Universidad de las Ciencias Informáticas

Facultad 7

Título: Implementación del módulo Registro de Enfermos Renales Crónicos.

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

Autores: Yanercy Zayas Genes.

Serguey Crespo Arias.

Tutores: Ing. Yanersy Díaz Colomé.

Ing. Renier Ricardo Figueredo.

Asesor: Ing. Jorge Carlos Yero

Ciudad de La Habana, Junio de 2008
"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

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

_____ _____

Yanercy Zayas Genes Serguey Crespo Arias

_____ _____

Yanersy Díaz Colomé Renier Ricardo Figueredo

Agradecimientos

Agradecemos:

A la Revolución Cubana por confiar en nosotros y en especial a nuestro invencible Comandante en Jefe por crear esta obra tan maravillosa, que es la UCI.

A nuestros padres, por ser nuestros creadores e inagotables fuentes de inspiración. Sin ellos siempre a nuestro lado, nuestros sueños no serían posibles.

A nuestros compañeros de aula, con los cuáles hemos compartidos muy buenos recuerdos y 5 años de estudios.

A la profesora Pura Miguel por su esmero, entrega y dedicación. Ella realmente es un Orgullo de Ser Maestro.

De Yanercy:

A mi madrecita santa que tanto la amo, a ella le debo todo lo que soy, gracias por dejarme ser tú. Te amo.

A mi familia, mis tíos, mis primos, mis amigos de la infancia, del pre-universitario, a todos, muchas gracias por siempre estar ahí para mi. Los adoro.

A mi novio Yordanis, por ser comprensivo y ofrecerme lo más bonito del mundo, el amor. Te quiero.

A mi compañero de tesis Serguey, por ser tan dedicado y preocupado.

De Serguey:

A todos los compañeros del proyecto que de una forma u otra me han brindado su apoyo, en especial a Dunior y a Daniel por su incondicional ayuda.

A toda mi familia que se ha preocupado tanto en que salga adelante.

A mí compañera de tesis por su apoyo y comprensión.

Especialmente, a todos mis amigos: Runer, Omar, Jorge Carlos, Daybert, Male, Lester, Josue, Mary, Yoky, Danaitis, Tranys, Oslo, Nay y Juan por compartir 5 maravillosos años aquí en la universidad.

Dedicatoria

De Yanercy:

A mis padres: Barbarita y Giraldo. Son mi vida.

A mis tíos: Mery, Alfredo, Rafelito y a mis primos.

A mi familia de Alamar. Los quiero mucho.

A mis amigos: Eduardito, Ashley, Michel, Ismaray, Isais, Anicia, Yusima, Adrian, Juan Guillermo, Ovidio, Dionis, Maykel, Jenny, Pedro Julio, Leyanny, Deyaniris, Kiki, Emilsis, Chadly, Yunet, Ana Sindy, Junior. Todos son parte de mí.

A mi grupo de baile Ilu Ashe, gracias por formar parte de mi sueño de baile, los quiero.

A mi novio querido, por ser una de las cosas más bonitas que me ha sucedido en la vida. Gracias por siempre estar ahí.

De Serguey:

A mis padres: Mario y Ada por brindame todo su apoyo y confianza, los AMO.

A toda mi familia: mis abuelos, tíos y primos.

A todas los que han hecho posible que haya logrado mi sueño.

A todos, muchas gracias.

RESUMEN

Cuba cuenta con 47 servicios nefrológicos que brindan atención médica a enfermos renales. Toda la información de estos se almacena en un archivo en la Dirección Nacional de Atención al Programa Enfermedad Renal, Diálisis y Trasplante; registrándose y actualizándose de forma manual. Esta situación provoca que puedan introducirse errores humanos en la información, ocasionando el retraso de un trasplante, la pérdida de órganos, o que el tratamiento indicado no sea el más óptimo.

El objetivo de este trabajo es implementar un sistema informático que mejore el proceso de gestión de la información de los pacientes con Enfermedad Renal Crónica atendidos por los servicios nefrológicos. Para el desarrollo de la aplicación, se utilizan tecnologías y lenguajes de programación de código abierto, que proporcionan librerías para la reutilización de código.

La aplicación digitaliza la Historia Clínica (HC) de los Enfermos Renales Crónicos. En ella se guardarán todos los datos necesarios del paciente, para lograr un buen diagnóstico de la enfermedad. Los datos abarcan la información general del paciente, así como todos los análisis a realizarse o los que se hará durante el transcurso del tratamiento. De esta forma, se facilitará el acceso de los médicos a la HC para darle un mejor seguimiento al paciente, en el servicio de nefrología o en otro. Además, posibilita agilizar la entrega de los partes e informes a la Dirección Nacional de Atención al Programa Enfermedad Renal, Diálisis y Trasplante.

Tabla de Contenidos

INTRODUCCIÓN..... 1

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA. 5

1.2. Tendencias y tecnologías actuales..... 7

1.3. Lenguajes de programación WEB 9

1.4. Navegadores WEB 14

1.5. Sistemas gestores de bases de datos (SGBD)..... 15

1.6. XML / Servicios WEB 19

1.7. Servidores WEB 21

1.8. Metodologías de desarrollo 22

1.9. Entornos de desarrollo integrado..... 27

1.10. Tecnología y herramientas a utilizar..... 29

CAPÍTULO 2. ELEMENTOS DE ARQUITECTURA..... 31

2.1. Requisitos no funcionales del sistema propuesto. 31

2.2. Estándares de codificación 34

2.3. Patrones de arquitectura 42

2.4. Vista de despliegue 48

2.5. Vista de implementación..... 50

CAPÍTULO 3. DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA. 57

3.1. Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados. 57

3.2. Descripción de clases RERC. 58

3.3. Diseño de la BD 66

3.4. Descripción de las tablas de la base de datos. 67

CONCLUSIONES 71

RECOMENDACIONES 72

BIBLIOGRAFÍA 77

GLOSARIO DE TÉRMINOS 83

ANEXOS 88

INTRODUCCIÓN

Por estos días, la aplicación de las Tecnologías de la Informática y las Comunicaciones, TIC, se ha hecho indispensable para el desarrollo de cualquier país. Resulta de gran importancia la informatización y automatización de todos los procesos cuya demanda sea alta y perfectamente sustituible por las tecnologías de la información.

Cuba está consciente de que una sociedad para ser más eficaz, eficiente y competitiva debe aplicar la informatización en todas sus esferas y procesos. En este sentido, se ha identificado desde muy temprano la conveniencia y necesidad de dominar e introducir en la práctica social las Tecnologías de la Información y las Comunicaciones y lograr una cultura digital como una de las características imprescindibles del hombre nuevo.

Desde los primeros años del Triunfo de la Revolución Cubana, fue una estrategia política e interés del gobierno revolucionario y el Ministerio de Salud Pública (MINSAP), el estudio y procesamiento de los hechos vitales y sanitarios, inicialmente de forma manual, después con equipos mecanizados IBM en algunas provincias y en el nivel central.

Al triunfo de la revolución en 1959, existía en el país un solo hospital para la atención nefrológica. Ya en 1961, se realiza la primera hemodiálisis en un enfermo portador de un fallo renal agudo. En 1966, se crea el Instituto de Nefrología (INEF), como cuna de la especialidad, centro de referencia. En 1968, comienza el tratamiento por diálisis en pacientes crónicos, el cuál, un año después, se extendía a 4 territorios del país. En 1970, se realizó el primer programa nacional de Nefrología para integrar el manejo de los pacientes, con los objetivos de diagnosticar, tratar científicamente, prevenir en lo posible el desarrollo de enfermedades renales, desarrollar las investigaciones, la docencia y preparar a los pacientes para un posible trasplante renal con vistas a una completa rehabilitación y reintegración a la vida social; se realizó el primer trasplante renal exitoso el 24 de febrero 1970. (1)

En 1992 se creó la Red Telemática de la Salud, INFOMED, dentro de la estructura del Centro Nacional de Información de Ciencias Médicas, utilizando los recursos disponibles y permitiendo enlazar a todo el sistema de salud, para dar una respuesta más eficiente en la esfera de la información científica a los profesionales y técnicos de la salud y a la situación sanitaria del país; dentro de esta red se encuentra una rama bien importante que es la nefrología, definida por el estudio de la estructura y función renal

tanto en la salud como en la enfermedad, incluyendo la prevención y tratamiento de las enfermedades que afectan al riñón y al tracto urinario. (2)

En la actualidad la demanda de servicios de diálisis y trasplante se ha incrementado, por lo que se está orientando a un nuevo método de salud renal donde se tratan factores como prevención primaria que vincula factores de riesgo, prevención secundaria enfocada en la detección de la enfermedad y el diagnóstico adecuado, y la prevención terciaria cuyo objetivo es retrasar el desarrollo de las complicaciones de la enfermedad existente. (3)

Para dar cumplimiento a dicho método los diferentes servicios nefrológicos se encuentran divididos en áreas que atienden a los enfermos según la categoría de su enfermedad. Específicamente el área de Enfermos Renales Crónicos (ERC) atiende de manera ambulatoria u hospitalaria a pacientes con insuficiencia renal crónica en etapa preterminal y con tratamiento dialítico iterado que presenten complicaciones. Para ello cuenta con consultas ambulatorias, cuerpo de guardia las 24 horas, diálisis de urgencia y con varias salas de hospitalización.

Aunque la atención a los pacientes con ERC se ha convertido en una tarea cada vez de mayor importancia para el MINSAP y se invierten a diario muchos recursos materiales y humanos en este sentido, aún subsisten problemas asociados a este proceso.

La gestión de la información referente a cada ERC se realiza de forma manual en el servicio nefrológico donde el mismo se atiende. Esto provoca que sea engorroso el proceso de búsqueda de alguna información específica del paciente, sobre todo si se desea medir la evolución de algún parámetro médico del mismo. Además si el paciente por algún motivo necesita atenderse de urgencia en otro servicio nefrológico que no sea el suyo, no habrá forma de que los especialistas de ese nuevo servicio conozcan toda la información del paciente que pueda resultar de interés en ese momento.

El estado de apto para trasplante de cada paciente se conoce en la Dirección Nacional de Atención al Programa Enfermedad Renal, Diálisis y Trasplante, mediante un parte telefónico que brinda cada servicio nefrológico. Este proceso es lento e introduce como consecuencia que no siempre se conozca el estado de aptitud de un paciente en tiempo real. La vida útil de un riñón para trasplante es de 24 horas, en ocasiones se llega a perder el órgano, porque se le asigna a una persona que realmente no está apta para ser trasplantada en ese momento, por lo que es de vital importancia hallar una solución

para darle facilidad, eficiencia y rapidez al trabajo que realiza el personal que labora en los distintos servicios de nefrología, ya que en ocasiones de esto depende la vida de muchos pacientes.

Dada la situación anterior se identificó el siguiente ***problema científico***: ¿Cómo facilitar el proceso de gestión de la información de los pacientes con Enfermedad Renal Crónica que se atienden en cada uno de los servicios nefrológicos del país?

Este problema se enmarca en el ***objeto de estudio*:** Proceso de gestión de la información de los pacientes con disfunciones renales.

El ***campo de acción*** abarcado es: Gestión de la información de los pacientes con Enfermedad Renal Crónica que se atienden en cada uno de los servicios nefrológicos del país.

Objetivo general*:* Implementar un sistema informático que mejore el proceso de gestión de la información de los pacientes con Enfermedad Renal Crónica que se atienden en cada uno de los servicios nefrológicos del país.

Las ***tareas*** que se llevarán a cabo para darle cumplimiento al objetivo trazado son:

1. Valorar la técnica de programación, lenguaje, plataforma y librerías propuesta por el cliente y la universidad, así como otras tecnologías más utilizadas en el desarrollo de aplicaciones web.
2. Obtener Modelo de Implementación y los artefactos necesarios que describan la Base de Datos.
3. Realizar el diseño de la Base de Datos y la implementación del módulo
4. Brindar una interfaz gráfica orientada al usuario y a las exigencias del cliente.

El presente trabajo de diploma consta de tres capítulos, el primero aborda toda la fundamentación teórica, donde se analizan las tecnologías, y las herramientas de programación a utilizar para el desarrollo de aplicaciones web, así como las plataformas de desarrollo que la soportan.

El segundo capítulo abarca detalladamente los elementos de la arquitectura propuesta por los analistas del módulo, se argumentan los requisitos no funcionales del sistema. Se explica la estrategia de codificación a utilizar, y los patrones arquitectónicos presentes en la propuesta de solución. También en este capitulo se muestra la vista de despliegue y la vista de implementación. El tercer capítulo comprende la descripción y análisis de la solución propuesta, en este se describen todas las clases necesarias para la implementación del sistema, además de las clases de diseño y las tablas de la base de datos de la aplicación.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.

En este capítulo, se definen las tecnologías y herramientas a utilizar en la implementación del sistema. También se realiza un análisis de los distintos sistemas que existen a nivel nacional e internacional, que responden a la misma problemática del presente trabajo.

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. (4)

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 Pacientes en 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 está 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. (5)

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. (6)

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 Único 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. (8)

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 como: agregar o modificar pacientes que se encuentran en el plan de crónicos, insertar en el sistema los pacientes que serán atendidos con todos los datos necesarios e importantes para su adecuada atención, así como modificar algún dato de estos; dar baja en el plan de crónicos, reingresar pacientes en el plan y agregar o modificar pacientes con necesidad de diálisis sin ser ERC.
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, 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; y en el caso del EMALEX que es un sistema nacional, no cumple con las expectativas del MINSAP.

1.2. Tendencias y tecnologías actuales

¿Qué es Internet?

Internet es una red mundial de computadoras con un conjunto de protocolos, el más destacado, el TCP/IP ("Transfer Control Protocol / Internet Protocol). Internet es definida en ocasiones como “La Red de Redes” o “La Autopista de la Información”. Ofrece distintos servicios, como el envío y recepción de correo electrónico (e-mail), la posibilidad de ver información en las páginas Web, de participar en foros de discusión (News), de enviar y recibir ficheros mediante FTP, de charlar en tiempo real, entre otras ventajas. (9)

Aplicaciones Web. Ventajas y desventajas

Una aplicación Web es un programa de computadora que el usuario invoca usando un navegador, el cual contacta algún servidor para ejecutar lógica de negocio o lo que es lo mismo: modificar el estado del negocio.

Las Aplicaciones Web utilizan las tecnologías existentes para generar contenidos dinámicos y permitir a los usuarios del sistema modificar la lógica del negocio en el servidor. Si no existe lógica de negocios en el servidor, el sistema no puede ser considerado una aplicación Web, en ese caso se considera como un sitio Web. En esencia una aplicación Web usa un sitio Web como entrada (front-end) a una aplicación típica.

Ventajas:

1. Extrapolación absoluta. El hecho de que todas las aplicaciones se realicen sobre Web, va a permitir que entre ellas se pueda compartir toda la información (principalmente gracias a XML). Esto garantiza:

    1.1. Propagación inmediata de contenido e información, que va a permitir el desarrollo de una mejor manera de la estructura de red.

    1.2. Uso de otras fuentes para desarrollar nuevas aplicaciones. Esta cuestión va a permitir que el desarrollo de nuevas aplicaciones se centre en la aportación de valor añadido, centrando los recursos en lo nuevo, y sacando partido de lo hecho por otros.

2. Aplicaciones (software) como servicio y no como producto.

3. Ubicación. La Web ya se ha consagrado como el canal de interoperabilidad por excelencia. Es decir, las aplicaciones basadas en Web pueden desarrollarse en cualquier terminal (y no necesariamente en los PC): ordenadores, móviles, PDAs, TV digital; esto va a permitir tener la información en todo momento y desde cualquier terminal con conexión a Internet.

Desventajas:

1. La conexión a Internet. La dependencia del sistema a la conexión de Internet sigue siendo una barrera a su adopción. (10)

1.3. Lenguajes de programación Web

Son herramientas que permiten crear programas y software. Los lenguajes de programación de una computadora en particular se conocen como código de máquinas o lenguaje de máquinas. Existen muchos lenguajes para el desarrollo de aplicaciones Web. Estos lenguajes se dividen en:

Lenguajes de programación del lado del servidor:

Son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Entre los lenguajes del lado del servidor más utilizados para el desarrollo de aplicaciones Web se tienen: el PHP, el ASP y el JSP, entre otros. A continuación se explican algunos de ellos.

PHP

Es un lenguaje de programación interpretado usado normalmente para la creación de páginas web dinámicas. Acrónimo recursivo que significa "PHP Hypertext Pre-processor" (inicialmente PHP Tools, o, Personal Home Page Tools). Actualmente también se puede utilizar para la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica. (11)

Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Mediante extensiones es también posible la generación de archivos PDF, Flash, así como imágenes en diferentes formatos.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

También tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX (y de ese tipo, como Linux o Mac OS X) y Windows, y puede interactuar con los servidores web más populares, ya que existe en versión CGI, módulo para Apache.

Aunque su creación y desarrollo se da en el ámbito de los sistemas libres, bajo la licencia GNU, existe además un IDE (entorno de desarrollo integrado) comercial llamado Zend Studio. Recientemente, CodeGear (la división de lenguajes de programación de Borland) ha sacado al mercado un entorno integrado de desarrollo para PHP, denominado Delphi for PHP.

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 cuál 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:

- No posee una abstracción de base de datos estándar, sino bibliotecas especializadas para cada motor (a veces más de una para el mismo motor).
- No posee adecuado manejo de internacionalización, Unicode (estándar industrial cuyo objetivo es proporcionar el medio por el cuál un texto en cualquier forma e idioma pueda ser codificado para el uso informático), entre otros.
- Por su diseño dinámico no puede ser compilado y es muy difícil de optimizar. (12)

ASP

Significa *Active Server Pages*; estas páginas pueden ser escritas en VBScript que es a su vez un derivado de Visual Basic.

Las páginas ASP pueden hacer uso de objetos COM (*Component Object Model*) que son objetos en algún otro lenguaje (ejemplo: ejecutables en C++ o Java); de manera que si ya se tiene algo programado las páginas ASP a través del Internet Information Server pueden hacer uso de los métodos en estos objetos.

Para conectarse a una base de datos, normalmente se utiliza ADO (Data Access Object) Objeto de acceso a datos, que es un adaptador universal a bases de datos que se especializa posteriormente para hablar con una base de datos concreta.

El esquema de trabajo es crear objetos COM que ejecutan la lógica de la aplicación (*Business Objects*) y luego hacer la capa de interfaz con este. (13)

Ventajas:

- Usa Visual Basic Script, siendo fácil para los usuarios.
- Comunicación óptima con SQL Server.
- Soporta el lenguaje JScript (JavaScript de Microsoft).

Desventajas:

- Código desorganizado.
- Se necesita escribir mucho código para realizar funciones sencillas.
- Tecnología propietaria. (14)
- Hospedaje de sitios web costosos

JSP

Significa *Java Server Pages* es una invención de la SUN que provee de un lenguaje de scripting en el lado del servidor que se comunica con clases Java, objetos RMI, CORBA, entre otros. La metodología de trabajo esperada es la misma que con Visual Basic con la diferencia de que esta vez se trata de una plataforma mucho más abierta.

El código JSP se puede poner dentro de las páginas HTML, o se puede precompilar en Servlets. Un servlet es una subclase de servlet que tiene métodos para atender requerimientos. El servidor pasa las variables hacia y desde el ambiente de ejecución del Servlet.

Para programar en JSP se requiere conocer Java, ser metódico y ordenado. El lenguaje no relaja las condiciones de tipado que le son propias. (15)

Ventajas:

- Ejecución rápida del servlets.
- Crear páginas del lado del servidor.
- Multiplataforma.
- Código bien estructurado.
- Integridad con los módulos de Java.
- La parte dinámica está escrita en Java.

Desventajas:

- Complejidad de aprendizaje. (16)

Lenguajes del lado del cliente:

Son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un pre-tratamiento, como son el HTML, el Java y el JavaScript, entre otros, los cuales son simplemente incluidos en el código HTML.

HTML

Hyper Text Markup Language (Lenguaje de marcado e Hipertexto) es el lenguaje de marcas de texto utilizado normalmente en la www (World Wide Web). HTML no es propiamente un lenguaje de programación como C++, Visual Basic, u otros, sino un sistema de etiquetas. No presenta ningún compilador, por lo tanto algún error de sintaxis que se presente, éste no lo detectará y se visualizará en la forma como éste lo entienda. (17)

El entorno para trabajar HTML es simplemente un procesador de texto, como el que ofrecen los sistemas operativos Windows (Bloc de notas), UNIX (el editor vi o ed) o el que ofrece Microsoft Office (Word). El conjunto de etiquetas que se creen, se deben guardar con la extensión .htm o .html.

Estos documentos pueden ser mostrados por los visores o browsers de paginas Web en Internet, como Netscape Navigator, Mosaic, Opera y Microsoft Internet Explorer. (18)

Ventajas:

- Sencillo, permite describir hipertexto.
- Texto presentado de forma estructurada y agradable.
- No necesita de grandes conocimientos cuando se cuenta con un editor de páginas web.
- Archivos pequeños.
- Despliegue rápido.
- Lenguaje de fácil aprendizaje.
- Lo admiten todos los exploradores.

Desventajas:

- Lenguaje estático.
- La interpretación de cada navegador puede ser diferente.
- Guarda muchas etiquetas que pueden convertirse en “basura” y dificultan la corrección.
- El diseño es más lento.
- Las etiquetas son muy limitadas.

Java

Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros. (19)

Características:

1-Orientado a objetos (OO)

Se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que use estén unidos a sus operaciones.

2-Independencia de la plataforma

Significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Es lo que significa ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo. (20)

JavaScript

Este es un lenguaje interpretado, no requiere compilación. Fue creado por Brendan Eich en la empresa Netscape Communications. Utilizado principalmente en páginas web. Es similar a Java, aunque no es un lenguaje orientado a objetos, el mismo no dispone de herencias. La mayoría de los navegadores en sus últimas versiones interpretan código JavaScript.

El código JavaScript puede ser integrado dentro de nuestras páginas web. Para evitar incompatibilidades el World Wide Web Consortium (W3C) diseñó un estándar denominado DOM (en inglés Document Object Model, en su traducción al español Modelo de Objetos del Documento). (21)

Ventajas:

- Lenguaje de scripting seguro y fiable.
- Los script tienen capacidades limitadas, por razones de seguridad.
- El código JavaScript se ejecuta en el cliente.

Desventajas:

- Código visible por cualquier usuario.
- El código debe descargarse completamente.
- Puede poner en riesgo la seguridad del sitio, con el actual problema llamado XSS (significa en inglés Cross Site Scripting renombrado a XSS por su similitud con las hojas de estilo CSS). (22)

1.4. Navegadores Web

Un navegador web o explorador web (del inglés, navigator o browser) es una aplicación que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web de todo el mundo a través de Internet.

La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados. Los documentos pueden estar ubicados en la computadora en donde está el usuario, pero también pueden estar en cualquier otro dispositivo que esté conectado a la computadora del usuario o a través de Internet, y que tenga los recursos necesarios para la transmisión de los documentos (un software servidor web).

El seguimiento de enlaces de una página a otra, ubicada en cualquier computadora conectada a la Internet, se llama navegación; que es de donde se origina el nombre de navegador. Por otro lado, hojeador es una traducción literal del original en inglés, browser, aunque su uso es minoritario. (23)

Mozilla Firefox

Es un navegador de Internet, con interfaz gráfica de usuario, desarrollado por la Corporación Mozilla. El programa es multiplataforma y está disponible en versiones para Microsoft Windows, Mac OS X y GNU/Linux. El código fuente de Firefox está disponible libremente bajo la triple licencia de Mozilla como un programa libre y de código abierto. (24)

Características: (25)

1. Pestañas. La navegación mediante pestañas consiste en poder abrir en una sola ventana del programa varias páginas a la vez, pudiendo ir de una a otra a través de las pestañas (o fichas). De este modo, la navegación resulta más cómoda y organizada y se consumen menos recursos en el equipo.

2. Bloqueador de ventanas emergentes (popups). Firefox incluye un bloqueador de ventanas emergentes integrado personalizable; por defecto, bloquea todas las ventanas emergentes que considere no solicitadas de cualquier página. Además, permite definir el nivel de protección ante las ventanas emergentes en cada caso.

3. Compatibilidad con estándares de programación. Mozilla Firefox es compatible con varios estándares web, incluyendo HTML, XML, XHTML, SVG, entre otros.

4. Marcadores. Firefox incluye la opción de almacenar sitios de la preferencia del usuario, lo que facilita la navegación de sitios visitados con frecuencia. Además, dentro de los marcadores, está la opción de palabra clave (keyword) para que el usuario escriba sólo una palabra en la barra de direcciones y el navegador comience a cargar el sitio. Puede añadirse a Firefox una barra de herramientas con algunos marcadores con los fines de acceder a los sitios correspondientes de manera más rápida. Una ventaja de los keyword es el reemplazo de texto, ideal para búsquedas rápidas.

1.5. Sistemas Gestores de Bases de Datos (SGBD)

Es un conjunto de programas que permiten crear y mantener una Base de Datos, asegurando su integridad, confidencialidad y seguridad. Por tanto debe permitir:

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

Algunas de las características deseables en un Sistema Gestor de base de datos SGBD son: (26)

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

PostGreSQL

Es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. El director de este proyecto es el profesor Michael Stonebraker, y fue patrocinado por Defense Advanced Research Projects Agency (DARPA), el Army Research Office (ARO), el National Science Foundation (NSF), y ESL, Inc. (27)

Características de PostGreSQL (28)

1. Implementación del estándar SQL92/SQL99.
2. Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, entre otros. También permite la creación de tipos propios.
3. Incorpora una estructura de datos array.
4. Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, entre otras.
5. Permite la declaración de funciones propias, así como la definición de disparadores.
6. Soporta el uso de índices, reglas y vistas.
7. Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales.
8. Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.

MySQL

Es un sistema de gestión de bases de datos relacionales, licenciado bajo la GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. Fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca.

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Características de MySQL (29)

Las principales características de este gestor de bases de datos son las siguientes:

1. Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
2. Soporta gran cantidad de tipos de datos para las columnas.
3. Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, entre otros).
4. Gran portabilidad entre sistemas.
5. Soporta hasta 32 índices por tabla.
6. Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

Oracle

Es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos. Es un producto vendido a nivel mundial, aunque la gran potencia que tiene y su elevado precio hacen que sólo se vea en empresas muy grandes y multinacionales, por norma general. En el desarrollo de páginas web pasa lo mismo: como es un sistema muy caro no está tan extendido como otras bases de datos, por ejemplo, Access, MySQL, SQL Server, entre otros. (30)

Oracle es un SGBD fabricado por la Corporación Oracle. Se considera como uno de los sistemas de bases de datos más completos, destacándose entre sus características generales: (31)

1. Soporte de transacciones, que no son más que una interacción con una estructura de datos que, aún siendo compleja y estando compuesta por varios procesos que se han de aplicar uno después del otro, se pretende que sea equivalente a una interacción atómica. Es decir, que se realice de una sola vez.
2. Estabilidad, o sea, que su nivel de fallos disminuye, en dependencia de la estabilidad que se requiera.
3. Escalabilidad, es decir, su posibilidad de estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos, siendo capaz de cambiar su tamaño o configuración para adaptarse a las circunstancias cambiantes.
4. Es multiplataforma.

SQL Server

Microsoft SQL Server es un sistema de gestión de bases de datos relacionales (SGBD) capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, PostgreSQL o MySQL. Es una plataforma global de base de datos que ofrece administración de datos empresariales con herramientas integradas de inteligencia empresarial. El motor de la BD SQL Server ofrece almacenamiento más seguro y confiable tanto para datos relacionales como estructurados, lo que le permite crear y administrar aplicaciones de datos altamente disponibles.

Entre sus características figuran:

1. Soporte de transacciones.
2. Gran estabilidad
3. Gran seguridad
4. Escalabilidad

Una de sus más significativas desventajas es la condición de software propietario, y hoy la tendencia es usar software libre por las librerías que ofrece. Además no es multiplataforma, pues sólo está disponible en Sistemas Operativos de Microsoft. (32)

1.6. XML / Servicios Web

XML

Es un lenguaje de marcado que ofrece un formato para la descripción de datos estructurados. Esto facilita unas declaraciones de contenido más precisas y unos resultados de búsquedas más significativos en varias plataformas. Además, XML habilitará una nueva generación de aplicaciones para ver y manipular datos basadas en el Web.

Representación estructural de los datos:

- XML ofrece una representación estructural de los datos que se puede implementar ampliamente y es fácil de distribuir. Definido por el World Wide Web Consortium (W3C) (en inglés), XML garantiza que los datos estructurados sean uniformes e independientes de aplicaciones o fabricantes. La interoperabilidad resultante está creando rápidamente una nueva generación de aplicaciones de comercio electrónico en la Web.
- XML, que proporciona un estándar de datos que puede codificar el contenido, la semántica y los esquemas de una gran variedad de casos, desde los más simples a los más complejos; sirve para marcar lo siguiente:
  - Un documento normal.
  - Un registro estructurado, como un registro de citas o un pedido de compra.
  - Un objeto con datos y métodos, como el formulario permanente de un objeto Java o de un control ActiveX.
  - Un registro de datos, como el conjunto de resultados de una consulta.
  - Metacontenido sobre un sitio Web, como el formato de definición de canal (CDF).
  - Representaciones gráficas, como la interfaz de usuario de una aplicación.
  - Entidades y tipos de esquema estándar.
  - Todos los vínculos entre datos y personas que hay en la Web.
- Cuando los datos llegan al escritorio del cliente, se pueden manipular, editar y presentar en varias vistas, sin tener que regresar al servidor. Ahora los servidores pueden ser más escalables, gracias a la reducción de las cargas de ancho de banda y computación. Además, dado que los datos se intercambian en el formato XML, se pueden combinar fácilmente desde distintas fuentes.
- XML es muy valioso para Internet, así como para los entornos de intranets corporativas de gran tamaño, pues proporciona interoperabilidad mediante un formato basado en estándares flexible

y abierto, con formas nuevas de acceso a las bases de datos existentes y de entregar datos a clientes de Web. Las aplicaciones se pueden generar más rápidamente, su mantenimiento es más sencillo y pueden ofrecer fácilmente varias vistas de los datos estructurados.

- En un principio, no rivalizarán HTML y XML, estos se complementarán el uno al otro, anudándose ambas gramáticas. Este Lenguaje de marcas extensible (XML) es una versión abreviada del SGML (Standard Generalized Markup Language). (33)

Servicios Web (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. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares. (34)

Ventajas de los servicios Web:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

Inconvenientes de los servicios Web

- Para realizar transacciones no pueden compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA.
- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como RMI o CORBA (XML no está diseñado para el rendimiento)

- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewalls cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.
- Existe poca información de servicios web para algunos lenguajes de programación.

Protocolos conocidos

- XML: Es el formato estándar para los datos que se vayan a intercambiar.
- SOAP o XML-RPC: Protocolos sobre los que se establece el intercambio.
- HTTP, FTP, o SMTP: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales ya bien conocidos.
- WSDL: Es el lenguaje de la interfaz pública para los servicios Web.
- UDDI: Protocolo para publicar la información de los servicios Web.
- WS-Security: Protocolo de seguridad aceptado como estándar por OASIS. (35)

1.7. Servidores Web

Apache

Es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa.

Es un proyecto nacido para crear un servidor web estable, fiable y veloz para plataformas Unix. Esta licencia permite hacer múltiples funciones con el código fuente. Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.

Es una tecnología gratuita fuente de código abierto. Esto le da transparencia al software, de manera que permite ver lo que se está instalando en el servidor, sin necesidad de acceder a través de puertas traseras.

Este servidor es un software que está estructurado en módulos, Módulos Base, Módulos Multiproceso y Módulos Adicionales. El Módulo Multiproceso se ha modificado en varias ocasiones para optimizar su rendimiento y rapidez en los distintos sistemas operativos sobre los que se puede ejecutar Apache. El resto de las funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar con solo ser añadidos al mismo.

Ventajas: (36)

- Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar para que ejecute un determinado script cuando ocurra un error en concreto.

- Tiene una alta configurabilidad en la creación y gestión de logs. Permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor.

1.8. Metodologías de Desarrollo

RUP

El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.
El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización. (37)

Visual Paradigm para UML

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software de modelado, ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales, demostraciones interactivas y proyectos. La versión 3.0 del Visual Paradigm tiene soporte para UML versión 6.0. (38)

Modelo de físico de la Base de datos.

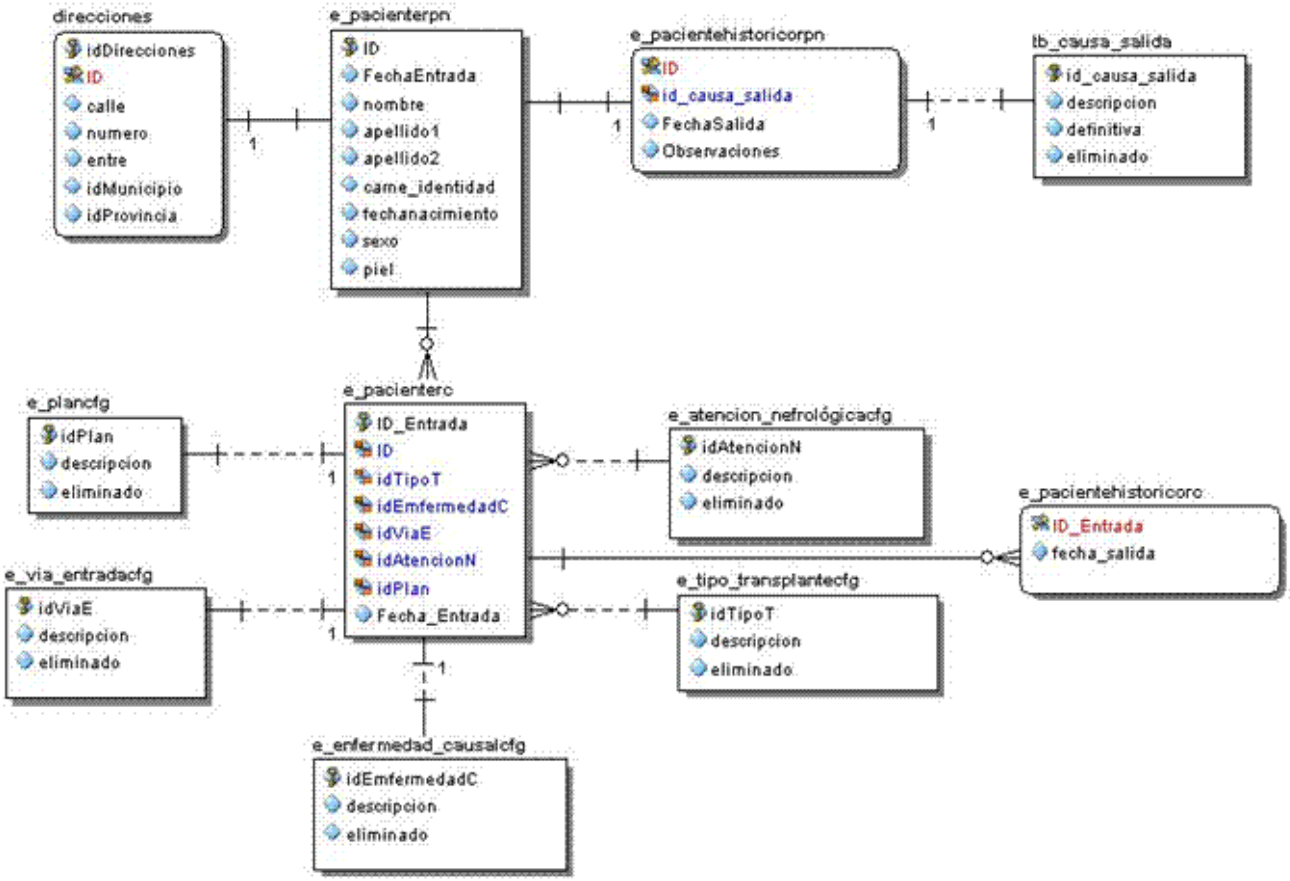


Fig.1: Modelo Físico de la Base de datos, Movimiento de Pacientes.

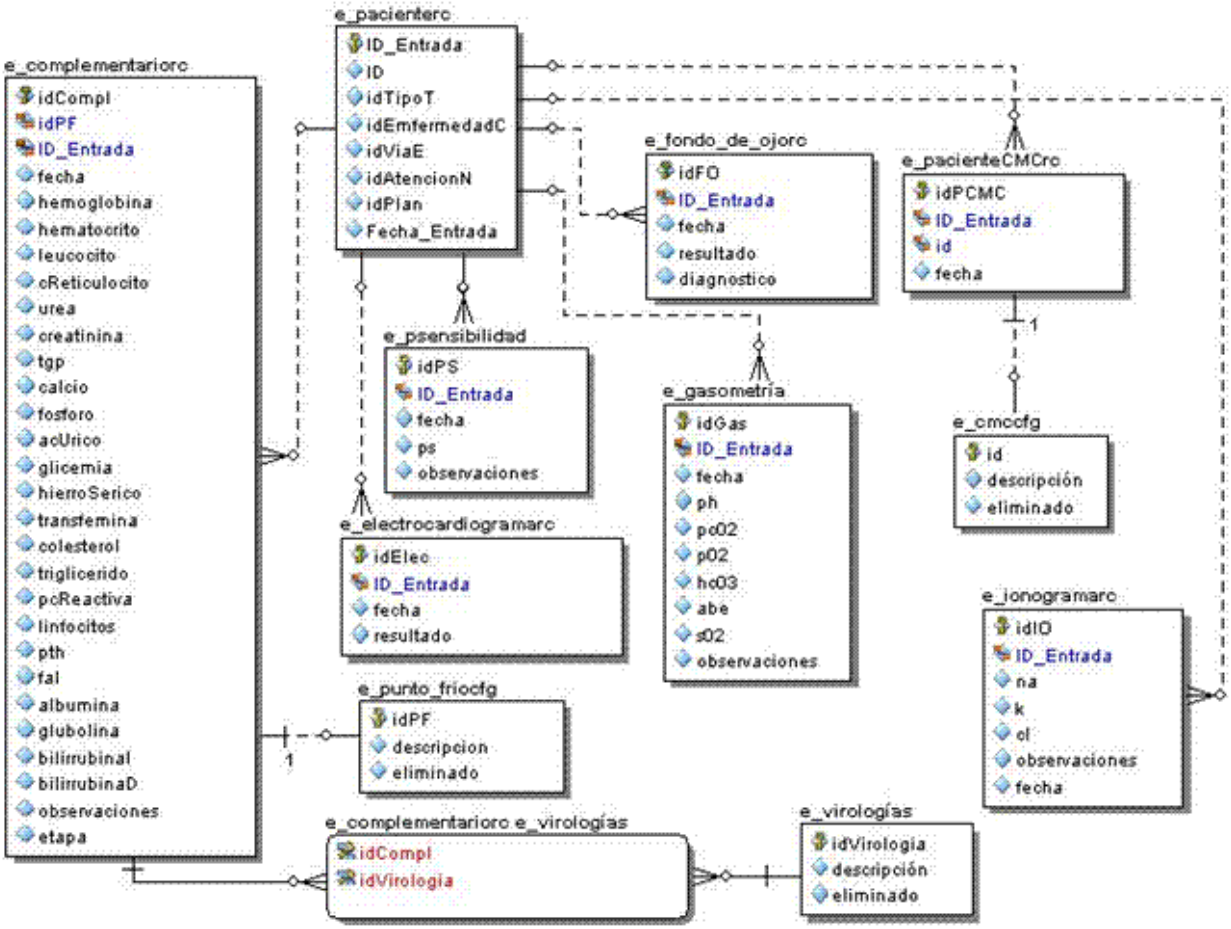


Fig.2: Modelo Físico de la Base de datos, Estudios.

Diagrama Entidad Relación de la Base de Datos.

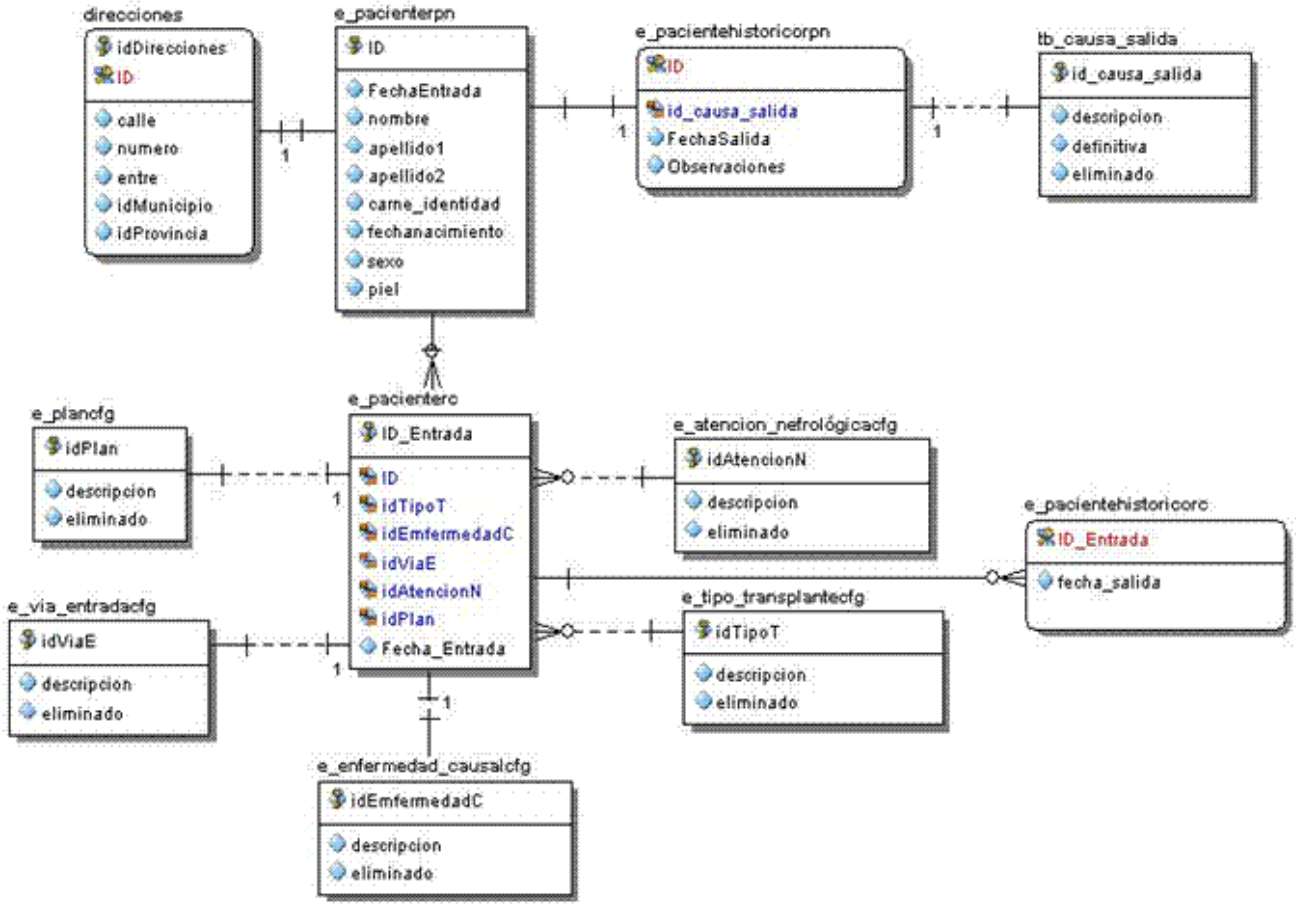


Fig.3: Diagrama Entidad Relación de la Base de datos, Movimiento de Pacientes.

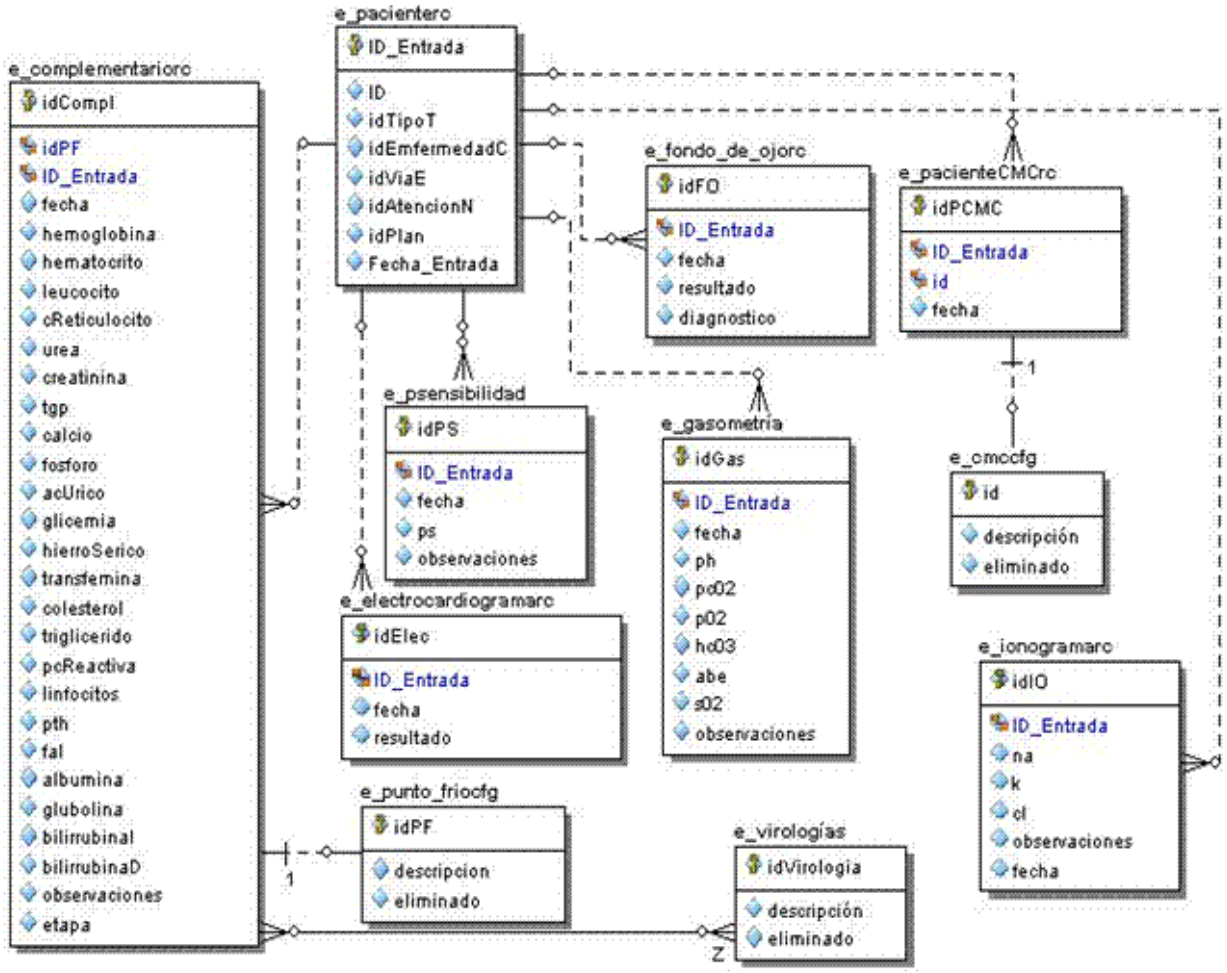


Fig.4: Diagrama Entidad Relación de la Base de datos, Estudios.

1.9. Entornos De Desarrollo Integrado

Zend Studio

Se trata de un programa de la casa Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web. 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.

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.

- El editor

La parte del programa que permite escribir los scripts es bastante útil para la programación en PHP. La interfaz está compuesta por varias partes: un explorador de archivos, una ventana de depuración, los menús y otra para mostrar el código de las páginas. Una de las ayudas que ofrece a la hora de escribir son las típicas en editores avanzados, como permitir editar varios archivos, y moverse fácilmente entre ellos, marcar a qué elementos corresponden los inicios y cierres de las etiquetas, paréntesis o llaves, moverse al principio o al final de una función, identificación automática del código, etc.

- La herramienta de depuración

Sin duda, más de una vez los programadores de PHP se han visto en un duro problema por no encontrar un error en algún script que esté haciendo que devuelva resultados inesperados. En estos casos lo que se suele hacer es escribir el contenido de diversas variables en la página web para que den algún indicio del lugar donde está el error. Para hacer el trabajo más fácil, Zend Studio dispone de una herramienta muy interesante de debug o depuración.

Gracias a ella se ejecutan páginas y conocer en todo momento el contenido de las variables de la aplicación y las variables del entorno como las cookies, las recibidas por formulario o en la sesión. Se colocan puntos de parada de los scripts y realizar las acciones típicas de depuración. (39)

Frameworks

En el desarrollo de software, un framework es una estructura de soporte definida en la cuál otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
Son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. (40)

CodeIgniter
Es un framework para PHP pensado para ofrecer un alto rendimiento, además es bastante ligero y fácilmente configurable.
Hay muchos frameworks para PHP y este podría pasar por una alternativa más, pero si la aplicación que se desarrolla requiere de una plataforma que no marque mucho la aplicación y que necesite un alto rendimiento, CodeIgniter puede ser la mejor elección.
Cumple perfectamente el fin de cualquier framework, una estructura definida que da soporte a un proyecto web y ayuda a que este proyecto sea organizado y desarrollado.
Implementa el patrón de arquitectura, MVC (Modelo, Vista Controlador). (41)

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). (42)

Yahoo! User Interface Library
Yahoo user Interface (YUI) Library es un conjunto de utilidades y controles, escrito en JavaScript para desarrollar aplicaciones Web interactivas (RIA - Rich Internet Application) usando técnicas DOM, DHTML y AJAX. Además incluye un conjunto de fuentes CSS. Todos sus componentes han sido desarrollados bajo Licencia BSD Open Source y están disponibles para todo tipo de uso. (43)

Características:
Están disponibles dos tipos de componentes diferentes: Utilidades y Controles.
Los componentes de 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 la página. (44)

1.10. Tecnología y Herramientas a utilizar.

El sistema que se está desarrollando está diseñado bajo la misma arquitectura que está concebida para el desarrollo de todos los módulos que formarán parte del Sistema Integral de Salud, con el objetivo de estandarizar el diseño de dichos módulos y facilitar el mantenimiento de los mismos. Esta Arquitectura es 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 3.0, ya que esta herramienta es muy completa y ofrece amplias potencialidades.

Para la implementación se tendrá como servidor Web Apache 2.0, el cuál es un servidor libre de código abierto y bajo la licencia apache/GPL, con PHP 5; servidor de bases de datos MySQL 5, el IDE de desarrollo Zend Studio; también se van a utilizar los frameworks CodeIgniter y Yahoo User Interface (YUI). Todas estas herramientas se van a utilizar por política del Área Temática “Sistemas Especializados”, encargada del desarrollo del sistema.

Conclusiones

En este capítulo se definieron las tecnologías y herramientas a utilizar en la implementación del sistema. También se realizó un análisis de los distintos sistemas que existen a nivel nacional e internacional, que responden a la misma problemática del presente trabajo, estos no fueron utilizados porque las licencias para su uso tienen un alto costo.

Capítulo 2. Elementos de arquitectura

En este capítulo se valora la arquitectura de la aplicación, así como los principales requisitos no funcionales que se deben tener en cuenta para el correcto funcionamiento del sistema. Se analizan los estándares de codificación a utilizar por la aplicación. Se profundiza en los patrones arquitectónicos presentes en la propuesta de solución, así como sus características y por último se muestran una vista de despliegue y otra de implementación.

2.1. Requisitos no funcionales del sistema propuesto.

Requisitos no funcionales del sistema propuesto
Para lograr un buen funcionamiento de la aplicación como tal, es necesario que se cumplan una serie de requerimientos funcionales y no funcionales. Estos constituyen condiciones que al cumplirlas se satisfacen las necesidades del usuario.

Requisitos No Funcionales

Los requerimientos no funcionales son aquellas cualidades que debe tener la aplicación. Existen diferentes clasificaciones de los requerimientos no funcionales, por ejemplo se pueden clasificar en: requerimientos de rendimiento, de seguridad y privacidad, requerimientos de confiabilidad, entre otros. A continuación se mencionan las propiedades que debe tener la aplicación para su correcto funcionamiento.

- Habilidades de Uso
  - 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 seg. y garantizar de esta manera una respuesta rápida del sistema.

- Soporte
  - El sistema contará con una ayuda para el usuario con la cuál podrá aprender rápidamente a utilizar la aplicación Web.
  - Estará bien documentado para garantizar futuros mantenimientos.

- Portabilidad
  - El sistema podrá ejecutarse sobre plataforma Linux o Windows.

- 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:

- Tendrán acceso al sistema a través de cualquier 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.
- Servidor Web Apache 2.0 y PHP 5 instalados.
- Servidor de Base de Datos MySQL 5.

- Hardware

Para el cliente:

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

Para el servidor:

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

- Restricciones para el diseño y la implementación:
  - Utilizar los patrones de diseño establecidos.
  - Para el análisis y el diseño del sistema debe ser utilizada 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

- 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.

2.2. Estándares de codificación

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 indentado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la PC o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deber estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción. Nunca colocar { en la línea de un código cualquiera, esto requiere una línea propia.	
Comentarios, separadores, líneas, espacios en blanco y márgenes. Objetivo: Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez		
Ubicación de comentarios	Al inicio de cada clase o función y al final de cada bloque de código.	Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas.
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. Antes de un punto y coma.
Variables y constantes		
Apariencia de variables	Las variables tendrán un prefijo para el tipo de datos en minúscula.	El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, la cuál 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 deben declarar las constantes con todas sus letras en mayúscula.
Declaración de constantes y asignación a	Una por cada línea	

variables		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;
Aspectos generales	Nombres de las variables y constantes	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.
Clases y Objeto Objetivo: Nombrar las clases e instancias de forma estándar para todas las aplicaciones.		
Apariencia de clases y objetos	Primera letra en mayúscula	Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*. Ejemplo: MiClase(). Para el caso de las instancias se comenzara con un prefijo que identificara el tipo de dato, este se escribirá en minúscula.
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*.
Declaración de parámetro en funciones	Agrupados por tipos Poner los string 1 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, Campos, $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*.
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 „v_via_entrada“;
Apariencia de las tablas	Las 2 primeras letras representan el tipo .Todas las letras en minúscula	El nombre a emplear para las tablas debe comenzar con el prefijo tb seguido de underscoard y luego debe escribirse todas las letras en minúscula, en caso de que sea un nombre compuesto se utilizara underscoard para separarlo.

		Ejemplo: create table “tb_causa_salida”
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
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
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 BDs, 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

2.3. Patrones de Arquitectura

La Arquitectura Cliente/Servidor se puede describir como:
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.

IBM define al modelo Cliente/Servidor. (45)

- "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".
- "Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información"

Características del Modelo Cliente/Servidor

1. El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.
2. Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.
3. Un servidor da servicio a múltiples clientes en forma concurrente.
4. Cada plataforma puede ser escalable independientemente. Los cambios realizados en las plataformas de los Clientes o de los Servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
5. La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formatos de datos y de los protocolos.
6. Un sistema de servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo sistema a las estaciones Clientes. Esto se logra combinando los recursos de cómputo que se

encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final.

7. También es importante hacer notar que las funciones Cliente/Servidor pueden ser dinámicas. Ejemplo, un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red.

8. Su capacidad para permitir integrar los equipos ya existentes en una organización, dentro de una arquitectura informática descentralizada y heterogénea.

9 Además se constituye como el nexo de unión más adecuado para reconciliar los sistemas de información basados en mainframes o minicomputadores, con aquellos otros sustentados en entornos informáticos pequeños y estaciones de trabajo.

10. Designa un modelo de construcción de sistemas informáticos de carácter distribuido.

11. Su representación típica es un centro de trabajo (PC), en donde el usuario dispone de sus propias aplicaciones de oficina y sus propias bases de datos, sin dependencia directa del sistema central de información de la organización, al tiempo que puede acceder a los recursos de este host central y otros sistemas de la organización ponen a su servicio.

En conclusión, Cliente/Servidor puede incluir múltiples plataformas, bases de datos, redes y sistemas operativos. Estos pueden ser de distintos proveedores, en arquitecturas propietarias y no propietarias y funcionando todos al mismo tiempo. Por lo tanto, su implantación involucra diferentes tipos de estándares: APPC, TCP/IP, OSI, NFS, DRDA corriendo sobre DOS, OS/2, Windows o PC UNIX, en TokenRing, Ethernet, FDDI o medio coaxial, sólo por mencionar algunas de las posibilidades.

Modelo Cliente – Servidor de dos Capas (Two – Tier)

Este modelo divide una aplicación entre un cliente y un servidor, estableciendo un middleware que controla las comunicaciones entre ambos. Divide la aplicación en dos entidades: la interfaz por un lado y las reglas de negocio junto con el acceso a bases de datos por otro.

Este modelo suele ser costoso de mantener, difícil de escalar, pesado de depurar y la lógica de la aplicación no puede ser rehusada ya que está ligada a la interfaz de usuario o al manejo de persistencia de datos. (46)

Modelo Cliente – Servidor de tres Capas (Three – Tier)

El modelo cliente – servidor de tres capas se refiere a un diseño que introduce una capa intermedia entre el cliente y el servidor de bases de datos. Esta capa consiste en un servidor de aplicaciones que contiene el grueso de la lógica de la aplicación. Con esta arquitectura la lógica de la aplicación reside en una sola capa que puede ser fácilmente mantenida, ya que cada capa es un proceso separado y bien definido corriendo en plataformas separadas.

Este modelo tiene la ventaja de que:

• Se puede diferenciar claramente la interfaz de usuario de la lógica de aplicación. Esta separación permite tener diferentes presentaciones accediendo a la misma lógica.

• Se pueden realizar cambios en las diferentes capas sin necesidad de modificarlas todas.

• La redefinición del almacenamiento de información no tiene influencia sobre la interfaz de usuario.

Arquitectura basada en componentes

El desarrollo de aplicaciones basado en componentes emergió como una importante solución al problema de desarrollo de sistemas grandes y complejos. La Arquitectura Basada en Componentes tiene como objetivo construir aplicaciones complejas mediante ensamblado de módulos (componentes), que han sido previamente diseñados por otras personas a fin de ser rehusados en múltiples aplicaciones. Y deben operar correctamente con independencia de los mecanismos internos que utilicen para soportar la funcionalidad de la interfaz. Su premisa es que los componentes cumplan con alta cohesión y bajo acoplamiento.

El surgimiento de la tecnología de componentes distribuidos es la clave de las arquitecturas de n-capas. Estos sistemas de computación utilizan un número variable de componentes individuales que se comunican entre ellos utilizando estándares predefinidos y frameworks de comunicación como:

CORBA - (Common Object Request Broker Architecture) del Object Management Group (OMG)

DNA - (Distributed interNet Architecture) de Microsoft (incluye COM/DCOM y COM+ además de MTS y MSMQ.)

EJB - (Enterprise Java Beans) de Sun Microsystems

XML - (extensible Markup Language) del World Wide Web Consortium (W3C)

Estas y otras tecnologías en rápida evolución proporcionan la infraestructura necesaria y la fontanería relacionada que permite a las compañías operar en un entorno complejo, multiplataforma y con

capacidades de computación distribuida, tanto interna como externamente según se requiera en cada caso. (47)

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 la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la Lógica de negocio.
Descripción del patrón
Modelo: Esta es la representación específica de la información con la cuál el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.
Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.
Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos. En MVC corresponde al modelo.

Esta separación permite construir y probar el modelo independientemente de la representación visual. La separación entre vista y controlador puede ser secundaria en aplicaciones de clientes ricos y, de hecho, muchos frameworks de interfaz implementan ambos roles en un solo objeto.
Por otra parte, en aplicaciones Web, la separación entre la vista (el browser) y el controlador (los componentes del lado del servidor que manejan los requerimientos de HTTP) está mucho más concretamente definida. (48)

Entre las ventajas de este patrón están las siguientes:

- Soporte de vistas múltiples. Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar

múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación de Web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes.

- Adaptación al cambio. Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo. Este patrón sentó las bases para especializaciones ulteriores, tales como Page Controller y Front Controller.

Entre las desventajas, se han señalado:

- Complejidad. El patrón introduce nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad de la solución. También se profundiza la orientación a eventosdel código de la interfaz de usuario, que puede llegar a ser difícil de depurar. En rigor, la configuración basada en eventos de dicha interfaz corresponde a un estilo particular (arquitectura basada en eventos) que aquí se examina por separado.
- Costo de actualizaciones frecuentes. Desacoplar el modelo de la vista no significa que los desarrolladores del modelo puedan ignorar la naturaleza de las vistas.
  Si el modelo experimenta cambios frecuentes, por ejemplo, podrían desbordar las vistas con una lluvia de requerimientos de actualización. (49)

2.4. Vista de despliegue

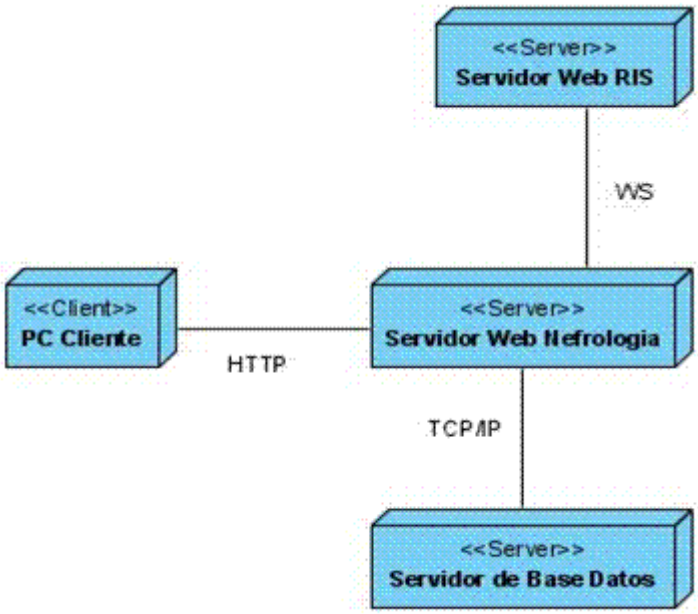


Fig.3 Diagrama de Despliegue.

Nodos

Nodo PC Cliente

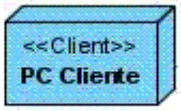


Fig. 4 PC Cliente.

Nodo Servidor Web RIS.

Fig. 5 Servidor Web RIS.

Nodo Servidor Web Nefrología

Fig. 6 PC Servidor Web Nefrología.

Nodo Servidor de Base Datos.

Fig. 7 PC Servidor de Base Datos.

2.5. Vista de implementación

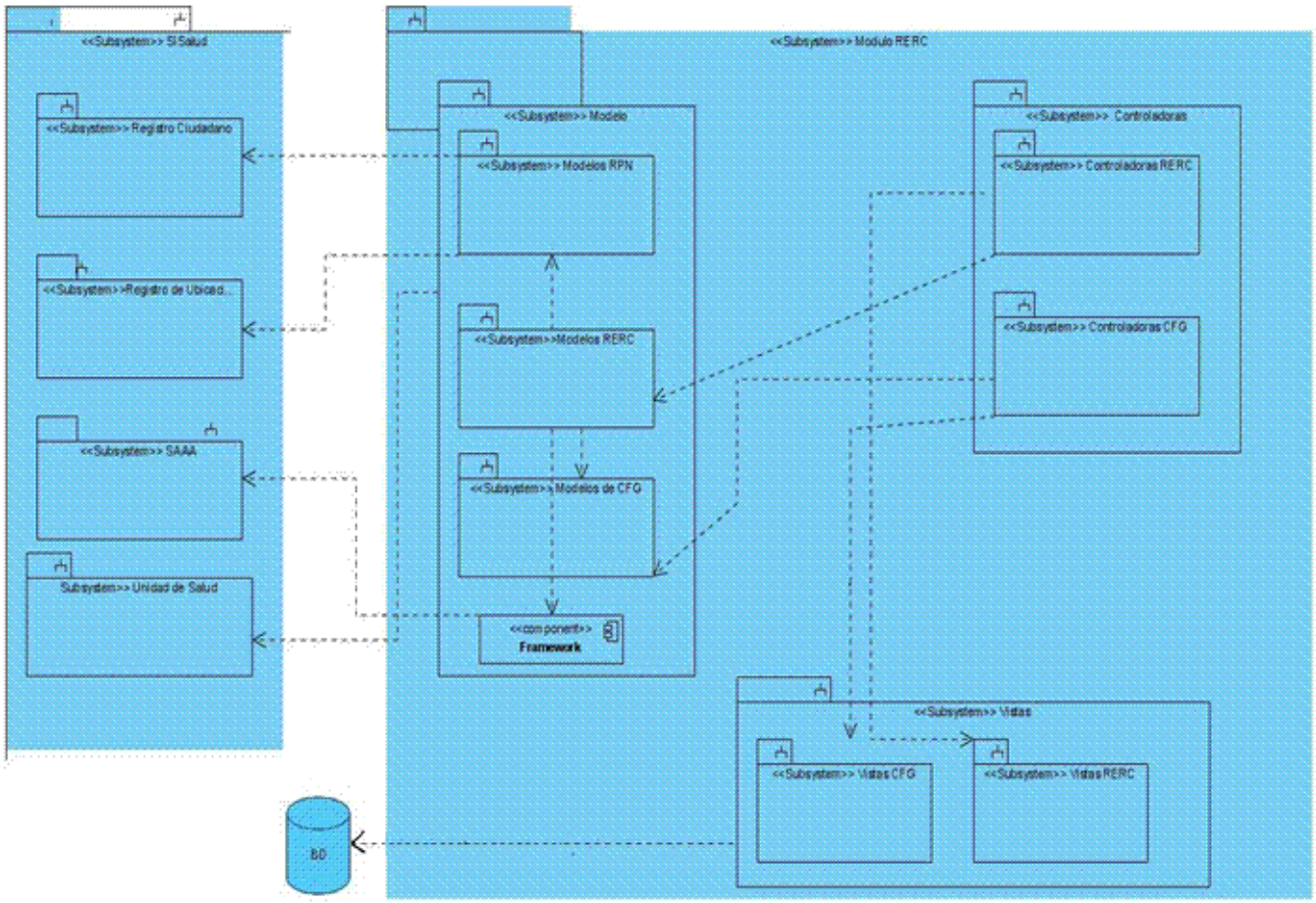


Fig. 8 Diagrama de Componentes.

Subsistemas

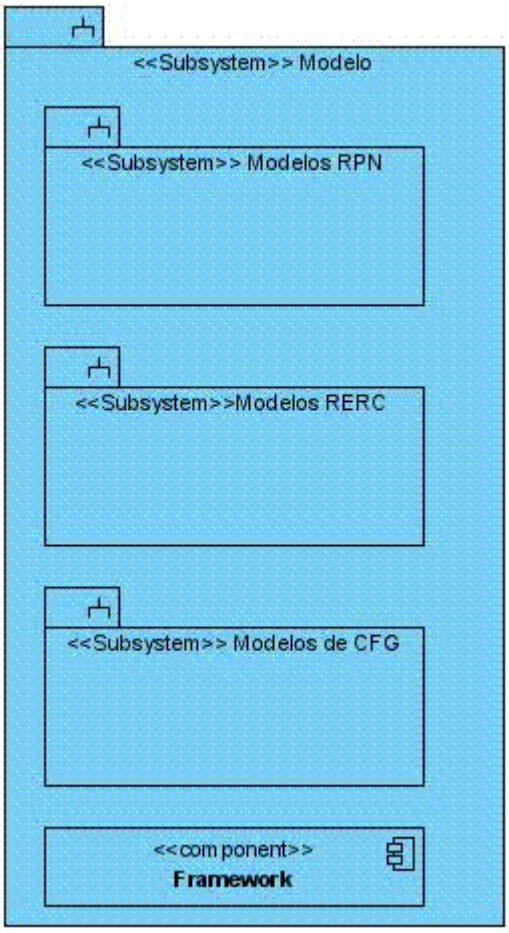


Fig. 9. Subsistemas de las clases modelos.

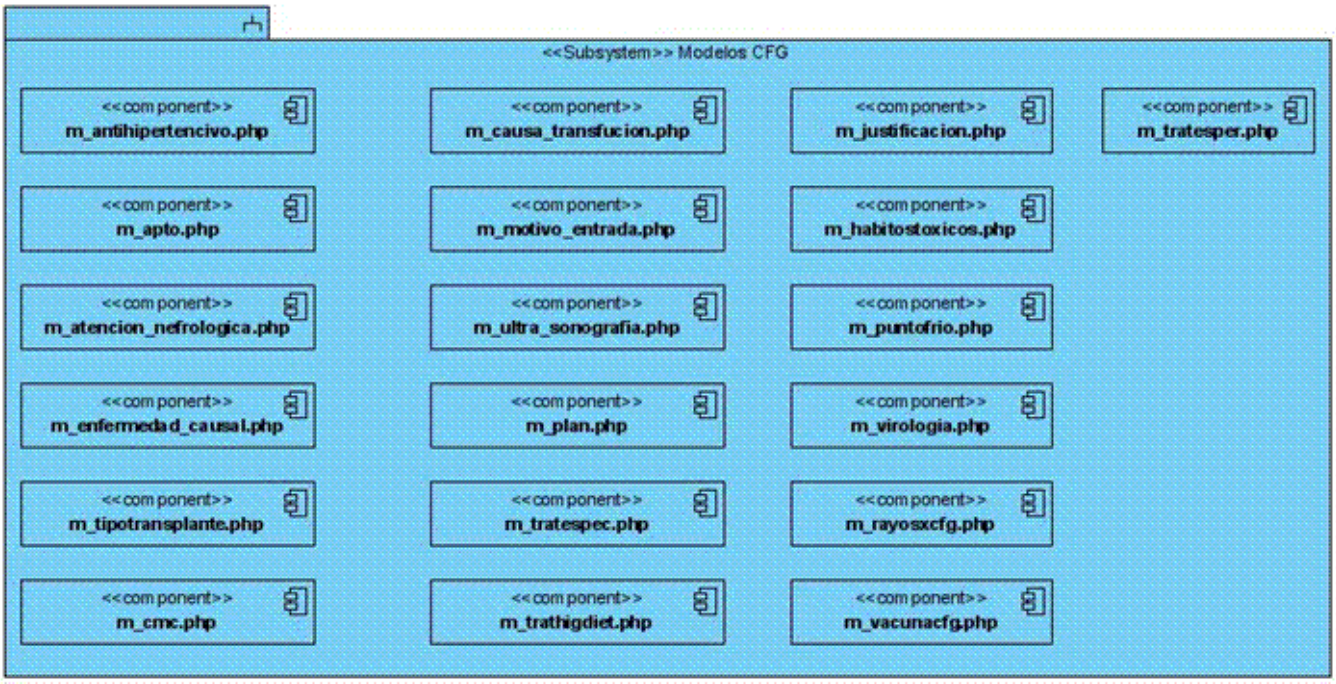


Fig. 10. Subsistema de las clases modelos del módulo configuración.

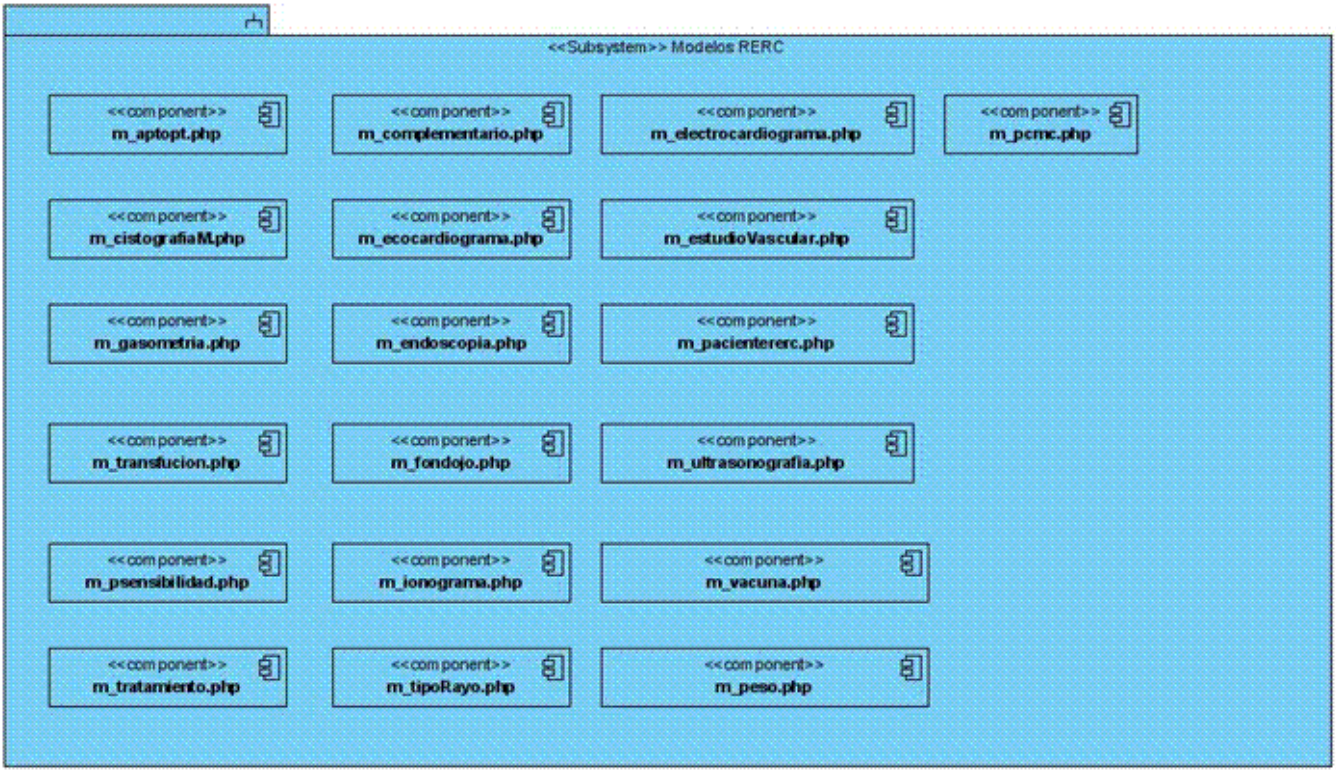


Fig. 11. Subsistema de las clases modelos del módulo RERC.

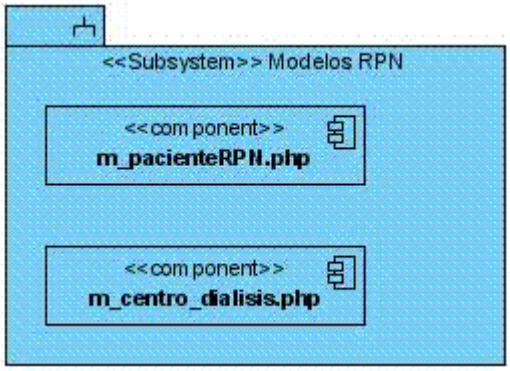


Fig. 12. Subsistema de las clases modelos del módulo RPN.

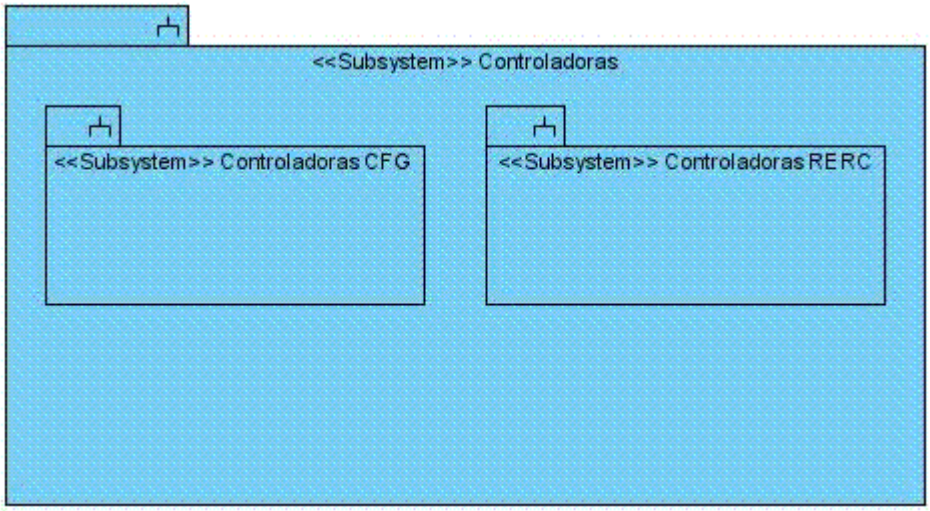


Fig. 13. Subsistema de las clases controladoras.

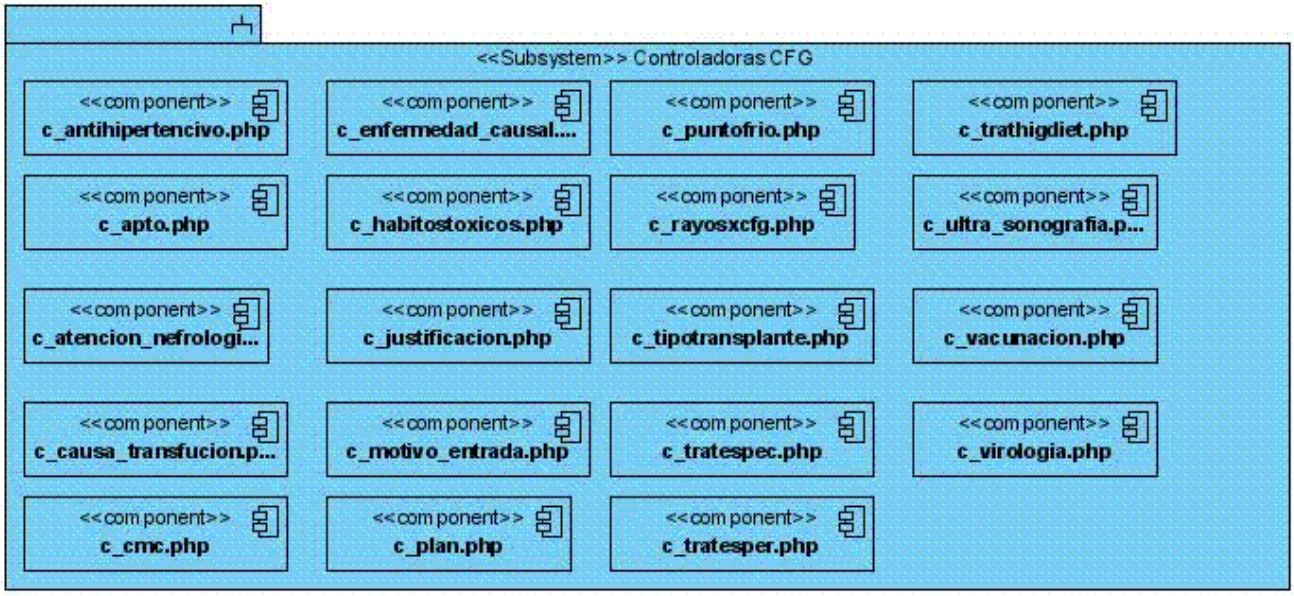


Fig. 14. Subsistema de las clases controladoras del módulo configuración.

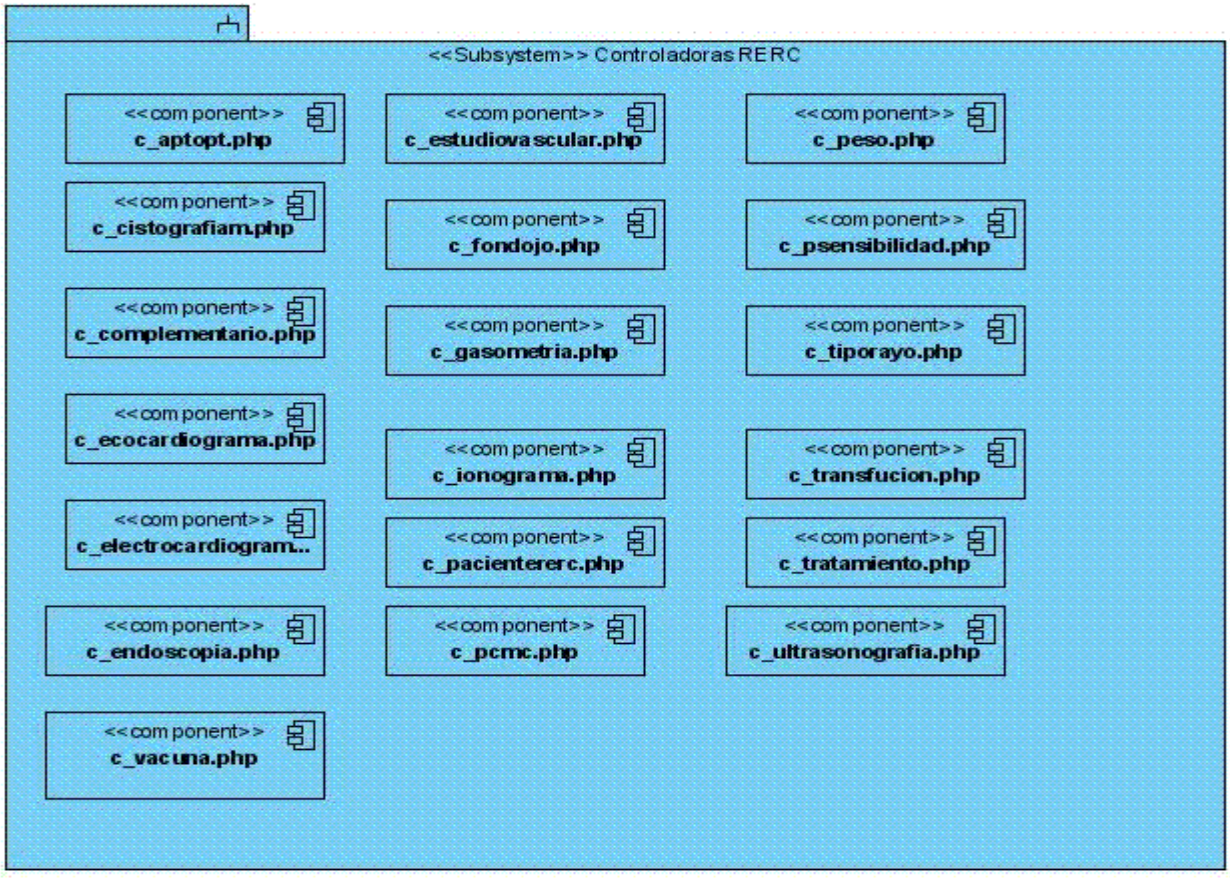


Fig. 15. Subsistema de las clases controladoras del módulo RERC.

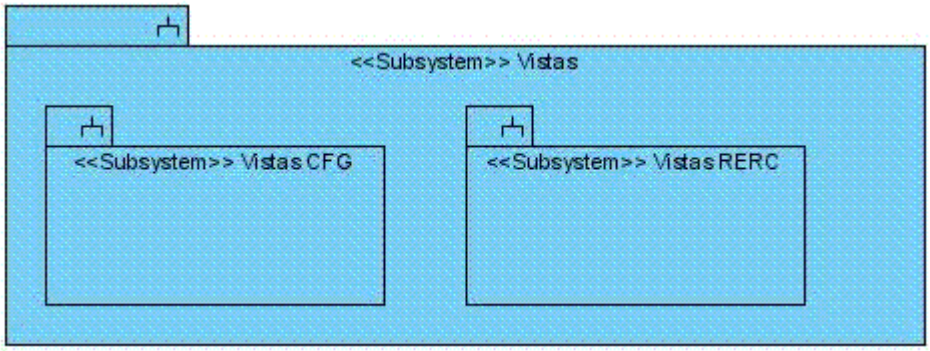


Fig. 16. Subsistema de las clases vistas.

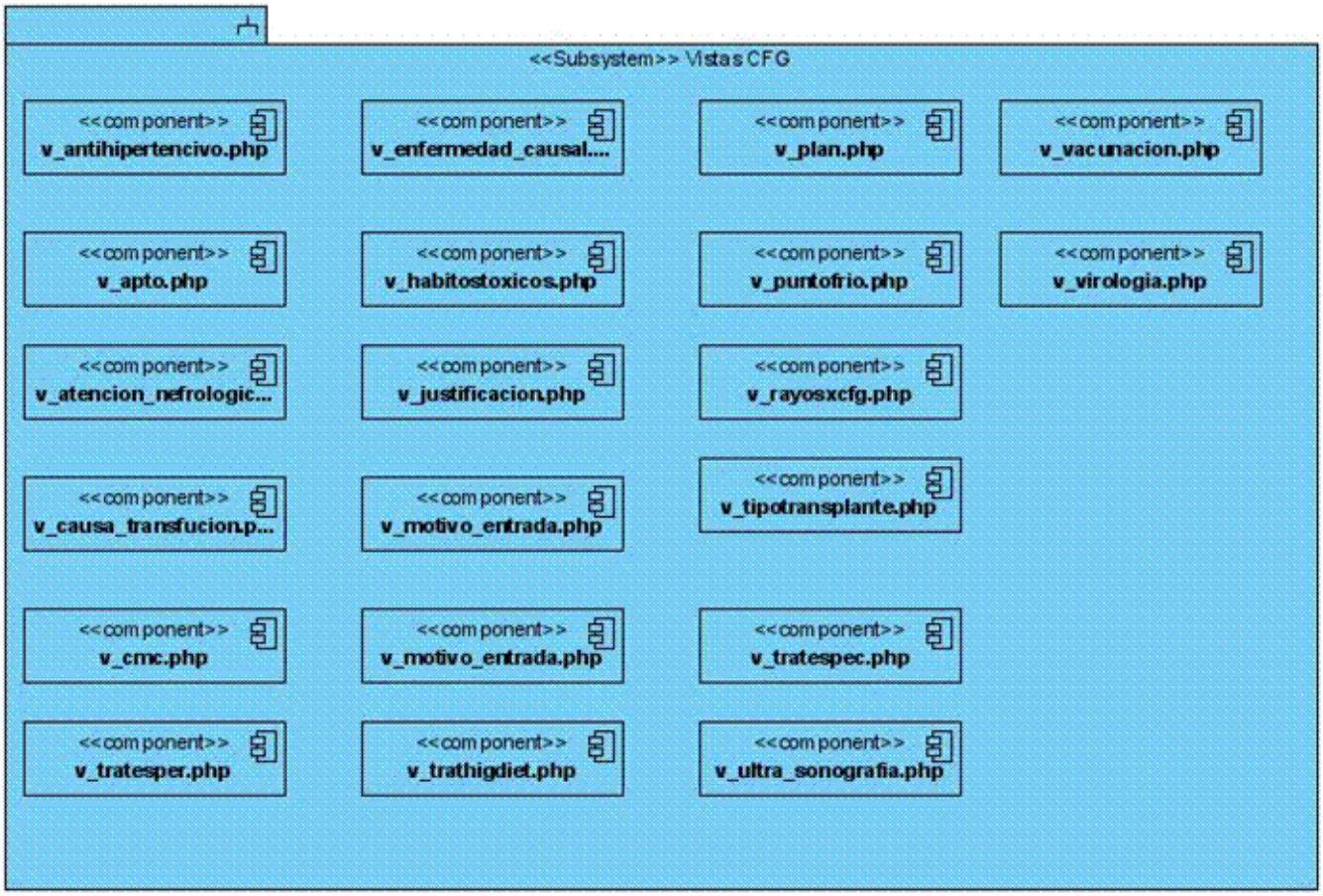


Fig. 17. Subsistema de las clases vistas del módulo configuración.

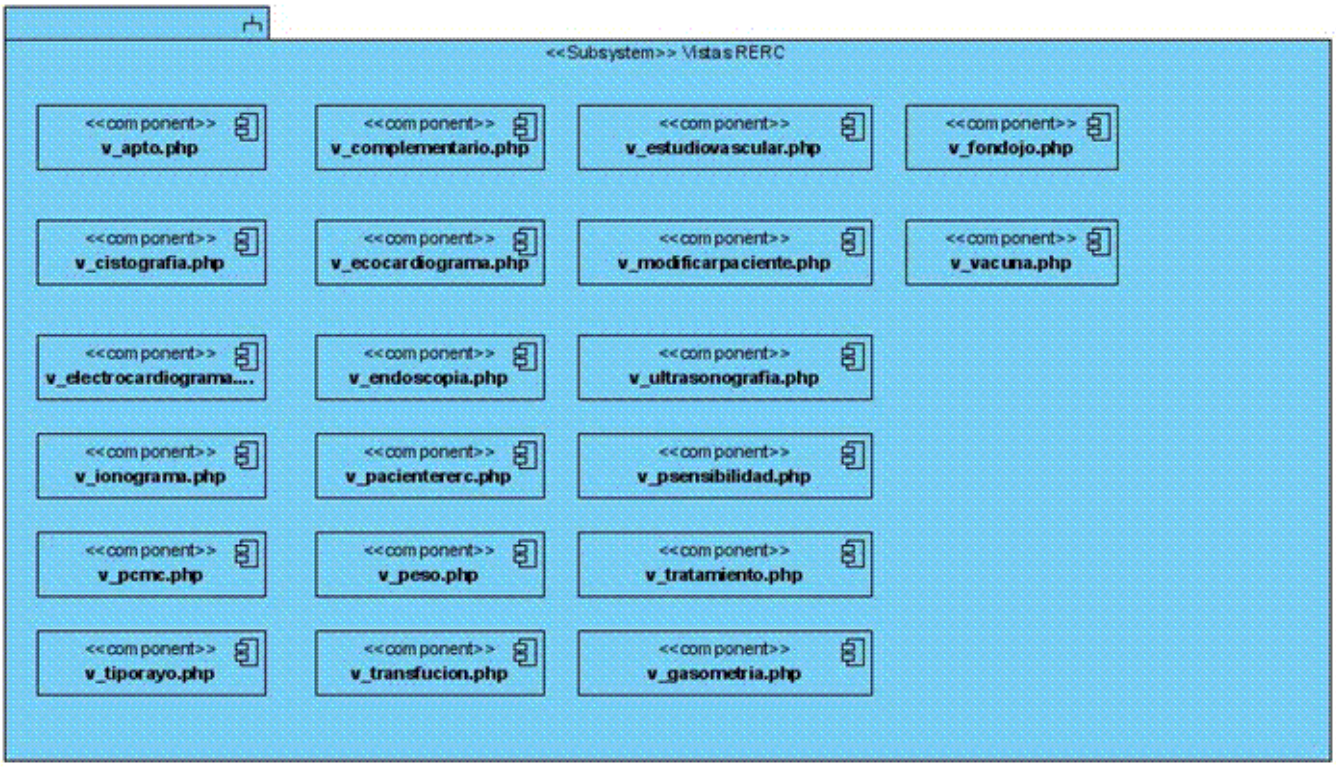


Fig. 18. Subsistema de las clases vistas del módulo RERC.

Conclusiones

Este capítulo se refirió a la arquitectura del sistema. También se expusieron los principales requisitos no funcionales que se deben tener en cuenta para el desarrollo de la aplicación. Se analizaron los estándares de codificación a utilizar por la aplicación. Se profundizó en los patrones arquitectónicos presentes en la propuesta de solución, así como en sus características. Por último se mostró una vista de despliegue y otra de implementación.

CAPÍTULO 3. DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA.

En este capítulo se analiza y fundamenta la estrategia que se tiene concebida para integrar el módulo Registro de Enfermos Renales Crónicos con otros módulos del Alas NefroRed u otros componentes del SISalud. También se realiza un análisis detallado de todas las tablas de la BD, y de todas las clases de diseño del sistema.

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

Integración con otros sistemas de SISalud.

Entre los diferentes componentes que existen en el Registro Informatizado de la Salud (RIS), se encuentra un módulo que se encarga del control de la seguridad, el Sistema de Autenticación, Auditoría y Autorización (SAAA), el cual brinda a través de WebServices una serie de métodos que facilitan todos los servicios necesarios para garantizar que acceda a un sistema solo el personal autorizado y con los niveles de acceso que se definan.

Para integrar el sistema a este componente se crearon diferentes clases que encapsulan el manejo de la seguridad así como la comunicación con dicho componente, de manera que esta queda de forma global para todo el sistema sin necesidad de ser implementada en cada subsistema, pero garantizando que sea utilizada por todos y de manera particular por cada uno.

El módulo Registro de Enfermos Renales Crónicos (RERC) se comunicará internamente con el Registro de Paciente Nefrológicos (RPN) y con el módulo Configuración(CFG), ya que este es el modulo donde están todos los nomencladores del sistema, y la comunicación con este es a través de la controladora de cualquier estudio a la modelo del nomenclador. Cuando se vaya a realizar el proceso de registrar a un paciente en el RERC, RPN consumirá el servicio web, servicio registro de ciudadano, que brinda el Registro Ciudadano (RC) de SISalud, para tomar los datos generales de la

persona: su nombre, apellidos, carnet de identidad, sexo y el identificador con que se identificará en la base de datos así como también la provincia y el municipio del Registro de Ubicación (RU).

3.2. Descripción de Clases RERC.

<<PacienteRERC>>

Capa de Negocio.RERC.1. E_PacienteRERC

Nombre: E_PacienteRERC	
Tipo de clase: Entidad	
Atributo	Tipo
idciudadano	int
fechaentrada	dateTime
viaentrada	string
plan	string
atencionNefrologica	string
nombre	string
ape1	string
ape2	string
carnet	int
sexo	string
tipoT	string
piel	string
fechaNacimiento	dateTime
Para cada responsabilidad:	
Nombre:	function __construct($idciudadano, $fechaentrada, $viaentrada, $plan, $atencionNefrologica, $nombre, $ape1, $ape2, $carnet, $sexo,$tipoT,$piel,$fechaNacimiento)
Descripción:	Constructor por parámetros.
Nombre:	SetIDEntrada(idE)
Descripción:	Cambia el numero de id de la fecha de entrada del paciente.
Nombre:	GetIDEntrada()
Descripción:	Devuelve el numero de id de la fecha de entrada del paciente.

Nombre:	GetFechaEntrada()
Descripción:	Devuelve la fecha de la entrada del paciente.
Nombre:	SetFechaEntrada(fechaentrada)
Descripción:	Cambia la fecha de entrada del paciente.
Nombre:	GetEnfermedadCausal()
Descripción:	Devuelve la enfermedad causal del paciente.
Nombre:	SetEnfermedadCausal (enfermedadcausal)
Descripción:	Cambia la enfermedad causal del paciente.
Nombre:	GetViaEntrada()
Descripción:	Devuelve la vía de entrada del paciente.
Nombre:	SetViaEntrada(viaentrada)
Descripción:	Cambia la vía de entrada del paciente.
Nombre:	GetAtencionNefrologica()
Descripción:	Devuelve la atención nefrológica dada al paciente.
Nombre:	SetAtencionNefrologica(atencionNefrologica)
Descripción:	Cambia la atención nefrológica dada al paciente.
Nombre:	GetPlan()
Descripción:	Devuelve el plan dado al paciente.
Nombre:	SetPlan(plan)
Descripción:	Cambia la atención nefrológica dada al paciente.

Capa de Negocio.RERC.2. M_PacienteRERC

Nombre: M_PacienteRERC	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	BuscarPacienteRegistrar(EstructuraEntrada, Paciente)
Descripción:	Método que lista los pacientes que están registrados, teniendo en cuenta la estructura de entrada introducida.
Nombre:	InsertarPaciente(Paciente,direccion,virologias,peso,cretinina,etapa)

Descripción:	Registra pacientes con los parámetros específicos.
Nombre:	CambiarPaciente($paciente)
Descripción:	Cambia un paciente.
Nombre:	ListarP(Paciente, EstructuraEntrada)
Descripción:	Lista pacientes según su estructura de entrada.
Nombre:	ModificarDatos(paciente,creatinina,etapa,pesoC,virologias)
Descripción:	Modifica los datos que fueron introducidos a un paciente determinado.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Método que registra si el paciente está en el módulo.
Nombre:	PerteneceOtroCD($paciente)
Descripción:	Método que devuelve, si o no, el paciente pertenece a otro centro de diálisis diferente al del usuario que está logueado.
Nombre:	DarSalida($paciente)
Descripción:	Método que da salida a un paciente del modulo.

<< Imageneología >>

Capa de Negocio. RERC. 3. E_TipoRayo

Nombre: E_TipoRayo	
Tipo de clase: Entidad	
Atributo	Tipo
fecha resultado diagnóstico paciente rayo	datetime string string string string
Para cada responsabilidad:	
Nombre:	E_TipoRayo ()
Descripción:	Constructor por defecto
Nombre:	E_TipoRayo ($resultado, $diagnostico, $rayo, $fecha)

Descripción:	Constructor por parámetros
Nombre:	Get_Fecha(): datetime
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Get_Resultado (): string
Descripción:	Devuelve el valor del atributo resultado.
Nombre:	Set_Resultado ($resultado): void
Descripción:	Permite cambiar el valor del atributo resultado.
Nombre:	Get_Paciente(): string
Descripción:	Devuelve el valor del atributo paciente.
Nombre:	Get_Rayo (): string
Descripción:	Devuelve el valor del atributo rayo.
Nombre:	Set_Rayo ($rayo): void
Descripción:	Permite cambiar el valor del atributo rayo.
Nombre:	Get_Diagnostico (): string
Descripción:	Devuelve el valor del atributo diagnóstico.
Nombre:	Set_Diagnostico ($diagnostico): void
Descripción:	Permite cambiar el valor del atributo diagnóstico.

Capa de Negocio. RERC. 4. M_TipoRayo

Nombre: M_TipoRayo	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Adicionar($rayo,$idE)
Descripción:	Adiciona una imageneología a la BD.
Nombre:	Editar($rayo,$ID)
Descripción:	Actualiza en la BD una imageneología seleccionado
Nombre:	Listar($offset, $cant, $orden, $idPaciente)
Descripción:	Lista ordenadamente todas las imageneologías que existen en la BD.

Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

Capa de Presentación. RERC. 5. C_TipoRayo

Nombre: C_TipoRayo	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Adicionar()
Descripción:	Adiciona un imageneología con los datos que captura de la vista.
Nombre:	Editar()
Descripción:	Actualiza un imageneaología con los nuevos datos.
Nombre:	Listar($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todas las imageneologías.

<< Electrocardiograma >>

Capa de Negocio. RERC. 6. E_ Electrocardiograma

Nombre: E_Electrocardiograma	
Tipo de clase: Entidad	
Atributo	Tipo
paciente fecha resultado	string datetime string
Para cada responsabilidad:	
Nombre:	E_Electrocardiograma()
Descripción:	Constructor por defecto.
Nombre:	E_Electrocardiograma($fecha, $resultado, $paciente)
Descripción:	Constructor por parámetros.

Nombre:	Get_Paciente(): string
Descripción:	Devuelve el valor del atributo paciente.
Nombre:	Get_Fecha(): datetime
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Get_ Resultado (): string
Descripción:	Devuelve el valor del atributo resultado.
Nombre:	Set_ Resultado (resultado): void
Descripción:	Permite cambiar el valor del atributo resultado.

Capa de Negocio. RERC. 7. M_ Electrocardiograma

Nombre: M_Electrocardiograma	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Adicionar($electro)
Descripción:	Adiciona un electrocardiograma a la BD.
Nombre:	Editar($electro, $iDElec)
Descripción:	Actualiza un electrocardiograma seleccionado de la BD.
Nombre:	Listar($offset, $cant, $orden, $iDPaciente)
Descripción:	Lista todos los electrocardiogramas que se encuentran en la BD.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

Capa de Presentación. RERC. 8. C_ Electrocardiograma

Nombre: C_ Electrocardiograma	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Adicionar()

Descripción:	Captura los datos de la vista, y los adiciona en la BD.
Nombre:	Editar()
Descripción:	Actualiza un electrocardiograma seleccionado con los datos nuevos.
Nombre:	Listar ($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todos los electrocardiogramas.

<< Aptitud >>

Capa de Negocio. RERC. 9. E_AptoPT

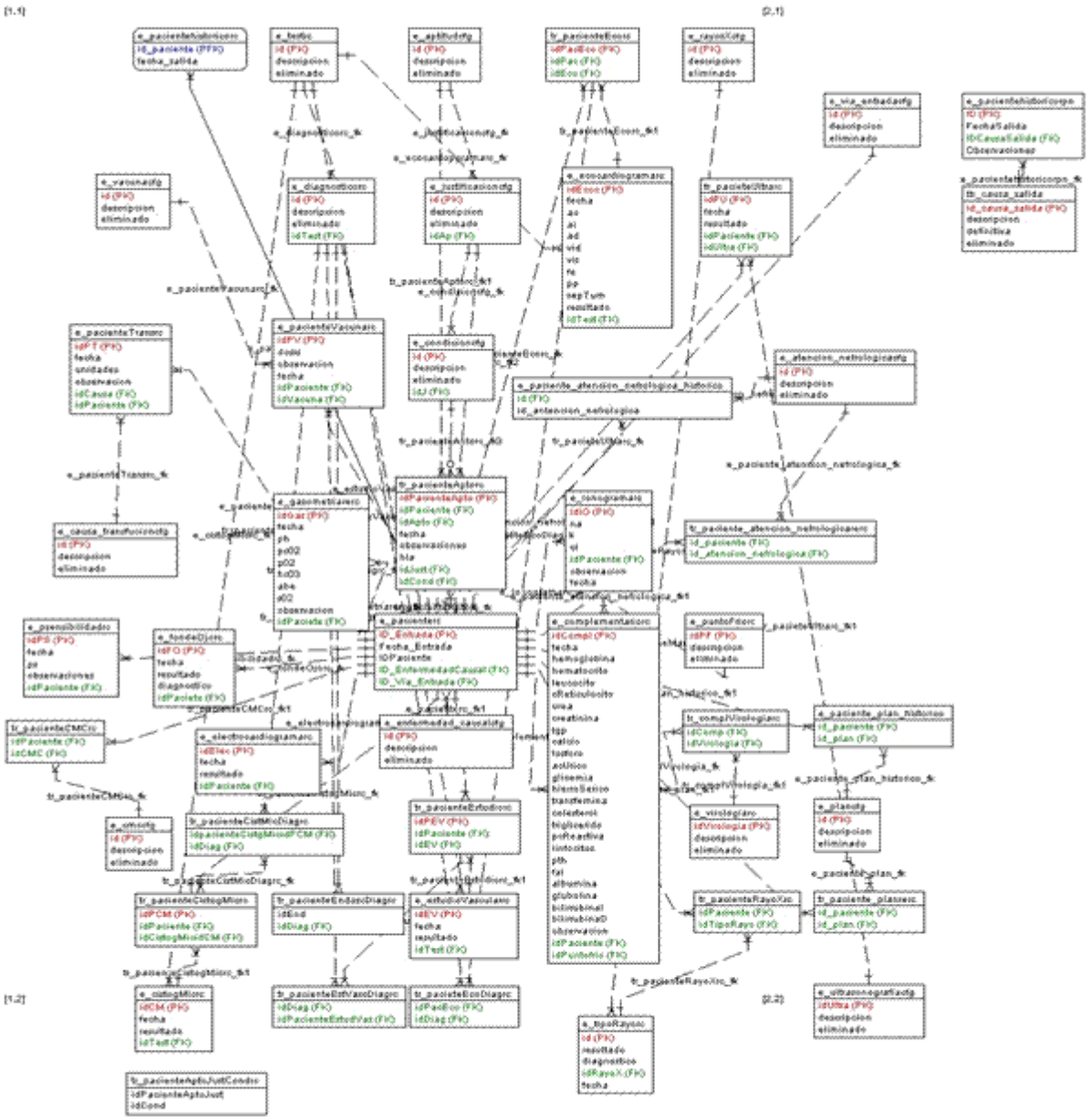
Nombre: E_AptoPT	
Tipo de clase: Entidad	
Atributo	Tipo
idJust fecha paciente cond observaciones apto hla	Int datetime string string varchar string int
Para cada responsabilidad:	
Nombre:	E_AptoPT ()
Descripción:	Constructor por defecto.
Nombre:	E_AptoPT ($fecha, $observacion, $paciente, $apto, $hla, $idJust, $cond)
Descripción:	Constructor por parámetros.
Nombre:	Get_IdJust(): int
Descripción:	Devuelve el valor del atributo idJust.
Nombre:	Get_Fecha(): varchar
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Get_ Paciente(): string
Descripción:	Devuelve el valor del atributo paciente.
Nombre:	Get_Condiciones (): string

Descripción:	Devuelve el valor del atributo cond.
Nombre:	Set_Condiciones (cond): void
Descripción:	Permite cambiar el valor del atributo cond.
Nombre:	Get_Observaciones (): varchar
Descripción:	Devuelve el valor del atributo observaciones.
Nombre:	Set_Observaciones (observaciones): void
Descripción:	Permite cambiar el valor del atributo observaciones.
Nombre:	Get_Apto (): string
Descripción:	Devuelve el valor del atributo apto.
Nombre:	Set_Apto (apto): void
Descripción:	Permite cambiar el valor del atributo apto.
Nombre:	Get_HLA (): int
Descripción:	Devuelve el valor del atributo hla.
Nombre:	Set_HLA (hla): void
Descripción:	Permite cambiar el valor del atributo hla.

Capa de Negocio. RERC. 10. M_AptoPT

Nombre: M_AptoPT	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Adicionar($apto)
Descripción:	Adiciona un paciente apto a la BD.
Nombre:	Editar($apto, $idPacienteApto)
Descripción:	Actualiza en paciente seleccionado en la BD.
Nombre:	Listar($offset, $cant, $orden, $idPaciente)
Descripción:	Lista en un orden especifico todos los pacientes con aptitud.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

3.3. Diseño de la BD



3.4. Descripción de las tablas de la Base de Datos.

Nombre: e_rayosXcfg		
Descripción: Tabla que guarda los rayosX que pueden tener los pacientes.		
Atributo	Tipo	Descripción
id	INTEGER	Identificador de la tabla, numero único.
descripcion	VARCHAR	Guarda la descripción de los rayosX
eliminado	TINYINT	Muestra si fue eliminado o no.

Nombre: tr_pacienteAptorc		
Descripción: Entidad que relaciona a la tabla paciente con la tabla aptitud, o sea, guarda los pacientes aptos para ser trasplantados en cada fecha.		
Atributo	Tipo	Descripción
idPacienteApto	INTEGER	Identificador de la tabla, numero único.
idPaciente	INTEGER	Guarda el id de la tabla e_pacienterc.
idApto	INTEGER	Guarda el id de la tabla e_aptitudcfg.
fecha	VARCHAR	Guarda la fecha.
observaciones	VARCHAR	Guarda las observaciones hechas al paciente.
hla	INTEGER	Guarda el tipo de análisis hla.
idJust	INTEGER	Guarda el id de la tabla e_justificacioncfg.
idCond	INTEGER	Guarda el id de la tabla e_condicioncfg.

Nombre: e_fondeOjorc		
Descripción: Tabla que guarda los análisis de fondos de ojos de los pacientes.		
Atributo	Tipo	Descripción
idFO	INTEGER	Identificador de la tabla, numero único.
fecha	VARCHAR	Guarda la fecha.
resultado	VARCHAR	Guarda el resultado de los análisis.
diagnostico	VARCHAR	Guarda el diagnóstico indicado.
idPaciete	INTEGER	Guarda el id de la tabla e_pacienterc.

Nombre: e_pacienterc		
Descripción: Tabla que guarda los datos del paciente enfermo renal crónico.		
Atributo	Tipo	Descripción
ID_Entrada	INTEGER	Guarda el id de entrada.
Fecha_Entrada	VARCHAR	Guarda la fecha de entrada del paciente.
IDPaciente	INTEGER	Identificador de la tabla, numero único.
ID_EnfermedadCausal	INTEGER	Guarda el id de la tabla e_enfermedad_causalcfg.
ID_Via_Entrada	INTEGER	Guarda el id de la tabla e_via_entradacfg.

Nombre: e_puntoFriorc		
Descripción: Tabla que guarda los análisis de punto frio de los pacientes.		
Atributo	Tipo	Descripción
idPF	INTEGER	Identificador de la tabla, numero único.
descripcion	VARCHAR	Guarda la descripción del punto frio.
eliminado	TINYINT	Muestra si fueron eliminados o no.

Nombre: e_electrocardiogramarc		
Descripción: Tabla que guarda los análisis de electrocardiograma de los pacientes		
Atributo	Tipo	Descripción
idElec	INTEGER	Identificador de la tabla, numero único.
fecha	VARCHAR	Guarda la fecha
resultado	VARCHAR	Guarda el resultado de los análisis.
idPaciente	INTEGER	Guarda el id de la tabla e_pacienterc.

Nombre: tr_pacienteRayoXrc		
Descripción: Entidad que relaciona a la tabla paciente con la tabla rayosX, o sea que a cada paciente le corresponde al menos un rayosX.		
Atributo	Tipo	Descripción
idPaciente	INTEGER	Guarda el id de la tabla e_pacienterc.
idTipoRayo	INTEGER	Guarda el id de la tabla e_tipoRayorc.

Nombre: e_tipoRayorc		
Descripción: Tabla que guarda los tipos de rayosX que pueden realizarse los pacientes.		
Atributo	Tipo	Descripción
id	INTEGER	Identificador de la tabla.
resultado	VARCHAR	Guarda el resultado de los análisis.
diagnostico	VARCHAR	Guarda el diagnóstico de los análisis.
idRayoX	INTEGER	Guarda el id de la tabla e_rayosXcfg.
fecha	VARCHAR	Guarda la fecha.

Nombre: e_aptitudcfg		
Descripción: Tabla que guarda el estado de aptitud de los pacientes		
Atributo	Tipo	Descripción
id	INTEGER	Identificador de la tabla, numero único.
descripcion	VARCHAR	Guarda la descripción de la aptitud del paciente
eliminado	INTEGER	Muestra si fue eliminado o no, la aptitud.

Conclusiones

En este capitulo se analizó y justificó la estrategia para integrar el modulo Registro de Enfermos Renales Crónicos con otros módulos u otros sistemas. Se realizó además un análisis detallado de todas las tablas de la BD, y de todas las clases para el mejor entendimiento de las mismas.

CONCLUSIONES

Una vez concluida la investigación, se han cumplido los objetivos planteados, obteniéndose los siguientes resultados:

- Se realizó un estudio de las técnicas de programación, lenguajes y librerías para el desarrollo de la aplicación.
- Se implementó el módulo Registro de Enfermos Renales Crónicos utilizando los patrones de diseños establecidos.
- Con la puesta en práctica del sistema se obtendrá mayor rapidez, eficacia y exactitud en el trabajo con los datos de los ERC, logrando un mejor aprovechamiento de los órganos disponibles para los trasplantes renales.

Con el funcionamiento de este sistema se logra el incremento de la capacidad organizativa de los servicios nefrológicos del país, el aumento de la calidad de la asistencia médica a este tipo de pacientes y una mejor atención a los mismos, específicamente en lo referente a su evolución y atención individualizada.

RECOMENDACIONES

Luego de la presentación de todo lo trabajado para la Implementación del Módulo Registro de Enfermos Renales Crónicos, se listan a continuación una serie de recomendaciones para la ampliación, modificación, mejora y construcción de nuevas versiones de este sistema, además de que evitarán que el sistema se vuelva obsoleto y fuera de las necesidades del cliente:

- Proporcionar mantenimiento a la aplicación en un período de tiempo de 6 meses como máximo.
- Concluir con el módulo de reportes que la aplicación necesita.
- Impartir cursos de capacitación a las personas que laboran en los servicios nefrológicos y trabajarán con la aplicación.

REFERENCIAS BIBLIOGRÁFICAS

1. Dr. Alexánder Mármol, Dr. Alexis Pérez, Dr. Juan C. Pérez de Prado, Dra. Mercedes Herrera, Dr. Saúl Molina, Dra. Yanet Parodis y Dr. Raúl Herrera. Nefrología. *Trasplante renal en Cuba.* [Online] 03 03, 2005. [Cited: 05 03, 2008.] http://www.bvs.sld.cu/revistas/med/vol44_1-2_05/med10105.htm.

2. Introduccion. *RED DE INFORMATIZACION EN NEFROLOGIA.* [Online] [Cited: 05 03, 2008.] http://www.ucmh.sld.cu/rhab/vol5_num2/rhcm07206.htm.

3. Torres, Mylenys. Mas Nefrología. *Exhibe la Nefrología en Cuba nuevos y mejores indicadores.* [Online] [Cited: 05 03, 2008.] http://www.visiontunera.co.cu/Salud/7.11.05%20sala%20de%20nefrologia%20en%20las%20tunas%20cuba.htm.

4. SISDIA. *Eficiencia y seguridad para los centros de diálisis.* [Online] [Cited: 05 05, 2008.] http://www.sisdia.com.uy/index.html.

5. Nefronet. *Renal Web .* [Online] [Cited: 05 05, 2008.] http://www.renal.org.ar/calidad/informaticos.htm.

6. Nefrolink. *Tecnologías de gestión clínica para una red asistencial digital.* [Online] [Cited: 05 05, 2008.] http://www.aiqei.com/index.php?ids=305&lang=es.

7. Dr. Miguel Suria y Dr. José Luis Gorriz, y la empresa informática Visual-Limes S.L. dirigida por los informáticos D. Emilio Navarro y D. Francisco Orellana. Nefrosoft. *NEFROSOFT.* [Online] [Cited: 05 05, 2008.] http://www.visual-limes.com/.

8. SINTRA. *Institucional Líneas de acción .* [Online] [Cited: 05 05, 2008.] http://www.incucai.gov.ar/institucional/lineas_accion/sintra.jsp.

9. Cabanes, José Ignacio. Internet. [Online] 05 21, 1998. [Cited: mayo 05, 2008.] http://usuarios.lycos.es/Resve/diccioninform.htm.

10. William, Edu. Aplicaciones Web. [Online] 04 06, 2007. [Cited: 05 05, 2008.] http://www.hosteltur.com/blogs/349_aplicaciones-basadas-web-ventajas-desventajas-nivel-economico-empresarial.html.

11. PHP. *Nuevas Tecnologías.* [Online] 01 16, 2008. [Cited: 05 06, 2008.] http://elmanusito.wordpress.com/2008/01/16/nuevas-tecnologias/.

12. PHP 5. [Online] [Cited: 05 06, 2008.] http://es.wikipedia.org/wiki/.php.

13. ASP. *Manual ASP .* [Online] [Cited: 05 06, 2008.] http://www.monografias.com/trabajos5/asp/asp.shtml.

14. Castillo, Carlos. REF. ASP. *Tejedores del Web.* [Online] [Cited: 05 06, 2008.] http://www.tejedoresdelweb.com/307/article-1883.html.

15. JSP. *Java Server Pages.* [Online] [Cited: 05 06, 2008.] http://es.wikipedia.org/wiki/Java_Server_Pages.

16. Guervos, Juan Julian Merelo. Mas JSP. *Programando con JSPs.* [Online] octubre 22, 2004. [Cited: 05 06, 2008.] http://geneura.ugr.es/~jmerelo/JSP/.

17. HTML. *Lenguaje de programación para páginas web HTML.* [Online] [Cited: 05 06, 2008.] http://www.monografias.com/trabajos7/html/html.shtml.

18. Ref. Html. *HTML.* [Online] [Cited: 05 06, 2008.] http://es.wikipedia.org/wiki/C%C3%B3digo_HTML.

19. Nyquist, Er. Java. *Lenguajes.* [Online] [Cited: 05 06, 2008.] http://nyquist.galeon.com/quees.htm.

20. Cortez, Ernesto Alonso Lopez. Ref.Java. *Lenguajes de Programación.* [Online] [Cited: 05 06, 2008.] http://www.edukativos.com/apuntes/archives/232.

21. Valdés, Damián Pérez. JavaScript. *Los diferentes lenguajes de programación para la web.* [Online] 11 02, 2007. [Cited: 05 06, 2008.] http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/.

22. Mas sobre javaScript. *JavaScript.* [Online] [Cited: 05 06, 2008.] http://es.wikipedia.org/wiki/JavaScript.

23. Navegadores Web. *Navegador Web.* [Online] [Cited: 05 06, 2008.] http://es.wikipedia.org/wiki/Browser.

24. López, Alejandro Cadavid. Concepto_Mozilla Firefox. *Mozilla Firefox, el navegador web del momento.* [Online] 06 16, 2004. [Cited: 05 06, 2008.] http://www.maestrosdelweb.com/editorial/firefox/.

25. Caracteristicas de mozilla. *Conocimientos Web.net.* [Online] [Cited: 05 06, 2008.] http://www.conocimientosweb.net/dt/article4880.html.

26. SGBD. *Garballe Collector.* [Online] 11 01, 2004. [Cited: 05 06, 2008.] http://www.error500.net/garbagecollector/bases_de_datos/sistema_gestor_de_base_de_dato.html.

27. Todo de PostgreSQL. *PostGreSQL vs. MySQL.* [Online] [Cited: 05 06, 2008.] http://www.netpecos.org/docs/mysql_postgres/x15.html.

28. Bases de datos. Postgre. *Ubuntu Bases de Datos.* [Online] [Cited: 05 06, 2008.] https://help.ubuntu.com/ubuntu/serverguide/es/databases.html.

29. MySQL. *PostGreSQL vs. MySQL.* [Online] [Cited: 05 06, 2008.] http://www.netpecos.org/docs/mysql_postgres/x57.html.

30. Oracle. [Online] [Cited: 05 06, 2008.] http://www.uaem.mx/posgrado/mcruz/cursos/miic/oracle.pdf.

31. Masip, David. Mas de Oracle. *Qué es Oracle .* [Online] [Cited: 05 06, 2008.] http://www.telepolis.com/cgi-bin/web/DISTRITODOCVIEW?url=/basededatos/doc/articulos/oracle1.htm.

32. Sql Server. *Microsoft SQL Server.* [Online] [Cited: 05 06, 2008.] http://es.wikipedia.org/wiki/SQL_Server.

33. Francisco, Mendoza. XML. *Introducción a XML.* [Online] [Cited: 05 06, 2008.] http://www.monografias.com/trabajos6/ixml/ixml.shtml?monosearch.

34. Servicios Web. *Los servicios web (web services).* [Online] [Cited: 05 06, 2008.] http://www.banespyme.org/imagesWeb/ArchivoMultimedia/Documentacion/17/serviciosweb.pdf.

35. Blog, Daniel. Ventajas y desv. Web service. *Web Services.* [Online] 26 07, julio. [Cited: 05 06, 2008.] http://danielvn7.wordpress.com/2007/07/26/web-services/.

36. Servidores Web. *Una Introducción a APACHE.* [Online] [Cited: 05 06, 2008.] http://linux.ciberaula.com/articulo/linux_apache_intro/.

37. Rup. *Proceso Unificado de Rational.* [Online] [Cited: 05 06, 2008.] http://es.wikipedia.org/wiki/RUP.

38. Visual Paradigm. *Sitio de descargas de Software.* [Online] 03 05, 2007. [Cited: 05 06, 2008.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5Bcuenta_de_Linux_14716_p/.

39. Zend Studio. *Zend Studio.* [Online] [Cited: 05 06, 2008.] http://www.desarrolloweb.com/articulos/1178.php.

40. frame_work. *FrameWork.* [Online] [Cited: 05 06, 2008.] http://es.wikipedia.org/wiki/Framework.

41. Diego. CodeIgniter-Framework-PHP. *CodeIgniter-Framework-PHP.* [Online] 06 25, 2007. [Cited: 05 06, 2008.] http://pixelco.us/blog/codeigniter-framework-php/.

42. SCAFFOLDING. *Sistemas aplicados a la Web.* [Online] 02 2008. [Cited: 05 06, 2008.] http://sistemasweb.wordpress.com/2008/02/24/que-es-framework-mvc-scaffolding/.

43. yui. *Yahoo! UI Library.* [Online] [Cited: 05 07, 2008.] http://es.wikipedia.org/wiki/Yahoo!_UI_Library.

44. YUI. *Herramienta para programadores: Yahoo User Interface Library.* [Online] [Cited: 05 07, 2008.] http://www.baluart.net/articulo/herramienta-para-programadores-yahoo-user-interface-library.

45. VALLE, JOSE GUILLERMO. Patrones de Arquitectura. *Definición arquitectura cliente servidor.* [Online] 2005. [Cited: 05 08, 2008.] http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml.

46. Rodriguez, Alberto. Modelo-Cliente-Servidor(En dos capas). *Diseño de Aplicaciones Three Tier.* [Online] [Cited: 05 06, 2008.] http://www.fpress.com/revista/Num9711/Nov97.htm.

47. Rodríguez, Caridad Salazar Alea y Antonio Cortina. ABC. *Aplicaciones Distribuidas.* [Online] 2007. [Cited: 05 06, 2008.] http://www.monografias.com/trabajos14/aplicacion-distrib/aplicacion-distrib.shtml.

48. MVC. *Patrón "Modelo-Vista-Controlador".* [Online] [Cited: 05 08, 2008.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

49. Sevilla, Iván Ruiz. Patron Modelo-Vista-Controlador. *una de Codigo.* [Online] [Cited: 05 08, 2008.] http://www.unadecodigo.com/2007/05/30/el-paradigma-modelo-vista-controlador-tutorial-ror-ii/.

BIBLIOGRAFÍA

- ✓ Dr. Alexánder Mármol, Dr. Alexis Pérez, Dr. Juan C. Pérez de Prado, Dra. Mercedes Herrera, Dr. Saúl Molina, Dra. Yanet Parodis y Dr. Raúl Herrera. Nefrología. Trasplante renal en Cuba. [En línea] 03 de 03 de 2005. [Citado el: 03 de 05 de 2008.] http://www.bvs.sld.cu/revistas/med/vol44_1-2_05/med10105.htm.
- ✓ Introduccion. RED DE INFORMATIZACION EN NEFROLOGIA. [En línea] [Citado el: 03 de 05 de 2008.] http://www.ucmh.sld.cu/rhab/vol5_num2/rhcm07206.htm.
- ✓ Torres, Mylenys. Mas Nefrología. Exhibe la Nefrología en Cuba nuevos y mejores indicadores. [En línea] [Citado el: 03 de 05 de 2008.] http://www.visiontunera.co.cu/Salud/7.11.05%20sala%20de%20nefrologia%20en%20las%20tunas%20cuba.htm.
- ✓ Sistema Informatizado Seguimiento de Diálisis. [En línea] Humana IT Developers. [Citado el: 10 de 12 de 2007.] http://www.sisdia.com.uy/index.html.
- ✓ Nefronet. Renal Web . [En línea] [Citado el: 05 de 05 de 2008.] http://www.renal.org.ar/calidad/informaticos.htm.
- ✓ Nefrolink. Tecnologías de gestión clínica para una red asistencial digital. [En línea] [Citado el: 05 de 05 de 2008.] http://www.aiqei.com/index.php?ids=305&lang=es.
- ✓ Dr. Miguel Suria y Dr. José Luis Gorriz, y la empresa informática Visual-Limes S.L. dirigida por los informáticos D. Emilio Navarro y D. Francisco Orellana. Nefrosoft. NEFROSOFT. [En línea] [Citado el: 05 de 05 de 2008.] http://www.visual-limes.com/.
- ✓ SINTRA. Institucional Líneas de acción . [En línea] [Citado el: 05 de 05 de 2008.] http://www.incucai.gov.ar/institucional/lineas_accion/sintra.jsp.
- ✓ Cabanes, José Ignacio. Internet. [En línea] 21 de 05 de 1998. [Citado el: 05 de mayo de 2008.] http://usuarios.lycos.es/Resve/diccioninform.htm.
- ✓ William, Edu. Aplicaciones Web. [En línea] 06 de 04 de 2007. [Citado el: 05 de 05 de 2008.] http://www.hosteltur.com/blogs/349_aplicaciones-basadas-web-ventajas-desventajas-nivel-economico-empresarial.html.
- ✓ PHP. Nuevas Tecnologías. [En línea] 16 de 01 de 2008. [Citado el: 06 de 05 de 2008.] http://elmanusito.wordpress.com/2008/01/16/nuevas-tecnologias/.
- ✓ PHP 5. [En línea] [Citado el: 06 de 05 de 2008.] http://es.wikipedia.org/wiki/.php.

- ✓ ASP. Manual ASP . [En línea] [Citado el: 06 de 05 de 2008.] http://www.monografias.com/trabajos5/asp/asp.shtml.
- ✓ Castillo, Carlos. REF. ASP. Tejedores del Web. [En línea] [Citado el: 06 de 05 de 2008.] http://www.tejedoresdelweb.com/307/article-1883.html.
- ✓ JSP. Java Server Pages. [En línea] [Citado el: 06 de 05 de 2008.] http://es.wikipedia.org/wiki/Java_Server_Pages.
- ✓ Guervos, Juan Julian Merelo. Mas JSP. Programando con JSPs. [En línea] 22 de octubre de 2004. [Citado el: 06 de 05 de 2008.] http://geneura.ugr.es/~jmerelo/JSP/.
- ✓ HTML. Lenguaje de programación para páginas web HTML. [En línea] [Citado el: 06 de 05 de 2008.] http://www.monografias.com/trabajos7/html/html.shtml.
- ✓ Ref. Html. HTML. [En línea] [Citado el: 06 de 05 de 2008.] http://es.wikipedia.org/wiki/C%C3%B3digo_HTML.
- ✓ Nyquist, Er. Java. Lenguajes. [En línea] [Citado el: 06 de 05 de 2008.] http://nyquist.galeon.com/quees.htm.
- ✓ Cortez, Ernesto Alonso Lopez. Ref.Java. Lenguajes de Programación. [En línea] [Citado el: 06 de 05 de 2008.] http://www.edukativos.com/apuntes/archives/232.
- ✓ Valdés, Damián Pérez. JavaScript. Los diferentes lenguajes de programación para la web. [En línea] 02 de 11 de 2007. [Citado el: 06 de 05 de 2008.] http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/.
- ✓ Mas sobre javaScript. JavaScript. [En línea] [Citado el: 06 de 05 de 2008.] http://es.wikipedia.org/wiki/JavaScript.
- ✓ Navegadores Web. Navegador Web. [En línea] [Citado el: 06 de 05 de 2008.] http://es.wikipedia.org/wiki/Browser.
- ✓ López, Alejandro Cadavid. Concepto_Mozilla Firefox. Mozilla Firefox, el navegador web del momento. [En línea] 16 de 06 de 2004. [Citado el: 06 de 05 de 2008.] http://www.maestrosdelweb.com/editorial/firefox/.
- ✓ Caracteristicas de mozilla. Conocimientos Web.net. [En línea] [Citado el: 06 de 05 de 2008.] http://www.conocimientosweb.net/dt/article4880.html.
- ✓ SGBD. Garballe Collector. [En línea] 01 de 11 de 2004. [Citado el: 06 de 05 de 2008.] http://www.error500.net/garbagecollector/bases_de_datos/sistema_gestor_de_base_de_dato.html.

- ✓ Todo de PostgreSQL. PostGreSQL vs. MySQL. [En línea] [Citado el: 06 de 05 de 2008.] http://www.netpecos.org/docs/mysql_postgres/x15.html.
- ✓ Bases de datos. Postgre. Ubuntu Bases de Datos. [En línea] [Citado el: 06 de 05 de 2008.] https://help.ubuntu.com/ubuntu/serverguide/es/databases.html.
- ✓ MySQL. PostGreSQL vs. MySQL. [En línea] [Citado el: 06 de 05 de 2008.] http://www.netpecos.org/docs/mysql_postgres/x57.html.
- ✓ Oracle. [En línea] [Citado el: 06 de 05 de 2008.] http://www.uaem.mx/posgrado/mcruz/cursos/miic/oracle.pdf.
- ✓ Masip, David. Mas de Oracle. Qué es Oracle . [En línea] [Citado el: 06 de 05 de 2008.] http://www.telepolis.com/cgi-bin/web/DISTRITODOCVIEW?url=/basededatos/doc/articulos/oracle1.htm.
- ✓ Sql Server. Microsoft SQL Server. [En línea] [Citado el: 06 de 05 de 2008.] http://es.wikipedia.org/wiki/SQL_Server.
- ✓ Francisco, Mendoza. XML. Introducción a XML. [En línea] [Citado el: 06 de 05 de 2008.] http://www.monografias.com/trabajos6/ixml/ixml.shtml?monosearch.
- ✓ Servicios Web. Los servicios web (web services). [En línea] [Citado el: 06 de 05 de 2008.] http://www.banespyme.org/imagesWeb/ArchivoMultimedia/Documentacion/17/serviciosweb.pdf.
- ✓ Blog, Daniel. Ventajas y desv. Web service. Web Services. [En línea] 07 de 26 de julio. [Citado el: 06 de 05 de 2008.] http://danielvn7.wordpress.com/2007/07/26/web-services/.
- ✓ Servidores Web. Una Introducción a APACHE. [En línea] [Citado el: 06 de 05 de 2008.] http://linux.ciberaula.com/articulo/linux_apache_intro/.
- ✓ Rup. Proceso Unificado de Rational. [En línea] [Citado el: 06 de 05 de 2008.] http://es.wikipedia.org/wiki/RUP.
- ✓ Visual Paradigm. Sitio de descargas de Software. [En línea] 05 de 03 de 2007. [Citado el: 06 de 05 de 2008.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5Bcuenta_de_Linux_14716_p/.
- ✓ Zend Studio. Zend Studio. [En línea] [Citado el: 06 de 05 de 2008.] http://www.desarrolloweb.com/articulos/1178.php.
- ✓ frame_work. FrameWork. [En línea] [Citado el: 06 de 05 de 2008.] http://es.wikipedia.org/wiki/Framework.

- ✓ Diego. CodeIgniter-Framework-PHP. CodeIgniter-Framework-PHP. [En línea] 25 de 06 de 2007. [Citado el: 06 de 05 de 2008.] http://pixelco.us/blog/codeigniter-framework-php/.
- ✓ SCAFFOLDING. Sistemas aplicados a la Web. [En línea] 02 de 2008. [Citado el: 06 de 05 de 2008.] http://sistemasweb.wordpress.com/2008/02/24/que-es-framework-mvc-scaffolding/.
- ✓ YUI. Yahoo! UI Library. [En línea] [Citado el: 07 de 05 de 2008.] http://es.wikipedia.org/wiki/Yahoo!_UI_Library.
- ✓ YUI. Herramienta para programadores: Yahoo User Interface Library. [En línea] [Citado el: 07 de 05 de 2008.] http://www.baluart.net/articulo/herramienta-para-programadores-yahoo-user-interface-library.
- ✓ VALLE, JOSE GUILLERMO. Patrones de Arquitectura. Definición arquitectura cliente servidor. [En línea] 2005. [Citado el: 08 de 05 de 2008.] http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml.
- ✓ Rodriguez, Alberto. Modelo-Cliente-Servidor(En dos capas). Diseño de Aplicaciones Three Tier. [En línea] [Citado el: 06 de 05 de 2008.] http://www.fpress.com/revista/Num9711/Nov97.htm.
- ✓ Rodríguez, Caridad Salazar Alea y Antonio Cortina. ABC. Aplicaciones Distribuidas. [En línea] 2007. [Citado el: 06 de 05 de 2008.] http://www.monografias.com/trabajos14/aplicacion-distrib/aplicacion-distrib.shtml.
- ✓ MVC. Patrón "Modelo-Vista-Controlador". [En línea] [Citado el: 08 de 05 de 2008.] http://www.proactiva-calidad.com/java/patrones/mvc.html.
- ✓ Sevilla, Iván Ruiz. Patron Modelo-Vista-Controlador. una de Codigo. [En línea] [Citado el: 08 de 05 de 2008.] http://www.unadecodigo.com/2007/05/30/el-paradigma-modelo-vista-controlador-tutorial-ror-ii/.
- ✓ SISDIA. Eficiencia y seguridad para los centros de diálisis. [En línea] [Citado el: 05 de 05 de 2008.] http://www.sisdia.com.uy/index.html.
- ✓ Arquitectura Basada em 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.
- ✓ Arquitectura Orientada a Servicio. [En línea] [Citado el: 05 de 02 de 2008.] http://www.logiclibrary.com/.

- ✓ Rational Unified Process(RUP). [En línea] 2006. https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducción%20a%20RUP.doc.
- ✓ Rational Rose: Procedimientos básicos para desarrollar un proyecto con UML . [En línea] 2006. http://www.vico.org/TallerRationalRose.pdf.
- ✓ Rational Rose Enterprise Edition. [En línea] [Citado el: 05 de 02 de 2008.] http://www.ciao.es/Rational_Rose_Enterprise_Edition__Opinion_612900.
- ✓ Programación Extrema. [En línea] 2007. http://deigote.blogspot.com/2006/03/extreme-programming.htm.
- ✓ Modelo Vista Controlador. [En línea] [Citado el: 05 de 02 de 2008.] http://java.sun.com/blueprints/patterns/MVC-detailed.html.
- ✓ Microsoft Solution Framework. [En línea] [Citado el: 05 de 02 de 2008.] http://www.gpicr.com/msf.aspx.
- ✓ InfoDial. [En línea] [Citado el: 10 de 12 de 2007.] http://www.renal.org.ar/revista/53/5303.htm.
- ✓ Extreme Programming. [En línea] [Citado el: 05 de 02 de 2008.] http://ele-zeta.com.ar/2004/08/27/extreme-programming-xp/.
- ✓ Enterprise Architect. [En línea] [Citado el: 05 de 02 de 2008.] http://www.sparxsystems.com.ar/products/ea_features.html.
- ✓ Gil, Sandra Victoria Hurtado. Representación de la arquitectura de software usando UML. 2000.
- ✓ Garzón, Darwin Jiménez. RUP. [En línea] [Citado el: 05 de 02 de 2008.] http://codeticainge.googlepages.com/guiaing.pdf..
- ✓ Visual Architect. [En línea] [Citado el: 05 de 06 de 2008.] http://www.visual-paradigm.com/product/bpva/.
- ✓ UML-Análisis y diseño de Software. [En línea] 2006. http://cfrela.en.eresmas.com/uml/umlanalisis.htm.
- ✓ UML. [En línea] [Citado el: 05 de 02 de 2008.] http://gidis.ing.unlpam.edu.ar/personas/glafuente/uml/uml.html.
- ✓ Gutierrez, Jorge A. Saavedra. Patrones GRASP. [En línea] [Citado el: 05 de 02 de 2008.] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.

- ✓ Paradigm, Visual. Visual Paradigm UML for Enterprise Edition. [En línea] [Citado el: 05 de 06 de 2008.] http://www.visual-paradigm.com/product/vpuml/.
- ✓ NefroSoft. [En línea] [Citado el: 05 de 02 de 2008.] http://www.nefrosoft.com/ .
- ✓ NefroNet. [En línea] [Citado el: 05 de 02 de 2008.] http://www.renal.org.ar/calidad/informaticos.htm.
- ✓ Xavier Ferré Grau, María Isabel Sánchez Segura. Desarrollo Orientado a Objetos con UML.
- ✓ 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.
- ✓ Rivero, Jorge Alid Erazo. Patrones GRASP. [En línea] [Citado el: 5 de 2 de 2008.] http://jorgeerazo.blogspot.com/2006/08/patrones-grasp.html.
- ✓ Molpeceres, Alberto. Procesos de desarrollo, RUP,XP y FDD. [En línea] 15 de 12 de 2002. [Citado el: 05 de 02 de 2008.] http://www.willydev.net/InsiteCreation/v1.0/Descargas/articulos/general/cualxpfddrup.pdf.
- ✓ Umbrelo. [En línea] 2006. http://es.wikipedia.org/wiki/Umbrello.
- ✓ Grady Booch, Ivar Jacobon, James Rumbaugh. El Lenguaje Unificado de Modelado. s.l. : Addison Wesley, 2000. 9788478290444.
- ✓ Autores, Colectivo de. Buenas Prácticas en Hemodíalisis. Ciudad Habana : Editora Política, 2003.
- ✓ Larman, Craig. UML y Patrones. Segunda Edición. s.l. : Addison Wesley, 2002. 9788420534381.
- ✓ Arquitectura en tres capas. [En línea] [Citado el: 05 de 02 de 2008.] http://www.linuxjournal.com/article/3508 .
- ✓ Visual Paradigm. [En línea] [Citado el: 05 de 04 de 2008.] http://www.visual-paradigm.com/news/vpsuite33/vpuml63.jsp.
- ✓ Autores, Coelctivo de. Guías SENEFRO. España : Sociedad Española de Nefrología.
- ✓ Autores, Colectivo de. Buenas Prácticas en Hemodíalisis. Ciudad Habana : Editora Política, 2003.
- ✓ Larman, Craig. UML y Patrones. Primera Edición. s.l. : Addison Weley, 1999.
- ✓ Lovelle, Juan Manuel Cuevas. Introducción a UML. 1999.
- ✓ PRESSMAN, R. S. Ingeniería de Software. Un enfoque Práctico. Quinta. La Habana : Editorial Félix Varela, 2005.

GLOSARIO DE TÉRMINOS

Access: Herramienta para la administración de bases de datos relacionales que permite almacenar, compartir, buscar o manejar datos, así como analizar e imprimir información.

AJAX: Es una técnica de desarrollo web para crear aplicaciones web interactivas.

Algoritmo: Secuencia finita de instrucciones, cada una de las cuales tiene un significado preciso y puede ejecutarse con una cantidad finita de esfuerzo en un tiempo finito.

Ambulatoria: Atención medica que se brinda y permite al paciente regresar a su casa en pocas horas de ser atendido.

APIs: Application Programming Interface. Interfaz de Programación de Aplicaciones.

Base de datos: Conjunto de datos almacenados en memoria auxiliar que permite acceso directo y un conjunto de programas que manipulan esos datos.

Código fuente: Conjunto de instrucciones que componen un programa, escrito en cualquier lenguaje.

Compatibilidad: Similitud entre las características de donante y su receptor, que permiten que se pueda trasplantar un órgano.

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.

Diálisis peritoneal: La diálisis peritoneal es un procedimiento que permite depurar líquidos y electrolitos en pacientes que sufren insuficiencia renal aguda de distinta etiología y en otras patologías como alteraciones metabólicas e intoxicaciones.

Disfunción: Alteración de una función orgánica. Desarreglo en el funcionamiento de algún órgano o Sistema de órganos.

Enfermedad Renal Crónica: Es una enfermedad que se presenta cuando los riñones ya no pueden funcionar al nivel necesario para la vida diaria. Este padecimiento se presenta a medida que la insuficiencia renal crónica progresa a tal punto en que la capacidad de los riñones para excretar los desechos, concentrar la orina y regular los electrolitos es menos del 10% de su capacidad normal.

FTP: File Transfer Protocol. Protocolo de transferencia de archivos.

GNU: La letras GNU se refieren en inglés a No es UNIX (Not UNIX); es un sistema de programas compatible con UNIX, desarrollado por el Free Software Foundation (FSF).

GPL: Acrónimo en inglés de General Public License (Licencia Pública General). Esta licencia regula los derechos de autor de los programas de software libre (free software) promovido por el Free Software Foundation (FSF) en el marco de la iniciativa GNU. Permite la distribución de copias de programas (e incluso cobrar por ello), así como modificar el código fuente de los mismos o utilizarlo en otros programas.
Hyper Text Markup Language (Lenguaje de marcación de Hipertexto): es el lenguaje de marcas de texto.

Historia clínica: Documento que es utilizado por los profesionales de la salud para recoger información general de un paciente y anotar todas las acciones realizadas por los profesionales. Este consta de varias partes para su confección que puede variar en dependencia de la especialidad**.**

Hospitalaria: Atención medica que se brinda a los pacientes que se encuentras en el hospital.

IDEs: Integrated/Intelligent Drive Electronics. Es una especificación ATA. Es la interface de disco más comun para discos duros, CD ROMS, etc. Es fácil de usar, pero también tiene muchas limitaciones.

Informatización: Dotar a un servicio, organismo, etc., de medios informáticos, asegurar su gestión mediante medios informáticos. Utilizar la informática para tratar con ayuda de ordenador las necesidades de un sector profesional o para solucionar un problema.

Interfaz: Zona de contacto o conexión entre dos componentes de "hardware"; entre dos aplicaciones; o entre un usuario y una aplicación. Apariencia externa de una aplicación informática.

Insuficiencia renal: Es una pérdida súbita de la capacidad del riñón para excretar los residuos, concentrar la orina y conservar los electrolitos.

J2EE: Java2 Enterprise Edition.

Linux: Versión de libre distribución del sistema operativo UNIX el cual tiene todas las características que se pueden esperar de un moderno y flexible UNIX.

Logs: Registro de todos los hits que un servidor ha recibido en un período de tiempo dado el cual puede ser utilizado por auditores externos para registrar el uso del sitio.

Método dialítico: Es un procedimiento que se realiza para retirar los elementos tóxicos (impurezas o desechos) de la sangre cuando los riñones no pueden hacerlo.

MINSAP: Ministerio de Salud Pública.

Nefrólogo: Especialista en el tratamiento de enfermedades de los riñones.

Servicios nefrológicos: Centros que brindan atención médica a los pacientes que presentan una enfermedad renal.

Paciente ERC: Paciente con enfermedades renales crónicas.

Servidor: Un servidor es una computadora que maneja peticiones de data, email, servicios de redes y transferencia de archivos de otras computadoras (clientes). También puede referirse a un software específico, como lo es el servidor WWW.

Servlets: objeto que se ejecuta en un servidor o contenedor JEE(Java Enterprise Edition).

Sesión: Uso de los recursos de una computadora desde una terminal la cual no se encuentra cercana a dicha computadora.

Sistemas Operativos: Operating System (OS) en inglés. Programa especial el cual se carga en una computadora al prenderla, y cuya función es gestionar los demás programas, o aplicaciones, que se

ejecutarán, como por ejemplo, un procesador de palabras o una hoja de cálculo, un juego o una conexión a Internet. Windows, Linux, Unix, MacOS son todos sistemas operativos.

Software: Soporte lógico, dígase programas, utilidades, aplicaciones, sistemas operativos, que hacen posible que el usuario pueda trabajar con la máquina.

Software libre: Código fuente abierto, software libre se refiere a un programa cuyo código fuente está disponible al público general, gratis, para usar y modificar. El software libre no es siempre software gratuito (equivocación bastante habitual que tiene su origen de la palabra en inglés "free" que significa tanto "libre" como "gratuito").

Trasplante renal o trasplante de riñón: es un procedimiento quirúrgico para implantar un riñón sano en un paciente con insuficiencia renal.

UML: Lenguaje Unificado de Modelado

Unix: Sistema operativo especializado en capacidades de multiusuario y multitarea. Alta portabilidad al estar escrito en lenguaje C, lo que lo hace independiente del hardware.

URL: Acrónimo de Uniform Resource Locator. Localizador Uniforme de Recurso. Es el sistema de direcciones en Internet. El modo estándar de escribir la dirección de un sitio específico o parte de una información en el Web.

Windows: Sistema operativo desarrollado por la empresa Microsoft cuyas diversas versiones (3.1, 95, 98, NT, 2000, XP, ME, etc) han dominado de forma abrumadora el mercado de las computadoras personales, aunque no se puede decir lo mismo del mercado de redes corporativas.

WWW (World Wide Web): Es el sistema de información basado en hipertexto, cuya función es buscar y tener acceso a documentos a través de la red de forma que un usuario pueda acceder usando un navegador web.

WYSIWYG: La edición visual, también llamada WYSIWYG por What You See Is What You Get (en inglés, Lo Que Ve Es Lo Que Obtendrá), consiste en dar formato a su texto a medida que lo escribe. El editor visual va creando el código HTML "por detrás" mientras usted escribe. Tipos de letra, enlaces e imágenes se ven tal y como aparecerán en Internet, en el caso de un editor de HTML. Otros ejemplos, Microsoft Word es un procesador de palabras wysiwyg.

ANEXOS

Descripción de las clases

<< Gestionar Complementarios>>

Capa de Negocio. RERC. 1. E_Complementario

Nombre: E_Complementario	
Tipo de clase: Entidad	
Atributo	Tipo
paciente	string
fecha	datetime
hemoglobina	int
hematocrito	int
leucocito	int
cReticulocito	int
urea	int
creatinina	int
tgp	int
calcio	int
fosforo	int
acUrico	int
glicemia	int
hierroSerico	int
transfemina	int
colesterol	int
triglicérido	int
pcReactiva	int
linfocitos	int
pth	int
fal	int
albumina	int
glubolina	int

bilirrubinal bilirrubinaD observacion	int int string
Para cada responsabilidad:	
Nombre:	E_Complementario()
Descripción:	Constructor por defecto.
Nombre:	E_Complementario(paciente, fecha, hemoglobina, hematocrito, leucocito, cReticulocito, urea, creatinina, tgp, calcio, fosforo, acUrico, glicemia, hierroSerico, transfemina, colesterol, triglicérido, pcReactiva, linfocitos, pth, fal, albumina, glubolina, bilirrubinal, bilirrubinaD, observacion)
Descripción:	Constructor por parámetros.
Nombre:	Get_Fecha (): datetime
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Get_Paciente (): string
Descripción:	Devuelve el valor del atributo paciente.
Nombre:	Get_Hemoglobina (): int
Descripción:	Devuelve el valor del atributo hemoglobina.
Nombre:	Set_Hemoglobina (hemoglobina): void
Descripción:	Permite cambiar el valor del atributo hemoglobina.
Nombre:	Get_Hematocrito (): int
Descripción:	Devuelve el valor del atributo hematocrito.
Nombre:	Set_Hematocrito (hematocrito): void
Descripción:	Permite cambiar el valor del atributo hematocrito.
Nombre:	Get_Leucocito (): int
Descripción:	Devuelve el valor del atributo leucocito.
Nombre:	Set_Leucocito (leucocito): void
Descripción:	Permite cambiar el valor del atributo leucocito.
Nombre:	Get_CReticulocito (): int
Descripción:	Devuelve el valor del atributo cReticulocito.
Nombre:	Set_CReticulocito (cReticulocito): void
Descripción:	Permite cambiar el valor del atributo cReticulocito.
Nombre:	Get_Urea (): int

Descripción:	Devuelve el valor del atributo urea.
Nombre:	Set_Urea (urea): void
Descripción:	Permite cambiar el valor del atributo urea.
Nombre:	Get_Creatinina (): int
Descripción:	Devuelve el valor del atributo creatinina.
Nombre:	Set_Creatinina (creatinina): void
Descripción:	Permite cambiar el valor del atributo creatinina.
Nombre:	Get_TGP (): int
Descripción:	Devuelve el valor del atributo tgp.
Nombre:	Set_TGP (tgp): void
Descripción:	Permite cambiar el valor del atributo tgp.
Nombre:	Get_Calcio (): int
Descripción:	Devuelve el valor del atributo calcio.
Nombre:	Set_Calcio (calcio): void
Descripción:	Permite cambiar el valor del atributo calcio.
Nombre:	Get_Fosforo (): int
Descripción:	Devuelve el valor del atributo fosforo.
Nombre:	Set_Fosforo (fosforo): void
Descripción:	Permite cambiar el valor del atributo fosforo.
Nombre:	Get_ACUrico (): int
Descripción:	Devuelve el valor del atributo acUrico.
Nombre:	Set_ACUrico (acUrico): void
Descripción:	Permite cambiar el valor del atributo acUrico.
Nombre:	Get_Glicemia (): int
Descripción:	Devuelve el valor del atributo glicemia.
Nombre:	Set_Glicemia (glicemia): void
Descripción:	Permite cambiar el valor del atributo glicemia.
Nombre:	Get_HierroSerico (): int
Descripción:	Devuelve el valor del atributo hierroSerico.
Nombre:	Set_HierroSerico (hierroSerico): void
Descripción:	Permite cambiar el valor del atributo hierroSerico.
Nombre:	Get_Transfemina (): int

Descripción:	Devuelve el valor del atributo transfemina.
Nombre:	Set_Transfemina (transfemina): void
Descripción:	Permite cambiar el valor del atributo transfemina.
Nombre:	Get_Colesterol (): int
Descripción:	Devuelve el valor del atributo colesterol.
Nombre:	Set_Colesterol (colesterol): void
Descripción:	Permite cambiar el valor del atributo colesterol.
Nombre:	Get_Triglicérido (): int
Descripción:	Devuelve el valor del atributo triglicérido.
Nombre:	Set_Triglicérido (triglicérido): void
Descripción:	Permite cambiar el valor del atributo triglicérido.
Nombre:	Get_PCReactiva (): int
Descripción:	Devuelve el valor del atributo pcReactiva.
Nombre:	Set_PCReactiva (pcReactiva): void
Descripción:	Permite cambiar el valor del atributo pcReactiva.
Nombre:	Get_Linfocitos (): int
Descripción:	Devuelve el valor del atributo linfocitos.
Nombre:	Set_Linfocitos (linfocitos): void
Descripción:	Permite cambiar el valor del atributo linfocitos.
Nombre:	Get_PTH (): int
Descripción:	Devuelve el valor del atributo pth.
Nombre:	Set_PTH (pth): void
Descripción:	Permite cambiar el valor del atributo pth.
Nombre:	Get_FAL (): int
Descripción:	Devuelve el valor del atributo fal.
Nombre:	Set_FAL (fal): void
Descripción:	Permite cambiar el valor del atributo fal.
Nombre:	Get_Albumina (): int
Descripción:	Devuelve el valor del atributo albumina.
Nombre:	Set_Albumina (albumina): void
Descripción:	Permite cambiar el valor del atributo albumina.
Nombre:	Get_Glubolina (): int

Descripción:	Devuelve el valor del atributo glubolina.
Nombre:	Set_Glubolina (glubolina): void
Descripción:	Permite cambiar el valor del atributo glubolina.
Nombre:	Get_Bilirrubinal (): int
Descripción:	Devuelve el valor del atributo bilirrubinal.
Nombre:	Set_Bilirrubinal (bilirrubinal): void
Descripción:	Permite cambiar el valor del atributo bilirrubinal.
Nombre:	Get_BilirrubinaD (): int
Descripción:	Devuelve el valor del atributo bilirrubinaD.
Nombre:	Set_BilirrubinaD (bilirrubinaD): void
Descripción:	Permite cambiar el valor del atributo bilirrubinaD.
Nombre:	Get_Observacion (): string
Descripción:	Devuelve el valor del atributo observacion.
Nombre:	Set_Observacion (observacion): void
Descripción:	Permite cambiar el valor del atributo observacion.

Capa de Negocio y Capa de Acceso a Datos. RERC. 2. M_Complementario

Nombre: M_Complementario	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Adicionar($complem,$idV)
Descripción:	Adiciona un complementario a la BD, teniendo en cuenta la virología de los pacientes.
Nombre:	Editar($complem,$IDComp,$idV)
Descripción:	Actualiza en la BD un complementario seleccionado.
Nombre:	Listar($offset, $cant, $orden, $idPaciente)
Descripción:	Lista ordenadamente todos los complementarios que existen en la BD, dado un paciente.

Nombre:	ListarVirologias($offset, $cant, $orden, $idCom)
Descripción:	Listar ordenadamente las virologías de un paciente en determinado estudio complementario.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

Capa de Presentación. RERC. 3. C_Complementarios

Nombre: C_ Complementario	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Adicionar()
Descripción:	Adiciona un complementario con los datos que captura de la vista.
Nombre:	Editar()
Descripción:	Actualiza un complementario con los nuevos datos.
Nombre:	Listar($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todos los Complementarios.
Nombre:	ListarVirologias($offset, $cant, $orden, $dir,$id)
Descripción:	Muestra en una lista ordenada todas las Virologías dado el estudio complementario.
Nombre:	ListarV($startIndex, $results, $orden, $dir)
Descripción:	Muestra en una lista ordenada todas las Virologías para poder llenar el estudio complementario.

<< Gestionar Gasometría >>

Capa de Negocio. RERC. 4. E_ Gasometria

Nombre: Gasometría	
Tipo de clase: Entidad	
Atributo	**Tipo**
paciente	string

fecha	datetime
ph	int
pcO2	int
pO2	int
hcO3	int
abe	int
sO2	int
observacion	string

Para cada responsabilidad:	
Nombre:	E_Gasometria()
Descripción:	Constructor por defecto
Nombre:	E_Gasometria(fecha, ph, pcO2, pO2, hcO3, abe, sO2, observacion)
Descripción:	Constructor por parámetros
Nombre:	Get_Fecha(): datetime
Descripción:	Devuelve el valor del atributo fecha
Nombre:	Get_Paciente (): string
Descripción:	Devuelve el valor del atributo paciente.
Nombre:	Get_PH(): int
Descripción:	Devuelve el valor del atributo ph
Nombre:	Get_PCO2(): int
Descripción:	Devuelve el valor del atributo pcO2
Nombre:	Get_PO2(): int
Descripción:	Devuelve el valor del atributo pO2
Nombre:	Get_HCO3(): int
Descripción:	Devuelve el valor del atributo hcO3
Nombre:	Get_ABE(): int
Descripción:	Devuelve el valor del atributo abe
Nombre:	Get_SO2(): int
Descripción:	Devuelve el valor del atributo sO2
Nombre:	Set_ Fecha(fecha): void
Descripción:	Permite cambiar el valor del atributo fecha
Nombre:	Set_PH(ph): void

Descripción:	Permite cambiar el valor del atributo ph
Nombre:	Set_PCO2(pcO2): void
Descripción:	Permite cambiar el valor del atributo pcO2
Nombre:	Set_PO2(pO2): void
Descripción:	Permite cambiar el valor del atributo pO2
Nombre:	Set_HCO3(hcO3): void
Descripción:	Permite cambiar el valor del atributo hcO3
Nombre:	Set_ABE(abe): void
Descripción:	Permite cambiar el valor del atributo abe
Nombre:	Set_SO2(sO2): void
Descripción:	Permite cambiar el valor del atributo sO2
Nombre:	Get_Observacion(): string
Descripción:	Devuelve el valor del atributo observacion
Nombre:	Set_Observacion(observacion): void
Descripción:	Permite cambiar el valor del atributo observacion

Capa de Negocio. RERC. 5. M_ Gasometria

Nombre: M_Gasometria	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Adicionar($gasometria)
Descripción:	Adiciona una gasometría a la BD.
Nombre:	Editar($gasometria, $IDGas)
Descripción:	Actualiza una Gasometría seleccionada de la BD.
Nombre:	Listar($offset, $cant, $orden, $idPaciente)
Descripción:	Lista todas las Gasometrías que se encuentran en la BD.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

Capa de Presentación. RERC. 6. C_ Gasometria

Nombre: C_ Gasometria	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Adicionar()
Descripción:	Captura los datos de la vista, y los adiciona en la BD.
Nombre:	Editar()
Descripción:	Actualiza una Gasometría seleccionado con los datos nuevos.
Nombre:	Listar($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todas las Gasometrías.

<< Ecocardiograma >>

Capa de Negocio. RERC. 10. E_ Ecocardiograma

Nombre: E_PacienteEcoc	
Tipo de clase: Entidad	
Atributo	Tipo
fecha	datetime
ao	int
ai	int
ad	int
vid	int
vis	int
fe	int
pp	int
septum	int
resultado	string
Para cada responsabilidad:	
Nombre:	E_PacienteEcoc()
Descripción:	Constructor por defecto
Nombre:	E_PacienteEcoc($fecha,$ao,$ai,$ad,$vid,$vis,$fe,$pp,$sepTum,$resultado,$idTes

	t)
Descripción:	Constructor por parámetros
Nombre:	Get_Fecha(): datetime
Descripción:	Devuelve el valor del atributo
Nombre:	Set_Fecha(fecha): void
Descripción:	Permite cambiar el valor del atributo
Nombre:	Get_ AO(): int
Descripción:	Devuelve el valor del atributo
Nombre:	Set_AO(ao): void
Descripción:	Permite cambiar el valor del atributo
Nombre:	Get_AI(): int
Descripción:	Devuelve el valor del atributo
Nombre:	Set_AI(ai): void
Descripción:	Permite cambiar el valor del atributo
Nombre:	Get_AD(): int
Descripción:	Devuelve el valor del atributo
Nombre:	Set_AD(ad): void
Descripción:	Permite cambiar el valor del atributo
Nombre:	Get_VID(): int
Descripción:	Devuelve el valor del atributo
Nombre:	Set_VID(vid): void
Descripción:	Permite cambiar el valor del atributo
Nombre:	Get_VIS(): int
Descripción:	Devuelve el valor del atributo
Nombre:	Set_VIS(vis): void
Descripción:	Permite cambiar el valor del atributo
Nombre:	Get_FE(): int
Descripción:	Devuelve el valor del atributo
Nombre:	Set_FE(fe): void
Descripción:	Permite cambiar el valor del atributo
Nombre:	Get_PP(): int
Descripción:	Devuelve el valor del atributo

Nombre:	Set_PP(pp): void
Descripción:	Permite cambiar el valor del atributo
Nombre:	Get_SepTum(): int
Descripción:	Devuelve el valor del atributo
Nombre:	Set_SepTum(sepTum): void
Descripción:	Permite cambiar el valor del atributo
Nombre:	Get_Resultado(): int
Descripción:	Devuelve el valor del atributo
Nombre:	Set_ Resultado(resultado): void
Descripción:	Permite cambiar el valor del atributo

Capa de Negocio. RERC. 11. M_Ecocardiograma

Nombre: M_Ecocardiograma	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Adicionar($eco,$idE)
Descripción:	Adiciona un Ecocardiograma a la BD.
Nombre:	Editar($eco,$ID,$idDiag)
Descripción:	Actualiza en la BD un Ecocardiograma seleccionado
Nombre:	Listar($offset, $cant, $orden, $idPaciente,$idTest)
Descripción:	Lista ordenadamente todos los Ecocardiogramas existentes en la BD.
Nombre:	ListarDiagnosticos($offset, $cant, $orden,$idEco)
Descripción:	Listar los diagnósticos según el Ecocardiograma seleccionado.
Nombre:	AdicionarDiagnostico($idDiag)
Descripción:	Agrega el listado de diagnósticos al Ecocardiograma que se le registra al paciente.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

Capa de Presentación. RERC. 12. C_ Ecocardiograma

Nombre: C_ Ecocardiograma	
Tipo de clase (interfaz, controladora, entidad--)	
Para cada responsabilidad:	
Nombre:	Adicionar ()
Descripción:	Adiciona un Ecocardiograma con los datos que captura de la vista.
Nombre:	Editar ()
Descripción:	Actualiza un Ecocardiograma con los nuevos datos.
Nombre:	Listar ($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todos los Ecocardiogramas.
Nombre:	ListarDignosticos($offset, $cant, $orden, $dir, $id)
Descripción:	Lista los diagnósticos según el Ecocardiograma seleccionado.
Nombre:	ListarD($startIndex, $results, $orden, $dir)
Descripción:	Lista los diagnósticos para poder llenar el Ecocardiograma que se le registra al paciente.

<<Cistografía Miccional >>

Capa de Negocio. RERC. 13. E_CistografiaM

Nombre: E_CistografiaM	
Tipo de clase: Entidad	
Atributo	Tipo
idTest fecha resultado	int datetime string
Para cada responsabilidad:	
Nombre:	E_CistografiaM()
Descripción:	Constructor por defecto.
Nombre:	E_CistografiaM($fecha,$resultado,$idTest)
Descripción:	Constructor por parámetros.
Nombre:	Get_IdTest(): dateTime

Descripción:	Devuelve el valor del atributo idTest.
Nombre:	Get_Fecha(): dateTime
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Get_Resultado ():string
Descripción:	Devuelve el valor del atributo resultado.
Nombre:	Set_Resultado (resultado): void
Descripción:	Permite cambiar el valor del atributo resultado.

Capa de Negocio. RERC. 14. M_CistografiaM

Nombre: M_ CistografiaM	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Adicionar($cist,$idE)
Descripción:	Adiciona una cistografía miccional a la BD.
Nombre:	Editar($cist,$IDCist,$idDiag)
Descripción:	Actualiza en la BD una cistografía seleccionada
Nombre:	Listar ($offset, $cant, $orden, $idPaciente,$idTest)
Descripción:	Lista ordenadamente todas las cistografías existentes en la BD.
Nombre:	AdicionarDiagnostico($idDiag)
Descripción;	Agrega los posibles diagnósticos a la Cistografía que se le registra al paciente.
Nombre:	ListarDiagnosticos($offset, $cant, $orden,$idCist)
Descripción:	Lista los diagnósticos según las Cistografía Miccional seleccionada.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

Capa de Presentación. RERC. 15. C_CistografiaM

Nombre: C_ CistografiaM	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Adicionar ()

Descripción:	Adiciona una cistografía con los datos que captura de la vista.
Nombre:	Editar ()
Descripción:	Actualiza una cistografía con los nuevos datos.
Nombre:	Listar ($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todas las cistografías miccionales.
Nombres:	ListarD($startIndex, $results, $orden, $dir)
Descripción:	Lista los diagnósticos para poder llenar la Cistografía Miccional que se le registra al paciente.

<< Estudio Vascular >>

Capa de Negocio. RERC. 16. E_ EstudioVascular

Nombre: E_ EstudioVascular	
Tipo de clase: Entidad	
Atributo	Tipo
idTest fecha resultado	Int varchar string
Para cada responsabilidad:	
Nombre:	E_PacienteEstudVasc()
Descripción:	Constructor por defecto
Nombre:	E_PacienteEstudVasc($fecha,$resultado,$idTest)
Descripción:	Constructor por parámetros
Nombre:	Get_Fecha(): datetime
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Get_IdTest(): int
Descripción:	Devuelve el valor del atributo idTest.
Nombre:	Get_Resultado(): varchar
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Set_Resultdao(resultado): void
Descripción:	Permite cambiar el valor del atributo fecha.

Capa de Negocio. RERC. 17. M_ EstudioVascular

Nombre: M_ EstudioVascular	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Adicionar($Estudio,$idE)
Descripción:	Adiciona un Estudio Vascular a la BD.
Nombre:	Editar($cist,$IDEst,$idDiag)
Descripción:	Actualiza en la BD un Estudio Vascular seleccionado
Nombre:	Listar($offset, $cant, $orden, $idPaciente,$idTest)
Descripción:	Lista ordenadamente todos los Estudios Vasculares que existen en la BD.
Nombre:	AdicionarDiagnostico($idDiag)
Descripción:	Agrega los diagnósticos al estudio que se le registra al paciente.
Nombre:	ListarDiagnosticos($offset, $cant, $orden, $idEst)
Descripción:	Lista los diagnósticos según el estudio seleccionado.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

Capa de Presentación. RERC. 18. C_ Estudio Vascular

Nombre: C_EstudioVascular	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Adicionar()
Descripción:	Adiciona un Estudio Vascular con los datos que captura de la vista.
Nombre:	Editar()
Descripción:	Actualiza un Estudio Vascular con los nuevos datos.
Nombre:	Listar($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todos los Estudios Vasculares.
Nombre:	ListarD($startIndex, $results, $orden, $dir)

Descripción:	Lista los diagnósticos para poder llenar el estudio que se le registra al paciente.
Nombre:	ListarDignosticos($offset, $cant, $orden, $dir,$id)
Descripción:	Lista los diagnósticos pertenecientes al estudio seleccionado.

.

<< Trasfusión >>

Capa de Negocio. RERC. 28. E_Transfucion

Nombre: E_Transfucion	
Tipo de clase: Entidad	
Atributo	Tipo
idCausa paciente fecha observación unidades	int string datetime string int
Para cada responsabilidad:	
Nombre:	E_Transfucion ()
Descripción:	Constructor por defecto.
Nombre:	E_Transfucion ($unidades, $observacion, $fecha, $paciente, $idCausa)
Descripción:	Constructor por parámetros.
Nombre:	Get_Fecha(): varchar
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Get_Unidades(): int
Descripción:	Devuelve el valor del atributo unidades.
Nombre:	Set_Unidades (unidades): void
Descripción:	Permite cambiar el valor del atributo unidades.
Nombre:	Get_Observacion(): string
Descripción:	Devuelve el valor del atributo observacion.
Nombre:	Set_Observacion (observacion): void
Descripción:	Permite cambiar el valor del atributo observacion.
Nombre:	Get_Causas(): string

Descripción:	Devuelve el valor del atributo idCausa.

Capa de Negocio. RERC. 29. M_Transfucion

Nombre: M_Transfucion	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Adicionar($trans)
Descripción:	Adiciona una transfusión a la BD.
Nombre:	Editar($trans,$idT)
Descripción:	Actualiza en la BD una transfusion seleccionada
Nombre:	Listar($offset, $cant, $orden, $idPaciente)
Descripción:	Lista todas las transfusiones que se encuentran en la BD.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

Capa de Presentación. RERC. 30. C_Transfucion

Nombre: C_Transfucion	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Adicionar()
Descripción:	Adiciona una transfusión con los datos que captura de la vista.
Nombre:	Editar()
Descripción:	Actualiza una transfusión con los nuevos datos.
Nombre:	Listar($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todas las transfusiones

<< Ultrasonografía >>

Capa de Negocio. RERC. 31. E_Ultrasonografia

Nombre: E_Ultrasonografia	
Tipo de clase: Entidad	
Atributo	Tipo
fecha resultado paciente ultrasonografia	datetime string string string
Para cada responsabilidad:	
Nombre:	E_Ultrasonografia ()
Descripción:	Constructor por defecto
Nombre:	E_Ultrasonografia ($fecha, $resultado, $paciente, $ultrasonografia)
Descripción:	Constructor por parámetros
Nombre:	Get_Fecha(): datetime
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Get_Resultado (): string
Descripción:	Devuelve el valor del atributo resultado.
Nombre:	Set_Resultado (resultado): void
Descripción:	Permite cambiar el valor del atributo resultado.
Nombre:	Get_Paciente(): string
Descripción:	Devuelve el valor del atributo paciente.
Nombre:	Get_Ultrasonografia (): string
Descripción:	Devuelve el valor del atributo Ultrasonografía.
Nombre:	Set_Ultrasonografia(ultrasonografia): void
Descripción:	Permite cambiar el valor del atributo Ultrasonografía.

Capa de Negocio. RERC. 32. M_Ultrasonografia

Nombre: M_Ultrasonografia
Tipo de clase: Modelo
Para cada responsabilidad:

Nombre:	Adicionar($ultra)
Descripción:	Adiciona una ultrasonografía a la BD.
Nombre:	Editar($ultra, $iDPU)
Descripción:	Actualiza en la BD una ultrasonografía seleccionada
Nombre:	Listar($offset, $cant, $orden, $idPaciente)
Descripción:	Lista ordenadamente todas las Ultrasonografías que existen en la BD.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

Capa de Presentación. RERC. 33. C_Ultrasonografia

Nombre: C_Ultrasonografia	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Adicionar()
Descripción:	Adiciona una Ultrasonografía con los datos que captura de la vista.
Nombre:	Editar()
Descripción:	Actualiza una Ultrasonografía con los nuevos datos.
Nombre:	Listar($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todas las Ultrasonografías.

<< Ionograma >>

Capa de Negocio. RERC. 37. E_Ionograma

Nombre: E_ Ionograma	
Tipo de clase: Entidad	
Atributo	Tipo
paciente	string
fecha	datetime
na	int
k	int
cl	int

observacion	string
Para cada responsabilidad:	
Nombre:	E_Ionograma()
Descripción:	Constructor por defecto.
Nombre:	E_Ionograma($fecha, $paciente, $na, $k, $cl, $observacion)
Descripción:	Constructor por parámetros.
Nombre:	Get_Paciente(): string
Descripción:	Devuelve el valor del atributo paciente.
Nombre:	Get_Fecha(): datetime
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Get_Na ():int
Descripción:	Devuelve el valor del atributo na.
Nombre:	Set_Na(na): void
Descripción:	Permite cambiar el valor del atributo na.
Nombre:	Get_K(): int
Descripción:	Devuelve el valor del atributo k.
Nombre:	Set_K(k): void
Descripción:	Permite cambiar el valor del atributo k.
Nombre:	Get_Cl():int
Descripción:	Devuelve el valor del atributo cl.
Nombre:	Set_Cl(cl): void
Descripción:	Permite cambiar el valor del atributo cl.
Nombre:	Get_Observacion():string
Descripción:	Devuelve el valor del atributo observacion.
Nombre:	Set_Observacion(observacion): void
Descripción:	Permite cambiar el valor del atributo observacion.

Capa de Negocio. RERC. 38. M_Ionograma

Nombre: M_ Ionograma
Tipo de clase: Modelo

Para cada responsabilidad:	
Nombre:	Adicionar($ionograma)
Descripción:	Adiciona un Ionograma a la BD.
Nombre:	Editar($ionograma, $IDIO)
Descripción:	Actualiza un Ionograma seleccionado de la BD.
Nombre:	Listar($offset, $cant, $orden, $idPaciente)
Descripción:	Lista todos los Ionogramas que se encuentran en la BD.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

Capa de Presentación. RERC. 39. C_ Ionograma

Nombre: C_Ionogramas	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Adicionar()
Descripción:	Captura los datos de la vista, y los adiciona en la BD.
Nombre:	Editar()
Descripción:	Actualiza un Ionograma seleccionado con los datos nuevos.
Nombre:	Listar($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todos los Ionogramas.

<< Vacuna >>

Capa de Negocio. RERC. 43. E_ Vacuna

Nombre: E_ Vacuna	
Tipo de clase: Entidad	
Atributo	Tipo
idVacuna	int

paciente fecha dosis observaciones	string datetime int string
Para cada responsabilidad:	
Nombre:	E_Vacuna()
Descripción:	Constructor por defecto.
Nombre:	E_Vacuna($dosis, $observacion, $fecha, $paciente, $idVacuna)
Descripción:	Constructor por parámetros.
Nombre:	Get_IdVacuna(): int
Descripción:	Devuelve el valor del atributo idVacuna.
Nombre:	Get_Paciente(): string
Descripción:	Devuelve el valor del atributo paciente.
Nombre:	Get_Fecha(): datetime
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Get_Dosis(): int
Descripción:	Devuelve el valor del atributo dosis.
Nombre:	Set_Dosis(dosis): void
Descripción:	Permite cambiar el valor del atributo dosis.
Nombre:	Get_Observacion():string
Descripción:	Devuelve el valor del atributo observacion.
Nombre:	Set_Observacion(observacion): void
Descripción:	Permite cambiar el valor del atributo observacion.

Capa de Negocio. RERC. 44. M_Vacuna

Nombre: M_Vacuna	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Adicionar($vacuna)
Descripción:	Adiciona una vacuna a la BD.

Nombre:	Editar($vacuna,$idV)
Descripción:	Actualiza en la BD una vacuna seleccionada
Nombre:	Listar($offset, $cant, $orden, $idPaciente)
Descripción:	Lista ordenadamente todas las vacunas existentes en la BD.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.

Capa de Presentación. RERC. 45. C_Vacuna

Nombre: C_ Vacuna	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Adicionar()
Descripción:	Adiciona una vacuna con los datos que captura de la vista.
Nombre:	Actualizar ()
Descripción:	Actualiza una vacuna con los nuevos datos.
Nombre:	Listar($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todas las vacunas.

<< Tratamiento >>

Capa de Negocio. RERC. 46. E_Tratamiento

Nombre: E_Tratamiento	
Tipo de clase: Entidad	
Atributo	Tipo
paciente fecha observaciones	string datetime string
Para cada responsabilidad:	
Nombre:	E_Tratamiento()
Descripción:	Constructor por defecto.

Nombre:	E_Tratamiento($fecha, $paciente, $obser)
Descripción:	Constructor por parámetros.
Nombre:	Get_Paciente(): string
Descripción:	Devuelve el valor del atributo paciente.
Nombre:	Get_Fecha(): datetime
Descripción:	Devuelve el valor del atributo fecha.
Nombre:	Get_Observacion():string
Descripción:	Devuelve el valor del atributo observacion.
Nombre:	Set_Observacion(observacion): void
Descripción:	Permite cambiar el valor del atributo observacion.

Capa de Negocio. RERC. 47. M_Tratamiento

Nombre: M_Tratmiento	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Adicionar($tratamiento)
Descripción:	Adiciona una vacuna a la BD.
Nombre:	Editar($tratamiento, $IDT, $ant, $espcEC, $espcER, $higD)
Descripción:	Actualiza en la BD una vacuna seleccionada
Nombre:	Listar($offset, $cant, $orden, $idPaciente)
Descripción:	Lista ordenadamente todas las vacunas existentes en la BD.
Nombre:	EstaPacienteModulo($paciente)
Descripción:	Convierte al paciente RPN en paciente RERC.
Nombre:	InsertarAntih($antih)
Descripción:	Agrega los tratamientos antihipertensivos a la BD.
Nombre:	InsertarEspcEC($EspcEC)
Descripción:	Agrega los tratamientos específicos para la enfermedad que conllevo al paciente a la enfermedad renal en la BD.
Nombre:	InsertarEspcER($EspcER)
Descripción:	Agrega los tratamientos específicos para la enfermedad renal de la cual

	padece el paciente en la BD.
Nombre:	InsertarHigD($HigD)
Descripción:	Agrega los tratamientos en cuanto a higiene y alimentación se refiere, del paciente en la BD.

Capa de Presentación. RERC. 48. C_Tratamiento

Nombre: C_Tratamiento	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	Adicionar()
Descripción:	Adiciona una vacuna con los datos que captura de la vista.
Nombre:	Actualizar ()
Descripción:	Actualiza una vacuna con los nuevos datos.
Nombre:	Listar($offset, $cant, $orden, $dir)
Descripción:	Muestra en una lista ordenada todas las vacunas.
Nombre:	ListarAntihip($startIndex, $results, $orden, $dir)
Descripción:	Lista los tratamientos Antihipertensivos para poder llenar el registro de tratamientos del paciente.
Nombre:	ListarEspcEC($startIndex, $results, $orden, $dir)
Descripción:	Lista los tratamientos específicos de las enfermedades causales para poder llenar el registro de tratamientos del paciente.
Nombre:	ListarEspcER($startIndex, $results, $orden, $dir)
Descripción:	Lista los tratamientos específicos de la enfermedad renal para poder llenar el registro de tratamientos del paciente.
Nombre:	ListarHig($startIndex, $results, $orden, $dir)
Descripción:	Lista los tratamientos referentes a la higiene y alimentación del paciente para poder llenar el registro de tratamientos.

Descripción de las tablas de la BD

Nombre: e_pacienteTransrc		
Descripción: Tabla que guarda las transfusiones de los pacientes.		
Atributo	**Tipo**	**Descripción**
idPT	INTEGER	Identificador de la tabla.
fecha	VARCHAR	Guarda la fecha.
unidades	VARCHAR	Guarda las unidades, o sea, la cantidad a transfundir.
observacion	VARCHAR	Guarda las observaciones hechas al paciente.
idCausa	INTEGER	Guarda el identificador de la tabla e_causa_transfucioncfg.
idPaciente	INTEGER	Guarda el id de la tabla e_pacienterc.

Nombre: e_pacienteVacunarc		
Descripción: Tabla que guarda los tipos de vacunas de los pacientes.		
Atributo	**Tipo**	**Descripción**
idPV	INTEGER	Identificador de la tabla.
dosis	INTEGER	Guarda las dosis, o sea, la cantidad a vacunar.
observacion	VARCHAR	Guarda las observaciones hechas al

		paciente.
fecha	VARCHAR	Guarda la fecha.
idPaciente	INTEGER	Guarda el id de la tabla e_pacienterc.
idVacuna	INTEGER	Guarda el id de la tabla e_vacunacfg

Nombre: e_gasometriarerc		
Descripción: Tabla que guarda las gasometrías de los pacientes.		
Atributo	**Tipo**	**Descripción**
idGas	INTEGER	Identificador de la tabla, numero único.
fecha	VARCHAR	Guarda la fecha.
ph	INTEGER	Guarda el tipo de análisis ph.
pc02	INTEGER	Guarda el tipo de análisis pc02.
p02	INTEGER	Guarda el tipo de análisis p02.
hc03	INTEGER	Guarda el tipo de análisis hc03.
abe	INTEGER	Guarda el tipo de análisis abe.
s02	INTEGER	Guarda el tipo de análisis s02.
observacion	VARCHAR	Guarda las observaciones hechas al paciente.
idPaciete	INTEGER	Guarda el id de la tabla e_pacienterc.

Nombre: e_pacientehistoricorc		
Descripción: Tabla que guarda los datos de un paciente histórico o sea que se atendió en un tiempo pasado.		
Atributo	**Tipo**	**Descripción**
id_paciente	INTEGER	Identifica al paciente, es un numero único
fecha_salida	VARCHAR	Guarda la fecha de salida del paciente histórico.

Nombre: e_ecocardiogramarc		
Descripción: Tabla que guarda los ecocardiogramas de los pacientes.		
Atributo	**Tipo**	**Descripción**
idEcoc	INTEGER	Identificador de la tabla, número único.
fecha	VARCHAR	Guarda la fecha.
ao	INTEGER	Guarda el valor del tipo de análisis ao.
ai	INTEGER	Guarda el valor del tipo de análisis ai.
ad	INTEGER	Guarda el valor del tipo de análisis ad.
vid	INTEGER	Guarda el valor del tipo de análisis vid.
vis	INTEGER	Guarda el valor del tipo de análisis vis.
fe	INTEGER	Guarda el valor del tipo de análisis fe.
pp	INTEGER	Guarda el valor del tipo de análisis pp.
sepTum	INTEGER	Guarda el valor del tipo de análisis septum.

resultado	VARCHAR	Guarda el valor del resultado.
idTest	INTEGER	Guarda el id de la tabla e_testrc.

Nombre: e_ionogramarc		
Descripción: Tabla que guarda los ionogramas de los pacientes.		
Atributo	**Tipo**	**Descripción**
idIO	INTEGER	Identificador de la tabla, numero único.
na	INTEGER	Guarda el tipo de análisis na.
k	INTEGER	Guarda el tipo de análisis k.
cl	INTEGER	Guarda el tipo de análisis cl.
idPaciente	INTEGER	Guarda el id de la tabla e_pacienterc.
observacion	VARCHAR	Guarda las observaciones hechas al paciente.
fecha	VARCHAR	Guarda la fecha.

Nombre: e_psensibilidadrc		
Descripción: Tabla que guarda el por ciento de sensibilidad de los pacientes.		
Atributo	**Tipo**	**Descripción**
fecha	VARCHAR	Guarda la fecha.
ps	FLOAT	Guarda el tipo de análisis ps.

observaciones	VARCHAR	Guarda las observaciones hechas al paciente.
idPaciente	INTEGER	Guarda el id de la tabla e_pacienterc.
idPS	INTEGER	Identificador de la tabla, número único.

Nombre: e_complementariorc		
Descripción: Tabla que guarda los análisis complementarios de los pacientes.		
Atributo	**Tipo**	**Descripción**
idCompl	INTEGER	Identificador de la tabla, numero único.
fecha	VARCHAR	Guarda la fecha.
hemoglobina	INTEGER	Guarda el tipo de análisis hemoglobina.
hematocrito	FLOAT	Guarda el tipo de análisis hematocrito.
leucocito	INTEGER	Guarda el tipo de análisis leucocito.
cReticulocito	INTEGER	Guarda el tipo de análisis cReticulocito.
urea	INTEGER	Guarda el tipo de análisis urea.
creatinina	INTEGER	Guarda el tipo de análisis creatinina.
tgp	INTEGER	Guarda el tipo de análisis tgp.
calcio	INTEGER	Guarda el tipo de análisis calcio.
fosforo	INTEGER	Guarda el tipo de análisis fosforo.
acUrico	INTEGER	Guarda el tipo de análisis acUrico.

glicemia	INTEGER	Guarda el tipo de análisis glicemia.
hierroSerico	INTEGER	Guarda el tipo de análisis hierroSerico.
transfemina	INTEGER	Guarda el tipo de análisis transfemina.
colesterol	INTEGER	Guarda el tipo de análisis colesterol.
triglicerido	INTEGER	Guarda el tipo de análisis triglicerido.
pcReactiva	INTEGER	Guarda el tipo de análisis pcReactiva.
linfocitos	INTEGER	Guarda el tipo de análisis linfocitos.
pth	INTEGER	Guarda el tipo de análisis pth.
fal	INTEGER	Guarda el tipo de análisis fal.
albumina	INTEGER	Guarda el tipo de análisis albumina.
glubolina	INTEGER	Guarda el tipo de análisis glubolina.
bilirrubinal	INTEGER	Guarda el tipo de análisis bilirrubinal.
bilirrubinaD	INTEGER	Guarda el tipo de análisis bilirrubinaD.
observacion	VARCHAR	Guarda las observaciones.
idPaciente	INTEGER	Guarda el id de la tabla e_pacienterc.
idPuntofrio	INTEGER	Guarda el id de la tabla e_puntoFriorc.

Nombre: e_estudioVascularrc		
Descripción: Tabla que guarda los estudios vasculares de los pacientes.		
Atributo	**Tipo**	**Descripción**

idEV	INTEGER	Identificador de la tabla, numero único.
fecha	VARCHAR	Guarda la fecha
resultado	VARCHAR	Guarda el resultado de los estudios realizados
idTest	INTEGER	Guarda el id de la tabla e_testrc.

Nombre: e_cistogMicrc		
Descripción: Tabla que guarda los análisis de cistografías miccionales de los pacientes.		
Atributo	**Tipo**	**Descripción**
idCM	INTEGER	Identificador de la tabla, numero único.
fecha	VARCHAR	Guarda la fecha
resultado	VARCHAR	Guarda el resultado de los análisis.
idTest	INTEGER	Guarda el id de la tabla e_testrc.