamiento del mismo.

Nombre: editSuccess	
Tipo de clase: interfaz	
Atributo	**Tipo**
Mes	Combobox
Listado de equipos	Grid
Fecha de inicio	Datefield
Fecha de fin	Datefield
Procedimiento	Combobox
Responsable	Combobox
Funcionalidades:	
Nombre:	setIdEquipo(idEquipo)
Descripción:	Esta función tiene como objetivo guardar el identificador del equipo seleccionado por el usuario y actualizar el Combobox de especialista con el responsable del equipo.

Tabla 4. Descripción de la clase editSuccess.php

Nombre: guiaTecnicaSuccess	
Tipo de clase: interfaz	
Atributo	**Tipo**
Nombre	Textfield
Descripción	Htmleditor
Especialidad	Combobox
Denominación	Combobox
Marca	Combobox
Modelo	Combobox
Cuerpo de la Guía	Htmleditor
Responsable	Combobox
Guía Técnica Definida	Fileuploadfield
Funcionalidades:	
Nombre:	updateIdNomeclador
Descripción:	Esta función tiene como objetivo actualizar el identificador del nomenclador de equipo seleccionado por el especialista.

Tabla 5. Descripción de la clase guiaTecnicaSuccess.php

Nombre: mantProcedimientoSuccess	
Tipo de clase: interfaz	
Atributo	**Tipo**
Responsable	Combobox
Nombre	Textfield
Frecuencia	Combobox
Especialidad	Combobox
Marca	Combobox
Denominación	Combobox
Modelo	Combobox
Piezas	Grid
Herramientas	Grid
Guías	Grid
Funcionalidades:	
Nombre:	Submit(idForm, idPage)
Descripción:	Esta función tiene como objetivo enviar todos datos ertrados por el usuario a la clase controladora.
Nombre:	addPiezas
Descripción:	Esta función tiene como objetivo adicionar al grid Piezas la pieza seleccionada por el usuario para formar parte del procedimiento.
Nombre:	addHerramientas
Descripción:	Esta función tiene como objetivo adicionar al grid Herramientas la herramienta seleccionada por el usuario para ser utilizada en el procedimiento.
Nombre:	addHerramientas
Descripción:	Esta función tiene como objetivo adicionar al grid Herramientas la herramienta seleccionada por el usuario para ser utilizada en el procedimiento.

Tabla 6. Descripción de la clase mantProcedimientoSuccess.php

Nombre: editValoracionSuccess	
Tipo de clase: interfaz	
Atributo	**Tipo**
Valoración	Slider
Funcionalidades:	
Nombre:	cambiarValor
Descripción:	Esta función tiene como objetivo mostrarle al usuario el valor de la valoración realizada cuando cambie la misma.

Tabla 7. Descripción de la clase editValoracionSuccess.php

3.3 Modelo de Datos

El Modelo de Datos está enfocado en la representación lógica y física de los datos persistentes, de los cuales se hará uso para el desarrollo y funcionamiento del sistema propuesto. Se utiliza para definir la correlación entre las clases de diseño y las estructuras de datos. A continuación se representa el nuevo modelo de datos obtenido, que gestiona toda la información referente al proceso de mantenimiento, identificando la relación entre cada una de las entidades.

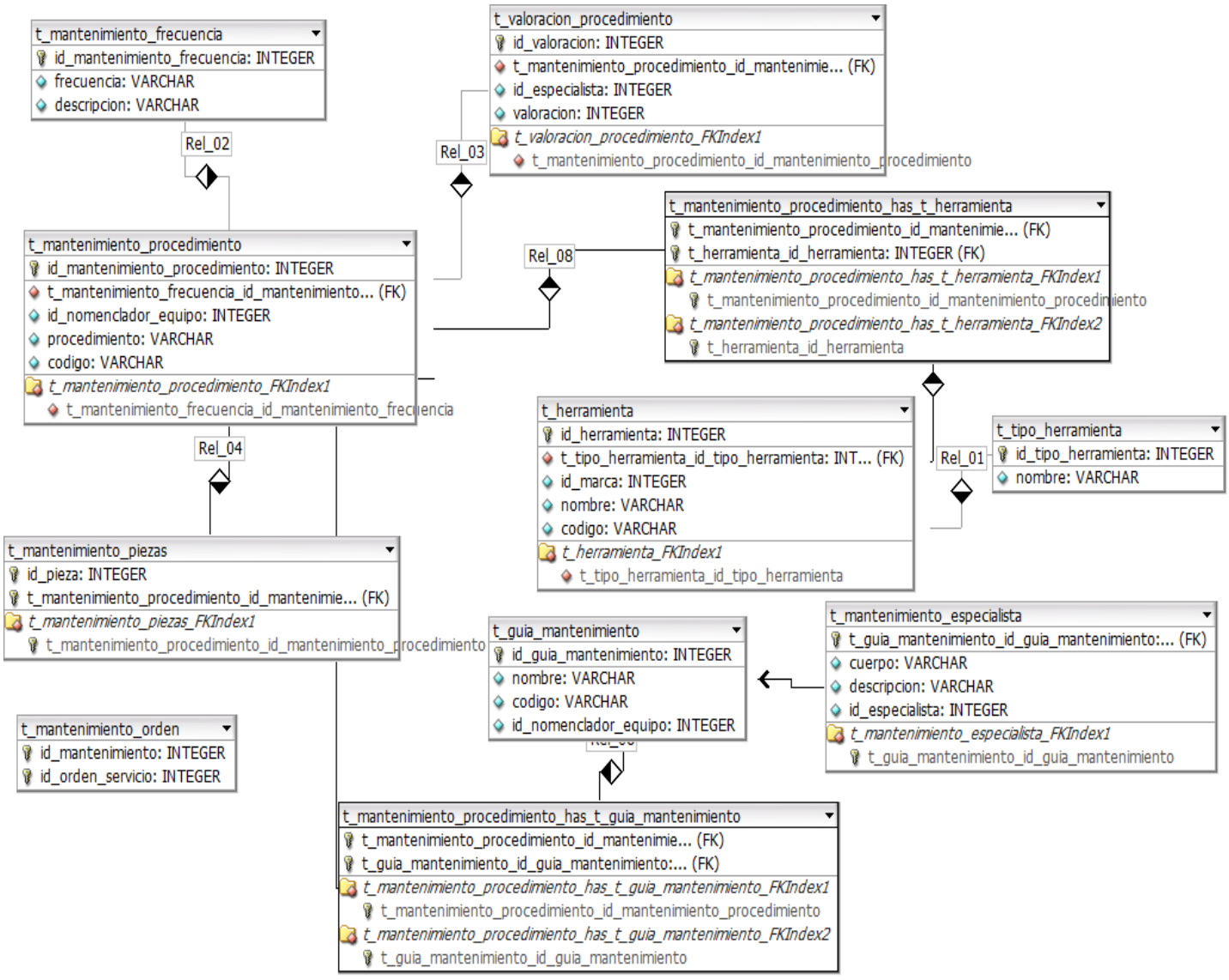

Figura 18. Modelo de Datos

3.4 Breve valoración de las técnicas de validación para la base de datos

Integridad

La integridad es uno de los factores más importantes a la hora de realizar el diseño de una Base de Datos. Esta se divide en varios aspectos como son:

- **Integridad referencial:** garantiza interrelaciones válidas entre entidades. Implica que los datos sean correctos, sin duplicaciones, pérdida de datos o relaciones mal resueltas. Todas las bases de datos relacionales incluyen esta propiedad pues el software gestor es responsable de su cumplimiento.
- **Integridad de Dominio:** la integridad de dominio viene dada por la validez de las entradas de los datos para una columna determinada. Se puede exigir la integridad de dominio para restringir el tipo mediante tipos de datos, el formato mediante reglas y restricciones CHECK, o el intervalo de valores posibles mediante restricciones FOREIGN KEY, definiciones DEFAULT, definiciones NOT NULL.
- **Integridad de la Entidad:** define una fila como entidad única para una tabla determinada. La integridad de entidad, exige la integridad de las columnas de los identificadores o la clave principal de una tabla, mediante índices y restricciones UNIQUE, o restricciones PRIMARY KEY.

Restricción de PRIMARY KEY (llave primaria)
Toda entidad debe tener una llave primaria la cual identifica una tupla unívocamente de las demás.

Garantizar la integridad y confidencialidad de la información es fundamental para una buena elaboración de un sistema. En la solución desarrollada esto se lleva a cabo mediante el tratamiento de errores, el cual permite evitar que sucedan situaciones indeseadas que se conviertan en problemas mayores y afecten la funcionalidad de la aplicación. Con la validación de los datos enviados por los usuarios al servidor se evitan muchos de estos errores; la validación de formularios se puede realizar tanto en el servidor como en el cliente. En el servidor es obligatoria para no corromper la base de datos con datos incorrectos y en el lado del cliente es opcional, pero mejora la experiencia de usuario.

Para minimizar estos errores se cuenta con cuadros de opción (ver Figura 19) y menú de selección que facilita la entrada de datos por parte del usuario al sistema.

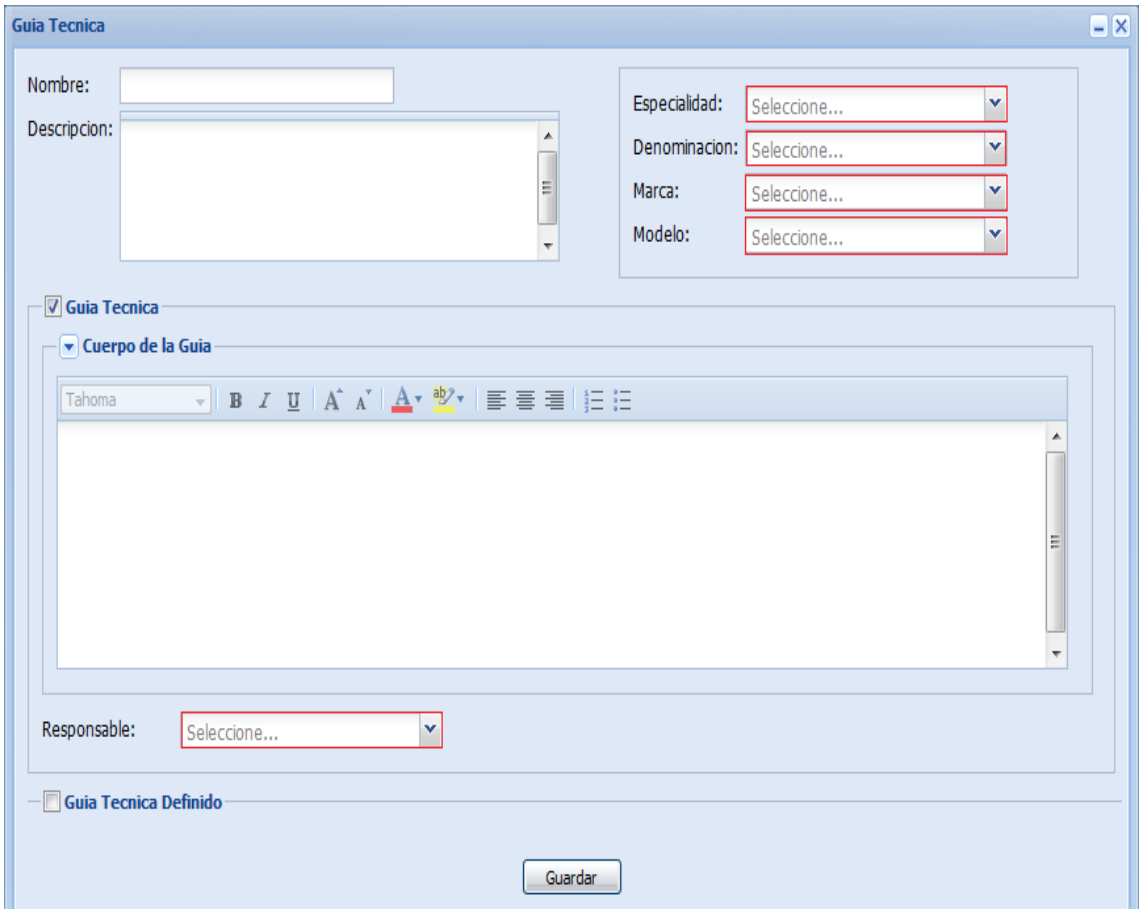

Figura 19. Ejemplo de cuadros de selección utilizado

Normalización de la Base de datos

La normalización es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y estables, son más fáciles de mantener. También se puede entender la normalización, como una serie de reglas que sirven para ayudar a los diseñadores de bases de datos a desarrollar un esquema, que minimice los problemas de lógica. Cada regla está basada en la que le antecede. La normalización se adoptó porque el viejo estilo de poner toda la información en un solo lugar, como un archivo o una tabla de la base de datos, era ineficiente y conducía a errores de lógica cuando se trataban de manipular la información. (31)

Existen básicamente tres niveles de normalización: Primera Forma Normal (1NF, por sus siglas en inglés), Segunda Forma Normal (2NF, por sus siglas en inglés) y Tercera Forma Normal (3NF, por sus siglas en inglés). Cada una de estas formas tiene sus propias reglas. Cuando una base de datos se conforma a un nivel, se considera normalizada a esa forma de normalización.

Primera Forma Normal

La regla de la Primera Forma Normal establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas. (32)

Segunda Forma Normal

La regla de la Segunda Forma Normal establece que todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas. Una dependencia parcial es un término que describe a aquellos datos que no dependen de la llave primaria de la tabla para identificarlos. (33)

Tercera Forma Normal

Una tabla está normalizada en Tercera Forma Normal si está en (2NF) y todas las columnas que no son llave son funcionalmente dependientes por completo de la llave primaria y no hay dependencias transitivas. Una dependencia transitiva es aquella en la cual existen columnas que no son llave que dependen de otras columnas que tampoco son llave. (34)

La base de datos del módulo en cuestión se encuentra en Tercera Forma Normal, con lo que se previenen errores de lógica cuando se insertan o borran registros. Cada columna en una tabla está identificada de manera única por la llave primaria y no debe haber datos repetidos. Esto provee un esquema limpio y elegante, que es fácil de trabajar y expandir.

3.5 Descripción de las tablas de la base de datos

A continuación serán expuestas algunas de las tablas de la base de datos con sus atributos para lograr un mejor entendimiento de la forma de almacenar la información en el sistema.

Nombre: t_mantenimiento_frecuencia

Descripción: Almacena información referente a todas las posibles frecuencias en las que se pueda realizar un mantenimiento.		
Atributo	**Tipo**	**Descripción**
id_mantenimiento_frecuencia(PK)	integer	Identificador único para las tuplas.
frecuencia	varchar	Nombre de la frecuencia.
descripción	varchar	Descripción de la frecuencia.

Tabla 8. Descripción de tabla Mantenimiento Frecuencia

Nombre: t_herramienta		
Descripción: Almacena información referente a todas las herramientas utilizadas en la realización de un determinado mantenimiento.		
Atributo	**Tipo**	**Descripción**
id_herramienta (PK)	integer	Identificador único para las tuplas.
id_tipo_herramienta(FK)	integer	Identificador asociado al tipo de herramienta.
id_marca(FK)	interger	Identificador asociado a la marca.
nombre	varchar	Nombre de la herramienta.
código	varchar	Código único asociado a cada herramienta.

Tabla 9. Descripción de tabla Herramienta

Nombre: t_mantenimiento_procedimiento		
Descripción: Almacena información referente a todos los procedimientos que intervienen en el proceso de mantenimiento.		
Atributo	**Tipo**	**Descripción**
id_mantenimiento_procedimiento (PK)	integer	Identificador único para las tuplas.
id_mantenimiento_frecuencia(FK)	integer	Identificador asociado a la frecuencia.
id_nomenclador_equipo(FK)	interger	Identificador asociado al nomenclador de equipo.
procedimiento	varchar	Nombre del procedimiento.
código	varchar	Código único asociado a cada procedimiento de mantenimiento.

Tabla 10. Descripción de tabla Procedimiento de Mantenimiento

Nombre: t_guia_mantenimiento		
Descripción: Almacena información referente a todas las guías técnicas que intervienen en el proceso de mantenimiento.		
Atributo	**Tipo**	**Descripción**
id_guia_mantenimiento(PK)	integer	Identificador único para las tuplas.
id_nomenclador_equipo(FK)	interger	Identificador asociado al nomenclador de equipo.
nombre	varchar	Nombre de la guía técnica.
código	varchar	Código único asociado a cada guía técnica.

Tabla 11. Descripción de tabla Guía de Mantenimiento

Nombre: t_guia_mantenimiento_especialista		
Descripción: Almacena información referente a todas las guías técnicas definidas por los especialistas.		
Atributo	**Tipo**	**Descripción**
id_guia_mantenimiento(PK)	integer	Identificador único para las tuplas.
id_especialista(FK)	integer	Identificador asociado al especialista encargado.
cuerpo	text	Pasos a seguir para la realización de un mantenimiento.
descripción	varchar	Descripción de la guía técnica definida por el especialista.

Tabla 12. Descripción de tabla Guía de Mantenimiento del Especialista

Nombre: t_valoracion_procedimiento		
Descripción: Almacena información referente a todas las valoración realizadas sobre un determinado procedimiento.		
Atributo	**Tipo**	**Descripción**

id_valoracion(PK)	integer	Identificador único para las tuplas.
id_mantenimiento_procedimiento(FK)	integer	Identificador asociado procedimiento de mantenimiento.
id_especialista(FK)	integer	Identificador asociado al especialista que realizó la valoración.
valoración	integer	Valor numérico sobre la calidad del procedimiento.

Tabla 13. Descripción de tabla Valoración de Procedimiento

3.6 Vista de Implementación

El modelo de implementación representa la composición física de la implementación en términos de subsistemas de implementación, y elementos de implementación (directorios y archivos, incluyendo código fuente, datos y archivos ejecutables). (35)

Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente o de otro tipo (36). Teniendo en cuenta la utilización del MVC y la división de los archivos y funcionalidades que implementa el Framework Symfony, se han definido tres paquetes de componentes generales: la Vista, el Controlador y el Modelo, donde se agrupan otros componentes más pequeños en dependencia de las funciones que realizan.

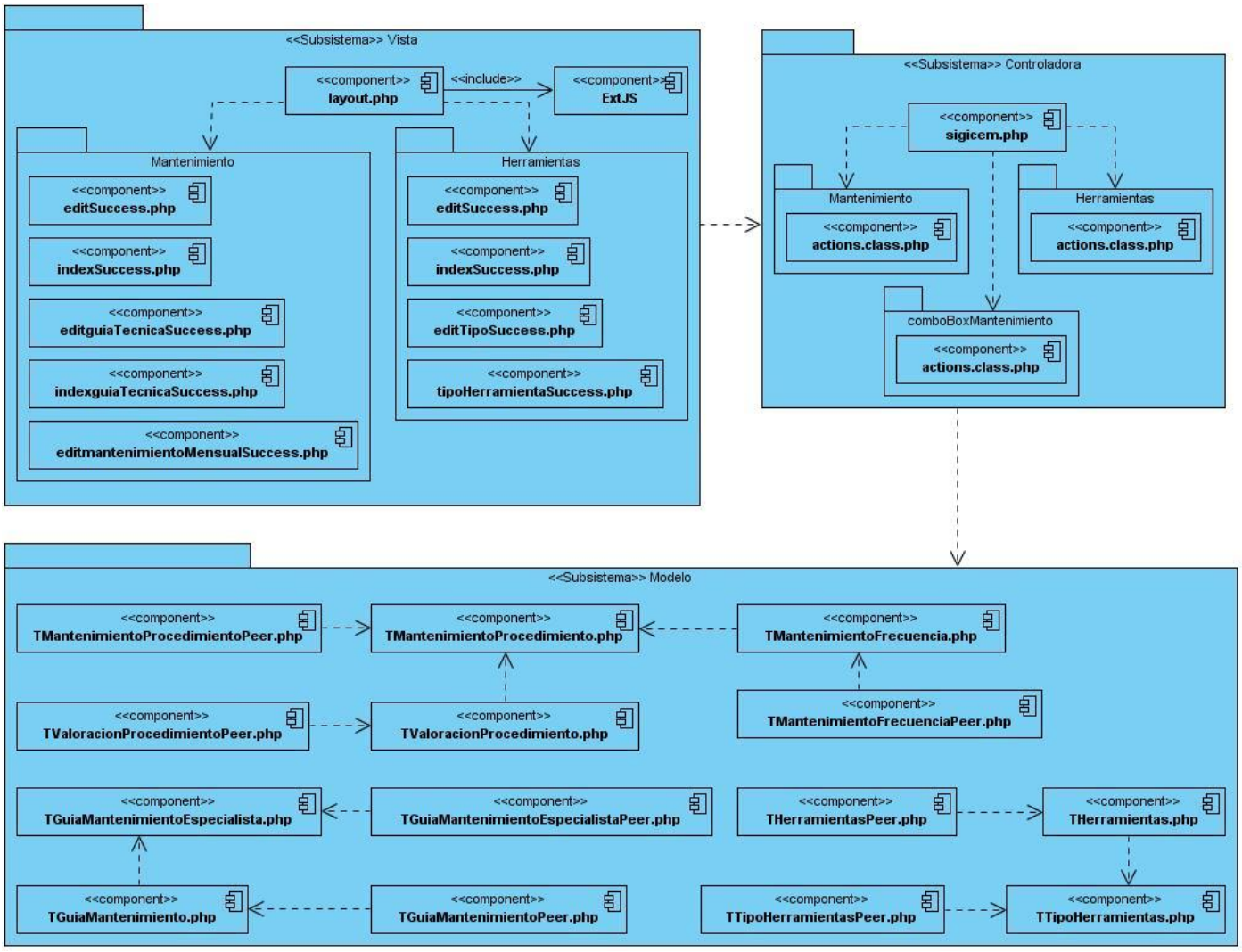

Figura 20. Diagrama de componentes módulo de Mantenimiento

En este capítulo se realizó una valoración del diseño propuesto por el analista del sistema refinándose los diagramas de clases del diseño y los diagramas de interacción, que permiten obtener el diseño de la interacción de los componentes y una mayor comprensión de la solución propuesta. Se obtuvo el modelo de implementación, el cual permitió detallar las organizaciones y dependencias lógicas de cada uno de sus componentes. Con la actualización del modelo de datos y la descripción de las nuevas tablas, se logró un mejor entendimiento de la forma de almacenar la información en el sistema.

Conclusiones

Con la realización del presente trabajo se concluye que:

- El estudio de las diferentes técnicas de programación, posibilitó seleccionar la más idónea a utilizar en la implementación del módulo desarrollado.
- El análisis del proceso de gestión de la información de los mantenimientos de equipos médicos permitió establecer el punto de partida para la especificación de los requerimientos que debía cumplir el módulo a desarrollar.
- A partir de la descripción de la arquitectura se obtuvo la base sobre la cual estaría soportado el funcionamiento del sistema, con el fin de obtener la propuesta de solución que más se adecue a las necesidades del cliente.
- La solución desarrollada garantiza la gestión de la información relacionada al mantenimiento de equipos médicos, contribuyendo así a elevar los niveles de disponibilidad técnica de los mismos.

Recomendaciones

Luego de haber concluido el presente trabajo se recomienda:

- Realizar la funcionalidad Predecir Mantenimiento.

Referencias Bibliográficas

1. Informática Salud. [En línea] [Citado el: 10 de mayo de 2013.] www.informatica2013.sld.cu/index.php/informaticasalud/2013/paper/download/166/126.

2. Emagister. [En línea] [Citado el: 10 de mayo de 2013.] http://www.emagister.com/curso-ingenieria-mantenimiento-hospitalario/historia-mantenimiento.

3. **Dr. José Ángel Córdova Villalobos.** PROGRAMA DE ACCIÓN ESPECÍFICO. [En línea] [Citado el: 4 de Junio de 2013.] http://www.cenetec.salud.gob.mx/descargas/PAES/PEDM.pdf.

4. **García, Gilberto.** *Metodología de la investigación educacional. Primera parte de Gastón Pérez.* La Habana : Pueblo y Educación, 1996.

5. Tipos De Mantenimiento. [En línea] [Citado el: 18 de marzo de 2013.] http://www.mitecnologico.com/Main/TiposDeMantenimiento\.

6. Guia. [En línea] [Citado el: 10 de abrir de 2013.] http://www.definicionabc.com/general/guia.php.

7. Procedimiento. [En línea] [Citado el: 11 de abrir de 2013.] http://www.slideshare.net/SagaJasper/definiciones-15256489.

8. **Curto, Josep.** *Information Management.* [En línea] 28 de Noviembre de 2006. [Citado el: 19 de marzo de 2013.] http://informationmanagement.wordpress.com/category/gestion/gestion-de-la-informacion..

9. Revistas Médicas Cubanas. [En línea] [Citado el: 12 de mayo de 2013.] http://bvs.sld.cu/revistas/aci/vol18_1_08/aci07708.htm.

10. Paradigmas de programacion. [En línea] [Citado el: 12 de mayo de 2013.] http://wozgeass.wordpress.com/2010/01/10/paradigmas-de-programacion/.

11. Informática. [En línea] [Citado el: 20 de marzo de 2013.] http://www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/Prog_Obj01.html.

12. Biblioteca de la Universidad Nacional del Santa. [En línea] [Citado el: 10 de marzo de 2013.] http://biblioteca.uns.edu.pe/saladocentes/archivoz/curzoz/programacion_orientada_a_objetos.pdf.

13. DICCIONARIO INFORMÁTICO. [En línea] [Citado el: 15 de marzo de 2013.] http://www.lawebdelprogramador.com/diccionario/mostrar.php?letra=A.

14. **Alfaro, Félix Murillo. Blearning.** [En línea] 08 de Julio de 2011. [Citado el: 21 de marzo de 2013.] http://blearning.itmina.edu.mx/dep/sada/carreras/Ingenieria%20Electronica/3er%20Semestre/Programacion%201/Programacion1_IE/unidad1.pdf..

15. Unidad Informática. [En línea] [Citado el: 16 de marzo de 2012.] http://www.sites.upiicsa.ipn.mx/polilibros/portal/Polilibros/P_terminados/Ingenieria_de_software/polilibro/capitulos/unidad_4/4_3.htm.

16. Unidad Docente de Ingeniería del Software . [En línea] [Citado el : 16 de marzo de 2013.] is.ls.fi.upm.es/docencia/proyecto/docs/curso/04CursoOO_herencia.doc.

17. Programación Estructurada . [En línea] [Citado el: 2 de abril de 2013.] http://programacion-estructurada.wikispaces.com/.

18. Universidad Técnica de Manabí. [En línea] [Citado el: 2 de abril de 2013.] http://www.sisman.utm.edu.ec/libros/FACULTAD DE CIENCIAS ZOOTC389CNICAS/CARRERA DE INGENIERIA EN INFORMATICA AGROPECUARIA/03/ESTRUCTURAS DE DATOS I/FUNDAMENTOS DE PRODUCCION ALGORITMOS Y ESTRUCTURA 0DE DATOS..

19. **Pressman, Roger S.** *Ingeniería de software, un enfoque práctico.* Madrid : Biblioteca Especializada, 2005.

20. **Aragón, Ismaray Morera.** *Módulo de Reportes Estadísticos del Sistema de Gestión para la Ingeniería Clínica y Electromedicina.* Sistemas de Apoyo a la Salud, Universidad de las Ciencias Informáticas. La Habana : s.n., 2011. pág. 18, Tesis de pregrado.

21. **Eguiluz, Javier.** Symfony. [En línea] 5 de Octubre de 2008. [Citado el: 23 de marzo de 2013.] http://www.symfony.es/2009/10/05/netbeans-ya-incluye-soporte-para-symfony..

22. **Potencie, Fabien y Zaninotto, François.** *Symfony 1.2, la guía definitiva.* 2008. pág. 425.

23. **Sommerville, Ian.** *Ingeniería del software.* Madrid : Pearson Educación, 2005.

24. EducaNet. [En línea] [Citado el: 3 de mayo de 2013.] http://www.educanet.ec/cursos-online-gratis/139-cursos-presenciales-programacion/php-avanzado-modelo-mvc/540-acerca-de-php-mvc.

25. **Gonzalez, Omar.** [En línea] 5 de Octubre de 2011. [Citado el: 24 de marzo de 2013.] http://www.scribd.com/doc/53315954/Patrones-de-Diseno-GRASP.

26. PATRONES GRASP (Patrones de Software para la asignación General de Responsabilidad). [En línea] [Citado el: 11 de marzo de 2013.] http://jorgesaavedra.wordpress.com/2007/05/08/patrones-grasp-patrones-de-software-para-la-asignacion-general-de-responsabilidadparte-ii..

27. **ZANINOTTO, F. y POTENCIER, F.** *Symfony 1.1, la guía definitiva.* 2009.

28. Symfony. [En línea] [Citado el: 12 de marzo de 2013.] http://www.symfony-project.org/plugins/sfDoctrineGuardPlugin.

29. TIC. [En línea] [Citado el: 10 de mayo de 2013.] http://indalog.ual.es/mtorres/LP/FundamentosDiseno.pdf.

30. *Rational Unified Process. Extended Help for RUP.* 2003.

31. Escuela de Educación Secundaria Técnica Nº 2. [En línea] [Citado el: 24 de marzo de 2013.] http://www.eet2mdp.edu.ar/alumnos/MATERIAL/MATERIAL/info/infonorma.pdf.

32. Instituto Tecnológico Superior de Lerdo. [En línea] [Citado el: 11 de mayo de 2013.] http://www.slideshare.net/Sircomun/normalizacin-presentation.

33. Universidad Autónoma del Estado de Hidalgo. [En línea] [Citado el: 12 de mayo de 2013.] http://virtual.uaeh.edu.mx/repositoriouaeh/paginas/Normalizacion de Base de 0Datos/segunda_forma_normal_2fn.html.

34. Slidershare. [En línea] [Citado el: 12 de mayo de 2013.] http://www.slideshare.net/HANZOSLASHER/normalizacion-de-base-de-datos.

35. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Adison-Wesley.

36. Modelo de implementación. Diagramas de componentes y Despliegue. [En línea] [Citado el: 25 de marzo de 2013.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

Bibliografía

Alfaro, Félix Murillo. Blearning. [En línea] 08 de Julio de 2011. [Citado el: 21 de marzo de 2013.] http://blearning.itmina.edu.mx/dep/sada/carreras/Ingenieria%20Electronica/3er%20Semestre/Programacion%201/Programacion1_IE/unidad1.pdf.

Aragón, Ismaray Morera. *Módulo de Reportes Estadísticos del Sistema de Gestión para la Ingeniería Clínica y Electromedicina.* Sistemas de Apoyo a la Salud, Universidad de las Ciencias Informáticas. La Habana : s.n., 2011. pág. 18, Tesis de pregrado.

Castro Alianny Valdivia y Betanco Alvarez, Nuria Isabel. Sistema para la Cooperación Médica: Módulo de Administración. Universidad de las Ciencias Informáticas. La Habana: s.n, 2011, Tesis de pregrado.

Córdova Villalobos, Dr. José Ángel. Programa de acción específico 2007-2012. Gestión de Equipo Médico. [En línea] [Citado el: 4 de junio de 2013] http://www.cenetec.salud.gob.mx/descargas/PAES/PEDM.pdf.

Curto, Josep. *Information Management.* [En línea] 28 de Noviembre de 2006. [Citado el: 19 de marzo de 2013.] http://informationmanagement.wordpress.com/category/gestion/gestion-de-la-informacion.

Definición. [En línea] [Citado el: 18 de marzo de 2013.] http://definicion.de/procedimiento.

Desarrollo en web [En línea] 22 de Octubre de 2008 [Citado el: 29 de enero de 2013.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo.

Eguiluz, Javier. Symfony. [En línea] 5 de Octubre de 2008. [Citado el: 23 de marzo de 2013.] http://www.symfony.es/2009/10/05/netbeans-ya-incluye-soporte-para-symfony.

Escuela de Educación Secundaria Técnica Nº 2. [En línea] [Citado el: 24 de marzo de 2013.] http://www.eet2mdp.edu.ar/alumnos/MATERIAL/MATERIAL/info/infonorma.pdf.

Eva. [En línea] 7 de Julio de 2011. [Citado el: 10 de enero de 2013.] http://eva.uci.cu/file.php/104/Tema_3/Bibliografia/Diseno_metodol_de_la_invest-poblacion_y_muestra-_Metodos_y_diseno_experimental_-_Tema_3.pdf.

ExtJS [En línea]. http://www.extjs.com.

Gonzalez, Omar. [En línea] 5 de Octubre de 2011. [Citado el: 24 de marzo de 2013.] http://www.scribd.com/doc/53315954/Patrones-de-Diseno-GRASP.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software.* s.l. : Adison-Wesley.

Lozano Letvin R. Diagramación y programación. México: McGraw Hill.

Martínez, Juan Francisco Javier. Eva. [En línea] 14 de Junio de 2011. [Citado el: 25 de marzo de 2013.] http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_1/Diseno_de_software/Patrones/Decorador.pdf.

Modelo de implementación. Diagramas de componentes y Despliegue. [En línea] [Citado el: 25 de marzo de 2013.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

Monografias. [En línea] [Citado el: 20 de marzo de 2013.] http://www.monografias.com/trabajos/objetos/objetos.shtml.

Monografías. [En línea] [Citado el: 22 de marzo de 2013.] http://www.monografias.com/trabajos/progestructu/progestructu.shtml.

NetBeans IDE. [En línea] [Citado el: 27 de febrero de 2013.] http://netbeans.org/features/index.html.

NetBeans ya incluye soporte para Symfony [En línea] [Citado el: 26 de marzo de 2013.] http://www.symfony.es/2009/10/05/netbeans-ya-incluye-soporte-para-symfony/.

Norton Peter. Introducción a la computación. México : McGraw Hill.

PATRONES GRASP (Patrones de Software para la asignación General de Responsabilidad). [En línea] [Citado el: 11 de marzo de 2013.] http://jorgesaavedra.wordpress.com/2007/05/08/patrones-grasp-patrones-de-software-para-la-asignacion-general-de-responsabilidadparte-ii.

PHP [En línea]. http://www.php.net.

PhpMyadmin [En línea] 2010. [Citado el: 28 de febrero de 2013.] http://www.phpmyadmin.net/home_page/index.php.

Potencie, Fabien y Zaninotto, François. *Symfony 1.2, la guía definitiva.* 2008. pág. 425.

Pressman, Roger S. *Ingeniería de software, un enfoque práctico.* Madrid : Biblioteca Especializada, 2005.
Rational Unified Process [En línea] [Citado el: 15 de diciembre de 2012.] http://www.rational.com.ar/herramientas/rup.html.

Rational Unified Process. 2003.

Sanders Donal H. Informática presente y futuro. México: McGraw Hill.

Sommerville, Ian. *Ingeniería del software.* Madrid : Pearson Educación, 2005.

Symfony. [En línea] [Citado el: 12 de marzo de 2013.] http://www.symfony-project.org/plugins/sfDoctrineGuardPlugin.

Técnicas de programación. [En línea] [Citado el: 17 de febrero de 2013.] http://catedraprogramacion.foroactivos.net/t117-que-es-tecnica-de-programacion.

Teleformacion. [En línea] [Citado el: 18 de febrero de 2013.] http://teleformacion.edu.aytolacoruna.es/PASCAL/document/modular.html.

Tipos De Mantenimiento. [En línea] [Citado el: 18 de marzo de 2013.] http://www.mitecnologico.com/Main/TiposDeMantenimiento\.

Visual paradigm [En línea]. 13 de Marzo de 2012. [Citado el: 28 de febrero de 2013.] http://www.visual-paradigm.com/product/vpuml.

Glosario de términos

Caso de uso: es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

Diagrama: gráfico que representa en forma esquematizada información relativa e inherente a algún tipo de ámbito. Los diseñadores las usan para organizar sus ideas e ilustrar el concepto de un proyecto.

Framework: Se conoce como marco de trabajo y constituye un conjunto de conceptos, metodologías y herramientas de administración y diseño para el desarrollo de forma estandarizada de una aplicación.

Herramientas CASE: Herramientas de Ingeniería Asistida por Computadora, del inglés *"Computer Aided Software Engineering"* (CASE, por sus siglas en inglés), son aquellas que brindan un entorno de trabajo a todo el equipo de desarrollo de software con el uso de las tecnologías de la información. Permite elevar la productividad durante todo el ciclo de vida del producto y la estructuración del mismo en leguaje UML.

MINSAP: Ministerio de Salud Pública.

Notación PascalCasing: Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, cada palabra con letra mayúscula.

RNF: Requerimientos No Funcionales.

SNS: Sistema Nacional de Salud.

Software: conjunto de los programas de cómputo, procedimientos, regla, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

TIC: Tecnologías de la Información y las Comunicaciones.

UML: Lenguaje Unificado de Modelado.