Universidad de las Ciencias Informáticas

Facultad 7

Título: Implementación del módulo Registro de Aseguramiento para Diálisis perteneciente a la Red Cubana de Nefrología

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

AUTORES:

Adilson Victor Jorda Orraca

Jorge Carlos Iglesias Alvarez

TUTORES:

Ing. Yanersy Díaz Colomé

Ing. Renier Ricardo Figueredo

ASESOR:

Lic. Hugo Vargas Calzado

Ciudad de la Habana

Junio del 2008

Declaración de Auditoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los 26 días del mes de junio del año 2008.

Adilson Victor Jorda Orraca

_____

Firma del Autor

Ing. Renier Ricardo Figueredo

_____

Firma del Tutor

Jorge Carlos Iglesias Alvarez

_____

Firma del Autor

Ing. Yanersy Díaz Colomé.

_____

Firma del Tutor

La educación es el gasto para la defensa más efectivo que existe.

Kofi Annan

DEDICATORIA

Antes que nada, esta tesis no existiera sin mis padres, a ellos dos Julia Orraca Mitjans y Fulgencio Jorda Valdés mis más sinceros respetos y gracias les doy desde el corazón de un hijo en el cuál han puesto la fé, y mientras me quede un gota de sangre les estaré agradecido por siempre.

Agradecer a Adelys Jorda Orraca, a Adis Marilda, ellas son la esperanza y el corazón de esta mi gran familia. A mi abuelos a todos, los que puedo ver y los que no, a ellos que se que me pueden ver , y estarían orgullosos de verme .

A Juana Barbara, la barbie de la familia, a sus hijos que son mis hermanos junto con mi Adelys, a Leysi y Lewys que los quiero con la vida, y se que esperan mucho de mi, como yo de ellos, desde ahora decirles desde lo mas profundo agradezco a dios por ser su familia y no los defraudare.

Por ultimo pero no menos, quisiera agradecer a Evelyn Ramirez Armenteros, mi chica, quien ha estado a mi lado con la intensidad de un sol de cariño y ternura, quien me ha alentado hasta el final y me ha brindado su ayuda incondicionalmente. Te amo.

No quiero terminar sin agradecer a Jorge Carlos, compañero de tesis y amigo, a todos los que han tratado de estar a mi lado en buenas y malas , lo que han soportado este apocalipsis de persona y lo han sobrevivido para contarlo , al "Yoe", a Hector Rojas, a Maurich, a Obert, al Team-Cuba(Hembras y Varones) .

Adilson Victor Jorda Orraca.

Nunca pensé que llegaría el momento de agradecer a todos los que me han ayudado siempre en esta larga vida de estudiante que comenzó en Camagüey siendo muy pequeño y culmina muy lejos, en la Universidad de las Ciencias Informática (UCI), en La Habana, donde por fin se ve el fruto de todo el esfuerzo realizado en estos años, donde se llega a la cúspide de todo estudiante, convertirse en un graduado universitario.

Ha sido mucho el tiempo y las personas desde los comienzos y no se si sea capaz de tan solo con unas líneas reflejar todo el agradecimiento que merecen y que quiero darles, en especial a mi familia la que más cerca ha estado de mi y la que en estos últimos años ha estado más lejos, a mis amigos del barrio con los que junto crecí y a toda esta gran familia que es la UCI, a los que conocí ya hace cinco años y hoy puedo decir que son mis amigos, a todos, por haber compartido conmigo tantos momentos, muchas gracias.

Jorge Carlos Iglesias Alvarez.

A todos nuestros familiares y amigos.

En especial a nuestro Comandante en Jefe, el creador de esta obra de la Revolución. Por convertir en realidad los sueños de miles de estudiantes cubanos.

Resumen

En la actualidad, para lograr la operatividad y calidad que el país necesita en los servicios de Nefrología, se deben corregir algunos problemas y controlar más los cuantiosos recursos que son asignados a esta actividad.

Por todo lo antes mencionado, el presente trabajo de diploma tiene como objetivo la Implementación de un sistema informático que mejore el proceso de gestión de la información relacionada con el aseguramiento complementario para la realización de las diálisis en los servicios nefrológicos del país.

En el desarrollo se utilizó PHP5 como lenguaje de programación, Zend Studio como IDE de desarrollo integrado, con MySql como gestor de base de datos y corriendo como Servidor Web, Apache.

Con el uso de la aplicación se obtendrá una eficiente gestión del equipamiento e insumos necesarios para realizar el proceso de diálisis en cada uno de los servicios y una reducción de los gastos en insumos propios de la especialidad de Nefrología en el país.

Palabras Claves

Equipos no Médicos. Hemodiálisis.Nefrología.Aseguramientos para Diálisis. Gestión de información. Equipos Médicos.

Contenido

Introducción

Fundado en noviembre de 1966, el Instituto de Nefrología Dr. Abelardo Buch López está dedicado a la investigación, la docencia, la asistencia médica calificada de las enfermedades renales y allí radica además, la Dirección Nacional de Atención al Programa Enfermedad Renal, Diálisis y Trasplante.

Actualmente en el país, existen 47 servicios de nefrología, diseminados en las 14 provincias y el municipio especial, equipados con la más alta tecnología para desarrollar el tratamiento nefrológico de forma eficiente y con calidad, todos con el precepto de acercarlos a donde viven los pacientes con afecciones renales en terapia de remplazo dialítico de la función renal. Esto ha posibilitado que los pacientes que antes se tenían que mover a grandes distancias hoy no lo hagan y encuentren los servicios más cercanos.

Cuba dentro del marco del desarrollo alcanzado en la salud pública, comparte con el mundo desarrollado muchos de los indicadores del mismo, entre ellos un elevado número de pacientes diabéticos, hipertensos, cardiópatas complicados y cada vez más una población envejecida.

Los anteriores son las principales causas de enfermedad renal crónica con requerimientos de diálisis en el mundo y en nuestro país, de ahí que se realicen esfuerzos a nivel de la asistencia primaria, Consultorios del Médico de la Familia y en los policlínicos para detectar a tiempo estas enfermedades crónicas no trasmisibles y dentro de ellas las renales, para evitar que se compliquen y progresen hasta la necesidad de diálisis y trasplante renal.

Pese a existir el Registro de Equipos Médicos y no Médicos del Sistema Informatizado para la Salud (SISalud), estos gestionan solo las generalidades y no los datos específicos acorde a las necesidades de los servicios de nefrología y en particular de la gestión de diálisis.

Por otra parte, el control del reuso de los dializadores se lleva hoy en formato duro, lo que trae como consecuencia que sea difícil gestionar la información necesaria para seguir el reuso de los dializadores de un paciente en hemodiálisis, donde la interrogante: ¿Cómo mejorar el proceso de gestión de la información relacionada con el aseguramiento complementario para la realización de las diálisis en los servicios nefrológicos del país?, es el **Problema a Resolver**.

Luego, este problema se enmarca en el **Objeto de Estudio**: Proceso de Gestión de Diálisis en Cuba. El **Campo de Acción** abarcado es: Proceso de gestión de aseguramientos complementarios para la realización de hemodiálisis en los servicios nefrológicos de Cuba.

El **Objetivo General** sería: Implementar un sistema informático que mejore el proceso de gestión de la información relacionada con el aseguramiento complementario para la realización de las diálisis en los servicios nefrológicos del país.

Como tareas para darle cumplimiento al objetivo se plantean:

1. Realizar un análisis crítico y valorativo de la técnica de programación, lenguaje, plataforma y librerías propuestas por el cliente y la universidad, así como de otras tecnologías más utilizadas en el desarrollo de aplicaciones Web.
2. Obtener el Modelo de Implementación y los artefactos necesarios que describan la Base de Datos.
3. Implementar el módulo utilizando los patrones de diseño establecidos por los analistas.
4. Hacer el diseño de la Base de Datos.
5. Brindar una interfaz gráfica orientada al usuario.

El presente trabajo consta de tres capítulos, como se presenta a continuación:

CAPÍTULO 1: Contiene la fundamentación teórica del tema tratado en la investigación. Se describen los conceptos fundamentales asociados al dominio del problema. Se hace una descripción de las tendencias y tecnologías actuales que se utilizaron como soporte de la propuesta.

CAPÍTULO 2: Contiene los elementos de arquitectura del tema tratado en la investigación. Se incluyen los requisitos no funcionales del sistema, así como los estándares de codificación utilizados, la vista de despliegue y de componentes del sistema.

CAPÍTULO 3: Contiene la descripción y el análisis de la solución propuesta, la estrategia de integración con otros módulos del Sistema de Información para la Salud (SISalud), como es el REM, así como la descripción de las clases.

Capítulo 1 Fundamentación Teórica.

Este capítulo está dedicado a realizar un análisis detallado de las distintas técnicas de programación que existen a nivel internacional, nacional y de la Universidad. Incluye las tendencias, técnicas, tecnologías, metodologías relacionadas con dichas técnicas así como de las plataformas de desarrollo que la soportan. Hace estudio de las técnicas de programación, plataforma y librerías usadas para el desarrollo de este módulo del proyecto.

1.1 Estado del Arte

Existen algunos sistemas a nivel internacional y nacional que responden a la situación problémica del presente trabajo y a la mayoría de los intereses del cliente, pero todos estos sistemas tienen inconvenientes por lo que no son utilizados por el sistema a desarrollar. Ejemplos de estos sistemas:

SISDIA. Este sistema a pesar de encontrarse disponible a través de Internet, favoreciendo así la administración del mismo a distancia, y de mantener un registro constante de la información referente a los pacientes, así como el uso que se le da a los dializadores; hay que realizar un pago de una licencia para adquirir el sistema para ser instalado de forma local en el centro o institución correspondiente.

Nefronet. Es un sistema de información con tecnología eXPlender Cliente Servidor de Tres Capas, con interfaz de escritorio o web (según las necesidades del usuario).
Información Básica que maneja el Sistema:
• Registro de Centros de Diálisis.
• Registro de Interrupción del Tratamiento.
• Registro de Profesionales.
• Generación de facturación por cada cobertura.
• Generación de soporte de facturación para enviar a los Nodos Provinciales por correo electrónico u otro medio. El sistema esta diseñado para países donde la medicina es un negocio más, es decir donde está privatizada, por lo que es capaz de generar la factura para el cobro a los pacientes.

Nefrolink. Es un Sistema de Información Renal que permite mantener por paciente un repositorio digital capaz de sustituir al 100% el papel, accesible para cualquier usuario

autorizado (según permisos), facilitando así la actividad asistencial y minimizar la carga administrativa en la atención a los pacientes renales. Dentro de sus objetivos generales se encuentra el de mantener un expediente clínico electrónico único por paciente, así como el apoyo a la investigación y el desarrollo del conocimiento.

Nefrosoft. Es una aplicación informática para la gestión clínica de una Unidad de Hemodiálisis. Realizada en el sistema operativo Microsoft Windows, utilizando la base de datos Microsoft Access. Compatible con Windows 95 / 98 / Me / NT4 / 2000 / XP. (7)

SINTRA (Sistema Nacional de Información de Procuración y Trasplante de la República Argentina)

• Fue desarrollado con tecnologías web de gran confiabilidad y flexibilidad en todos sus niveles (Linux, Oracle, J2EE). El hardware y software utilizado fue seleccionado y adquirido para uso exclusivo y se encuentra en la sala de servidores del INCUCAI (Instituto Nacional Centro Unico Coordinador de Ablación e Implante).

• Su diseño prioriza la confiabilidad y seguridad de la información, y se utilizan todas las medidas necesarias para lograrlo, como la autenticación de acceso mediante cuentas de usuario y la implementación de sesión segura.

• Siendo una aplicación web, su utilización es posible desde cualquier computadora que tenga acceso a Internet, sin la necesidad de instalar o configurar nada localmente. Sin embargo, el servicio de conexión a Internet se convierte en un factor fundamental en el rendimiento y el uso del sistema.

• La arquitectura del sistema SINTRA fue diseñada para funcionar las 24 horas del día los 365 días del año, y cuenta con las medidas de seguridad necesarias para resguardar el acceso, cuidado y confidencialidad de la información.

WGestNefro. Es un módulo no definitivo que persigue como objetivo describir la funcionalidad necesaria para lograr introducir la información de la Base de Datos de ERC para ayudar a la correcta determinación de la compatibilidad Donante – Receptor, por orden de compatibilidad, de los pacientes que se encuentran aptos para recibir el riñón donado.
Este sistema es multiusuario, está desarrollado para Windows, lo cual es uno de los inconvenientes para su uso debido a la necesidad de Cuba de migrar hacia el software libre; accede a una base de datos MySQL y necesita de al menos 10Gb de espacio libre en el servidor de la base de datos para su correcto funcionamiento.

EMALEX (Historia Clínica Automatizada para Pacientes con Enfermedad Renal Crónica)

Es una aplicación basada en la realización de una historia clínica automatizada de los enfermos renales crónicos en diálisis que brinda varios servicios para la atención a los pacientes que se encuentren en la Base de Datos

Debilidades:

- Está desarrollado en Delphi y usa como gestor de base de datos Access. Lo cuál no cumple con las políticas definidas por el MINSAP para las aplicaciones destinadas al sector de la Salud Pública.
- Al ser una aplicación desktop que se puede desplegar en cada uno de los servicios nefrológicos de forma independiente, permite tener en cada uno de los servicios solamente el control de los pacientes que allí se atienden, pero no existirá un control a nivel central de la información de dichos pacientes.

Los anteriores sistemas encontrados a nivel internacional y nacional como es el caso del EMALEX, son capaces de brindar solución a lo que se quiere con esta aplicación, pero no se pueden emplear, ya que son fabricados por empresas privadas, sobre software privados, lo que provoca la necesidad del pago de una licencia, para el uso de estas aplicaciones y muchos resuelven aspectos que no son significativos para la nación cubana.

1.2Tendencias y tecnologías actuales

1.2.1 ¿Qué es Internet?

Internet es una red de redes a escala mundial de millones de computadoras interconectadas con el conjunto de protocolos TCP/IP. Permite entonces comunicar y buscar y transferir información sin grandes requerimientos tecnológicos ni económicos relativos para el individuo.

La World Wide Web ha influenciado mucho a la popularidad de Internet, esta se dice que es la Telaraña Mundial o WWW.Esta permite desplegar gráficos y usar el mouse para "navegar" (visitar) los lugares en Internet. La WWW es sólo uno de los muchos servicios que se brindan el la red Internet, es decir la web emplea a esta como medio de trasmisión.

Internet funciona con la estrategia "Cliente/Servidor", lo que significa que en la Red hay ordenadores Servidores que dan una información concreta en el momento que se solicite, y por otro lado están los ordenadores que piden dicha información, los llamados Clientes.

Hoy en día, los servicios más usados en Internet son: Correo Electrónico, World Wide Web, FTP, Grupos de Noticias, IRC y Servicios de Telefonía.

1.2.2 Aplicaciones Web. Ventajas y desventajas.

Una aplicación web es un sistema informático que los usuarios utilizan accediendo a un servidor web a través de Internet o de una intranet. Las aplicaciones web son populares debido a la practicidad del navegador web como cliente ligero. La facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad. [1]

Aunque muchas variaciones son posibles, una aplicación web está comúnmente estructurada como una aplicación de tres-capas. En su forma más común, el navegador web es la primera capa, un motor usando alguna tecnología web dinámica es la capa de en medio, y una base de datos como última capa. El navegador web manda peticiones a la capa media, que la entrega valiéndose de consultas y actualizaciones a la base de datos generando una interfaz de usuario.

Las aplicaciones Web ofrecen grandes **ventajas** que pueden ser aprovechadas. Entre las ventajas que se pueden mencionar están: [2]

- No requieren instalación, pues usan tecnología Web, lo cual permite el aprovechamiento de todas las características del Internet.
- Son fáciles de usar (No se requieren complicadas combinaciones de Hardware/Software para utilizar estas aplicaciones. Solo un computador con un buen navegador web.).
- Una empresa puede migrar de sistema operativo o cambiar el Hardware libremente sin afectar el funcionamiento de las aplicaciones de servidor.
- Se facilita el trabajo a distancia. Se puede trabajar desde cualquier PC o computador portátil con conexión Internet.
- Actualizar o hacer cambios en el Software es sencillo y sin riesgos de incompatibilidades. Existe solo una versión en el servidor lo que implica que no hay que distribuirla entre los demás computadores. El proceso es rápido y limpio.
- Al funcionar en un navegador, se requiere un conocimiento básico de informática para utilizar una aplicación web.
- Alta disponibilidad, ya que puede realizar consultas en cualquier parte del mundo donde tenga acceso a Internet y a cualquier hora. [3]

DESVENTAJAS

- Acceso limitado, la necesidad de conexión permanente y rápida a Internet hacen que el acceso a estas aplicaciones no esté al alcance de todos.

- La interactividad no se produce en tiempo real, en las aplicaciones web cada acción del usuario conlleva un tiempo de espera hasta que se obtiene la reacción del sistema.
- Diferencias de presentación entre plataformas y navegadores. La falta de estándares ampliamente soportados dificulta el desarrollo de las aplicaciones. [4]

1.3 Sistemas de Gestión de Base de Datos.

Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan.

El objetivo fundamental de un SGBD consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado.

Un SGBD tiene los siguientes objetivos específicos:

- Independencia de los datos y los programas de aplicación
- Minimización de la redundancia
- Integración y sincronización de las bases de datos
- Integridad de los datos
- Seguridad y protección de los datos
- Facilidad de manipulación de la información

1.3.1 SQL Server.

Microsoft SQL Server 2000 es uno de los mejores SGDB base de datos para Windows, es el Sistema Administrador de Bases de Datos de elección para una amplia gama de clientes corporativos y Proveedores Independientes de Software (ISVs) que construyen aplicaciones de negocios. Las necesidades y requerimientos de los clientes han llevado a la creación de innovaciones de producto significativas para facilitar la utilización, escalabilidad, confiabilidad y almacenamiento de datos.

Ventajas:

- Soporta la configuración automática y la auto-optimización.
- Administración multiservidor para un gran número de servidores.
- Gran variedad de opciones de duplicación de cualquier base de datos.
- Acceso universal a los datos (Universal Data Access).
- Fácil de usar.
- Escalabilidad: Se adapta a las necesidades de la empresa, soportando desde unos pocos usuarios a varios miles.
- Potencia: Microsoft SQL Server es la mejor base de datos para Windows NT Server.
- Posee los mejores registros comparativos (TCP) tanto en transacciones totales como en coste por transacción.
- Gestión: Con una completa interfaz gráfica que reduce la complejidad innecesaria de las tareas de administración y gestión de la base de datos.

Desventajas:

- Licencias con costos altos.

  Plataformas Windows.

1.3.2 MySQL.

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL, es también uno de los sistemas gestores de bases de datos más populares desarrollados bajo la filosofía de código abierto.[5]

Características:

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones.

Poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

MySQL es software de fuente abierta. Fuente abierta significa que es posible para cualquier persona usarlo y modificarlo.

Por último, MySQL es rápido, estable, gratuito, y soporta múltiples lenguajes de programación.

Características de la versión 5.0

- Un amplio subconjunto de ANSI SQL 99, y varias extensiones.
- Soporte a multiplataforma
- Procedimientos almacenados
- Triggers
- Cursores
- Vistas actualizables
- Soporte a VARCHAR
- INFORMATION_SCHEMA
- Soporte X/Open XA de transacciones distribuidas; transacción en dos fases como parte de esto, utilizando el motor InnoDB de Oracle
- Motores de almacenamiento independientes (MyISAM para lecturas rápidas, InnoDB para transacciones e integridad referencial)
- Transacciones con los motores de almacenamiento InnoDB, BDB Y Cluster; puntos de recuperación(savepoints) con InnoDB
- Soporte para SSL
- Query Caché
- Selecciones Anidadas

- Réplica con un maestro por esclavo, varios esclavos por maestro, sin soporte automático para múltiples maestros por esclavo.
- Indexación y búsquedas de campos de texto completos usando el motor de almacenamiento MyISAM
- Biblioteca Interna de datos
- Soporte completo para Unicode
- MySQL soporta las reglas ACID usando los motores InnoDB, BDB y Cluster.

1.3.3 PostgreSQL.

PostgreSQL posee una amplia licencia BSD (esta licencia básicamente consiste en que el código puede ser redistribuido y modificado. La FSF lo considera, junto con la licencia GPL, Software libre) y ampliamente utilizado. Posee una estabilidad y confiabilidad legendaria nunca ha presentado caídas en varios años de operación de alta actividad. Fue diseñado para ambientes de alto volumen intentando estar a la altura de Oracle, Sybase o Interbase. Escala muy bien al aumentar el número de CPUs y la cantidad de RAM. Tiene mejor soporte para subconsultas, triggers, vistas y procedimientos almacenados en el servidor, además tiene ciertas características orientadas a objetos.

Ventajas:

- Soporta transacciones y desde la versión 7.0, llaves foráneas (integridad referencial).

Inconvenientes:

- Consume bastantes recursos y carga más el sistema.

1.4 Lenguajes de Programación Web.

Los Lenguajes de Programación orientados al Web se clasifican en lenguajes del lado del cliente y lenguajes del lado del servidor.

Entre los lenguajes que trabajan del lado del servidor se puede citar algunos, como PERL, ASP, PHP, Java, JSP, entre otros. Estos lenguajes desarrollan la lógica de negocio dentro del servidor, además se encargan de los accesos a los distintos Sistemas de Gestión de Bases de Datos. Dentro de los lenguajes que trabajan del lado del cliente se encuentran el JavaScript, XSLT y el Visual Basic Script, estos dos últimos al combinarse con el HTML

forman lo que se conoce como DHTML, es decir, salida estándar dinámica o HTML dinámico.

1.4.1 PERL.

Perl es un lenguaje de propósito general originalmente desarrollado para la manipulación de texto y que ahora es utilizado para un amplio rango de tareas incluyendo administración de sistemas, desarrollo web, programación en red, desarrollo de GUI y más.[6]

Es un lenguaje interpretado que tiene varias utilidades, pero está principalmente orientado a la búsqueda, extracción y formateado de ficheros de tipo texto. También es muy usado para manejo y gestión de procesos (estado de procesos, conteo y extracción de parámetros característicos). [7]

Sus principales características son que es fácil de usar, soporta tanto la programación estructurada como la programación orientada a objetos y la programación funcional, tiene incorporado un poderoso sistema de procesamiento de texto y una enorme colección de módulos disponibles. [8]

La estructura completa de Perl deriva ampliamente del lenguaje C. Es un lenguaje imperativo, con variables, expresiones, asignaciones, bloques de código delimitados por llaves, estructuras de control y subrutinas. Es gratuito, libre y aunque se desarrolló originalmente en un entorno UNIX.

Actualmente existen en casi todos los sistemas operativos: Windows, Mc OS, de manera que es un lenguaje Multiplataforma.

Algunas de las **ventajas** del uso del lenguaje PERL son las siguientes:

- Construcción de pequeños programas que pueden ser usados como filtros para obtener información de ficheros, realizar búsquedas.
- Se puede utilizar en varios entornos, como puede ser Windows 95, OS/2, sin realizar cambios de código, siendo únicamente necesario la introducción del interprete PERL correspondiente a cada sistema operativo.
- También es uno de los lenguajes mas utilizados en la programación de CGI scripts, que son guiones o scripts que utilizan el interface CGI (Common Gateway Interface), para intercambio de información entre aplicaciones externas y servicios de información.
- El mantenimiento y depuración de un programa en PERL es mucho más sencillo que la de cualquier programa en C. [9]

1.4.2 ASP. [10]

Active Server Pages (ASP, Active Server Pages) es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS). La tecnología ASP está estrechamente relacionada con el modelo tecnológico de su fabricante. Intenta ser solución para un modelo de programación rápida ya que programar en ASP es como programar en Visual Basic, por supuesto con muchas limitaciones ya que es una plataforma que no se ha desarrollado como lo esperaba Microsoft.

Traduciendo la definición de Microsoft: “Las Active Server Pages son un ambiente de aplicación abierto y gratuito en el que se puede combinar código HTML, scripts y componentes ActiveX del servidor para crear soluciones dinámicas y poderosas para el.

Lo interesante de este modelo tecnológico es poder utilizar diversos componentes ya desarrollados como algunos controles ActiveX. Otros problemas que han hecho evolucionar esta tecnología es el no disponer de información "que oriente a quienes desean aprenderla y resulta muy costosa en tiempo descubrir aquí y allá toda la información para volverla altamente útil".

ASP ha pasado por cuatro iteraciones mayores, ASP 1.0 (distribuido con IIS 3.0), ASP 2.0 (distribuido con IIS 4.0), ASP 3.0 (distribuido con IIS 5.0) y ASP.NET (parte de la plataforma .NET de Microsoft). Las versiones pre-.NET se denominan actualmente (desde 2002) como ASP clásico.

En el último ASP clásico, ASP 3.0, hay seis objetos integrados disponibles para el programador, Application, ASPError, Request, Response, Server y Session. Cada objeto tiene un grupo de funcionalidades frecuentemente usadas y útiles para crear páginas web dinámicas.

1.4.3 PHP.

Es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es un acrónimo recursivo que significa "PHP Hypertext Pre-processor" (inicialmente PHP Tools, o, Personal Home Page Tools.

PHP conocido como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones Web dirigidas a bases de datos. Es un potente lenguaje de secuencia de comandos diseñado específicamente para permitir a los programadores crear aplicaciones en Web con distintas prestaciones de forma rápida. MySQL es una base

de datos rápida y fiable que se integra a la perfección con PHP y que resulta muy adecuada para aplicaciones dinámicas basadas en Internet. [11]

Su interpretación y ejecución se da en el servidor en el cual se encuentra almacenada la página y el cliente solo recibe el resultado de la ejecución. Cuando el cliente hace una petición al servidor para que le envíe una página web, enriquecida con este código, el servidor interpretará las instrucciones mezcladas en el cuerpo de la página y las sustituirá con el resultado de la ejecución antes de enviar el resultado a la computadora del cliente. Además es posible utilizarlo para generar archivos PDF, Flash o JPG, entre otros. Permite la conexión a numerosas bases de datos de forma nativa tales como MySQL, Postgres, Oracle, ODBC, IBM DB2, Microsoft SQL Server y SQLite, lo cual permite la creación de Aplicaciones web muy robustas.

PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX, Linux, Windows y Mac OS X, y puede interactuar con los servidores de web más populares.

Su modelo puede ser visto como una alternativa al sistema de Microsoft que utiliza ASP.NET/C#/VB.NET, a ColdFusion de la compañía Macromedia, a JSP/Java de Sun Microsystems, y al famoso CGI/Perl. Aunque su creación y desarrollo se da en el ámbito de los sistemas libres, bajo la licencia GNU, existe además un compilador comercial denominado Zend Optimizer. [12]

Ventajas:

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL .
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones.

Desventajas:

- Por sus características promueve la creación de código desordenado y complejo de mantener.
- Todo el trabajo lo realiza el servidor. Por tanto puede ser más ineficiente a medida que aumenten las solicitudes.
- La orientación a objetos es aún muy deficiente para aplicaciones grandes. [13]

1.4.4 JSP. [14]

Java Server Pages (JSP) es la tecnología para generar páginas web de forma dinámica en el servidor, desarrollado por Sun Microsystems, basado en scripts que utilizan una variante del lenguaje java.

La tecnología JSP, o de JavaServer Pages, es una tecnología Java que permite a los programadores generar dinámicamente HTML, XML o algún otro tipo de página web. Esta tecnología permite al código Java y a algunas acciones predefinidas ser embebidas en el contenido estático.

La principal ventaja de JSP frente a otros lenguajes es que permite integrarse con clases Java (.class) lo que permite separar en niveles las aplicaciones web, almacenando en clases java las partes que consumen más recursos así como las que requieren más seguridad, y dejando la parte encargada de formatear el documento HTML en el archivo jsp.

Además Java se caracteriza por ser un lenguaje que puede ejecutarse en cualquier sistema, lo que sumado a jsp le da mucha versatilidad.

Sin embargo JSP no se puede considerar un script al 100% ya que antes de ejecutarse el servidor web compila el script y genera un servlet, por lo tanto se puede decir que aunque este proceso sea transparente para el programador no deja de ser una aplicación compilada. La ventaja de esto es algo más de rapidez y disponer del API de Java en su totalidad.

1.4.5 JavaScript.

JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.[15]

Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado. [16]

Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas web. La ventaja que presenta sobre el HTML es que permite crear páginas Web más dinámicas, lo que las hace más atractivas para el usuario.

1.4.6 Visual Basic Script.

Es un lenguaje que, a diferencia de JavaScript, es solamente compatible con Internet Explorer, aunque posee toda la funcionalidad que brinda JavaScript.

1.4.7 AJAX.

AJAX (XML y JavaScript asíncronos) no es una tecnología. Es realmente muchas tecnologías, cada una florecida por su propio mérito, uniéndose en poderosas nuevas formas. AJAX incorpora:

• Presentación basada en estándares usando XHTML y CSS.
• Exhibición e interacción dinámicas usando el Document Object Model.
• Intercambio y manipulación de datos usando XML y XSL.
• Recuperación de datos asincrónica usando XMLHttpRequest.
• JavaScript para manipular estas tecnologías.

1.5 XML / WebServices.

1.5.1 WebServices.

Un servicio web (en inglés Web service) es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.

Los Servicios Web necesitan diversas especificaciones y tecnologías, ahí la necesidad de apoyarse en los 4 principales estándares: XML, SOAP, UDDI y WSDL.

Web Services Protocol Stack: Así se denomina al conjunto de servicios y protocolos de los servicios Web:

XML (Extensible Markup Language): Es el formato estándar para los datos que se vayan a intercambiar.

SOAP (Simple Object Access Protocol) o XML-RPC (XML Remote Producer Call): Protocolos sobre los que se establece el intercambio.

Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), o SMTP (Simple Mail Transfer Protocol).

WSDL (Web Services Description Languages): Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.

UDDI (Universal Description, Discovery and Integration): Protocolo para publicar la información de los servicios Web. Permite comprobar qué servicios web están disponibles.

WS-Security (Web Service Security): Protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados. [17]

1.5.2 XML.

XML, sigla en inglés de Extensible Markup Language («lenguaje de marcas extensible»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Este lenguaje es la base de los servicios Web Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. Se emplea principalmente para separar el contenido de la presentación, esta usa sus etiquetas sólo para delimitar piezas de datos, y la interpretación de los datos, la hace la aplicación que los lee. Es gratis, independiente de la plataforma y ampliamente distribuido. [18]

1.5.3 SOAP.

SOAP (siglas de Simple Object Access Protocol/protocolo de acceso simple) es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML[19], por lo cual se dice que se utiliza para el intercambio de datos.

1.5.4 WSDL.

WSDL son las siglas de Web Services Description Language, describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. [20]

1.5.5 UDDI.

UDDI son las siglas del catálogo de negocios de Internet denominado Universal Description, Discovery and Integration. El registro en el catálogo se hace en XML. UDDI es una iniciativa industrial abierta entroncada en el contexto de los servicios Web.

UDDI es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL.[21]

1.5.6 Apache.

Apache es un servidor web gratuito, potente y que ofrece un servicio estable y sencillo de mantener y configurar. Es indiscutiblemente uno de los mayores logros del Software Libre. Apache se demuestra más rápido que otros webserver free.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (http) de la Apache Software Foundation.

Se destacan las siguientes características:

- Es multiplataforma, aunque idealmente está preparado para funcionar bajo Linux.
- Muy sencillo de configurar.
- Es Open-Source

- Muy útil para proveedores de Servicios de Internet que requieran miles de sitios pequeños con páginas estáticas
- Amplias librerías de PHP y Perl a disposición de los programadores
- Posee diversos módulos que permiten incorporarle nuevas funcionalidades, estos son muy simples de utilizar
- Es capaz de utilizar lenguajes como PHP, TCL, Pitón, entre otros

Apache presenta además entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft).

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

1.6 FrameWorks.

En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, librerías y un lenguaje de scripting entre otros softwares para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. [22]

Los frameworks simplifican el desarrollo de las aplicaciones mediante la automatización de muchas de las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al programador a crear código más legible y más fácil de mantener.

1.6.1 CodeIgniter.

CodeIgniter es un Framework para desarrollo de Aplicaciones- un toolkit para personas que quieran construir sitios usando PHP. Su objetivo es poder hacer los proyectos más rápidos de lo que usted puede hacerlos escribiendo código desde el principio, suministrando un rico conjunto de librerías para tareas comunes, con una simple interface y estructura lógica para acceder a estas librerías. CodeIgniter deja que su creatividad se centre en el proyecto, minimizando la cantidad de código necesitado para ejecutar una tarea. [23]

Entre sus características, se encuentra la compatibilidad con PHP 4 y 5, está basado en el Modelo Vista Controlador, posee soporte para múltiples bases de datos entre las que se pueden mencionar PostgreSQL, MySQL, MSSQL, además de que se lo pueden incluir drivers para otros gestores de bases de datos que usted programe, incluye también plantillas, validaciones, no requiere instalación, se encuentra una librería con un gran número de clases.

Una de las características más interesantes de CodeIgniter es el elevado número de clases que incluye para trabajar con distintos objetos: calendario, bases de datos, correo electrónico, manipulación de imágenes, FTP, lenguaje, tablas, sesiones, compresión ZIP, entre otros. A diferencia de otros frameworks, CodeIgniter cuenta con una documentación excelente que permite conocer todos los secretos de este entorno de trabajo. Está liberado bajo la licencia de código abierto Apache-BSD, lo que significa que es totalmente libre y puede ser usado.

1.6.2 Scaffolding.

El scaffolding es un método para construir aplicaciones basadas en bases de datos, esta técnica está soportada por algunos frameworks tal es el caso de CodeIgniter (tipo MVC), en el cuál el programador escribe una especificación que describe cómo debe ser usada la base de datos. Luego el compilador utiliza esa especificación para generar el código que la aplicación usará para crear, leer, actualizar y eliminar registros de la base de datos, esto es conocido cómo CRUD (create, read, update, delete).

1.6.3 Yahoo User Interface Library.

Están disponibles dos tipos de componentes diferentes: Utilidades y Controles: Las YUI Utilidades simplifican el desarrollo para la compatibilidad entre Navegadores basados en técnicas DOM, DHTML y AJAX.

Los controles de YUI proporcionan elementos visuales altamente interactivos del diseño para sus aplicaciones Web. Estos elementos se crean y se manejan íntegramente del lado del cliente (usuario) y nunca requieren de una recarga de página.

1.7 IDE de Programación.

Un entorno de desarrollo integrado o en inglés Integrated Development Environment ('IDE') es un programa compuesto por un conjunto de herramientas para un programador.

Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva.

Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación.

Los componentes que deben tener los IDEs son:

- Un editor de texto.
- Un compilador.
- Un intérprete.
- Herramientas de automatización.
- Un depurador.
- Posibilidad de ofrecer un sistema de control de versiones.
- Factibilidad para ayudar en la construcción de interfaces gráficas de usuarios.

1.7.1 Zend Studio.

Editor web orientado a la programación de páginas PHP, con ayudas en la gestión de proyectos y depuración de código.

Los expertos en PHP consideran a Zend Studio como el entorno IDE más maduro y con más características útiles. La última versión ofrece manipulación avanzada de bases de datos y otras mejoras. Se trata de un programa de la casa Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web, en lenguaje PHP.

El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código.

El programa entero está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones del producto para Windows, Linux y MacOS, aunque el desarrollo de las versiones de este último sistema se retrase un poco más.

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor, que instala Apache y el módulo PHP o, en caso de que estén instalados, los configura para trabajar juntos en depuración.

1.7.2 Notepad++.

Notepad++ es un poderoso editor de código fuente que soporta diversos lenguajes de programación.

Este interesante editor permite imprimir el código escrito con los diferentes colores de la sintaxis, posee la opción de auto completado con la posibilidad de definir una API personalizada, permite la edición de varios documentos al mismo tiempo, con sus correspondientes vistas separadas o integradas a la misma interfaz.

Notepad++ posee soporte para búsqueda y reemplazo de expresiones regulares, detección automática del estado del texto, incluye una herramienta de Zoom, trabaja con puntos de marca para editar y desplazarse más dinámicamente y permite la creación de marcos con su correspondiente secuencia de teclas para acceso rápido.

Con Notepad++ se programa en los siguientes lenguajes: C, C++, Java, C#, XML, HTML, PHP, CSS, makefile, ASCII art (.nfo), doxygen, ini file, batch file, JavaScript, ASP, VB/VBS, SQL, Objective-C, RC resource file, Pascal, Perl, pitón, Lua, TeX, TCL, Assembler, Ruby, Lisp, Scheme, Properties, Diff, Smalltalk, Postscript, VHDL, Ada, Caml, AutoIt, KiXtart, Matlab, Verilog, Haskell, InnoSetup y CMake.

Notepad++ es un excelente editor para tener en cuenta.

1.8 Tecnologías y Herramientas a utilizar.

El sistema a desarrollar está diseñado bajo la misma arquitectura que está concebida para el desarrollo de todos los módulos que formarán parte del Sistema Informatizado de Salud, con el objetivo de estandarizar el diseño de dichos módulos y facilitar el mantenimiento de los mismos. Esta arquitectura está basada en componentes y en tres capas lógicas;

haciendo uso de los WebServices, los cuáles posibilitan que diversas aplicaciones compartan información sin importar sus ubicaciones físicas o cómo se hayan creado.
El desarrollo del sistema está concebido bajo la metodología RUP que junto con el Lenguaje Unificado de Modelado (UML) constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas. Para documentar el desarrollo del software se utilizará la herramienta Visual Paradigm Suite 3.0, ya que esta herramienta es muy completa y ofrece amplias potencialidades.
Para la implementación se utilizará como servidor Web Apache 2.0 con PHP 5; servidor de bases de datos MySQL 5, el IDE de desarrollo Zend Studio 5.1; también se van a utilizar los frameworks CodeIgniter y Yahoo User Interface (YUI).
Se seleccionaron todas estas herramientas y tecnologías no solo por las ventajas antes mencionadas, sino también por política del Área Temática (Sistemas Especializados).

En este capítulo se profundizó en el conocimiento de algunos conceptos necesarios para la compresión de este trabajo. Además se realizó un análisis completo de las tecnologías que serán utilizadas a lo largo del desarrollo del sistema propuesto, y se fundamentaron las elecciones del lenguaje, el sistema gestor de bases de datos, y la metodología a utilizar.

2. Elementos de Arquitectura.

El siguiente capítulo pretende realizar una selección y argumentación de los requisitos no funcionales propuestos, se hace un análisis de los Patrones o estilos arquitectónicos presentes en la propuesta de solución, justificando así su uso. En este capítulo se puede encontrar la vista de despliegue y de implementación.

2.1 Requisitos no funcionales del sistema propuesto.

Interfaz externa.

- La interfaz de usuario será sencilla, amigable, intuitiva y de fácil navegación por el usuario, con el objetivo de evitar la resistencia humana al uso del nuevo sistema.
- Se seleccionará un esquema de colores a la vez atractivo pero que no canse.
- Paginación de reportes de búsqueda, y listados.
- Diseño perfectamente encuadrado para resoluciones de 1024 x 768, pero preparado para verse en otras resoluciones.

Usabilidad.

- La aplicación Web será flexible y de fácil aprendizaje, pues se trata en todo lo posible de mantener un estándar de operabilidad que logre que las interacciones del usuario con el sistema sean predecibles y familiares.
- El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.
- Deberá visualizarse bien en los principales navegadores que existen en el mundo.
- Estará disponible en todo momento.
-

Rendimiento.

Las pantallas estarán poco cargadas de imágenes para garantizar que el tiempo de ejecución de los hipervínculos, las adiciones, modificaciones y eliminaciones no excedan los 4 segundos y garantizar de esta manera una respuesta rápida del sistema.

Soporte.

- El sistema contará con una ayuda para el usuario con la cual podrá aprender rápidamente a utilizar la aplicación Web.

- Estará bien documentado para garantizar futuros mantenimientos.

Portabilidad.

El sistema podrá ejecutarse sobre plataforma Linux, Windows 98 o superior.

Seguridad.

La protección del sistema contra el acceso desautorizado y las modificaciones de información está garantizada por el Sistema de Autentificación, Autorización y Auditoría (SAAA).

Confiabilidad.

- Todas las partes del diseño del sistema serán realmente aplicadas y se hará una transformación correcta del diseño en un lenguaje de programación.
- Deberá prevenir los posibles fallos y/o errores que pudieran presentarse y posibilitar una rápida recuperación en dichos casos.

Software

Para el cliente:

- Un navegador Web, recomendados: Mozilla 1.5, Internet Explorer 4.0 o superior.
- Sistema operativo Linux o Windows 98 ó Superior

Para el servidor:

- Sistema operativo Linux, Windows XP o superior.
- Servidor Web Apache 2.0 y PHP 5.
- Framework CodeIgniter.
- Servidor de Base de Datos MySQL 5.1

Hardware

PC clientes:

- Procesador Pentium III o superior.
- 128 de memoria RAM o superior
- Monitor VGA o superior
- Tarjeta de red

Servidor:

- Procesador Pentium IV o superior.
- 512 de memoria RAM o superior.
- Disco Duro de 80 GB.

Diseño e implementación

Utilizar los patrones de diseño establecidos.

Para el análisis y el diseño del sistema debe ser utilizado la metodología RUP, usando el lenguaje de modelación UML y como herramienta para llevarlo a cabo el Visual Paradigm.

Implementado con el lenguaje de programación php 5

Desarrollado en Zend Studio.

2.2 Estándares de Codificación.

Estándar de codificación para PHP

Idioma:

Se debe utilizar como idioma el español, las palabras no se acentuarán.

Palabras Reservadas

Las palabras reservadas van en minúsculas sin excepción alguna.

Identación **Objetivo:** Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento		
0 espacios en blanco desde la izquierda en	require include class	No se empleará ningún espacio en blanco desde la izquierda para las instrucciones antes mencionadas. Se tomará como inicio de la página el tag PHP <?
2 espacio en blanco desde la izquierda en	function define	Se dejarán dos espacios en blanco desde la izquierda en las

		instrucciones antes mencionadas.
2 espacio en blanco desde la referencia en	Inicio y fin de bloque	Se recomienda dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones If, else, For, While, Do While, Switch, Foreach.
Niveles de anidación	Hasta 5 niveles	Se recomienda emplear hasta 5 niveles de anidación en instrucciones If, For, While.
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.	

Ejemplo de identación

```
<?
require ('class/Interface.php');
class MiClase
{
  function BuscarUnidad($sTabla, $sCampos, $iIndice)
  {
   if ($sTabla)
   {
…
   }
  for (…)
  {
```

```
  …
 }
}
}
?>
```

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, si existe.	Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma.
Separador de instrucciones	Se emplea el punto y coma.	Se recomienda usar el separador al final de cada instrucción y no en la línea de abajo. Ejemplo: define (“CONSTANT”,”value1”);
Líneas en blanco	Se emplean antes y después de métodos, clases y estructuras.	Se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura, y de la implementación de una función.
Espacios en blanco	Entre operadores lógicos y aritméticos.	Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código. Ejemplo: $sTabla = „nomproducto“; if (($sTabla) && ($sCampos))
Márgenes y líneas de continuidad	Sobre márgenes y líneas de continuación	Los márgenes de cada línea de código no deben exceder los 80 caracteres, pero puede exceptuarse si es para terminar la

		escritura de una palabra. Las líneas más largas deben cortarse en una o más líneas de continuación. Las líneas de continuación deben estar alineadas entre sí e identadas respecto al paréntesis abierto. Las líneas de continuación nunca deben comenzar con un operador binario.
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.
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 identificará el tipo de datos al que se refiere , en caso de que sea un

		nombre compuesto se empleará notación CamellCasing**. Ejemplo: $ sNombrePaciente
Apariencia de constantes	Todas sus letras en mayúscula	Se recomienda declarar una constante por cada línea y con las asignaciones a las variables sucede lo mismo. Ejemplo: define(“CONSTANT1”,”value1”); define(“CONSTANT2”,”value2”); $sTabla=“nomproducto“; $sIndice=0;
Declaración de constantes y asignación a variables	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 comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.
Apariencia de atributos	Primera letra en minúscula	El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que

		sea un nombre compuesto se empleará notación CamellCasing**.
Apariencia de las funciones	Primera letra en mayúscula	Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hace la función. Se empleará notación PascalCasing*. Ejemplo: function BuscarUnidad(). Si son funciones que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set
Declaración de parámetro en funciones	Agrupados por tipos Poner los string numéricos 2, además, agrupar según valores por defecto.	Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que estén agrupados por el tipo de dato que contienen, especificando el tipo de datos Ejemplo:BuscarUnidad($sTabla, sCampos, $iIndice).
Aspectos generales	Sobre las clases, los objetos, los atributos y las funciones.	El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos. Para los atributos:Ejemplo: $sTabla, este atributo denota el nombre de una tabla.
Bases de Datos, Tablas, esquemas y Campos		
Apariencia de la BD	Las 2 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 comienza con mayúscula y el

		resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*. Ejemplo: bd _BalanceMaterial.
Apariencia de los vistas	Las 2 primeras letras representan el tipo. Todas las letras en minúscula	El nombre a emplear para las vistas deben comenzar con el prefijo vt seguido de underscoard y el nombre debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor. Ejemplo: create view „vt_finanzas‟;
Apariencia de las tablas	Las 2 primeras letras representan el tipo. Todas las letras en minúscula	El nombre a emplear para las vistas deben comenzar con el prefijo vt seguido de underscoard y el nombre debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor. Ejemplo: create view „vt_finanzas‟;
Apariencia de los procedimientos almacenados.	Las 2 primeras letras representan el tipo .Todas las letras en minúscula.	El nombre a emplear para los procedimientos debe comenzar con el prefijo pa 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:

		pa _ paciente_especialidad.
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: „idproducto“;
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.
Sentencias SQL	Todas las letras en mayúscula.	Las palabras correspondientes a las sentencias SQL y sus parámetros deben ir en mayúsculas.
Aspectos generales	Sobre las BD, vistas, tablas atributos y procedimientos.	El nombre empleado para las Base 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, en caso de que sea un nombre compuesto

		se empleará notación CamellCasing**. Ejemplo: btnAceptar

Tabla 2.1 Estrategia y estándares de codificación.

2.3 Patrones Arquitectónicos.

2.3.1 Modelo Vista Controlador.

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde se separa la vista, de la modelo, de las controladoras. Para construir una aplicación utilizando el patrón MVC hay que definir tres clases de módulos:

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos o acceso a datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Lleva un registro de las vistas y controladores del sistema.
- Si se está ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

El controlador es responsable de:

- Responde a eventos y modifica la vista y el modelo. Recibe los eventos de entrada.
- Contiene reglas de gestión de eventos. Estas acciones pueden suponer peticiones al modelo o a las vistas.

Las vistas son responsables de:

Existen distintas formas de implementar el patrón, en este trabajo en la vista se reciben los datos del controlador y se muestra al usuario. Define la interfaz de usuario.

La principal ventaja de esta separación reside en la facilidad para realizar cambios en la aplicación puesto que:

Cuando se realiza un cambio de bases de datos, programación o interfaz de usuario solo se cambia uno de los componentes

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo en este sistema es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace).
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista.: La vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

2.3.2 Arquitectura Modelo Cliente – Servidor.

Esta arquitectura consiste básicamente en que un programa el Cliente informático realiza peticiones a otro programa, el servidor, que les da respuesta.

Con respecto a la definición de arquitectura cliente/servidor se encuentran las siguientes definiciones:

Cualquier combinación de sistemas que pueden colaborar entre sí para dar a los usuarios toda la información que ellos necesiten sin que tengan que saber donde está ubicada.

Es una arquitectura de procesamientos cooperativos donde uno de los componentes pide servicios a otro.

Es un procesamiento de datos de índole colaborativo entre dos o más computadoras conectadas a una red.

El término cliente/servidor es originalmente aplicado a la arquitectura de software que describe el procesamiento entre dos o más programas: una aplicación y un servicio soportante.

Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores. [24]

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

Ventajas de la arquitectura cliente-servidor.

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. El servidor no necesita tanta potencia de procesamiento, parte del proceso se reparte con los clientes.
- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado.
- Se reduce el tráfico de red considerablemente.
- En esta arquitectura el cliente se conecta al servidor cuando es necesario nada mas, obtiene los datos que necesita y cierra la conexión dejando libre la red.
- Se puede decir que todas las aplicaciones tienen la misma arquitectura básica y se pueden subdividir en tres partes:
- Interfaz del Usuario: La presentación al usuario, con las entradas de datos y las pantallas de consulta.
- Reglas de negocio: Sería el procesamiento de la información.
- Accesos a Datos: El control del almacén de datos.

2.3.3 Aplicaciones con Arquitectura Mono – Capa.

Se entiende por aplicaciones mono-capa, aquellas que tanto la propia aplicación como los datos que maneja se encuentran en la misma máquina y son administradas por la misma herramienta: son una sola entidad.

2.3.4 Aplicaciones con Arquitectura en Dos – Capas.

Estas aplicaciones son más conocidas como aplicaciones Cliente/Servidor y lo más característico es que dividen una aplicación entre un cliente y un servidor estableciendo un middleware que controla las comunicaciones entre ambos. Un programa Visual FoxPro que interroga a una Base de Datos SQLServer es un ejemplo de aplicación en dos capas.

En la raíz de las aplicaciones cliente/servidor está la separación de la aplicación en componentes encapsulados u objetos. La ventaja de romper una aplicación en trozos es que cualquier cambio de uno de esos componentes no tiene un impacto directo sobre los otros o en el resto de la aplicación.

En las arquitecturas two-tier, la aplicación se divide en dos entidades separadas.

La arquitectura two-tier dividida en dos entidades con el interfaz por un lado y las reglas de negocio junto con el Acceso a Bases de Datos. Encapsular las reglas de negocio junto con los datos tiene la ventaja de que se pueden cambiar sin tener que tocar los interfaces de los clientes que seguramente estarán muy distribuidos. El inconveniente es que normalmente los Servidores de Datos no son muy moldeables y es bastante complicado implementar reglas de negocio en los servidores.

En estas aplicaciones el Servidor de Datos procesa las Consultas y realiza todas las actividades relacionadas con la Base de Datos. Cada Cliente inicia y deja abierta una conexión al servidor para poder enviar las peticiones y poder procesar las respuestas.

Este modelo suele ser costoso de mantener, difícil de escalar y pesado de depurar.

Aspectos a tener en cuenta a la hora de pasar de una aplicación de una sola capa a otra en Dos Capas:

- Usa Vistas Locales en vez de acceder a tablas directamente
- Encapsula las reglas de negocio fuera de contenedores visuales. Las Reglas de Negocio deberían ir en clases no visuales tipo Custom.

2.3.5 Aplicaciones con Arquitectura en Tres – Capas.

Con la arquitectura en tres capas (three-tier) se añade una nueva capa entre el cliente y el servidor donde se implementa la lógica de la aplicación. De esta forma el cliente es básicamente una interface, que no tiene por qué cambiar si cambian las especificaciones de la base de datos o de la aplicación; queda aislado completamente del acceso a los datos. En este caso se tiene total libertad para escoger dónde se coloca la lógica de la aplicación: en el cliente, en el servidor de base de datos, o en otro(s) servidor(es). También se tiene total libertad para la elección del lenguaje a utilizar. [25]

Puede determinarse en qué servidor(es) se quiere hacer funcionar estos procedimientos. En aplicaciones críticas se pueden agregar tantos servidores de aplicación como sean necesarios, de forma simple, y sin comprometer en absoluto la integridad de la base de datos, obteniéndose una escalabilidad muy grande sin necesidad de tocar el servidor de dicha base de datos

En teoría, los sistemas en 3 capas son de más fácil ampliación y más robustos y flexibles. Además, pueden integrar datos de múltiples fuentes.

La programación por capas es un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles.

Diseño en tres niveles (o en tres capas).

Capas o niveles

1.- **Capa de presentación**: es la que ve el usuario (hay quien la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario.

2.- **Capa de negocio**: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

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

Todas estas capas pueden residir en un único ordenador, si bien lo más usual es que haya una multitud de ordenadores, si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores. En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares.

El término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

Presentación/ Lógica de Negocio/ Datos.

En cambio, el término "nivel" corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

Una solución de tres capas (presentación, lógica, datos) que residen en un solo ordenador (Presentación + lógica + datos). Se dice que la arquitectura de la solución es de tres capas y un nivel.

Una solución de tres capas (presentación, lógica, datos) que residen en dos ordenadores (Presentación + lógica, lógica + datos). Se dice que la arquitectura de la solución es de tres capas y dos niveles.

Una solución de tres capas (presentación, lógica, datos) que residen en tres ordenadores (presentación, lógica, datos). La arquitectura que la define es: solución de tres capas y tres niveles.

Ventajas de la arquitectura en 3 capas

- . Separación clara de la interfaz de usuario de la lógica de la aplicación. Esta separación permite tener diferentes presentaciones accediendo a la misma lógica
- La redefinición del almacenamiento de información no tiene influencia sobre la presentación.
- En contraste con una arquitectura en 2 capas, donde solamente datos están accesibles al público, los objetos de negocios pueden brindar servicios (lógica de la aplicación) por la red. [26]
- Se pueden realizar cambios en las diferentes capas sin necesidad de modificarlas todas.

2.3.6 Arquitectura Basada en Componentes. [27]

El objetivo de la tecnología de componentes software es construir aplicaciones complejas mediante ensamblado de módulos (componentes) que han sido previamente diseñados por otras personas a fin de ser rehusados en múltiples aplicaciones.

La arquitectura software de una aplicación basada en componentes consiste en uno o un número pequeño de componentes específicos de la aplicación (que se diseñan específicamente para ella), que hacen uso de otros muchos componentes prefabricados que se ensamblan entre sí para proporcionar los servicios que se necesitan en la aplicación. Es decir consiste en mapear los componentes funcionales en la arquitectura lógica de la aplicación.

Para que una arquitectura de componentes pueda operar es necesario disponer de un entorno normalizado que proporcione soporte a los mecanismos con que se comunican las interfaces.

Reusabilidad de Servicios: Reducción considerable de tiempos y costos de desarrollo de aplicaciones al utilizar servicios disponibles ya desarrollados, para resolver problemáticas comunes a otras aplicaciones. Aumentado por esta razón la robustez del nuevo sistema, al utilizarse software ya probado.

Interoperatibilidad de aplicaciones: Disminución de la complejidad en el proceso de integración, pues se interactúa con elementos que se abstraen de la tecnología y ubicación de los servicios.

Las características principales son la modularidad, la reusabilidad y compatibilidad, en la tecnología basada en componentes también se requiere robustez ya que los componentes han de operar en entornos mucho más heterogéneos y diversos.

Su premisa es que los componentes cumplan con alta cohesión y bajo acoplamiento.

2.3.7 Arquitectura Orientada a Servicios.

La Arquitectura Orientada a Servicios (en inglés Service Oriented Architecture o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.

Proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones identifican la utilización de Servicios Web (empleando SOAP y WSDL) en su implementación, no obstante se puede implementar una SOA utilizando cualquier tecnología basada en servicios.

Al contrario de las arquitecturas orientado a objetos, las SOAs están formadas por servicios de aplicación débilmente acoplados y altamente interoperables. Para comunicarse entre sí,

estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación La definición de la interfaz encapsula (oculta) las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo (como Plataforma Java o Microsoft.NET). Con esta arquitectura, se pretende que los componentes software desarrollados sean muy reusables, ya que la interfaz se define siguiendo un estándar; así, un servicio C Sharp podría ser usado por una aplicación Java.

2.4 Vista de Despliegue.

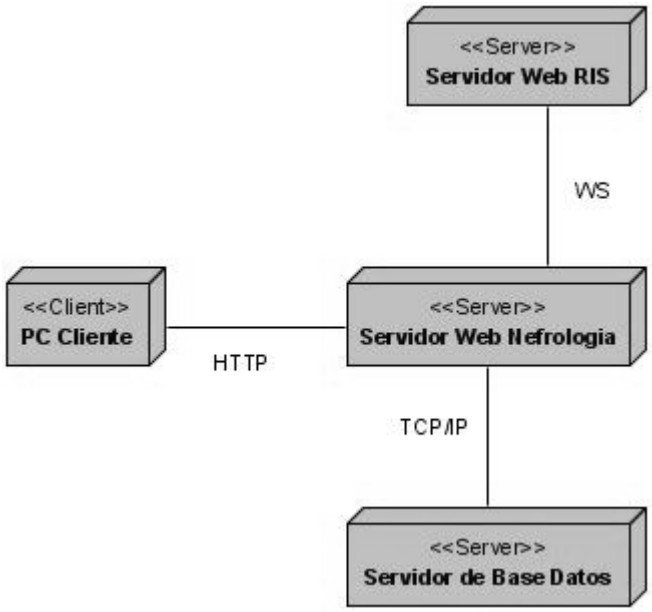


Fig. 2.1 Diagrama de la Vista de Despliegue

Servidor Web RIS: En este nodo se encuentran desplegados los componentes del RIS con los que debe interactuar la Red Cubana de Nefrología.

Servidor Web Nefrología: En este nodo se encuentra la aplicación Red Cubana de Nefrología.

Servidor de Base de Datos: En este nodo se encuentra el servidor de la BD del sistema Red Cubana de Nefrología.

PC cliente: Desde ella se accede a través de un navegador al sistema Red Cubana de Nefrología.

2.5 Vista de Implementación.

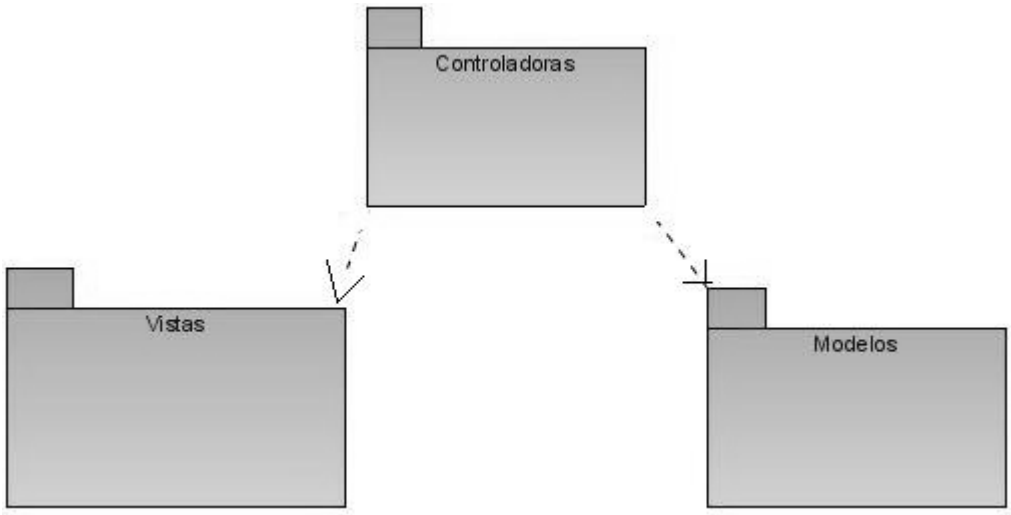


Fig. 2.2 Diagrama de paquetes.

En este capítulo se describen los requisitos no funcionales del sistema a tener en cuenta para realizar la aplicación. En el se muestran los estándares y las estrategias de codificación que están presente en el sistema, para que el código sea de una forma mas organizada, y fácil de entender. Además se hace un estudio de los patrones, y estilos necesarios, para definir la arquitectura, la que está centralizada en los servidores de Infomed, basada en componentes, en Tres capas lógicas, y basada en el patrón modelo-vista- controlador. En este capítulo se encontra la vista de despliegue y de implementación.

3 Descripción y análisis de la solución propuesta.

En este capítulo se aborda el análisis del diseño propuesto por los analistas. Se analizarán las posibles implementaciones de módulo. Se harán descripciones de las clases, los tipos de datos y las operaciones que se implementen para dar solución al problema.

3.1 Estrategia de integración con otros módulos o sistemas.

El Sistema Red Cubana de Nefrología pertenece al Sistema Informatizado de Atención Especializada (SIAE), perteneciente al SISalud, pues es aquí, donde se agruparán los módulos que pertenecen al nivel de atención terciario o especializado.

Siguiendo esta filosofía, la Red Cubana de Nefrología necesita para su buen funcionamiento integrarse a una serie de componentes ya desarrollados para el SISalud.Del Registro Informatizado de la Salud (RIS) para el control de la seguridad se integra con el:

- **SAAA** (Sistema de autenticación, autorización y auditoría)

Módulo de seguridad y Administración, está basado en un modelo de autenticación, autorización y auditoría (SAAA). Es el sistema que gestiona el nivel de acceso a los distintos componentes del SISalud, permitiendo que los usuarios tengan un solo usuario y una contraseña para acceder a todos los sistemas.

Este brinda a través de Web Services una serie de métodos que facilitan todos los servicios necesarios para asegurar el sistema. Para su uso se crearon diferentes clases que encapsulan el manejo de la seguridad así como la comunicación con este módulo, de manera que esta queda de forma global para todo el sistema sin necesidad de ser implementada en cada subsistema, pero sí utilizada por todos y de manera particular por cada uno.

- **REM** (Registro de Equipos Médicos) y **RENM** (Registro de Equipos No Médicos)

Constituye el componente del Registro informatizado de Salud, que permite el manejo de la información relacionada con los equipos existentes en el Sistema de Salud. El control de los equipos existentes, su estado técnico, su funcionalidad, en una Institución dada, municipio o provincia, constituye uno de los objetivos fundamentales del Sistema de Salud en todas sus instancias.

El uso de este componente permite identificar los principales problemas y limitaciones en la utilización y distribución de los equipos y tecnologías en el Sector. Permite conocer de cerca las realidades en la gestión de las tecnologías en el Sistema de Salud. Así como, la toma de

decisiones para modificaciones administrativas y funcionales para una mayor eficiencia en la utilización de los recursos.

Además el sistema está implementado usando el framework de clases de PHP CodeIgniter, y el framework de clases de JavaScript YUI, se hace rehúso de todas las librerías y clases que estos dos potentes frameworks ofrecen.

3.2 Descripción de las clases u operaciones necesarias.

Estas son clases y operaciones necesarias del módulo RAD.

- **Descripción de las clases del CU crítico Gestionar Insumos**

Nombre: E_Insumos	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_Insumos()
Descripción:	Constructor de la clase por defecto
Nombre:	E_Insumos(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_TInsumo	
Tipo de clase: Entidad	
Atributo	**Tipo**

id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_TInsumo()
Descripción:	Constructor de la clase por defecto
Nombre:	E_TInsumo(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_Fabricante	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_Fabricante()
Descripción:	Constructor de la clase por defecto
Nombre:	E_Fabricante(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción

Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_Paquetes	
Tipo de clase: Entidad	
Atributo	**Tipo**
cantidad actual lote fecha realización fecha vencimiento	int string date date
Para cada responsabilidad:	
Nombre:	Get_Lote(): string
Descripción:	Devuelve el valor del atributo lote
Nombre:	Get_FechaRealizacion(): date
Descripción:	Devuelve el valor del atributo fecha realización
Nombre:	Get_FechaVecimiento() :date
Descripción:	Devuelve el valor del atributo fecha vencimiento
Nombre:	Get_Fabricante(): string
Descripción:	Devuelve los objetos que se utilizan de la clase E_Fabricante
Nombre:	Get_TipoInsumo(): string
Descripción:	Devuelve los objetos que se utilizan de la clase E_TInsumo
Nombre:	SetCant_Actual(): void
Descripción:	Permite cambiar el estado del atributo cantidad
Nombre:	SetLote(): void
Descripción:	Permite cambiar el estado del atributo lote

Nombre: E_ CausaD	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción	int string

eliminado	bool
Para cada responsabilidad:	
Nombre:	E_ CausaD()
Descripción:	Constructor de la clase por defecto
Nombre:	E_ CausaD(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_ Paquete_Entrada	
Tipo de clase: Entidad	
Atributo	**Tipo**
cantidad inicial fecha entrada	int date
Para cada responsabilidad:	
Nombre:	Getcant_Inicial(): int
Descripción:	Devuelve el valor del atributo cantidad inicial
Nombre:	Get_FechaEntrada(): date
Descripción:	Devuelve el valor del atributo fecha entrada
Nombre:	Setcant_Inicial(): void
Descripción:	Cambia el estado del atributo cantidad
Nombre:	Set_FechaEntrada(): void
Descripción:	Cambia el estado del atributo fecha entrada

Nombre: E_ Paquete_Salida	
Tipo de clase: Entidad	
Atributo	**Tipo**

causa fecha salida	E_ CausaD date
Para cada responsabilidad:	
Nombre:	GetCausa(): E_ CausaD
Descripción:	Devuelve los objetos que se utilizan de la clase CausaD
Nombre:	Get_FechaSalida(): date
Descripción:	Devuelve el valor del atributo fecha salida
Nombre:	SetCausa (): void
Descripción:	Cambia el estado del atributo causa
Nombre:	Set_FechaSalida(): void
Descripción:	Cambia el estado del atributo fecha salida

La clase modelo Gestionar Insumo

Nombre: M_Gestionar_Insumos	
Tipo de clase : Modelo	
Para cada responsabilidad:	
Nombre:	DarEntrada(E_Paquete_Entrada): void
Descripción:	Adiciona un nuevo paquete de insumos para el centro de diálisis
Nombre:	DarSalida(E_Paquete_Salida): void
Descripción:	Elimina un paquete de insumos por una causa determinada en el centro de diálisis
Nombre:	ListarInsumosExistencia(): List
Descripción:	Muestra la lista de insumos útiles
Nombre:	MostrarAlertasVencimientoInsumos(): string
Descripción:	Muestra cuando se esta llegando a la fecha de vencimiento de un insumo.
Nombre:	GenerarInforme(date, date): void
Descripción:	Genera el informe de un insumo dado una fecha
Nombre:	Listado_Reportes(): List
Descripción:	Lista los reportes realizados

La clase Controladora Gestionar Insumo

Nombre: C_Gestionar Insumos	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Entrar_Paquete(E_Paquete_Entrada): void
Descripción:	Registra un paquete en un hospital.
Nombre:	Dar_Salida()
Descripción:	Extrae un paquete de insumos
Nombre:	ListadoExistencia()
Descripción:	Muestra el listado actual de insumos
Nombre:	Listado_Reportes()
Descripción:	Realiza un listado de los reportes
Nombre:	FormatFechaSalida()
Descripción:	Crea un formato de fecha válido para dar salida
Nombre:	Buscar()
Descripción:	Busca un paquete de insumos
Nombre:	Cargar_CausaSalida()
Descripción:	Carga los datos de la causa que vienen desde la vista
Nombre:	CargarDatos_Fabricante()
Descripción:	Carga los datos del fabricante que vienen desde la vista
Nombre:	CargarDatos_Insumo()
Descripción:	Carga los datos del insumo que vienen desde la vista

- **Descripción de las clases del Caso de Uso Gestionar Sillones Hemodiálico**

Nombre: E_SillonesH
Tipo de clase: Entidad

Atributo	**Tipo**
modelo	string
tiempo_explot	double
noserie	string
Para cada responsabilidad:	
Nombre:	GetModelo ():string
Descripción:	Devuelve el valor del atributo modelo
Nombre:	GetTiempo_Explot ():double
Descripción:	Devuelve el valor del atributo tiempo de explotación
Nombre:	GetNoSerie():string
Descripción:	Devuelve el valor del atributo número de serie
Nombre:	SetModelo (modelo):void
Descripción:	Cambia el estado del atributo modelo
Nombre:	SetTiempo_Explot (tiempo explot):void
Descripción:	Cambia el estado del atributo (tiempo de explotación)
Nombre:	SetNoSerie(noserie):void
Descripción:	Cambia el estado del atributo(número de serie)

Nombre: E_Revisiones	
Tipo de clase: Entidad	
Atributo	**Tipo**
observaciones	string
estado	string
fecha_revision	date
t_revisiones	Enum
Para cada responsabilidad:	
Nombre:	GetObservaciones():string
Descripción:	Devuelve el valor del atributo observaciones
Nombre:	GetEstado():string
Descripción:	Devuelve el valor del atributo estado
Nombre:	GetFecha_Realizacion():date
Descripción:	Devuelve el valor del atributo fecha de realización
Nombre:	GetTRevision():Enum

Descripción:	Devuelve los campos de la clase
Nombre:	SetObservaciones(observaciones):void
Descripción:	Cambia el estado del atributo observaciones
Nombre:	SetEstado(estado):void
Descripción:	Cambia el estado del atributo estado
Nombre:	SetFecha_Realizacion(fecha_realizacion):void
Descripción:	Cambia el estado del atributo fecha de realización
Nombre:	SetTRevision(trevision):void
Descripción:	Cambia el estado del atributo tipo de revisiones

Nombre: E_TLocal	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_TLocal()
Descripción:	Constructor de la clase por defecto
Nombre:	E_TLocal(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_Locales

Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_Locales()
Descripción:	Constructor de la clase por defecto
Nombre:	E_Locales(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_TSillonesH	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_TSillonesH()
Descripción:	Constructor de la clase por defecto
Nombre:	E_TSillonesH (descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void

Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

La clase modelo Gestionar Sillones Hemodiálicos

Nombre: M_Gestionar_SillonesH	
Tipo de clase : Modelo	
Para cada responsabilidad:	
Nombre:	RegistrarSillon(E_SillonH): void
Descripción:	Adiciona un nuevo un sillón en el centro de diálisis
Nombre:	RegistrarRevision (E_Revisiones): void
Descripción:	Registra una nueva revisión.
Nombre:	ModificarDatosSillon (E_SillonesH):void
Descripción:	Modifica los datos de un sillon
Nombre:	ModificarDatosRevision (E_Revisiones):void
Descripción:	Modifica los datos de una revisión.
Nombre:	MostrarTiempoExplotacion():double
Descripción:	Consiste en mostrar el tiempo de explotación de un sillón.

- **Descripción de las clases del Caso de Uso Gestionar Riñón Artificial**

Nombre: E_riñon

Tipo de clase: Entidad	
Atributo	**Tipo**
horas_empleo	string
marca	string
tipo	string
noserie	string
Para cada responsabilidad:	
Nombre:	Gethoras_empleo ():string
Descripción:	Devuelve el valor del atributo horas de empleo
Nombre:	Getmarca():string
Descripción:	Devuelve el valor del atributo marca
Nombre:	Gettipo ():string
Descripción:	Devuelve el valor del atributo tipo
Nombre:	Getnoserie ():string
Descripción:	Devuelve el valor del atributo número de serie
Nombre:	Getdestino():string
Descripción:	Devuelve el valor del atributo destino
Nombre:	Sethoras_empleo (horas_empleo)::void
Descripción:	Cambia el estado del atributo horas_empleo
Nombre:	Setmarca(marca)::void
Descripción:	Cambia el estado del atributo marca
Nombre:	Settipo (tipo)::void
Descripción:	Cambia el estado del atributo tipo
Nombre:	Setnoserie (noserie)::void
Descripción:	Cambia el estado del atributo número de serie
Nombre:	Setdestino(destino)::void
Descripción:	Cambia el estado del atributo destino

Nombre: E_Revisiones	
Tipo de clase: Entidad	
Atributo	**Tipo**
observaciones	string
estado	string

fecha_revision t_revisiones	date Enum
Para cada responsabilidad:	
Nombre:	GetObservaciones():string
Descripción:	Devuelve el valor del atributo observaciones
Nombre:	GetEstado():string
Descripción:	Devuelve el valor del atributo estado
Nombre:	GetFecha_Realizacion():date
Descripción:	Devuelve el valor del atributo fecha de realización
Nombre:	GetTRevision():Enum
Descripción:	Devuelve el valor del atributo tipo de revisión
Nombre:	SetObservaciones(observaciones):string
Descripción:	Cambia el estado del atributo observaciones
Nombre:	SetEstado(estado):string
Descripción:	Cambia el estado del atributo estado
Nombre:	SetFecha_Realizacion(fecha_realizacion):date
Descripción:	Cambia el estado del atributo fecha de realización
Nombre:	SetTRevision(trevision):Enum
Descripción:	Cambia el estado del atributo tipo de revisiones

Nombre: E_TLocal	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_TLocal()
Descripción:	Constructor de la clase por defecto
Nombre:	E_TLocal(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string

Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_Locales	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_Locales()
Descripción:	Constructor de la clase por defecto
Nombre:	E_Locales(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_Virologia
Tipo de clase: Entidad

Atributo	**Tipo**
id	int
descripción	string
eliminado	bool
Para cada responsabilidad:	
Nombre:	E_Virologia()
Descripción:	Constructor de la clase por defecto
Nombre:	E_Virologia (descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

- **Descripción de las clases del Caso de Uso Gestionar Dializadores**

Nombre: E_Locales	
Tipo de clase: Entidad	
Atributo	**Tipo**
id	int
descripción	string
eliminado	bool
Para cada responsabilidad:	
Nombre:	E_Locales()
Descripción:	Constructor de la clase por defecto
Nombre:	E_Locales(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string

Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_TLocal	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_TLocal()
Descripción:	Constructor de la clase por defecto
Nombre:	E_TLocal(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_Esterelizantes	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción	int string

eliminado	bool
Para cada responsabilidad:	
Nombre:	E_Esterelizantes()
Descripción:	Constructor de la clase por defecto
Nombre:	E_Esterelizantes(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_Insumos	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_Insumos()
Descripción:	Constructor de la clase por defecto
Nombre:	E_Insumos(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_TInsumo	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_TInsumo()
Descripción:	Constructor de la clase por defecto
Nombre:	E_TInsumo(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_ Paquete_Salida	
Tipo de clase: Entidad	
Atributo	**Tipo**
causa fecha salida	E_ CausaD date
Para cada responsabilidad:	
Nombre:	GetCausa(): E_ CausaD
Descripción:	Devuelve los campos de la clase
Nombre:	Get_FechaSalida(): date
Descripción:	Devuelve los campos de la clase
Nombre:	SetCausa (): void
Descripción:	Cambia el estado del atributo causa

Nombre:	Set_FechaSalida(): void
Descripción:	Cambia el estado del atributo fecha salida

Nombre: E_Fabricante	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_Fabricante()
Descripción:	Constructor de la clase por defecto
Nombre:	E_Fabricante(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_ CausaD	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_ CausaD()
Descripción:	Constructor de la clase por defecto
Nombre:	E_ CausaD(descripción)

Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_Paquetes	
Tipo de clase: Entidad	
Atributo	**Tipo**
cantidad actual lote fecha realización fecha vencimiento	int string date date
Para cada responsabilidad:	
Nombre:	Get_Lote(): string
Descripción:	Devuelve el valor del atributo lote
Nombre:	Get_FechaRealizacion(): date
Descripción:	Devuelve el valor del atributo fecha de realización
Nombre:	Get_FechaVecimiento() :date
Descripción:	Devuelve el valor del atributo fecha de vencimiento
Nombre:	Get_Fabricante(): string
Descripción:	Devuelve los objetos que se utilizan de la clase E_Fabricante
Nombre:	Get_TipoInsumo(): string
Descripción:	Devuelve los objetos que se utilizan de la clase E_TInsumo
Nombre:	SetCant_Actual(): void
Descripción:	Permite cambiar el estado del atributo cantidad
Nombre:	SetLote(): void
Descripción:	Permite cambiar el estado del atributo lote

Nombre: E_ CausaD	
Tipo de clase: Entidad	
Atributo	**Tipo**
id descripción eliminado	int string bool
Para cada responsabilidad:	
Nombre:	E_ CausaD()
Descripción:	Constructor de la clase por defecto
Nombre:	E_ CausaD(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	Get_Descripcion(): string
Descripción:	Devuelve el valor del atributo descripción
Nombre:	Set_Descripcion(descripción): void
Descripción:	Permite cambiar el valor del atributo descripción
Nombre:	Get_Eliminado(): bool
Descripción:	Devuelve el valor del atributo eliminado
Nombre:	Set_Eliminado(): void
Descripción:	Permite cambiar el valor del atributo eliminado

Nombre: E_Reuso	
Tipo de clase: Entidad	
Atributo	**Tipo**
fecha_reuso volumen_cebado_posd	date double
Para cada responsabilidad:	
Nombre:	E_ Reuso()
Descripción:	Constructor de la clase por defecto
Nombre:	E_ Reuso(descripción)
Descripción:	Constructor de la clase por parámetros
Nombre:	GetFecha_Reuso():date

Descripción:	Devuelve el valor del atributo fecha_reuso
Nombre:	SetFecha_Reuso(fecha_reuso): void
Descripción:	Permite cambiar el valor del atributo fecha_reuso
Nombre:	GetVolumen_cebado_posd(): double
Descripción:	Devuelve el valor del atributo volumen de cebado
Nombre:	SetVolumen_cebado_posd (volumen_cebado_posd): void
Descripción:	Permite cambiar el valor del atributo volumen de cebado

Nombre: E_PacienteD	
Tipo de clase: Entidad	
Atributo	**Tipo**
id	int
Para cada responsabilidad:	
Nombre:	E_ PacienteD()
Descripción:	Constructor de la clase por defecto
Nombre:	GetId():int
Descripción:	Devuelve el valor del atributo id

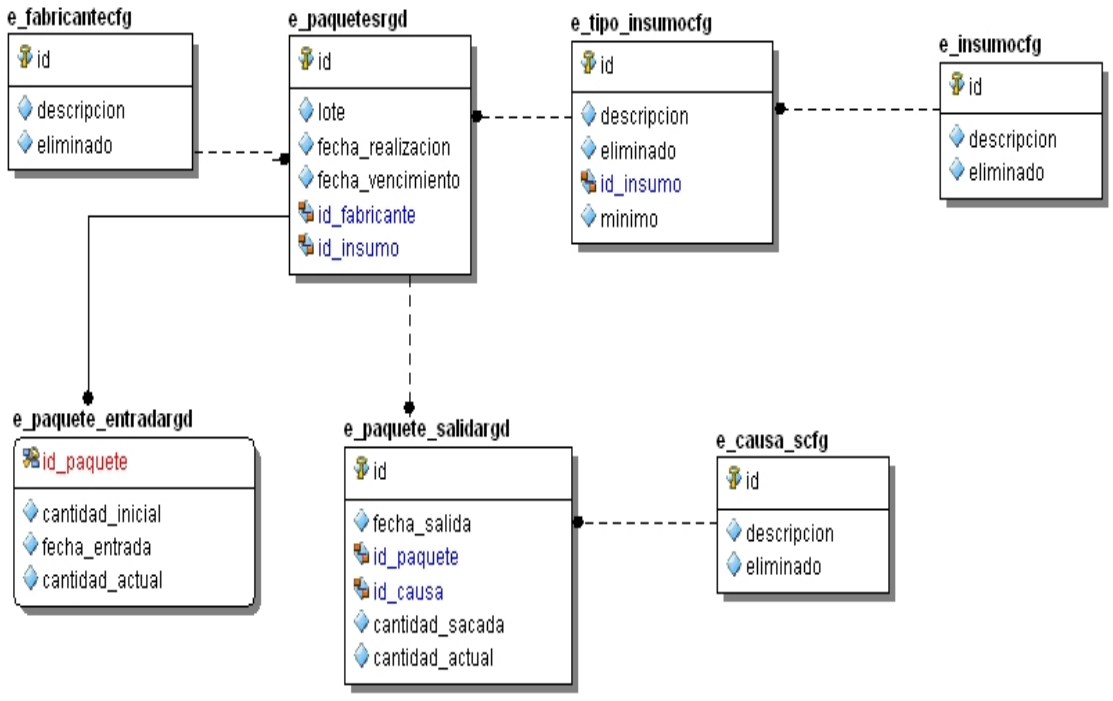
Nombre: E_Dializador	
Tipo de clase: Entidad	
Atributo	**Tipo**
fecha_inicio fecha_fin volumen_cebado_posd veces_usado	date date double int
Para cada responsabilidad:	
Nombre:	E_ Dializador()
Descripción:	Constructor de la clase por defecto
Nombre:	E_ Dializador(fecha_inicio,fecha_fin,volumen_ cebado_posd,veces_usado)
Descripción:	Constructor de la clase por parametros
Nombre:	GetFecha_inicio ():date
Descripción:	Devuelve el valor del atributo fecha_inicio

Nombre:	SetFecha_inicio(fecha_inicio): void
Descripción:	Permite cambiar el valor del atributo fecha_inicio
Nombre:	GetFecha_fin ():date
Descripción:	Devuelve el valor del atributo fecha_fin
Nombre:	SetFecha_fin(fecha_fin): void
Descripción:	Permite cambiar el valor del atributo fecha_fin
Nombre:	GetVolumen_cebado ():double
Descripción:	Devuelve el valor del atributo volumen_cebado
Nombre:	SetVolumen_cebado (volumen_cebado): void
Descripción:	Permite cambiar el valor del atributo volumen_cebado
Nombre:	GetVeces_usado ():int
Descripción:	Devuelve el valor del atributo veces_usado
Nombre:	SetVeces_usado(veces_usado): void
Descripción:	Permite cambiar el valor del atributo veces_usado

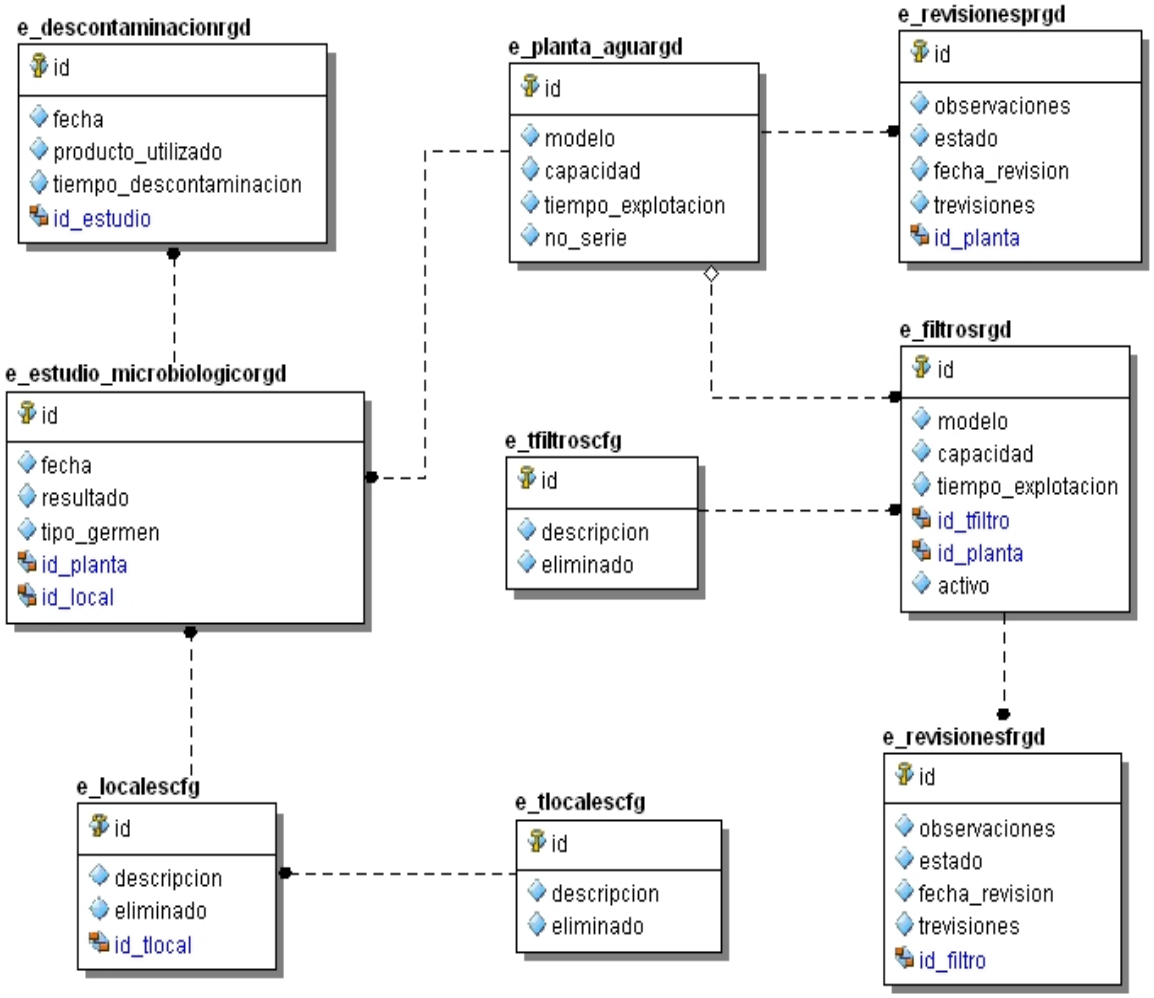
3.3 Diseño de la Base de Datos.

3.3.1 Diagrama Entidad Relación.

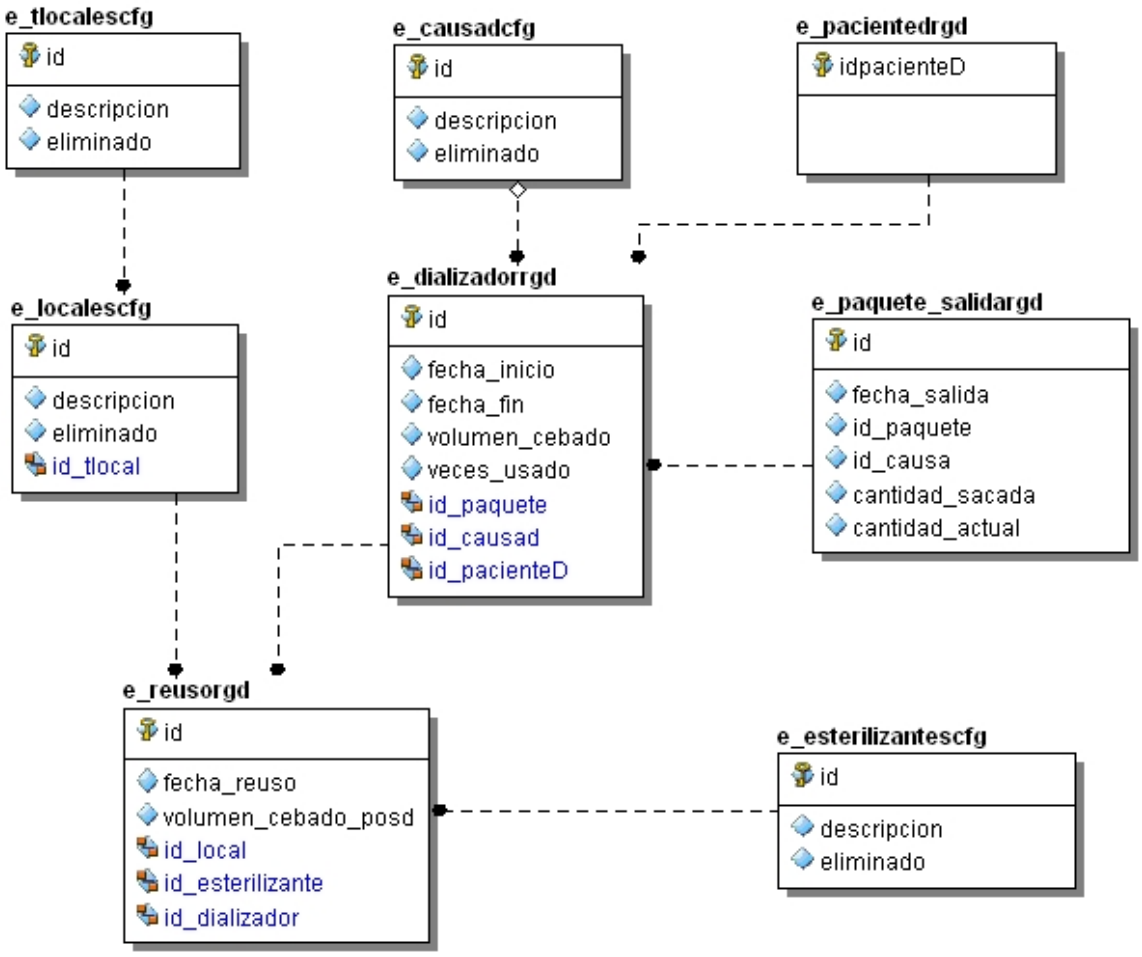
Caso de Uso Gestionar Insumos



Caso de Uso Gestionar Planta de Agua



Caso de Uso Gestionar Dializador



3.4 Descripción de las tablas de la Base de Datos.

A continuación se mostrarán las descripciones de algunas tablas de la base de datos (Las más usadas) .Las otras descripciones de las tablas se encontrarán en los anexos.

Descripción de las tablas.

Nombre: e_dializadorrgd		
Descripción: Se encarga de guardar los datos iníciales de un dializador.		
Atributo	**Tipo**	**Descripción**
id	integer	Guarda el identificador del dializador
id_paquete	integer	Guarda el identificador del paquete al que el dializador pertenece.
id_causaD	integer	Guarda un identificador de la causa.
id_pacienteD	integer	Guarda el identificador del paciente correspondiente.
fecha_inicio	date	Guarda la fecha de inicio de uso.
fecha_fin	date	Guarda la fecha de fin de uso.
volumen_cebado	double	Guarda el volumen de cebado.
veces_usado	integer	Guarda cantidad de veces de uso.

Nombre: e_equipos_medicoscfg		
Descripción: Se encarga de guardar los datos iníciales de un equipo medico.		
Atributo	**Tipo**	**Descripción**
id	integer	Guarda el identificador del equipo.
descripción	varchar	Guarda la descripción del equipo.
eliminado	tinyint	Guarda un resultado de si es

		eliminado.

Nombre: e_equipos_no_medicoscfg		
Descripción: Se encarga de guardar los datos iníciales de un equipo no medico.		
Atributo	**Tipo**	**Descripción**
id	integer	Guarda el identificador del equipo no médico.
descripción	varchar	Guarda la descripción del equipo no médico.
eliminado	tinyint	Guarda un resultado de si es eliminado.

Nombre: e_maquinareus		
Descripción: Se encarga de guardar los datos iníciales de un dializador.		
Atributo	**Tipo**	**Descripción**
id	integer	Guarda el identificador del dializador.
descripción	varchar	Guarda la descripción del equipo.
eliminado	tinyint	Guarda un resultado de si es eliminado.

Nombre: e_paquetesrgd		
Descripción: Se encarga de guardar los datos iníciales de un dializador.		
Atributo	**Tipo**	**Descripción**
id	integer	Guarda el identificador del paquete
id_fabricante	integer	Guarda el identificador del fabricante al que pertenece el paquete.
id_insumo	integer	Guarda el identificador del insumo.
fecha_realizacion	date	Guarda la fecha de realización
fecha_vencimiento	date	Guarda la fecha de vencimiento.
lote	varchar	Guarda el lote al que pertenece.

En este capítulo se valoró la propuesta realizada los analistas se han presentado los diagramas de clases de algunos de los casos de uso y los casos de usos arquitectónicamente significativos del sistema. Se describieron las clases, los tipos de datos y las operaciones que se implementaron.

4 Conclusiones.

Después de haber realizado el trabajo, se llegó a la siguiente conclusión:

- Se realizó un estudio de las técnicas de programación, lenguajes y librerías para el desarrollo de la aplicación.
- Se implementó el módulo Registro de Aseguramientos para Diálisis (RAD) utilizando los patrones de diseños establecidos.
- Con la puesta en práctica del sistema se obtendrá mayor rapidez en el trabajo con los datos de los equipos médicos y no médicos.

5 Recomendaciones.

Para evitar que el sistema se vuelva obsoleto se hacen las siguientes recomendaciones:

- Brindar soporte a la aplicación en un período determinado de tiempo, no muy largo.
- Desarrollar completamente el módulo de reportes que la aplicación necesita.
- Utilizar las nuevas herramientas como Yahoo User Interface para las futuras versiones.
- Capacitar al personal necesario para el trabajo con la aplicación.

6 Referencias Bibliográficas.

[1]-NefroNet. [En línea] [Citado el: 10 de 12 de 2007.] http://www.renal.org.ar/calidad/informaticos.htm.

[2]-NefroSoft. [En línea] [Citado el: 05 de 02 de 2008.] http://www.nefrosoft.com/

[3]-Gutierrez, Jorge A. Saavedra. Patrones GRASP. [En línea] [Citado el: 14 de 02 de 2008.] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.

[4]-Rivero, Jorge Alid Erazo. Patrones GRASP. [En línea] [Citado el: 22 de 01 de 2008.] http://jorgeerazo.blogspot.com/2006/08/patrones-grasp.html.

[5]-Umbrelo. [En línea] 2006. http://es.wikipedia.org/wiki/Umbrello

[6]-Extreme Programming. [En línea] [Citado el: 04 de 02 de 2008.] http://ele-zeta.com.ar/2004/08/27/extreme-programming-xp/.

[7]-Microsoft Solution Framework. [En línea] [Citado el: 04 de 02 de 2008.] http://www.gpicr.com/msf.aspx.

[8]-Paradigm, Visual. Visual Paradigm UML for Enterprise Edition. [En línea] [Citado el: 05 de 06 de 2008.] http://www.visual-paradigm.com/product/vpuml/.

[9]- Ídem a la Referencia 6.

[10]. Molpeceres, Alberto. Procesos de desarrollo, RUP,XP y FDD. [En línea] 15 de 12 de 2002. [Citado el: 15 de 03 de 2008.] http://www.willydev.net/InsiteCreation/v1.0/Descargas/articulos/general/cualxpfddrup.pdf.

[11]. Modelo Vista Controlador. [En línea] [Citado el: 15 de 03 de 2008.] http://java.sun.com/blueprints/patterns/MVC-detailed.html.

[12]. Arquitectura Orientada a Servicio. [En línea] [Citado el: 15 de 03 de 2008.] http://www.logiclibrary.com/.

[13]- Ídem a la Referencia 10.

[14]. Paradigm, Visual. Visual Paradigm UML for Enterprise Edition. [En línea] [Citado el: 05 de 06 de 2008.] http://www.visual-paradigm.com/product/vpuml/.

[15]. Rational Rose Enterprise Edition. [En línea] [Citado el: 05 de 02 de 2008.] http://www.ciao.es/Rational_Rose_Enterprise_Edition__Opinion_612900.

[16]- Visual Architect. [En línea] [Citado el: 05 de 06 de 2008.] http://www.visual-paradigm.com/product/bpva/.

[17]-systems, Renal. Guía de reprocesamiento de dializadores para el paciente. [En línea] [Citado el: 05 de 02 de 2008.] http://www.minntech.com/renal/patient/50096392B.pdf.

[18]-autores, Colectivo de. *Buenas Prácticas en Hemodíalisis.* Ciudad Habana : Editora Política, 2003.

[19]-Rational Unified Process(RUP). [En línea] 2006. https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducción%20a%20RUP.doc.

[20] -Rational Rose: Procedimientos básicos para desarrollar un proyecto con UML . [En línea] 2006. http://www.vico.org/TallerRationalRose.pdf.

[21]-Programación Extrema. [En línea] 2007. http://deigote.blogspot.com/2006/03/extreme-programming.htm.

[22]Grady Booch, Ivar Jacobon, James Rumbaugh. *El Lenguaje Unificado de Modelado.* s.l. : Addison Wesley, 2000. 9788478290444

[23]- Ídem a la Referencia 22.

[24]-Enterprise Architect. [En línea] [Citado el: 05 de 02 de 2008.] http://www.sparxsystems.com.ar/products/ea_features.html.

[25]-Gil, Sandra Victoria Hurtado. *Representación de la arquitectura de software usando UML.* 2000.

[26]-Garzón, Darwin Jiménez. RUP. [En línea] [Citado el: 05 de 02 de 2008.] http://codeticainge.googlepages.com/guiaing.pdf..

[27]-UML. [En línea] [Citado el: 05 de 02 de 2008.] http://gidis.ing.unlpam.edu.ar/personas/glafuente/uml/uml.html.

7 Bibliografía.

1. Arquitectura en tres capas. [En línea] [Citado el: 05 de 02 de 2008.] http://www.linuxjournal.com/article/3508 .

2. Arquitectura Orientada a Servicio. [En línea] [Citado el: 05 de 02 de 2008.] http://www.logiclibrary.com/.

3. Arquitectura Basada en Componente. [En línea] [Citado el: 05 de 02 de 2008.] http://msguayaquil.com/blogs/julioc/archive/2006/05/08/Desarrollo-de-Software-Basado-en-Componentes.aspx.

4. autores, Colectivo de. *Buenas Prácticas en Hemodíalisis.* Ciudad Habana : Editora Política, 2003.

5. Grady Booch, Ivar Jacobon, James Rumbaugh. *El Lenguaje Unificado de Modelado.* s.l. : Addison Wesley, 2000. 9788478290444.

6. Larman, Craig. *UML y Patrones. Segunda Edición.* s.l. : Addison Wesley, 2002. 9788420534381.

7. —. *UML y Patrones. Primera Edición.* s.l. : Addison Weley, 1999.

8. Lovelle, Juan Manuel Cuevas. *Introducción a UML.* 1999.

9. PRESSMAN, R. S. *Ingeniería de Software. Un enfoque Práctico. Quinta.* La Habana : Editorial Félix Varela, 2005.

10. Programación Extrema. [En línea] 2007.http://deigote.blogspot.com/2006/03/extreme-programming.htm.

8 Anexos.

Anexo 1

Algunas Tablas del Módulo Configuración

Nombre: e_estado_equipocfg		
Descripción: Se encarga de guardar los datos iníciales del estado de un equipo.		
Atributo	**Tipo**	**Descripción**
id	integer	Guarda el identificador del estado de un equipo.
descripción	varchar	Guarda la descripción del estado de un equipo.
eliminado	tinyint	Guarda un resultado de si es eliminado.

Nombre: e_fabricantecfg		
Descripción: Se encarga de guardar los datos iniciales del fabricante.		
Atributo	**Tipo**	**Descripción**
id	integer	Guarda el identificador del fabricante.
descripción	varchar	Guarda la descripción del fabricante.
eliminado	tinyint	Guarda un resultado de si es eliminado.

Nombre: e_insumocfg		
Descripción: Se encarga de guardar los datos iníciales del insumo.		
Atributo	**Tipo**	**Descripción**

id	integer	Guarda el identificador del insumo.
descripción	varchar	Guarda la descripción del insumo.
eliminado	tinyint	Guarda un resultado de si es eliminado.

Nombre: e_localescfg		
Descripción: Se encarga de guardar los datos iníciales de un local.		
Atributo	**Tipo**	**Descripción**
id	integer	Guarda el identificador del local.
descripción	varchar	Guarda la descripción del local.
eliminado	tinyint	Guarda un resultado de si es eliminado.
Id_tlocal	integer	Guarda el identificador del tipo de local.

Nombre: e_marca_rinoncfg		
Descripción: Se encarga de guardar los datos iníciales de una marca de un riñon.		
Atributo	**Tipo**	**Descripción**
id	integer	Guarda el identificador de una marca de un riñon.
descripción	varchar	Guarda la descripción de una marca de un riñon.
eliminado	tinyint	Guarda un resultado de si es eliminado.

Nombre: e_modelo_planta_aguacfg
Descripción: Se encarga de guardar los datos iníciales de una planta de agua.

Atributo	Tipo	Descripción
id	integer	Guarda el identificador de la planta de agua.
descripción	varchar	Guarda la descripción de la planta de agua.
eliminado	tinyint	Guarda un resultado de si es eliminado.

Anexo 2

Algunas tablas propias del modulo RAD

Nombre: e_descontaminacionrgd		
Descripción: Se encarga de guardar los datos iníciales de una descontaminación.		
Atributo	**Tipo**	**Descripción**
id	integer	Guarda el identificador de una descontaminación
id_estudio	integer	Guarda el identificador del estudio microbiológico
fecha_inicio	date	Guarda la fecha de inicio.
producto_utilizado	varchar	Guarda el nombre del producto utilizado.
tiempo_descontaminacion	double	Guarda cantidad de tiempo que demoro.

Nombre: e_filtrosrgd
Descripción: Se encarga de guardar los datos iníciales de un filtro

Atributo	Tipo	Descripción
id	integer	Guarda el identificador del filtro
id_tfiltro	integer	Guarda el identificador del tipo de filtro.
id_planta	integer	Guarda un identificador de la planta de agua.
modelo	varchar	Guarda el modelo del filtro.
capacidad	float	Guarda la capacidad del filtro.
tiempo_explotacion	Float	Guarda la cantidad de tiempo de explotación.
activo	tinyint	Guarda si esta activo o no.

9 Glosario de Términos.

A.

Apache:

Servidor de páginas Web de código abierto para diferentes plataformas (UNIX, Windows.)

B.

C.

D.

Diálisis:

Es un procedimiento que se realiza para retirar los elementos tóxicos (impurezas o desechos) de la sangre cuando los riñones no pueden hacerlo. La diálisis se puede llevar a cabo usando diferentes métodos: diálisis peritoneal y hemodiálisis.

H.

Hemodiálisis:

Es un método dialítico que se realiza al hacer circular la sangre a través de filtros especiales por fuera del cuerpo. La sangre fluye a través de una membrana semipermeable (dializador o filtro), junto con soluciones que ayudan a eliminar las toxinas. La hemodiálisis requiere un flujo de sangre de 400 a 500 mililitros por minuto, por lo que utiliza formas especiales para llevar la sangre a los vasos sanguíneos (vías de accesos)

HTTP: (HyperText Transfer Protocolo)

Protocolo cliente-servidor utilizado para el intercambio de páginas Web (HTML).

I.

INFOMED:

Nombre que identifica a la primera red electrónica cubana de información para la salud y surgió como parte de un proyecto del Centro Nacional de Información de Ciencias Médicas de Cuba. INFOMED es el Portal de Salud Cubano y la red de

personas e instituciones que comparten el propósito de facilitar el acceso a la información de salud en Cuba.

J.

***JavaScript*:**

Lenguaje de programación que se utiliza dentro del HTML. Lo interpreta el navegador y produce alguna acción determinada en la página Web donde está insertado. Java Script es un lenguaje basado en objetos y guiado por eventos, logrando con esto el dinamismo de las páginas que incluyan este tipo de código, útil para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet.

L.

O.

Open Source:

Código abierto o código libre. Software que distribuye de forma libre su código fuente, de forma que los desarrolladores pueden hacer variaciones, mejoras o reutilizarlo en otras aplicaciones. También conocido como *free software*.

P.

Programación Orientada a Objetos (POO):

La POO permite agrupar secciones de código con funcionalidades comunes. Con la programación orientada a objetos se pretende agrupar el código encapsulándolo y haciéndolo independiente, de manera que una modificación debida al crecimiento de la aplicación solo afecte a unas pocas líneas. El objetivo de POO es catalogar y diferenciar el código, en base a estructuras jerárquicas dependientes, al estilo de un árbol genealógico.

R.

RAD:

Registros de Aseguramientos para la Diálisis.

S.

SQL:

(Structured query language) Lenguaje de preguntas estructurado, lenguaje que utiliza bases de datos para pedir información de las mismas.

U.

W.

Web o WWW:

(World Wide Web) Telaraña o malla mundial. Sistema de información con mecanismos de hipertexto. Los usuarios pueden crear, editar y visualizar documentos de hipertexto. También llamado W3.

X.

XML:

Es el acrónimo de eXtensible Markup Language (lenguaje de marcado extensible) desarrollado por el World Wide Web Consortium (W3C).