Universidad de las Ciencias Informáticas

Facultad 7

Título: Implementación del Módulo Centro Coordinador Provincial de Emergencias Médicas

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Miguel Angel Fernández Marín

Andy Morfa Hernández

Tutor: Ing. Luis Mariano Reyna Soler

Asesora: Ing. Lourdes Escalona Peral

Ciudad de La Habana, junio del 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 27 días del mes de junio del año 2008.

_____ _____

Miguel Angel Fernández Marín

Autor

Andy Morfa Hernández

Autor

_____

Ing. Luis Mariano Reyna Soler

Tutor

DATOS DE CONTACTO

Lourdes Escalona Peral (lescalonap@uci.cu): Graduado de Ingeniero Informático en la Universidad de Holguín en el 2004. Ha impartido las asignaturas Ingeniería de Software 1 y 2 y Seminario de Tesis. Fue líder del proyecto Atención Primaria de Salud durante dos años consecutivos y asesora de calidad de la facultad durante un año. Ha tutorado tesis de perfil de análisis y diseño, implementación de sistema y atendiendo al área de calidad, Pruebas de software. Posee la categoría docente de Instructor y cursa la maestría Gestión de Proyectos en la UCI. Se desempeña como Jefe de Dpto. de la Especialidad de la Facultad 7 en los últimos 2 años.

Luis Mariano Reyna Soler (lreyna@uci.cu): Graduado de Ingeniero Informático en la CUJAE en el curso 2006. Imparte las asignaturas Introducción a la Programación y Programación I desde hace dos años en la Facultad 7. Posee la categoría docente de Instructor. Es jefe de las asignaturas Introducción a la Programación y Programación I en la Facultad 7.

"El hombre que pone corazón en lo que hace, consigue recursos donde los incapaces se dan por vencidos"

Simón Bolívar.

AGRADECIMIENTOS

Agradecemos:

Al Comandante en Jefe Fidel Castro Ruz y a la Revolución por haber transformado sueños y esperanzas en realidades.

A nuestros queridos profesores que han contribuido en nuestra formación profesional.

A la Universidad de las Ciencias Informáticas por confiarnos tantos recursos en pos del conocimiento.

Miguel Angel: *Gracias a mis amigos por apoyarme en todos estos años, a los profesores que han contribuido en mi formación como ingeniero, en especial agradezco a Lourdes Escalona Peral, a Pura Miguel Tamayo y a Orlenis Vega Rodríguez.*

A mi querida novia Debora González Tolmo que ha contribuido de una forma especial en la realización de esta investigación, su apoyo y su presencia han hecho de la desesperación una infinita perseverancia. Gracias mi amor por ser como eres.

A mis tres grandes amores, Elidianys, Blanca Rosa y Cecilia por haber confiado en mí y haber apoyado cada paso que he dado en la vida, las amo y doy gracias por tenerlas junto a mi por siempre.

Gracias a Andy por ser un excelente compañero y un intachable amigo.

A las analistas del presente trabajo investigación Yarielys Hernandez Fonticiella y Marisel Rivera Alarcón muchas gracias por su colaboración y por ser tan buenas amigas, las quiero mucho.

Andy: *Gracias a mi familia que siempre ha estado a mi lado en las buenas y en las malas.*

A mi compañero de tesis Miguel Angel que desde 1er año hemos sido como hermanos y que en los momentos más duros en la universidad siempre me tendió su mano.

A todos mis amigos que, mencionarlos hoy sería una hipocresía de mi parte porque siempre se me quedaría alguno pero que pueden estar seguros que los recordaré siempre pues marcaron en su momento muy profundamente mi vida.

A mi tutor por dedicarnos su tiempo para que la tesis concluyera satisfactoriamente, y por el ánimo que me daba cada día cuando se pensaba que no se iba a concluir a tiempo el trabajo.

Gracias a todas las personas que han llenado de alegría mi vida durante todos estos años de estudio.

DEDICATORIA

Dedico la presente investigación a mi familia, en especial a Blanca Rosa y a Cecilia por enseñarme el significado de la vida.

A mi querida hermana Elidianys.

A mi novia Debora González Tolmo

A mi amiga Gretel Rubio Ríos.

Miguel Angel Fernández Marín.

A mis padres por poner cada día con esmero y dedicación un granito de arena en mi formación.

A mi hermana por preocuparse tanto por mí y regalarme su cariño en todo momento.

A mi abuela Alicia la persona más dulce de este mundo que ni un momento de su vida ha dejado de tenerme presente.

A mi abuelo Tomás que aunque hoy no está entre nosotros siempre será un paradigma a seguir por mí.

Andy Morfa Hernández.

RESUMEN

En los Centros Coordinadores Provinciales de Emergencia Médica, las demandas se realizan de forma manual y se archivan en formato duro. Esto provoca demora en la recepción de datos, quejas de la población, pérdidas y deterioro de la información, cuando se necesita elaborar algún reporte se realiza una búsqueda de datos que es agotadora dado el volumen de la misma.

Para darle solución a la problemática anterior, se desarrolló este trabajo que tiene como objetivo implementar una aplicación Web que facilite la gestión de la información de las demandas en los Centros Coordinadores Provinciales de Emergencia Médica (CCPEM).

Para realizar el software se utilizó como lenguaje de programación PHP 5.2.3, el gestor de base de datos MySQL 5.0.36. Las herramientas usadas para su confección fueron Zend Studio 5.5.0 que es un potente IDE de desarrollo para PHP. Case Studio 2 para el diseño de la base de datos y Visual Paradigm for UML 3.0 para el análisis y diseño del software.

Como resultado de la investigación se obtuvo una herramienta que gestiona la información de los Centros Coordinadores Provinciales de Emergencia Médica, mejorando la calidad del almacenamiento de la información y la rapidez con que se atiende los casos de urgencias y emergencias.

Palabras claves: Sistema Informatizado de Urgencias Médicas, aplicación Web, gestión de la información, Centros Coordinadores Provinciales de Emergencia Médica.

TABLA DE CONTENIDOS

AGRADECIMIENTOS I
DEDICATORIA II
RESUMEN III
INTRODUCCIÓN 1
CAPITULO I: FUNDAMENTACIÓN TEÓRICA 5
1.1 Conceptos básicos asociados al problema. 5
1.2 Sistemas internacionales vinculados a la atención de emergencias. 6
1.3 Tendencias actuales de la tecnología en el desarrollo de software. 10
CAPITULO II: ELEMENTOS DE LA ARQUITECTURA. 19
2.1 Argumentación de los requisitos no funcionales del sistema propuesto. 19
2.2 Patrones o estilos arquitectónicos presentes en la propuesta de solución. 23
2.3 Valoración crítica del diseño propuesto por el analista. 31
2.4 Vista de despliegue. 31
2.5 Vista de implementación. 34
CAPÍTULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA 42
3.1 Análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser reutilizados. Estrategias de integración. 42
3.2 Descripción de las clases u operaciones necesarias. 43
3.3 Diseño de la BD. 68
3.4 Descripción de las tablas de la base de datos. 70
CONCLUSIONES 84
RECOMENDACIONES 85
BIBLIOGRAFÍA 86
DOCUMENTOS CITADOS 90
ANEXOS 94
GLOSARIO DE TÉRMINOS 101

INTRODUCCIÓN

En Cuba, el estado y el gobierno garantizan la salud a la población, teniendo como premisa que la salud es un derecho de todos los individuos y una responsabilidad del estado.

El Ministerio de Salud Pública (MINSAP) es el organismo rector del Sistema Nacional de Salud (SNS) y es el responsable de atender la salud a través de una red de servicios organizados en el sistema de salud, que tiene como características ser único, integral y regionalizado. [1]

El SNS en Cuba es gratuito para todos los cubanos, independientemente del sexo, raza, ideología y religión. Constituye una prioridad la atención al paciente grave, exigiendo una respuesta rápida y precisa en situaciones críticas, así como una constante actualización y entrenamiento del equipo médico, el cual tiene la responsabilidad de ofrecer tratamiento al paciente. [2]

Entre los años 1980 y 1989 con la epidemia del dengue, en el país se crean y desarrollan las unidades de Terapia Intensiva e Intermedias. Entre los años 1993 y1994 se desarrollan las Vice-direcciones de Urgencia Hospitalarias. En el año 1996, se inicia la Red de Urgencia en Atención Primaria. En 1997 se inicia la Emergencia Médica Móvil, dando inicio el Centro Nacional de Urgencias Médicas (CNUM). [3]

El CNUM está insertado dentro del SNS, y su misión es organizar la atención de urgencias y emergencias médicas desde la Comunidad, Consultorios, Policlínicos, Coordinaciones de Ambulancias de Urgencias y Emergencias hasta el sistema de Emergencia y Terapia Hospitalaria. Se establece mediante un proceso de evaluación y decisión médica, a través de los diferentes eslabones del Sistema Nacional de Salud. [4]

Figura 1: Órganos aseguradores del Sistema Nacional de Salud.

INTRODUCCIÓN

Actualmente, cada provincia del país tiene un Centro Coordinador Provincial de Emergencia Médica (CCPEM), que es atendido directamente por el CNUM, y en cada municipio existe una base de ambulancia que atiende los casos de urgencias y emergencias que le asigne.

El CNUM ha elaborado un Plan de Acción homogéneo para todos los CCPEM del país. Pero cada uno ellos basados en sus experiencias de trabajo diario, necesidades y características laborales, han adaptado el Plan de Acción. De forma tal que existe heterogeneidad en la información, no se ejecutan los procedimientos de la misma manera al atender un caso de urgencia o emergencia.

Actualmente, la solicitud de los servicios al CCPEM se realiza a través de una llamada telefónica al 104, que es el número de teléfono de la Sala de Coordinación de Emergencia Médica, la llamada es procesada por una tele-operadora experta, enfermera o licenciada en enfermería, preparada en casos de emergencias, la cual toma una serie de datos, conformando así una demanda por atender. La demanda es transferida al coordinador que se encuentra en la base de ambulancia, quien está encargado de asignar el tipo de ambulancia que requiere el caso.

La asignación de ambulancia procede de dos formas, puede asignarse una ambulancia que se encuentre en la base o puede ser una ambulancia que se encuentre en un recorrido cercano al caso solicitado.

La demanda recibe un seguimiento desde el centro de coordinación. El seguimiento lo realiza el coordinador; consiste en asignar una serie de claves según los estados por los que pase la ambulancia desde el inicio de atención al caso hasta el retorno del vehículo a la base. Al culminar el seguimiento la demanda se archiva.

Después de una investigación en el CCPEM, se han detectado un conjunto de dificultades:

- Demora en tomar, procesar y transferir datos. Esto es un problema sustancial pues estos factores podrían ocasionar pérdidas de vida humana que con un poco de rapidez se hubiera podido evitar.
- La gestión de la información se realiza de forma manual, siendo una vía desfavorable a la hora de emitir reportes de casos atendidos con anterioridad, pues la búsqueda de información se hace tediosa y no hay seguridad de un 100% de exactitud en la información.
- El volumen de información es vulnerable a pérdidas, estas pueden ser provocadas por factores naturales o directamente el factor humano.
- Quejas de la población por incumplimiento de recogidas a pacientes.

- Falta de control de móviles que viajan sin la debida autorización del centro o móviles que se desvían por una causa extra-laboral.

Se realizó un estudio de productos internacionales que gestionan un negocio similar al que se necesita informatizar la presente investigación. Estos sistemas son:

- GAMXXI: Fue desarrollado por Professional Soft en España.
- Ambuwin CS: Fue desarrollado por Informática y Comunicaciones Avanzadas de Málaga en España.
- Gestión TTS2000: Fue desarrollado por SEINCO de Salamanca en España.

Por todo lo anterior, se plantea como problema a resolver en la investigación: ¿Cómo facilitar la gestión de información de las demandas en los centros coordinadores provinciales de Emergencias médicas en Cuba? Donde el objeto de estudio es la gestión de la información en el CNUM y el campo de acción es el proceso de gestión de la información en los CCPEM del CNUM.

Como objetivo general del trabajo se propone: implementar una aplicación Web para facilitar la gestión de la información de las demandas en los centros coordinadores provinciales.

Como tareas de investigación se tiene:

- Valorar las nuevas tendencias de las tecnologías de la información relacionadas con las aplicaciones Web.
- Analizar los modelos de análisis y diseño propuestos por los analistas.
- Realizar un análisis crítico de la selección de la tecnología: metodología de desarrollo de software, el lenguaje de programación, el gestor de base de datos y las herramientas a utilizar.
- Valorar la integración con otros componentes ya existentes en el SISalud, así como la integración con otras partes del Sistema Informatizado de Urgencias Médicas.
- Obtener el Modelo de Implementación y los artefactos necesarios que describan la base de datos.
- Realizar la implementación del Módulo Centro Coordinador Provincial utilizando los patrones de diseño establecidos en el Análisis y Diseño, basándose en la propuesta de arquitectura definida por el MINSAP (Orientada a Servicios y Basada en Componentes) así como la implementación de la capa de acceso a datos y procedimientos almacenados.
- Realizar las pruebas que validen el correcto funcionamiento de la aplicación.

Como resultado de la investigación se ha obtenido el diseño de la Base de Datos y la implementación del Módulo CCPEM del Sistema Informatizado para el Centro Nacional de Urgencias Médicas.

El presente trabajo está estructurado en tres capítulos, en ellos se desarrolla el análisis de los procesos que realizan los Centros Coordinadores Provinciales de Emergencia Médica y proporciona

una vía de solución a sus problemas de gestión de información:

- El primer capítulo hace referencia a las herramientas, librerías, arquitectura, frameworks, gestor de base de datos, servidor de aplicaciones, lenguajes de programación y metodología de desarrollo que son usadas para la realización del software. También se hace un estudio del arte de los diferentes productos informáticos existentes en el mundo, que aborden el tema de atención a urgencias y emergencias médicas, y se definen algunos conceptos importantes.
- El segundo capítulo argumenta los requisitos no funcionales, los patrones o estilos arquitectónicos usados. Hace una valoración del diseño propuesto por los analistas. Muestra una vista del despliegue del software así como una vista de la implementación.
- El tercer capítulo hace un análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser reutilizados. Describe las clases más importantes del sistema, muestra una vista de la base de datos así como una descripción de sus tablas.

CAPITULO I: FUNDAMENTACIÓN TEÓRICA

Las tecnologías de la información son fundamentales para la superación y desarrollo de un país. Por eso, los países desarrollados basan su crecimiento en la aplicación y la programación estratégica de las herramientas computacionales, definiendo políticas que permitirán su permanencia en el dinamismo mundial de los próximos años. En el nuevo entorno económico-mundial, los países emergentes están obligados a preparar profesionales en áreas de la informática y las telecomunicaciones, capaces de enfrentar los retos que se tienen hoy en día. La presencia de la computación en los sectores productivos es un factor determinante para su funcionamiento.

Debido a esto, la Universidad de las Ciencias Informáticas ha desarrollado un plan estratégico de informatización de los principales sectores, tanto económicos como sociales del país, para de esta manera, mejorar la calidad de vida de la población, así como los servicios que se brindan.
En este capítulo se hace referencia a las herramientas, librerías, arquitectura, frameworks, gestor de base de datos, servidor de aplicaciones, lenguajes de programación y metodología de desarrollo que son usadas para la realización del software. Se realiza un estudio del arte de los diferentes productos informáticos existentes en el mundo, que aborden el tema de atención a urgencias y emergencias médicas, y se definen algunos conceptos importantes.

1.1 Conceptos básicos asociados al problema.

Emergencias

Son urgencias con afectaciones vitales o peligro vital potencial, que generan un peligro vital inmediato. Son la primera prioridad de urgencias, pues requieren de atención inmediata. Estos pacientes requieren de ambulancias intensivas o de emergencias. [5]

Urgencias

- Urgencia verdadera con requerimientos hospitalarios: paciente con enfermedad o lesiones que de no atenderse en un tiempo mediato puede generar complicaciones fatales. Constituye la segunda prioridad de urgencias. Estos se trasladan en ambulancias de urgencias.

- Urgencia banal o sin riesgos: pacientes que requieren ser vistos de urgencia pero sin peligro vital, mediato o tardío. Constituyen la tercera prioridad de urgencias. No deben trasladarse en ambulancias a no ser casos especiales Ej. Impedidos físicos.
- Urgencia social: no constituye una urgencia aunque es una urgencia sentida, es un problema para el paciente y sus familiares. [6]

Paciente grave

Un paciente se puede calificar de grave, cuando existe una amenaza importante para su vida o un alto riesgo de que ocurra la muerte. [7]

Código o tarjeta negra

Personas muertas o que se encuentra en Paro Cardio-Respiratorio. [8]

Gestión sin efecto

Término que identifica a un móvil que tiene establecida una demanda, pero antes de llegar al lugar se le ordena que no la realice, por lo que se dice que queda sin efecto. [9]

Gestión fallida

Término que identifica a un móvil cuando llega al lugar de la demanda a recoger el paciente y por alguna razón el paciente no se encuentra, ya sea porque se marchó en otro vehículo o por que no se encontró la dirección de la demanda. [10]

1.2 Sistemas internacionales vinculados a la atención de emergencias.

1.2.1 GAMXXI

Desarrollado por Professional Soft, España. El sistema cuenta con un conjunto de módulos entre ellos Módulo Coordinador, GPS, Turnos. [11]

Funcionalidades generales del sistema: [12]

- Control de datos de pacientes, servicios, programaciones, mutuas, aseguradoras, hospitales, tratamientos, ambulancias, diagnósticos, tipos de traslados, personal, motivos de anulación.
- Sistema de ayuda para la gestión rápida en la aplicación, como direcciones predeterminadas, poblaciones, kilómetros automáticos según origen-destino.
- Tratamiento diferenciado por tipos de servicio: Normales (Ida y/o Vuelta), Urgentes (Vía pública), Siniestros (Accidentes empresa), Colectivos (Ruta) y Programados (Se repiten periódicamente).
- Accesos restringidos según permisos del usuario.
- Botones de acceso rápido según usuario.
- Más de 50 listados, todos exportables a Excel, Notepad, e-mail.
- Agenda por teléfonos de interés y personal de la empresa.
- Para empresas de un gran volumen de datos, existe la opción de SQL Server de Microsoft.

Principales módulos:

Módulo coordinador: Módulo de adjudicación y finalización de servicios a través de la computadora. [13]

Principales Funcionalidades: [14]

- Base de datos de pacientes.
- Historial de servicios a pacientes.
- Compañías de seguros con sus tarifas.
- Centros de atención primaria.
- Rápida recepción de servicios.
- Servicios colectivos, programados, rutas, siniestros y urgencias.
- Generación automática de servicios.
- Asignación automática de ambulancia para servicios programados y colectivos.
- Asignación automática de tarifas para servicios de mutuas.
- Centro de adjudicación y finalización de servicios.
- Información actualizada de la localización y disponibilidad de ambulancias.
- Central de recepción de llamadas para servicios.
- Ayudas a introducción rápida servicios: direcciones predeterminadas, destinos, km.
- Mantenimiento de diagnósticos, tratamientos, tipos ambulancias, traslados, procedencias, destinos, tipos siniestros y urgencias, poblaciones habituales, conductor, ayudante, tarifas mutuas, direcciones pacientes, etc.
- Generación automática diaria, de servicios programados o colectivos.

Módulo Turnos: Generación automática del calendario anual según convenio.

Principales funcionalidades: [15]

- Programación de las necesidades del año.
- Trabajadores y vehículos con sus turnos.
- Asignación automática de vehículos al calendario.
- Asignación automática de la dotación del vehículo.
- Explotación del calendario.

Módulo GPS: Localización de vehículos por satélite.

Principales funcionalidades: [16]

- Visualización a central de un vehículo en tiempo real dentro de un mapa.
- Seguimiento en diferido de vehículos dentro de un mapa.
- Posibilidad de instalar en el vehículo un navegador donde se visualice la ruta a seguir.

1.2.2 Ambuwin CS

Desarrollado por Informática y Comunicaciones Avanzadas de Málaga, España. Es una aplicación basada en cliente/servidor, que permite la gestión completa de las empresas dedicadas al sector del transporte Sanitario. [17]

Sus principales características son: fácil manejo, manejo de servicios en tiempo real, sistema de alertas horarias, sistema de alertas de servicios urgentes, estatus de vehículos, control de pacientes, control de colegiados, control de vehículos (servicios, reparaciones, seguros, equipamiento), facturación, presupuestos, gestores de informes, sistema de control de accesos a la aplicación cumpliendo con lo establecido en la Ley de Protección de Datos. [18]

Principales funcionalidades: [19]

- Servicios: control exhaustivo de servicios a realizar en tiempo real, tipos de servicios, orígenes y destinos, estatus horarios, generación automático de sesiones a realizar (programados), colegiados que realiza el servicio, tipo de ambulancia que lo realiza, bloqueos y desbloqueos de servicios, anulación, pendientes.
- Urgencias: sistema de alertas para servicios urgentes.
- Vehículos: control de gastos, consumos, seguros, extintores, certificados sanitarios, datos de compra.

- Conductores: datos generales, datos académicos, control de cursos realizados, tipos de carnets.
- Ayudantes: igual que conductores.
- Aseguradoras y entidades: Datos generales, tipos de facturación, generador de listas de precios personalizable.
- Otras bases de datos: Tipos de servicios, servicios médicos, tipos de vehículos.
- Otras prestaciones: Sistema de alerta horaria para servicios personalizable, rangos provinciales de Km.
- Facturación y control de almacén, compras, control de cobros pagos, seguimiento de clientes.

1.2.3 Gestión TTS2000

Desarrollado por SEINCO de Salamanca, España. Es un programa específico destinado al sector del Transporte Sanitario para la realización de una gestión eficaz de todas las tareas que requiere. Es el más potente existente en el mercado, siendo un referente en el transporte sanitario y es utilizado actualmente por más de 50 empresas a nivel nacional. [20]

Permite una organización automática del transporte sanitario, optimizando los recursos de los que se dispone, ahorrando tanto medios técnicos como económicos. Además este programa permite la generación de facturación y estadísticas, puede ser enlazado con el programa de contabilidad de la empresa en cuestión. Este programa ofrece la posibilidad de llevar un control de flota exhaustivo, teniendo controlado en todo momento los vehículos. [21]

Ofrece la posibilidad de Coordinar tanto el trabajo con la Administración como el Privado, realizar Facturación, Estadísticas, Estructuración automática de Servicios, Control de Personal, Control de Flota. [22]

El estudio realizado ha contribuido a la experiencia del equipo de desarrollo. Estos sistemas son propietarios, lo que impide la modificación del código para la adaptación al sistema de salud cubano. Tienen un alto costo en el mercado, son aplicaciones de escritorio y fueron desarrollados con características de los sistemas de salud de otros países, donde la medicina es una empresa más. Por lo que se ha decidido crear una aplicación web que resuelva los problemas que están presentando, los Centros Coordinadores Provinciales de Emergencia Médica en Cuba.

1.3 Tendencias actuales de la tecnología en el desarrollo de software.

Al analizar todos lo problemas existentes en los CCPEM y estudiar como debiese fluir la información en estos; se comienza un estudio de las tendencias actuales del desarrollo de software para definir las herramientas, tecnologías y metodologías utilizar.

Después de consultar una amplia información sobre el tema, se define la realización de una aplicación Web. A continuación se muestra un conjunto de ventajas de este tipo de aplicaciones:

- Una empresa puede migrar de sistema operativo o cambiar el Hardware libremente sin afectar el funcionamiento de las aplicaciones de servidor.
- No se requieren complicadas combinaciones de Hardware/Software para utilizar estas aplicaciones. Solo un computador con un buen navegador web.
- Actualizar o hacer cambios en el Software es sencillo y sin riesgos de incompatibilidades. Existe solo una versión en el servidor lo que implica que no hay que distribuirla entre los demás computadores. El proceso es rápido y limpio.
- Al funcionar en un navegador, se requiere un conocimiento básico de informática para utilizar una aplicación web.

1.3.1 Metodología de Desarrollo

El Proceso Unificado de Racional (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.

Sus principales características se centran en: implementar las mejores prácticas en Ingeniería de Software, disciplinar la forma de asignar tareas y responsabilidades, administrar requisitos, usar arquitectura basada en componentes y controlar cambios y modelado visual del software.

RUP posee tres características fundamentales: su desarrollo es iterativo e incremental, por lo que divide el proceso de desarrollo en ciclos, teniendo un producto final al terminar cada ciclo. La segunda, es que está guiado por casos de uso. Un caso de uso será aquello que describe un fragmento de las funcionalidades del sistema que proporciona al usuario un resultado importante. Los casos de uso guían el diseño, la construcción y las pruebas del sistema, esto significa que guían el proceso de desarrollo. La tercera es, que está centrada en la arquitectura, lo que permite a los desarrolladores una

mayor visibilidad del sistema, pues la arquitectura es una vista del diseño completo del software, con las características más importantes resaltadas, dejando a un lado los detalles.

1.3.2 Lenguajes de Programación

1.3.2.1 PHP

Es un lenguaje de script interpretado en el lado del servidor, utilizado para la generación de páginas Web dinámicas, embebido en páginas HTML y ejecutado en el servidor. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl, con algunas características específicas de si mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. No es un lenguaje de marcas como es HTML, XML o WML. [23]

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador. [24]

Al ser un lenguaje libre dispone de características que lo convierten en la herramienta ideal para la creación de páginas web dinámicas: [25]

- Soporte para una gran cantidad de bases de datos: MySQL, PostgreSQL, Oracle, MSSQL Server, Sybase mSQL, Informix.
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- Ofrece una solución simple y universal para las paginaciones dinámicas de la web de fácil programación.
- Más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.
- Goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente.
- El código se pone al día continuamente con mejoras y extensiones del lenguaje, con el objetivo de ampliar las capacidades de PHP.
- Se puede hacer cualquier cosa que podemos realizar con un script CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas.

En cuanto a seguridad es un potente lenguaje y el intérprete, tanto incluido en el servidor Web, como módulo o ejecutado como un binario CGI, puede acceder a ficheros, ejecutar comandos y abrir comunicaciones de red en el servidor. Todas estas características hacen que lo que se ejecute en el servidor Web sea seguro por defecto. [26]

PHP ha sido diseñado específicamente para ser un lenguaje más seguro para escribir programas CGI, Perl o C y con la correcta selección de las opciones de configuración de tiempo de compilación y ejecución se consigue la exacta combinación de libertad y seguridad que se necesita. Ya que existen diferentes modos de utilizar PHP, existe también una multitud de opciones de configuración que permiten controlar su funcionamiento. [27]

Una gran selección de opciones garantiza que se pueda usar PHP para diferentes aplicaciones, pero también significa que existen combinaciones de estas opciones y configuraciones del servidor que producen instalaciones inseguras. [28]

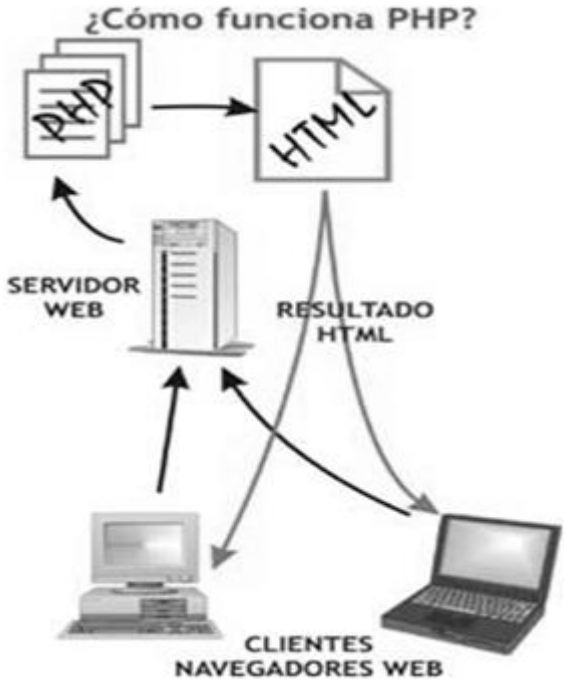


Figura 2: Cómo funciona PHP.

1.3.2.2 JavaScript

Es un lenguaje de scripts del lado del cliente desarrollado por Netscape para incrementar las funcionalidades del lenguaje HTML. Fundamentalmente se utiliza para realizar validaciones de datos del lado del cliente. Es un lenguaje interpretado pues no requiere compilación. El navegador del

usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente. [29]

Es un lenguaje orientado a eventos. Cuando un usuario pincha sobre un enlace o mueve el puntero sobre una imagen se produce un evento. Mediante este se pueden desarrollar scripts que ejecuten acciones en respuesta a estos eventos. [30]

Es un lenguaje orientado a objetos. El modelo de objetos de JavaScript está reducido y simplificado, pero incluye los elementos necesarios para que los scripts puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador. [31]

1.3.3 Grupo de tecnologías para desarrollo web.

1.3.3.1 Ajax

Acrónimo de Asynchronous JavaScript And XML (JavaScript y XML asíncronos), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente (en el navegador del usuario) y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad. [32]

Un grupo de tecnologías conforman a Ajax como XHTML, HTML, hojas de estilos en cascada (CSS) para el diseño que acompaña a la información, Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada. [33]

El objeto XMLHttpRequest para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios. [34]

XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML pre formateado, texto plano, JSON y hasta EBML. [35]

No constituye una tecnología en sí, sino que es un término que engloba a un grupo de tecnologías que trabajan conjuntamente. [36]

Estas aplicaciones se ejecutan en la máquina cliente, manipulando las páginas actuales dentro de sus navegadores, para ello usa métodos de Document Object Model. [37]

Es usado para multitud de tareas como actualizar o eliminar registros, expandir formularios web, devolver peticiones simples de búsqueda, o editar árboles de categorías; todo sin tener la necesidad de tener que recargar toda la página de HTML cada vez que se realiza un cambio. Generalmente sólo requiere enviar pequeñas peticiones al servidor, y se devuelven respuestas relativamente cortas. [38]

Se utilizan características bien documentadas presentes en todos los navegadores importantes existentes. Mientras que la plataforma está más restringida que la plataforma de Java. Las aplicaciones actuales de Ajax llenan con eficacia la parte de los Java applets: ampliar el navegador con mini-aplicaciones ligeras. [39]

Los navegadores que permiten Ajax son Microsoft Internet Explorer para Windows versión 5.0 y superiores. Navegadores basados en Gecko como Mozilla, Mozilla Firefox, SeaMonkey, Camino, Flock, Epiphany, Galeon y Netscape versión 7.1 y superiores. Navegadores con el API KHTML versión 3.2 y superiores implementado, incluyendo Konqueror versión 3.2 y superiores, y el Web Browser para S60 de Nokia tercera generación y posteriores. Opera versión 8.0 y superiores, incluyendo Opera Mobile Browser versión 8.0 y superiores. [40]

Los navegadores que no permiten Ajax son Opera 7 y anteriores. Microsoft Internet Explorer para Windows versión 4.0 y anteriores. Microsoft Internet Explorer para Macintosh (todas las versiones). Navegadores basados en texto como Lynx y Links. Navegadores para incapacitados visuales (braille). [41]

1.3.4 Servidor de Aplicaciones

1.3.4.1 Apache

El nombre de Apache viene de "A PAtCHy server", un servidor lleno de remiendos. Está basado en alguna codificación existente y en una serie de archivos "parche". Apache es el programa servidor HTTP. Con este se puede crear y publicar documentos php de la misma forma que se hace en Internet, con una estabilidad y eficacia ampliamente comprobada en la gran cantidad de servidores apache actualmente en uso. [42]

Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos (HTTP 1.1). Entre sus características destacan:

- Multiplataforma
- Es un servidor de web conforme al protocolo HTTP/1.1
- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con el API de programación de módulos, para el desarrollo de módulos específicos.

- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- Se desarrolla de forma abierta
- Extensible: gracias a que es modular, se ha desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor. [43]

1.3.5 Gestor de Base de datos

1.3.5.1 MySQL

Es el sistema de gestión de bases de datos SQL Open Source más popular, lo desarrolla, distribuye y soporta MySQL AB. MySQL AB es una compañía comercial, fundada por los desarrolladores de MySQL. Es una compañía Open Source de segunda generación que une los valores y metodología Open Source con un exitoso modelo de negocio. [44]

El servidor de base de datos MySQL es rápido, fiable y fácil de usar**.** Proporciona sistemas de almacenamiento transaccional y no transaccional. Permite un uso completo de multi-threaded mediante threads del kernel. Pueden usarse fácilmente múltiple CPUs si están disponibles. [45]

El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible. [46]

Contiene un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de contraseñas está encriptado cuando se conecta con un servidor. [47]

1.3.6 Frameworks

1.3.6.1 CodeIgniter

Es un framework de PHP basado en la arquitectura Modelo-Vista-Controlador (MVC), diseñado fundamentalmente para desarrolladores que necesiten crear aplicaciones web en poco tiempo y con un

alto rendimiento. Es ligero y flexible. Puede ser tan sólo VC (Vista-Controlador) pues no fuerza al usuario a utilizar una base de datos para un desarrollo. [48]

La configuración básica del CodeIgniter es:

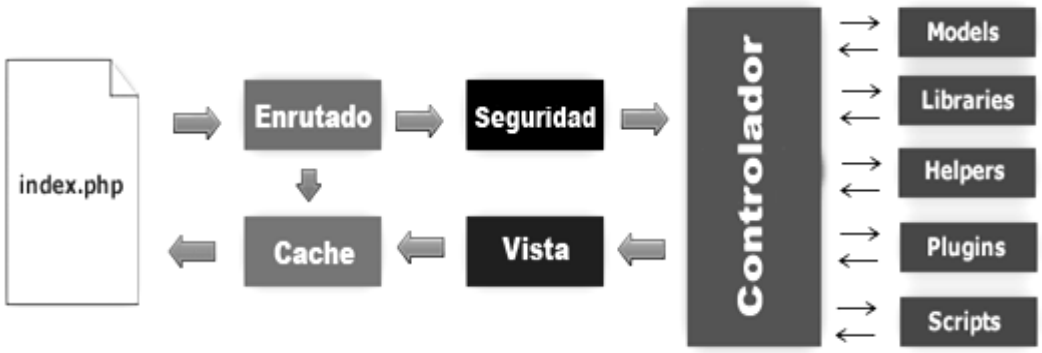


Figura 3: Diagrama de flujo de CodeIgniter.

El index.php inicializa el núcleo de CodeIgniter. El enrutado examina la petición HTTP y determina que se debe hacer. Si existe la cache devuelve el archivo HTML sin pasar por el sistema, ahorrando la carga que ello conlleva. La seguridad antes de que se cargue el controlador se filtra los datos enviados para que estos sean fiables. El controlador carga la modelo, las librerías, los plugins, los helpers y todos los recursos necesarios para satisfacer la petición. Una vez la vista está construida es enviada al navegador**,** si la cache está habilitada se almacena el resultado en cache para la próxima vez que el URL sea servida. [49]

1.3.6.2 Prototype

Es un framework de Ajax muy sencillo de entender, y bastante popular. Para usarlo sólo es necesario incluir el archivo prototype.js y con esto se puede utilizar los objetos que incluye la librería. Simplifica y reduce la cantidad de código JavaScript. Posee una estandarización del modelo de codificación. Se integra con el modelo de programación de la tecnología del lado de servidor. [50]

Es un framework del lado del cliente. No sólo incluye funcionalidad para Ajax, sino que también incluye funciones para manejo y optimización de arreglos, cadenas u objetos. [51]

1.3.7 Librerías

1.3.7.1 FPDF

Es una clase escrita en PHP, permite generar documentos PDF directamente desde PHP sin usar la biblioteca PDFlib. La ventaja es que, mientras PDFlib es de pago para usos comerciales, la F de FPDF significa Free (gratis y libre), puede ser usada para cualquier propósito y modificarla a gusto del programador para satisfacer sus necesidades. [52]

Posibilita la elección de la unidad de medida, formato de página y márgenes. Gestiona las cabeceras y pies de página. Posibilita salto de página automático, salto de línea y justificación del texto automáticos. Admite imágenes (JPEG y PNG), fuentes TrueType, Type1 y codificación. Permite la compresión de página FPDF, no necesita de ninguna extensión para PHP (excepto la biblioteca zlib si se va a activar la opción de compresión) y funciona con PHP4 y PHP5. [53]

1.3.8 Herramientas

1.3.8.1 Zend Studio

Potente IDE de desarrollo para PHP concebido con el fin de crear aplicaciones altamente fiables, posee una integración del uso y completado de código personalizado de Zend Framework y vista de la lista de las funciones del framework desde la Visualización de Funciones PHP. Aumenta la productividad con: Soporte PHP 5 completo, Analizador de Código, carpeta de Código, completado de Código, coloreado de Sintaxis, Administrador de Proyecto, Editor de Código, Depurador de gráficos y asistentes.

La documentación del código ocurre de forma más sencilla, aplicaciones, y proyectos con PHPDocumentor, la herramienta de documentación standard para PHP. Permite simplificar el despliegue con la integración FTP y SFTP de forma tal que posibilite a los programadores en forma segura subir y descargar archivos de proyectos de modo transparente hacia y desde servidores remotos.

Permite conectarse directamente con la bases de datos profesionales más utilizadas tales como IBM DB2/Cloudscape/ Derby/, MySQL, Oracle, Microsoft SQL Server, PostgreSQL y SQlite. Puede escribir y realizar consultas a servidores conectados usando el editor de consultas SQL de Zend con SQL92 y soporte de coloreado de Sintaxis. Permite visualizar las estructuras de la base de datos y administrar el contenido con el explorador SQL de Zend.

1.3.8.2 Case Studio 2

Herramienta profesional que se utiliza para diseñar bases de datos. Permite crear diagramas de relación, modelado de datos y gestión de estructuras. Soporta Oracle, SQL, MySQL, MS SQL, MaxDB, Firebird, PostgreSQL, Access. Permite la generación rápida de diagramas gráficos de bases de datos relacionales.

Su potencia se basa en la ingeniería inversa, que permite identificar y estructurar bases de datos ya existentes para poder trabajar con ellas sin problemas. Permite generar scripts SQL. Crea detallados informes en HTML y RTF. Para utilizar Case Studio se requiere de los sistemas operativos: Win95/98/98SE/Me/2000/NT/XP.

1.3.8.3 Visual Paradigm for UML 3.0(VP-UML)

Herramienta multiplataforma de modelado visual UML y una herramienta CASE muy potente y fácil de utilizar. VP-UML aporta a los desarrolladores de software una plataforma de desarrollo puntera para construir aplicaciones de calidad mejores y más baratas con rapidez. Aporta una excelente interoperabilidad con otras herramientas CASE y muchos de los entornos IDE líderes del mercado.

Contiene un entorno de creación de diagramas para UML 2.0. El diseño se centra en casos de uso y está enfocado al negocio que generan un software de mayor calidad. Proporciona un lenguaje estándar a todo el equipo de desarrollo, facilitando la comunicación entre ellos.

Posee capacidades de ingeniería directa (versión profesional) e inversa. Hay una sincronización durante todo el ciclo de desarrollo del modelo y el código. Dispone de múltiples versiones para cada necesidad. Posee una disponibilidad de integrarse a los principales IDEs. Desarrolla una disponibilidad en múltiples plataformas.

Este capítulo abordó un conjunto de conceptos necesarios para la comprensión del negocio. Se investigó acerca de productos actualmente desplegados con negocio similar al Centro Coordinador de Provincial de Emergencia Médica. Se determinó que ninguno cumplía las expectativas que se exigían dadas sus características, por ello se llegó a la conclusión que debía realizarse un software capaz de responder a las necesidades actuales de este órgano.

A través del estudio de las tendencias actuales de la tecnología en el desarrollo de software, se concretó que se haría una aplicación web utilizando el lenguaje php, como servidor de aplicaciones Apache, gestor de base de datos MySQL. Las herramientas especificadas para el desarrollo del software son Zend Studio v 5.5.0 para la implementación, Case Studio v 2 para modelar la base de datos y Visual Paradigm for UML v 3.0 para la realización de la ingeniería de software del producto.

CAPITULO II: ELEMENTOS DE LA ARQUITECTURA.

El presente capítulo especifica los requisitos no funcionales del sistema realizado. Se caracteriza los estilos arquitectónicos presentes en el diseño. Se hace una valoración del diseño propuesto por los analistas. Se ofrece una vista de cómo sería el despliegue de la aplicación así como una vista de la implementación realizada.

2.1 Argumentación de los requisitos no funcionales del sistema propuesto.

Los requerimientos no funcionales, son propiedades o cualidades que hacen al producto atractivo, usable, rápido y confiable. Estos influyen en el éxito del producto. Su importancia recae en que tanto clientes como usuarios puedan valorar las características no funcionales del producto

Aunque los requisitos no funcionales no formen parte de la razón fundamental del producto, son extremadamente necesarios para hacer funcionar el sistema. No modifica funcionalidades del producto, pero si las añade. Describen de manera global las experiencias del usuario cuando trabaja con el producto.

A continuación se muestra los requisitos no funcionales del software realizado:

Nº: RNF 1	
Nombre	Apariencia o Interfaz Externa.
Descripción	La interfaz del sistema no contiene numerosas imágenes para evitar demoras en la respuesta de cualquier acción del usuario. La misma será sencilla, amigable e intuitiva, de fácil navegación por parte del usuario. Estará diseñada para una óptima visualización siendo adaptable a cualquier resolución.

Tabla 2.1: RNF1 de apariencia o interfaz externa.

Nº: RNF 2	
Nombre	Apariencia o Interfaz Interna.
Descripción	Los componentes del sistema serán desarrollados siguiendo el principio de alta cohesión y bajo acoplamiento, y los que sean

	reutilizables en los diferentes módulos del sistema serán desarrollados como Servicios Web XML que interactuarán con otros componentes a través de SOAP.

Tabla 2.2: RNF2 de apariencia o interfaz interna.

Nº: RNF 3	
Nombre	Usabilidad
Descripción	La Aplicación Web facilita la interacción usuario–sistema con el objetivo de evitar rechazo en el uso de la misma. Mediante mensajes guía al usuario en las diferentes acciones que realice. El usuario deberá poseer conocimientos básicos de computación y estar familiarizado con el negocio del Centro Coordinador Provincial de Emergencia Médica.

Tabla 2.3: RNF3 de usabilidad.

Nº: RNF 4	
Nombre	Portabilidad
Descripción	Permite que el sistema se ejecute sobre el sistema operativo Windows XP y Linux.

Tabla 2.4: RNF4 de portabilidad.

Nº: RNF 5	
Nombre	Soporte
Descripción	La Aplicación Web contará con una ayuda, donde el usuario podrá suplir las dudas que se le puedan presentar durante su utilización. El usuario del módulo deberá recibir un adiestramiento previo en la utilización del sistema, con el fin de que pueda explotar las prestaciones del sistema sin contratiempos ocasionados por la falta de preparación técnica.

Tabla 2.6: RNF6 de soporte.

Nº: RNF 6	
Nombre	Seguridad
	6.1
Nombre	Confidencialidad
Descripción	La información que brinda el sistema estará protegida contra el acceso de usuarios no autorizados. Solamente los administradores del sistema podrán realizar cambios en la configuración y en la información.
	6.2
Nombre	Disponibilidad
Descripción	El sistema estará accesible cada vez que los usuarios del mismo lo requieran.

Tabla 2.7: RNF7 de seguridad.

Nº: RNF 7	
Nombre	Software
	7.1
Nombre	Software en el Servidor
Descripción	El servidor debe contar con sistema operativo Linux/Debian 4 Etch, con el servidor web appserver.
	7.2
Nombre	Software en el Cliente
Descripción	Para utilizar la Aplicación Web será necesaria una computadora con el Sistema Operativo Windows o Linux en cualquiera de sus versiones, recomendándose Windows XP o superior y Ubuntu 7.10 o Superior. Además se podrá acceder desde cualquier Navegador Web recomendándose Internet Explorer 6 o superior y

	Firefox 2.0 o superior.

Tabla 2.8: RNF8 de software.

Nº: RNF 8	
Nombre	Hardware
Descripción	Para garantizar el correcto funcionamiento de la aplicación, se necesita una computadora con un procesador Pentium II o superior, una memoria RAM de 512 MB o más, un Disco Duro de 10 GB o más y una tarjeta de red a 128Mbps o más.

Tabla 2.9: RNF9 de hardware.

Nº: RNF 9	
Nombre	Restricciones en el Diseño y la Implementación.
Descripción	• Se utilizarán los patrones de diseño establecidos. • La lógica de presentación constituirá una capa independiente de la lógica de negocio, centrando su función en la interfaz de usuario y validaciones simples de la entrada de datos. • Para el análisis y el diseño del sistema deberá ser utilizada la metodología RUP, usando el lenguaje de modelado UML y como herramienta para llevarlo a cabo el Visual Paradigm for UML v 3.0. • Para la implementación se utilizará como lenguaje de programación PHP 5 y como herramienta de desarrollo el Zend Studio v 5.5.0.

Tabla 2.10: RNF10 Restricciones en el diseño y la implementación.

2.2 Patrones o estilos arquitectónicos presentes en la propuesta de solución.

2.2.1 Arquitectura.

La arquitectura de software es la organización fundamental de un sistema formado por sus componentes, las relaciones entre ellos, el contexto en el que se implantarán y los principios que orientan su diseño y evolución. Tiene la responsabilidad de:

- Definir los módulos principales.
- Definir las responsabilidades que tendrá cada uno de estos módulos.
- Definir la interacción que existirá entre dichos módulos.
- Realizar el control y flujo de datos.
- Realizar secuenciación de la información.
- Definir protocolos de interacción y comunicación.
- Definir ubicación en el hardware.

El objetivo principal de la arquitectura de software, es aportar elementos que ayuden a la toma de decisiones, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto.

2.2.1.1 Arquitectura orientada a servicios.

La arquitectura SOA (Service Oriented Architecture), propone un modelo en el que el código de la función es independiente de la forma en que se resuelve la integración. Puede estar hecha en cualquier lenguaje de programación y residir en cualquier tipo de plataforma tecnológica, conservándose de esta manera los activos actuales de la empresa en sus sistemas de información. [54]

Desde el punto de vista de las aplicaciones externas, la función es una caja negra que recibe unos parámetros de llamada o solicitud de información, y responde de una manera que es reconocible según unos estándares. La integración se resuelve mediante una buena definición de los parámetros de llamada a la función y una buena definición de la naturaleza de la respuesta. Esta definición se hace mediante los estándares que engloban los Web Service los cuales son XML, SOAP, WSDL, UDDI. [55]

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

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red, como servicios independientes a los que tienen acceso de un modo estandarizado. La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP y WSDL) en su implementación.

SOA define las siguientes **capas de software**:

- **Aplicaciones básicas**: sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad.
- **Exposición de funcionalidades**: las funcionalidades de la capa aplicativas son expuestas en forma de servicios (webservices).
- **Integración de servicios**: facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración.
- **Composición de procesos**: define el proceso en términos del negocio y sus necesidades, y que varía en función del negocio;
- **De entrega**: los servicios son desplegados a los usuarios finales.

Ventajas de la arquitectura SOA:

- Ahorro de dinero, tiempo y esfuerzo mediante la reutilización de componentes**.**
- Elimina frustraciones con Tecnología de Información (TI) gracias a las soluciones flexibles y los tiempos más cortos de implementación de soluciones.
- Permite justificar más claramente las inversiones en TI, ya que estas están más alineadas con el negocio.
- Proporciona a los ejecutivos del negocio una visión clara de lo que hace la TI y su valor asociado.
- Permite la creación y cambio de servicios de forma incremental, evitando proyectos de larga duración y alto costo.
- La orientación a Servicios permite enfocarse en la descripción del problema de negocio y no en el uso de una tecnología de ejecución.

- El real valor de SOA se aprecia en las etapas posteriores de desarrollo, en las cuales es posible crear nuevas aplicaciones, casi en su totalidad, a partir de la composición de servicios existentes, lo cual reduce costos y tiempo al mercado, incrementa eficiencia y agiliza las operaciones de la empresa.

2.2.1.2 Arquitectura basada en componentes.

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

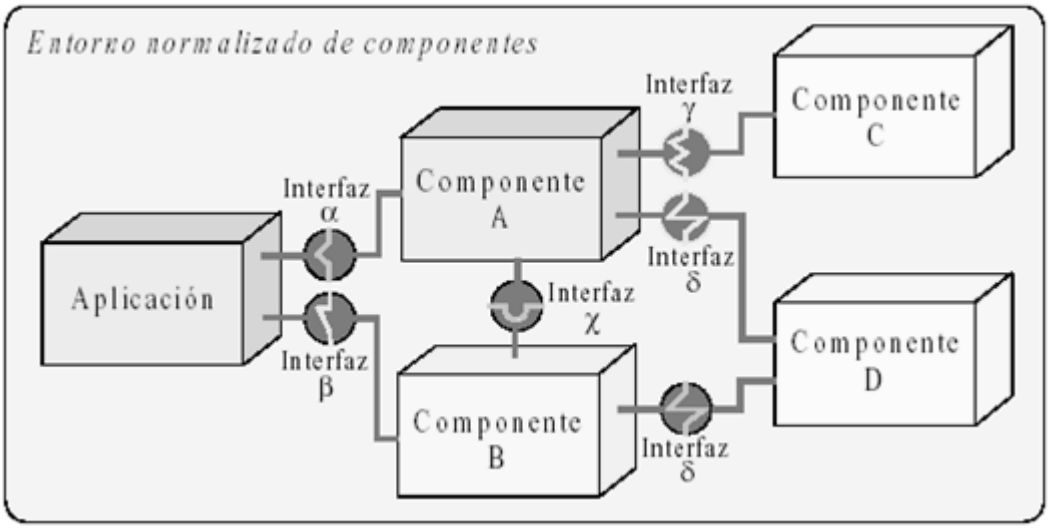


Figura 2.1: Arquitectura basada en componentes

Características de la arquitectura basada en componentes:

- Los componentes residen en diversas máquinas integrantes de una red.
- Se ejecutan en diferentes plataformas, sistemas operativos, escritos en diferentes lenguajes, diferentes desarrolladores.
- Modificables y ampliables añadiendo nuevas componentes.
- Sujetos a evolución por ampliación, desaparición, sustitución de componentes o reconfigurando las relaciones entre ellos.

2.2.1.2 Arquitectura en capas

Es la generalización de la arquitectura cliente-servidor. Es un estilo de programación cuyo objetivo es la separación de la lógica de negocios de la presentación y el acceso a datos.

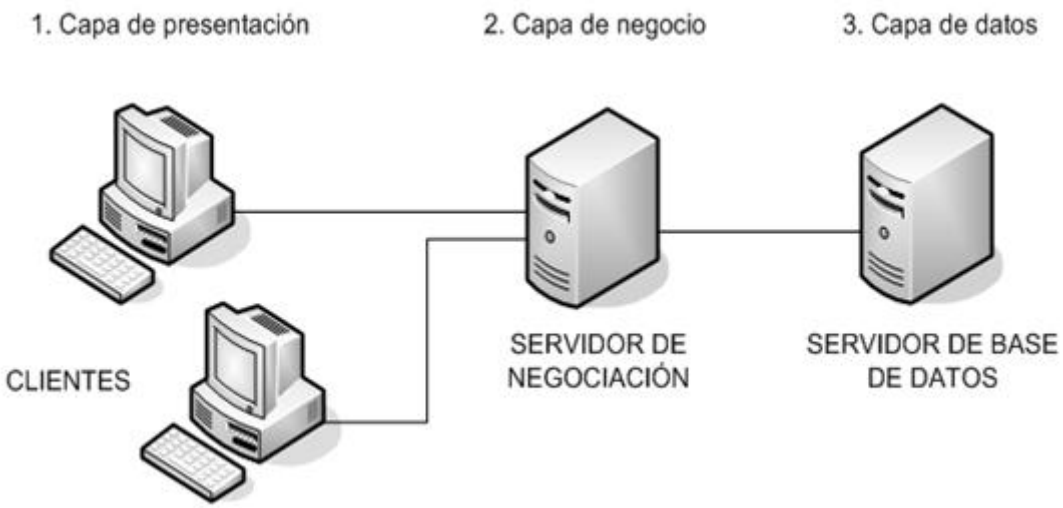


Figure 2.2: Arquitectura en 3 capas.

Características de la arquitectura en tres capas:

- La capa de presentación es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso. Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser amigable (entendible y fácil de usar) para el usuario.
- En la capa de negocio, es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio o lógica del negocio porque es aquí donde se establecen todas las reglas que deben cumplirse.

  Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

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

Las tres capas pueden residir en un único ordenador. Lo más usual es que haya una multitud de ordenadores en donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores.

Si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio. Si por el contrario, fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos.

En sistemas muy complejos, se llega a tener una serie de ordenadores sobre los cuales corre la capa de datos, y otra serie de ordenadores sobre los cuales corre la base de datos. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que se necesite algún cambio, sólo se transforma el nivel requerido.

2.2.2 Patrones de diseño.

Los patrones de diseño constituyen el punto de partida en la búsqueda de soluciones a problemas comunes en el desarrollo de software. Estos proponen soluciones a problemas concretos. Se basan en las experiencias acumuladas por lo que se usa con mucha frecuencia. La reutilización de código es una de las ventajas del mismo.

Los analistas del equipo de desarrollo de la presente investigación, ha hecho uso de algunos patrones de diseño, los cuales han evitado la reiteración de código, estandarizando la forma en que se realiza el diseño, lo cual ha permitido crear un código más resistente al cambio.

Se usaron 5 patrones fundamentales dentro de los patrones GRASP (Patrones generales de software para asignar responsabilidades)**:**

- El patrón experto.
- El creador.
- El controlador.

- Bajo acoplamiento.
- Alta cohesión.

Se hizo el diseño de la aplicación siguiendo el patrón de diseño Modelo-Vista-Controlador. A continuación se realiza una descripción de cada uno de estos patrones.

2.2.2.1 Patrón experto

Responde a la pregunta: ¿Quién asumirá la responsabilidad en el caso general? [56]

Al aplicar este patrón, surge la necesidad de asignar las responsabilidades a las clases que poseen la información necesaria. [57]

El uso de este patrón ha traído beneficios a la hora de implementar las funcionalidades del sistema: [58]

- Ha permitido conservar el encapsulamiento, pues los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.
- El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas, siendo más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

2.2.2.2 Patrón creador

Responde a la pregunta: ¿Quién debería ser responsable de crear una nueva instancia de alguna clase? [59]

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. [60]

El propósito fundamental de este patrón es asignar a una clase Z la responsabilidad de crear una instancia de otra clase W cumpliendo alguna de las condiciones siguientes: [61]

- Z contiene a W.
- Z agrega a W.
- Z tiene los datos de inicialización de W.

- Z registra a W.
- Z utiliza a W muy de cerca.

El uso de este patrón ha traído beneficios a la hora de implementar las funcionalidades del sistema: [62]

- Brinda soporte a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

2.2.2.3 Patrón controlador

Resuelve el problema de: ¿Quién debería encargarse de atender un evento del sistema? [63]

Un evento del sistema es un evento de alto nivel generado por un actor externo, es decir, es un evento de entrada externa. Se asocia a operaciones del sistema que son las que emiten respuestas a los eventos del sistema. [64]

Un Controlador es un objeto de interfaz no destinada al usuario, que se encarga de manejar un evento del sistema. Define además el método de su operación. [65]

El uso de este patrón ha traído beneficios a la hora de implementar las funcionalidades del sistema: [66]

- Facilita la centralización de actividades (validaciones, seguridad). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.
- Mayor potencial de los componentes reutilizables, garantizando que los procesos no sean manejados por la interfaz.

2.2.2.4 Patrón bajo acoplamiento:

Responde a la interrogante ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización? [67]

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Una clase con bajo o débil acoplamiento no depende de muchas otras. [68]

El uso de este patrón ha traído beneficios a la hora de implementar las funcionalidades del sistema:

- Los componentes no se afectan por cambios a otros componentes.
- Los componentes son fáciles de entender por separado.

- Los componentes poseen una fácil reutilización. [69]

2.2.2.5 Alta cohesión

Responde a la interrogante: ¿Cómo mantener la complejidad dentro de límites manejables? [70]

En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. El uso de este patrón trae asociado beneficios como:

- Mejora la claridad y la facilidad con que se entiende el diseño.
- Simplifica el mantenimiento y las mejoras en funcionalidades.
- Soporta una mayor capacidad de reutilización, pues una clase muy cohesiva puede destinarse a un propósito muy específico. [71]

2.2.2.6 Patrón Modelo-Vista-Controlador (MVC)

La arquitectura MVC (Modelo-Vista-Controlador) fue introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Su característica principal es que el Modelo, las Vistas y los Controladores se tratan cómo entidades separadas. [72]

- El modelo representa la estructura de datos con clases que contienen funciones, las cuales permiten insertar, eliminar y actualizar la información de la base de datos.
- La vista es la información que se presenta al usuario, una vista es a menudo una página web en sí misma, pero en CodeIgniter una vista también puede llegar a ser un fragmento de la página web (pie, cabecera, contenido principal). [73]
- El controlador sirve de intermediario entre los modelos y las vistas, y es el encargado de coordinar la petición y generar la página web. [74]

Es una arquitectura preparada para los cambios, que desacopla datos y lógica de negocio de la lógica de presentación, permitiendo la actualización y desarrollo independiente de cada uno de los citados componentes. [75]

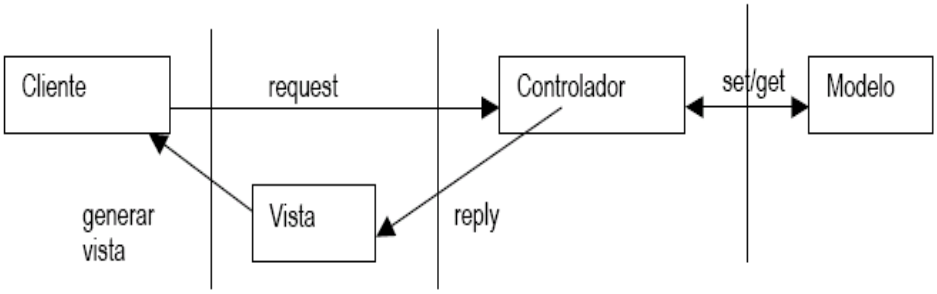


Figura 2.3: Patrón MVC.

2.3 Valoración crítica del diseño propuesto por el analista.

El diseño propuesto por los analistas significó un punto de partida para las actividades de implementación, pues favoreció la comunicación entre los miembros del equipo mediante el uso de un lenguaje gráfico. También en la comprensión de los requisitos funcionales. Los diagramas de clases aportaron una mejor visualización de las relaciones entre las clases que involucra el sistema.

Los diagramas de interacción mostraron en detalles los escenarios pertenecientes a cada caso de uso, permitiendo ver con mayor claridad el intercambio de mensajes entre los objetos. Estos describieron con precisión el comportamiento dinámico del sistema, permitiendo verificar la coherencia del mismo, validándolo con el modelo de clases.

2.4 Vista de despliegue.

El diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).

Cada nodo representa un recurso de cómputo, en el caso del Centro Coordinador Provincial de Emergencia Médica va a estar desplegado en dos capas físicas, una primera capa compuesta por la aplicación y una segunda capa compuesta por la base de datos.

Nodo Impresora

Representa las impresoras, pues se requiere del uso de las mismas con el objetivo de imprimir los reportes que se necesiten. No todas las computadoras clientes necesitan de una impresora, solo la que usa el jefe de guardia.

Nodo PC Cliente

Representa las computadoras que utilizaran los usuarios para interactuar con la aplicación. Establece comunicación con el servidor de aplicaciones a través del protocolo http.

Nodo Servidor de Aplicaciones

Representa el servidor donde estará la aplicación, constituye la primera capa física. Este nodo mantendrá comunicación con otros nodos a través de diferentes protocolos:

- Con el nodo Servidor de Base de Datos a través del protocolo ADO.
- Con el nodo Servidor SISalud a través del protocolo SOAP.
- Con el nodo Servidor de Operaciones a través de SOAP.

Nodo Servidor de Base de Datos

Representa al servidor donde estará la base de datos de la aplicación, constituye la segunda capa física.

Nodo Servidor SISalud

Representa el servidor que brindará servicios de algunos componentes que serán consumidos por la aplicación.

Nodo Servidor de Operaciones

Representa el servidor que brindará el servicio de la disponibilidad de ambulancias a la aplicación.

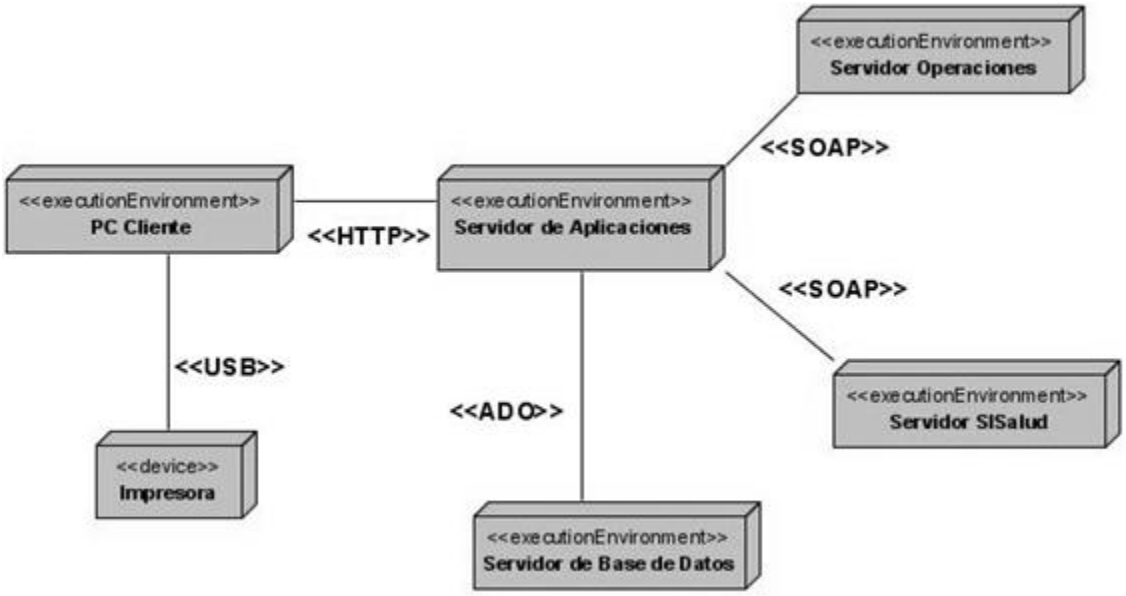


Figura 2.4: Vista de despliegue de la aplicación.

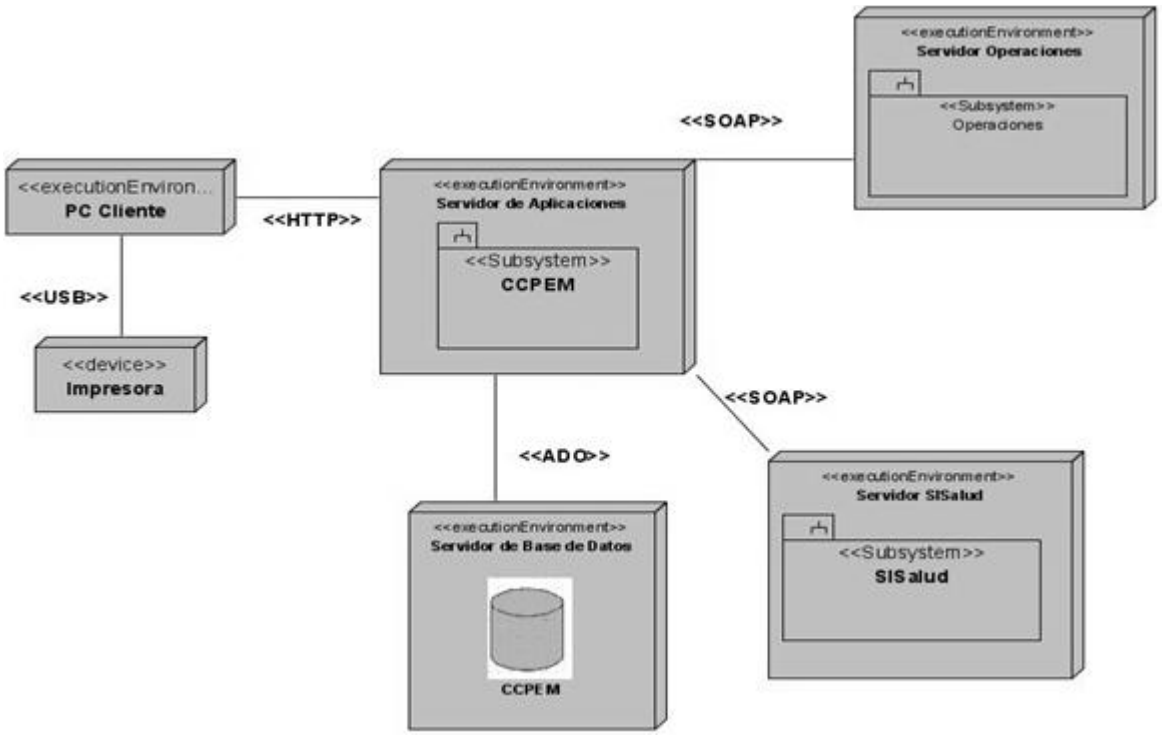


Figura 2.5: Visita de despliegue refinada.

2.5 Vista de implementación.

Lo que distingue a un diagrama de componentes de otros tipos de diagramas es su contenido. Normalmente contienen componentes, interfaces y relaciones entre ellos. Y como todos los diagramas, también puede contener paquetes utilizados para agrupar elementos del modelo.

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes.

La implementación del Centro Coordinador Provincial de Emergencia Médica se dividió en 4 subsistemas con el objetivo de hacer más factible la reutilización de código, disminuyendo el impacto que puede procurar un cambio. Se integra con SISalud y con el módulo Operaciones del Sistema Informatizado para el Centro Nacional de Urgencias Médicas.

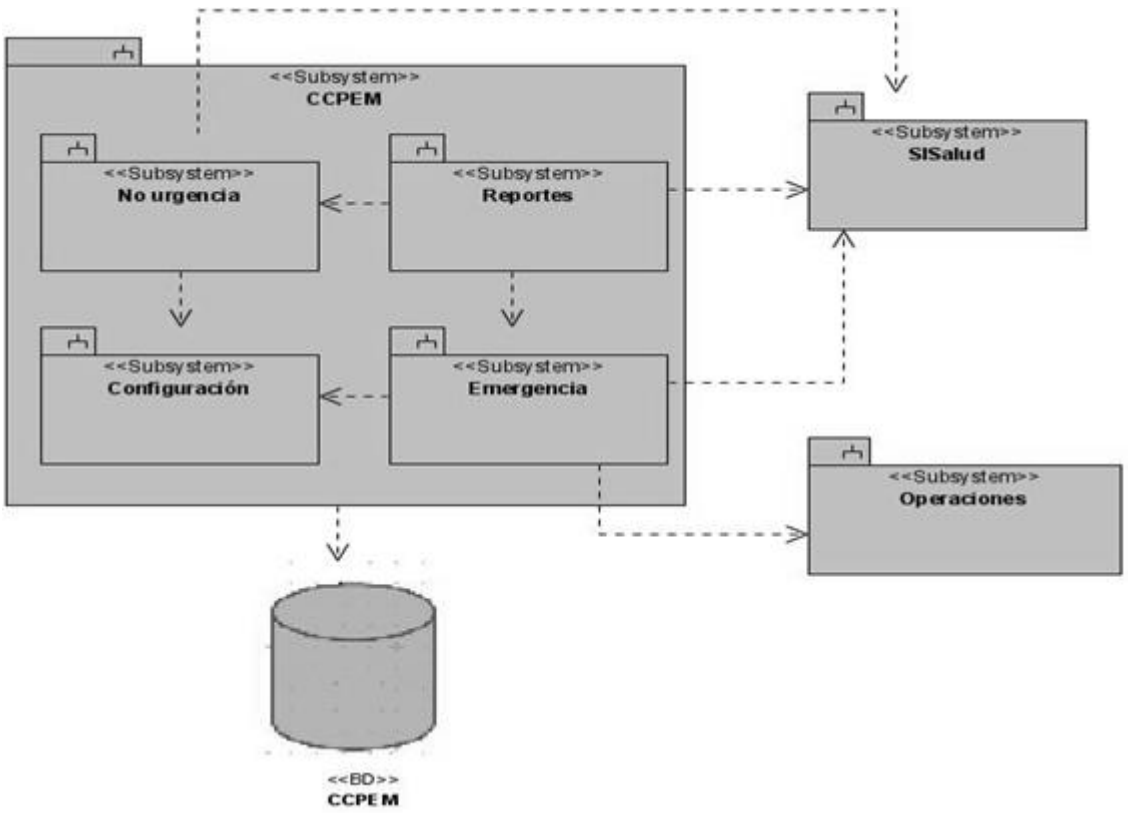


Figura 2.6: Organización en paquetes del módulo CCPEM.

El subsistema de Emergencia agrupa un conjunto de funcionalidades las cuales son:

- Gestionar demanda: esta funcionalidad pretende insertar, listar y modificar demandas.

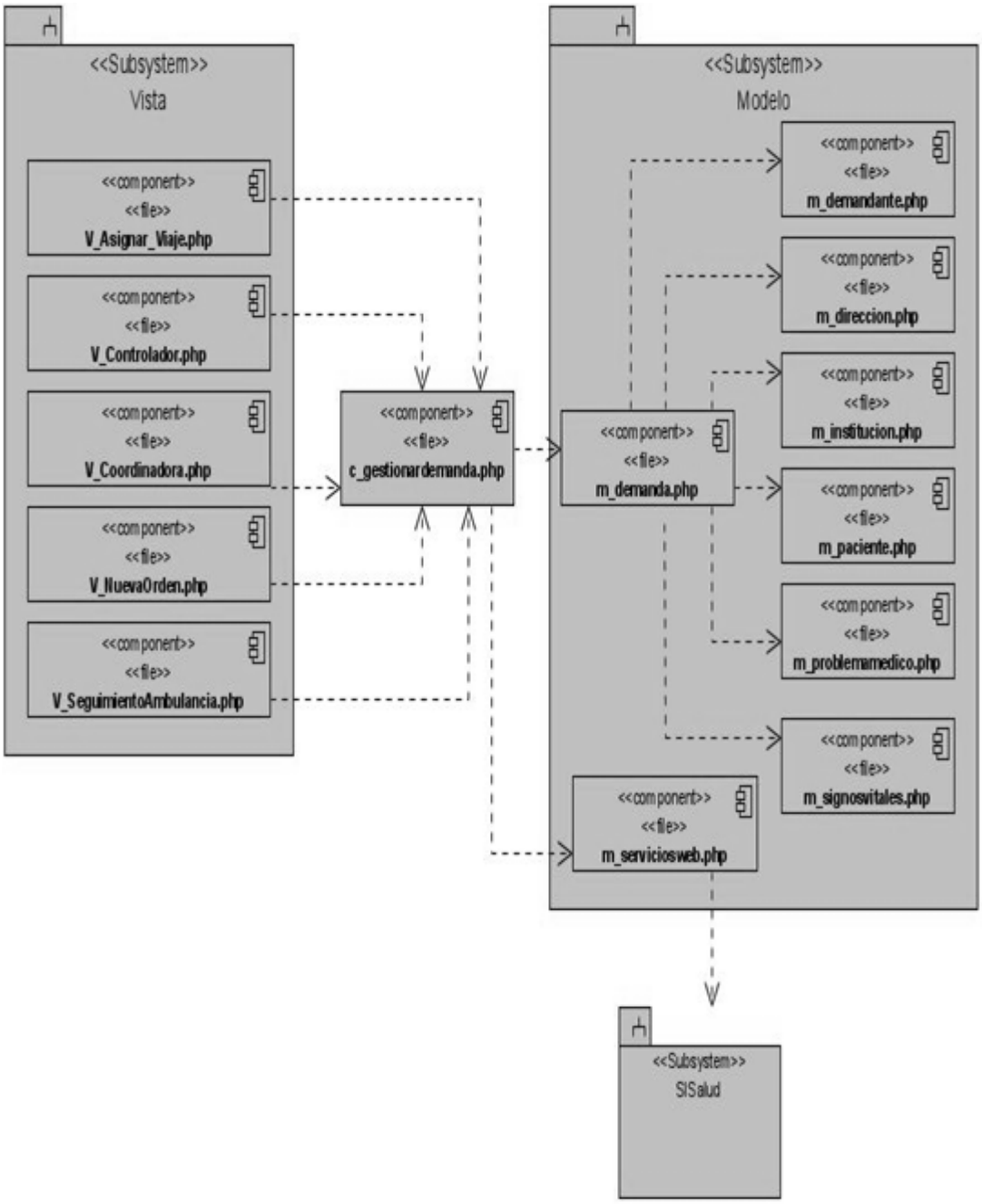


Figura 2.6: Vista de implementación del subsistema emergencia

El subsistema de no urgencia agrupa un conjunto de funcionalidades las cuales son:

- Gestionar certificado: esta funcionalidad pretende insertar, buscar, modificar y visualizar certificados.
- Gestionar demanda no urgente: pretende insertar, buscar, modificar y visualizar demandas no urgentes.

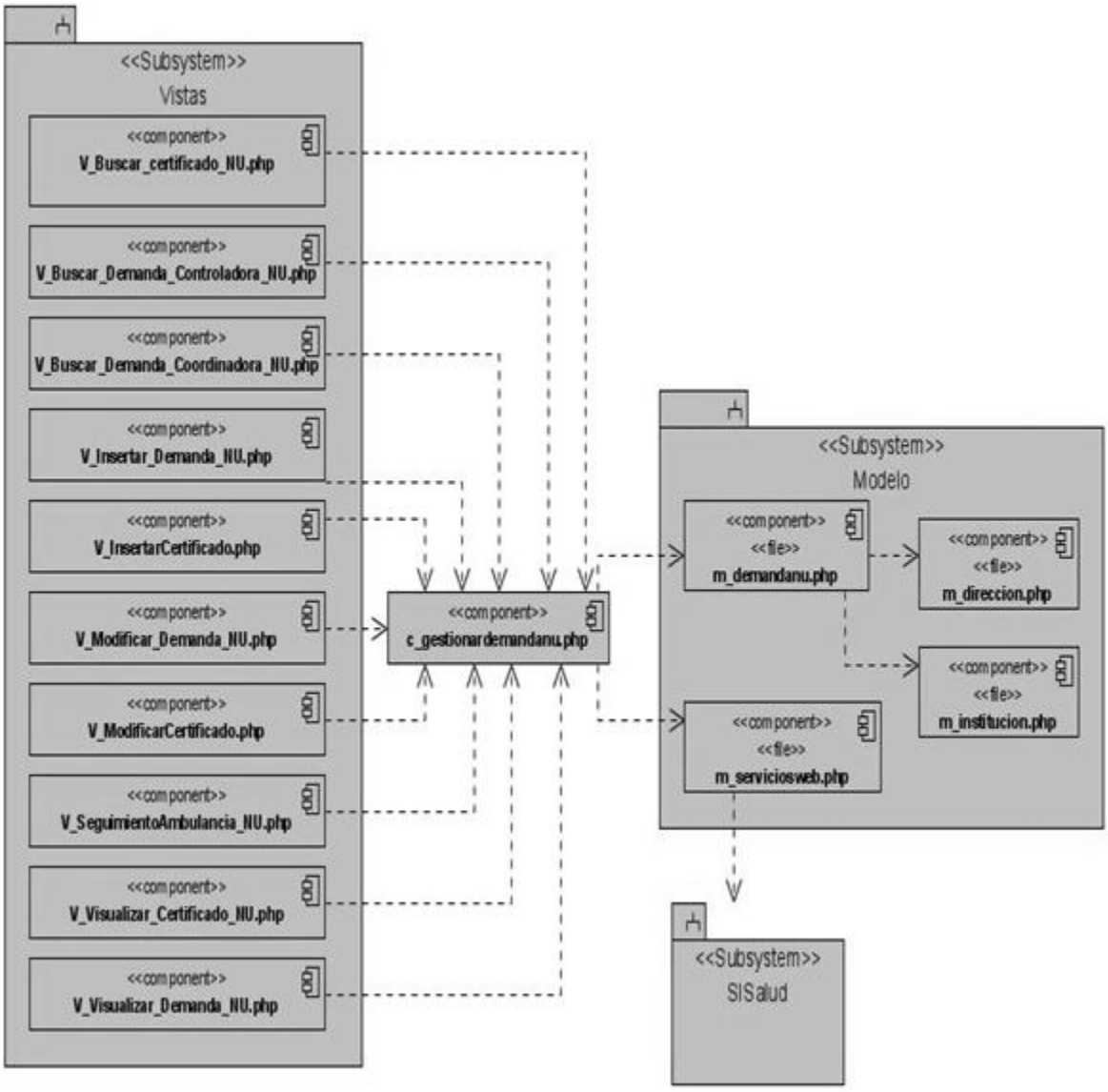


Figura 2.7: Vista de implementación del subsistema no urgencia

El subsistema de configuración agrupa un conjunto de funcionalidades las cuales son:

- Gestionar claves: esta funcionalidad pretende insertar, modificar, listar y eliminar claves en la base de datos.
- Gestionar estado_conciencia: esta funcionalidad pretende insertar, modificar, listar y eliminar estados de conciencia en la base de datos.
- Gestionar clasificación problema: esta funcionalidad pretende insertar, modificar, listar y eliminar clasificaciones del problema médico en la base de datos.
- Gestionar problema médico: esta funcionalidad pretende insertar, modificar, listar y eliminar problemas médicos en la base de datos.

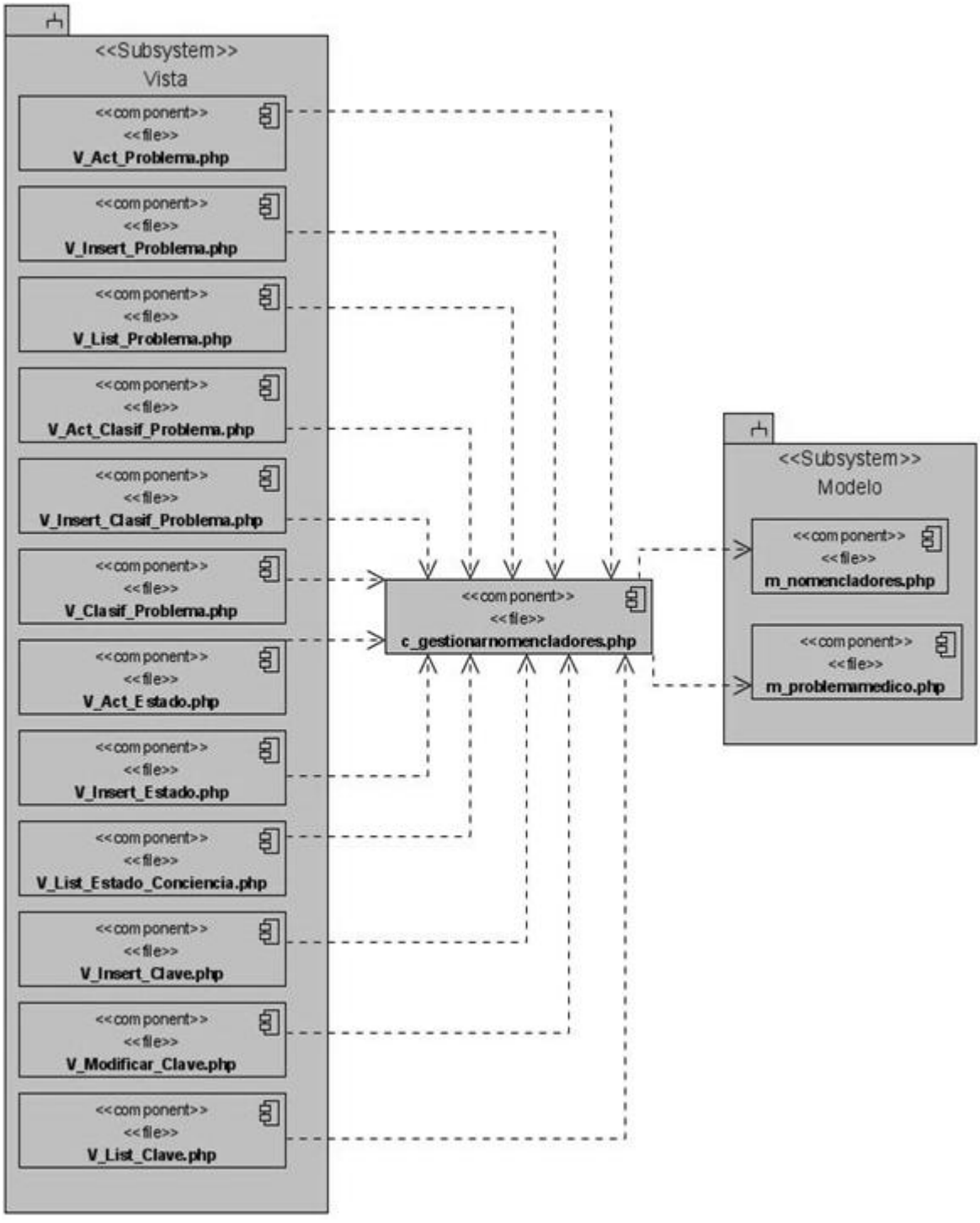


Figura 2.8: Vista de implementación del subsistema de configuración

El subsistema de reportes agrupa un conjunto de funcionalidades las cuales son:

- Generar reporte Hoja de Cargo: pretende realizar un reporte que pueda ser guardado o impreso por el personal que lo necesite.
- Generar reporte Parte Diario: pretende realizar un reporte que pueda ser guardado o impreso por el personal que lo necesite.
- Generar reporte Enfermedades Trazadoras: pretende realizar un reporte que pueda ser guardado o impreso por el personal que lo necesite.
- Generar reporte Entrega de Guardia no urgente: pretende realizar un reporte que pueda ser guardado o impreso por el personal que lo necesite.

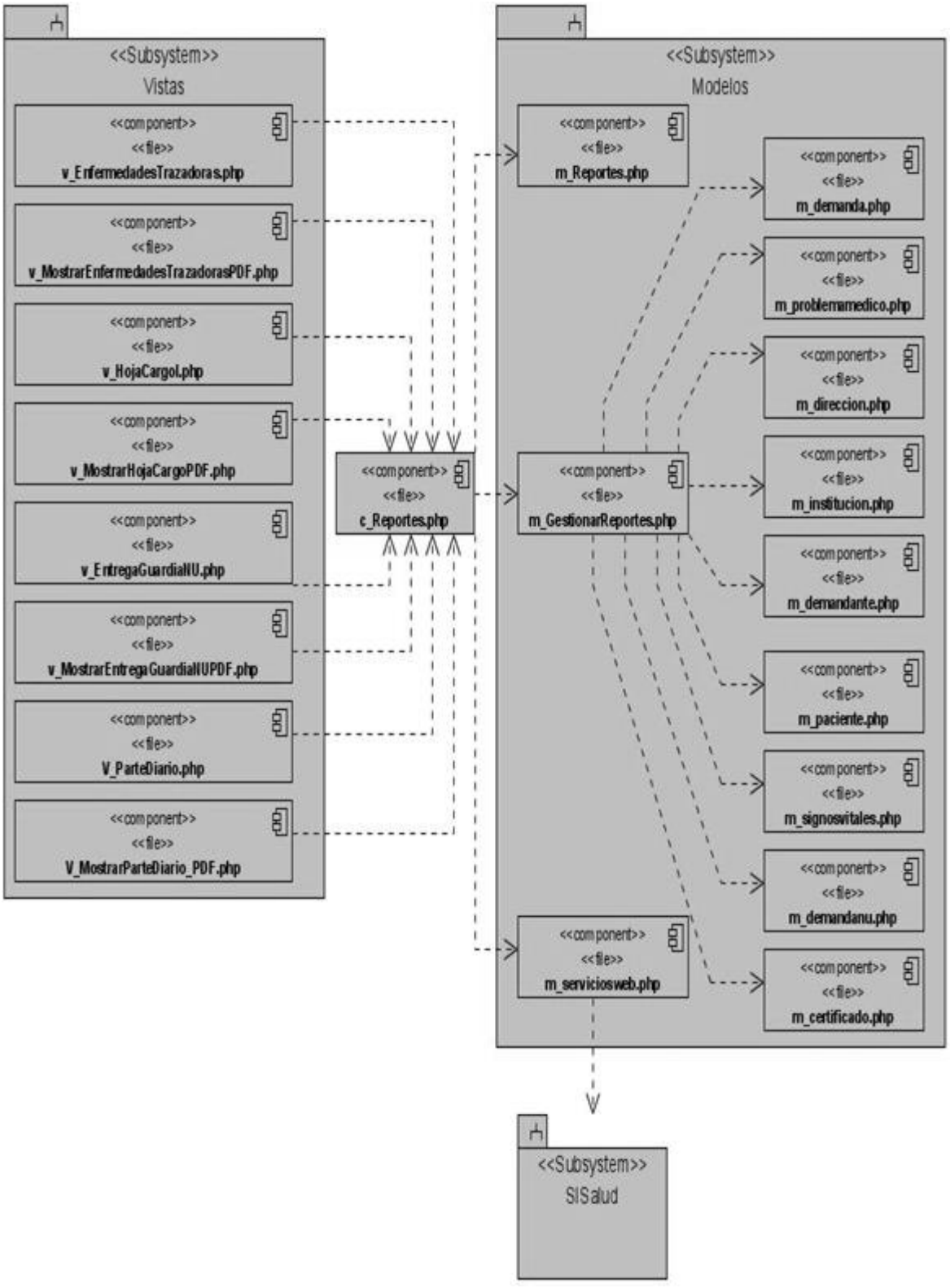


Figura 2.9: Vista de implementación del subsistema de Reportes

CAPITULO II: ELEMENTOS DE LA ARQUITECTURA

Con el desarrollo de este capítulo se muestra una perspectiva del sistema que se construyó en términos de requerimientos no funcionales y las funcionalidades agrupadas por subsistemas. Se identificó que el sistema tendría una arquitectura orientada a servicios y basada en componentes, también tendría una arquitectura en tres capas aplicando el patrón de diseño Modelo-Vista-Controlador (MVC). Se aplicaron otros patrones de diseño como el patrón experto, el creador, el controlador, se determinó que el diseño debe tener un bajo acoplamiento y una alta cohesión.

A través del estudio del diseño propuesto por los analistas se hizo una valoración crítica del mismo, señalando los beneficios para los implementadores. También se muestra una vista del despliegue del sistema donde queda claro, que estará desplegado en dos capas físicas.

Muestra una vista de la implementación del sistema, donde visualiza una organización en cuatro subsistemas, también muestra la organización interna de cada subsistema según el patrón de diseño MVC y siguiendo una arquitectura en tres capas.

CAPÍTULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

El presente capítulo tiene como objetivo describir los componentes que van a ser reutilizados. Hacer una descripción de las clases y operaciones más importantes en la implementación. Mostrar una vista del diseño de la base de datos y hacer una descripción de las tablas de la base de datos.

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

La reutilización de código se refiere al comportamiento y a las técnicas que garantizan, que una parte o la totalidad de un programa informático existente se puedan emplear en la construcción de otro programa, de esta forma se aprovecha el trabajo anterior, se economiza tiempo y se reduce la redundancia.

Se utilizó el framework CodeIgniter, que describe una estructura de software fácilmente ampliable y reutilizable, ya que aporta una rica biblioteca así como una interfaz simple y estructura lógica para acceder a estas. También permite modificar el código e insertar nuevas funcionalidades según necesidades del programador.

Se utilizó la librería FPDF, que es una clase escrita en PHP que permite generar documentos PDF directamente desde PHP sin usar la biblioteca PDFlib. La ventaja es que, mientras PDFlib es de pago para usos comerciales, la F de FPDF significa Free (gratis y libre), se puede usar para cualquier propósito y modificarla para satisfacer necesidades del programador.
Se utiliza la librería clientesoap que es una clase diseñada por el área temática Especialización, cuyo objetivo es hacer más amena la interacción entre el sistema y los servicios web a consumir.
Se integra con el módulo Operaciones, el mismo le brinda la disponibilidad de ambulancias que existe a través del método Disponibilidad (). Brinda algunos datos de la ambulancia como categoría, chapa y clave a través del método Datos_Ambulancia (). Posibilita que el módulo CCPEM pueda insertar una demanda en el Modulo operaciones a través del método Insertar_Demanda ().
También se integra con algunos componentes del Sistema de Información para la Salud (SISalud), como:

- **Componente de seguridad SAAA (Registro de Seguridad Autenticación Autorización y Auditoría)**: La autenticación debe ser la primera acción del usuario en el sistema. Consiste en suministrar un nombre de usuario único y una contraseña que debe ser de conocimiento

exclusivo de la persona que se autentica. Si el usuario autenticado no se encuentra registrado se reporta un error de acceso, en caso contrario se autoriza el acceso y se crea un certificado digital retornando los datos y permisos del usuario.

- **RU (Registro de Ubicación)**: Este componente tiene registrado las provincias del país y los municipios de cada una de ellas, así como otras informaciones. Se utiliza el método ListarProvincias() para obtener todos las provincias de nuestro país y el método ListarMunicipio() para obtener los municipios de una provincia dada
- **RPS (Registro de Personal de Salud)**: Este componente tiene registrado todo el personal de salud de Cuba. Se utiliza para cuando el demandante es una persona perteneciente al área de salud, los datos que el mismo brinda a la enfermera coordinadora serán verificados contra los datos que tiene almacenado el RPS. Esto se hace mediante el uso del método BuscarProfesionalesRapido (), que dado el número del registro profesional se obtienen los datos de esa persona.
- **RUS (Registro de Unidades de Salud)**: Es un componente que contiene todos los datos de las unidades de salud de Cuba. Es utilizado en situaciones cuando una demanda tiene como origen o como destino, o como ambos una institución médica. Se usa el método Buscar_Total () para obtener las instituciones de salud de un municipio dado y obtener dado una institución de salud sus características.
- **RC (Registro de Ciudadanos)**: Es un componente que contiene la misma información que registra la Oficina de Carné de Identidad de las personas en Cuba, como datos generales, datos de nacimiento, dirección y datos de los padres. La integración con RC se realiza con el fin de uniformar los datos de los usuarios del componente Centro Coordinador Provincial de Emergencia Médica con los del registro externo RC para validar la autenticidad de los mismos. Se utiliza los métodos BuscarTotalCiudadano () con el propósito de obtener los datos de un ciudadano dado su identificador, y BuscarCiudadano () para obtener todos los ciudadanos que cumplan con un criterio de búsqueda dado.

3.2 Descripción de las clases u operaciones necesarias.

Una clase es la parte de programación que nos ayuda a definir la funcionalidad que tendrá un objeto, dentro de ella se describen las capacidades del objeto y entre estas capacidades estarán principalmente las propiedades, los métodos, los eventos.

3.2.1 Descripción de las paginas clientes.

Nombre: V_Coordinadora
Tipo de clase: Vista
Descripción General: La *clase V_Coordinadora* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Permite capturar los datos de una demanda que luego serán insertados en la base de datos. Contiene un conjunto de validaciones de entrada en javascript. Es utilizada en el caso de uso: - Insertar demanda.

Tabla 3.2.1.1: Descripción de la vista V_Coordinadora.

Nombre: V_Controlador
Tipo de clase: Vista
Descripción General: La clase *V_Controlador* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Permite visualizar al controlador las demandas que deben atenderse, así como las que se están dando seguimiento. Contiene un conjunto de validaciones de entrada en javascript. Es utilizada en el caso de uso: - Gestionar demanda.

Tabla 3.2.1.2: Descripción de la vista V_Controlador.

Nombre: V_NuevaOrden
Tipo de clase: Vista
Descripción General: La clase *V_NuevaOrden* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Esta clase recibe información del servicio web que brinda el módulo operaciones, correspondiente a la disponibilidad de ambulancia, y a partir de esta información es responsabilidad del coordinador escoger la ambulancia que le va a dar respuesta al caso. Contiene un conjunto de validaciones javascript que no permiten realizar peticiones innecesarias. Es utilizada en el caso de uso: - Gestionar demanda.

Tabla 3.2.1.3: Descripción de la vista V_NuevaOrden.

Nombre: V_SeguimientoAmbulancia
Tipo de clase: Vista
Descripción General: La clase *V_SeguimientoAmbulancia* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Esta clase brinda la posibilidad de ir escogiendo claves por la que esta pasando una ambulancia y como seleccionar el estado del caso (sin efecto, fallido y fallecido). Contiene un conjunto de validaciones javascript que no permiten realizar peticiones innecesarias. Es utilizada en el caso de uso: - Gestionar demanda.

Tabla 3.2.1.4: Descripción de la vista V_SeguimientoAmbulancia.

Nombre: V_Act_Clasif_Problema
Tipo de clase: Vista
Descripción General: La clase *V_Act_Clasif_Problema* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Esta clase permite visualizar los datos del estado de conciencia seleccionado, posibilitando la modificación de los mismos. Contiene un conjunto de validaciones javascript que no permiten realizar peticiones innecesarias. Es utilizada en el caso de uso: - Gestionar clasificación problema.

Tabla 3.2.1.5: Descripción de la vista V_Act_Clasif_Problema.

Nombre: V_Act_Estado
Tipo de clase: Vista
Descripción General: La clase *V_Act_Estado* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Esta clase permite visualizar los datos de la clasificación del problema medico seleccionada, posibilitando la modificación de los mismos. Contiene un conjunto de validaciones javascript que no permiten realizar peticiones innecesarias. Es utilizada en el caso de uso: - Gestionar estado_conciencia.

Tabla 3.2.1.6: Descripción de la vista V_Act_Estado.

Nombre: V_Act_Problema
Tipo de clase: Vista
Descripción General: La clase *V_Act_Problema* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Esta clase permite visualizar los datos del problema médico seleccionado, posibilitando la modificación de los mismos. Contiene un conjunto de validaciones javascript que no permiten realizar peticiones innecesarias. Es utilizada en el caso de uso: - Gestionar problema médico.

Tabla 3.2.1.7: Descripción de la vista V_Act_Problema.

Nombre: V_Buscar_certificado_NU
Tipo de clase: Vista
Descripción General: La clase *V_Buscar_certificado_NU* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Esta clase permite buscar los certificados y escoger el parámetro de búsqueda, los datos encontrados son mostrados en la página. Contiene un conjunto de validaciones javascript que no permiten realizar peticiones innecesarias. Es utilizada en el caso de uso: - Gestionar certificado.

Tabla 3.2.1.8: Descripción de la vista V_Buscar_certificado_NU.

Nombre: V_Buscar_Demanda_Controladora_NU
Tipo de clase: Vista
Descripción General: La clase *V_Buscar_Demanda_Controladora_NU* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Esta clase permite buscar todas las demandas y certificados, permite escoger el parámetro de búsqueda dentro de los definidos y los datos encontrados son mostrados en la página. Contiene un conjunto de validaciones javascript que no permiten realizar peticiones innecesarias. Es utilizada en el caso de uso: - Gestionar demanda no urgente.

Tabla 3.2.1.9: Descripción de la vista V_Buscar_Demanda_Controladora_NU.

Nombre: V_Buscar_Demanda_Coordinadora_NU
Tipo de clase: Vista
Descripción General: La clase *V_Buscar_Demanda_Coordinadora_NU* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Esta clase permite buscar todas las demandas y certificados, permite escoger el parámetro de búsqueda dentro de los definidos y los datos encontrados son mostrados en la página. Contiene un conjunto de validaciones javascript que no permiten realizar peticiones innecesarias. Es utilizada en el caso de uso: - Gestionar demanda no urgente

Tabla 3.2.1.10: Descripción de la vista V_Buscar_Demanda_Controladora_NU.

Nombre: V_Clasif_Problema
Tipo de clase: Vista
Descripción General: La clase *V_Clasif_Problema* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Esta clase permite buscar todas las clasificaciones de los problemas médicos y los datos encontrados son mostrados en la página. Contiene un conjunto de validaciones javascript que no permiten realizar peticiones innecesarias. Es utilizada en el caso de uso: - Gestionar clasificación problema.

Tabla 3.2.1.11: Descripción de la vista V_Clasif_Problema.

Nombre: V_Insert_Clasif_Problema
Tipo de clase: Vista
Descripción General: La clase *V_Insert_Clasif_Problema* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Esta clase permite capturar los datos necesarios para insertar una clasificación de un problema medico. Contiene un conjunto de validaciones javascript que no permiten realizar peticiones innecesarias. Es utilizada en el caso de uso: - Gestionar clasificación problema.

Tabla 3.2.1.12: Descripción de la vista V_Insert_Clasif_Problema.

Nombre: V_Insert_Clave
Tipo de clase: Vista
Descripción General: La clase *V_Insert_Clave* es una página Web que se ejecuta del lado del cliente sobre un navegador Web. Esta clase permite capturar los datos necesarios para insertar clave. Contiene un conjunto de validaciones javascript que no permiten realizar peticiones innecesarias. Es utilizada en el caso de uso: - Gestionar clave.

Tabla 3.2.1.13: Descripción de la vista V_Insert_Clasif_Problema.

3.2.2 Descripción de las clases controladoras.

Nombre: c_gestionardemandanu	
Tipo de clase: controladora	
Responsabilidades:	
Nombre	c_gestionardemandanu
Descripción	Constructor de la clase.
Nombre	MostrarInsertarCertificado
Descripción	Carga la vista certificado con algunos datos previos como las provincias.
Nombre	InsertarCertificado
Descripción	Inserta el certificado en la base de datos.
Nombre	MostrarInsertarDemandaNU
Descripción	Carga la vista demanda no urgente con algunos datos previos como las provincias.
Nombre	InsertarDemandaNU
Descripción	Inserta la demanda no urgente en la base de datos.
Nombre	MostrarCoordinadoraNU

Descripción	Carga la página donde la coordinadora ve las demandas que tienen para una fecha y una base.
Nombre	ListarCoordinadoraNU
Descripción	Muestra un listado de todas las demandas que ahí en un día determinado para una base dada.
Nombre	MostrarVisualizar($id)
Descripción	Es para visualizar los datos de una demanda o de un certificado para la coordinadora.
Nombre	MostrarVisualizarControlador($id)
Descripción	Es para visualizar los datos de una demanda o de un certificado para el controlador.
Nombre	MostrarDatosCertificado($id)
Descripción	Muestra los datos de un certificado dado.
Nombre	MostrarActualizar($id, $pagina)
Descripción	Muestra los certificados que pueden ser actualizados dependiendo de un criterio de búsqueda.
Nombre	ActualizarCertificado
Descripción	Envía a la modelo m_demandanu los datos de un certificado para que sean actualizados en la Base de Datos.
Nombre	ActualizarDemandaNU
Descripción	Envía a la modelo m_demandanu los datos de una demanda para que sean actualizados en la Base de Datos.
Nombre	MostrarControladorNU
Descripción	Carga la página donde el controlador ve las demandas que tienen para una fecha y una base.
Nombre	ListarControladorNU
Descripción	Muestra un listado de todas las demandas que ahí en un día

	determinado para una base dada.
Nombre	MostrarModificarDemandaNU($id)
Descripción	Muestra los datos de una demanda para que sean actualizados el viaje y el móvil que va a trasladar a la demanda.
Nombre	MostrarClaveDemandaNU($id)
Descripción	Muestra los datos de una demanda para que se le inserte una clave en la base de datos.
Nombre	SeleccionarOpciones($id)
Descripción	Para saber si se va a mostrar la pantalla de modificar demanda o la de modificar clave.
Nombre	ModificarDemandaNU
Descripción	Envía a la modelo m_demandanu los datos para que sea modificada la demanda.
Nombre	InsertarClave
Descripción	Envía a la modelo m_demandanu los datos para que sea insertada la clave.
Nombre	MostrarBuscarCertificadoNU
Descripción	Muestra la pantalla para buscar las demandas por diferentes criterios de búsqueda.
Nombre	BuscarCertificadoNU
Descripción	Recoge los datos y se los envía a la modelo m_demandanu para que busque los certificados.
Nombre	BuscarCertificado
Descripción	Crea un listado con todos los certificados que cumplen un criterio de búsqueda dado.

Tabla 3.2.2.1: Gestionar demanda no urgente.

Nombre: c_gestionardemanda	
Tipo de clase: controladora	
Responsabilidades:	
Nombre	c_gestionardemanda
Descripción	Constructor de la clase.
Nombre	MostrarControlador
Descripción	Carga la vista V_Controlador
Nombre	MostrarCoordinadora
Descripción	Muestra la página de la coordinadora de emergencia.
Nombre	MostrarModificarDemanda($id)
Descripción	Carga la vista modificar demanda para en ella visualizar algunos datos e insertar la ambulancia que atenderá el caso.
Nombre	MostrarModificarClave($id)
Descripción	Carga la vista modificar clave con algunos datos previos como las claves existentes, el identificador de la ambulancia y el código de la demanda.
Nombre	Seleccionar($id)
Descripción	Selecciona si la demanda está aún sin ambulancia o si ya la tiene para definir que popup mostrarle al controlador.
Nombre	XMLMunicipios
Descripción	Crea un XML con los municipios de una provincia dada.
Nombre	XMLInstituciones
Descripción	Crea un XML con las instituciones de salud de un municipio dado.
Nombre	XMLProblema
Descripción	Crea un XML con los problemas médicos dado una clasificación.

Nombre	PersonaldeSalud
Descripción	Verifica si el registro profesional del demandante es válido.
Nombre	InsertarDemanda
Descripción	Inserta una demanda en la base de datos
Nombre	IdDemanda
Descripción	Muestra el número que tendrá en la base de datos, la próxima demanda a insertar.
Nombre	ListarDemanda
Descripción	Lista todas las demandas que pertenecen al controlador.
Nombre	ModificarDemanda
Descripción	Modifica el identificador de la ambulancia en la base de datos.
Nombre	hora
Descripción	Muestra la hora actual.
Nombre	InsertarClave
Descripción	Insertar la clave y la hora en que está actualmente una ambulancia atendiendo una demanda.

Table 3.2.2.2: Gestionar demanda.

Nombre: c_gestionarnomencladores	
Tipo de clase: controladora.	
Responsabilidades:	
Nombre	__construct
Descripción	Constructor de la clase.
Nombre	ListarClasificacion
Descripción	Lista las clasificaciones del problema médico.
Nombre	eliminarclasif

Descripción	Eliminar una clasificación del problema médico.
Nombre	MostrarActualizarClasificacion($id)
Descripción	Muestra la vista actualizar clasificación del problema médico.
Nombre	ActualizarClasificacion
Descripción	Le envía a la modelo m_nomencladores los datos de la clasificación del problema médico a actualizar.
Nombre	MostrarInsertarClasif
Descripción	Muestra la vista insertar clasificación del problema médico.
Nombre	InsertarClasif
Descripción	Le envía a la modelo m_nomencladores los datos de la clasificación del problema médico a insertar.
Nombre	MostrarInsertClave
Descripción	Muestra la vista insertar clave.
Nombre	InsertarClave
Descripción	Le envía a la modelo m_nomencladores los datos de la clave a insertar.
Nombre	ListarClave
Descripción	Lista todos las claves existentes en al base de datos.
Nombre	EliminarClave
Descripción	Le envía a la modelo m_nomencladores los datos para que elimine una clave.
Nombre	MostrarModificarClave
Descripción	Muestra una vista con los datos de la clave que se quiere modificar.
Nombre	ModificarClave
Descripción	Le envía a la modelo m_nomencladores los datos para que modifique una clave.
Nombre	ListarEstadoConciencia

Descripción	Lista los estados de conciencia existentes en la base de datos.
Nombre	EliminarEstadoConciencia
Descripción	Le envía a la modelo m_nomencladores los datos para que elimine un estado de conciencia.
Nombre	MostrarInsertarEstado
Descripción	Muestra la vista para insertar el estado de conciencia.
Nombre	InsertarEstado
Descripción	Recibe los datos del estado de conciencia y se los envía a la modelo m_nomencladores para que sean insertados.
Nombre	MostrarDatosEstado
Descripción	Muestra los datos de un estado de conciencia dado.
Nombre	ModificarEstado
Descripción	Recibe los datos del estado de conciencia y se lo envía a la modelo m_nomencladores para que sean modificados.
Nombre	ListarProblema
Descripción	Muestra un listado con todos los problemas médicos existentes en la Base de Datos.
Nombre	EliminarProblema
Descripción	Envía a la modelo m_nomencladores los datos del problema médico que va a ser eliminado.
Nombre	DatosProblema
Descripción	Carga los datos de un problema médico dado.
Nombre	ModificarProblema
Descripción	Recoge los datos del problema médico.
Nombre	MostrarInsertProblema
Descripción	Muestra la vista para insertar un nuevo problema médico.

Nombre	InsertarProblema
Descripción	Recoge los datos del problema médico y se los envía a la modelo m_nomencladores para ser insertados en la Base de Datos.

Tabla 3.2.2.3: Gestionar nomencladores.

Nombre: C_GestionarUsuario	
Tipo de clase: controladora.	
Responsabilidades:	
Nombre	C_GestionarUsuario
Descripción	Constructor de la clase.
Nombre	index
Descripción	Método que carga la página de Autenticarse.
Nombre	Autenticarse
Descripción	Método que envía los datos del usuario a la modelo m_usuario para verificar que es usuario del sistema.
Nombre	MostrarBienvenida
Descripción	Carga la pagina V_Bienvenida.
Nombre	Salir
Descripción	Método que permite cerrar la sesión que se está trabajando.
Nombre	MostrarListarCiudadano
Descripción	Carga la pagina V_Listar_Ciudadano.

Tabla 3.2.2.4: Gestionar usuario.

Nombre: c_Reportes	
Tipo de clase: controladora.	
Responsabilidades:	
Nombre	c_Reportes

Descripción	Constructor de la clase.
Nombre	hoja_cargo
Descripción	Muestra la vista v_HojaCargoI.
Nombre	f_HojaCargo
Descripción	Recoge los datos que viene por post que fue enviado por la vista v_HojaCargoI, y decide la ejecución de los métodos HojaCargo_guardar_PDF o HojaCargo_mostrar_PDF.
Nombre	HojaCargo_mostrar_PDF($fecha)
Descripción	Recibe una fecha y se encarga de obtener los resultados que necesita el reporte hoja de cargo, haciendo una llamada al método hojacargo () de la clase m_GestionarReportes. Luego se encarga de generar un reporte en formato pdf con los datos obtenidos.
Nombre	HojaCargo_guardar_PDF($fecha)
Descripción	Recibe una fecha y se encarga de obtener los resultados que necesita el reporte hoja de cargo, haciendo una llamada al método hojacargo de la clase m_GestionarReportes. Luego se encarga de generar un reporte en formato pdf con los datos obtenidos.
Nombre	parte_diario
Descripción	Muestra la vista V_ParteDiario.
Nombre	f_parte_diario
Descripción	Recoge los datos que viene por post que fue enviado por la vista V_ParteDiario, obtiene un arreglo con los municipios de la provincia seleccionada en la vista y decide la ejecución de los métodos parte_diario_mostrar_PDF o parte_diario_guardar_PDF.
Nombre	parte_diario_mostrar_PDF($fecha,$arreglo)
Descripción	Recibe una fecha y un arreglo de municipios y se encarga de obtener los resultados que necesita el reporte parte diario, haciendo una llamada al método partediario () de la clase m_GestionarReportes. Luego se encarga

	de generar un reporte en formato pdf con los datos obtenidos.
Nombre	parte_diario_guardar_PDF($fecha,$arreglo)
Descripción	Recibe una fecha y un arreglo de municipios y se encarga de obtener los resultados que necesita el reporte parte diario, haciendo una llamada al método partediario de la clase m_GestionarReportes. Luego se encarga de guardar un reporte en formato pdf con los datos obtenidos.
Nombre	entrega_de_guardia_no_urgente
Descripción	Carga la página v_EntregaGuardiaNU.
Nombre	f_entrega_de_guardia_no_urgente
Descripción	Recoge los datos que viene por post que fue enviado por la vista v_EntregaGuardiaNU, y decide la ejecución de los métodos entrega_de_guardia_no_urgente_mostrar_PDF o entrega_de_guardia_no_urgente_guardar_PDF.
Nombre	entrega_de_guardia_no_urgente_mostrar_PDF($fecha)
Descripción	Recibe una fecha y se encarga de obtener los resultados que necesita el reporte entrega de guardia no urgente, haciendo una llamada al método entrega_de_guardia_no_urgente de la clase m_GestionarReportes. Luego se encarga de generar el reporte en formato pdf con los datos obtenidos.
Nombre	entrega_de_guardia_no_urgente_guardar_PDF($fecha)
Descripción	Recibe una fecha y se encarga de obtener los resultados que necesita el reporte entrega de guardia no urgente, haciendo una llamada al método entrega_de_guardia_no_urgente de la clase m_GestionarReportes. Luego se encarga de guardar el reporte en formato pdf con los datos obtenidos.

Tabla 3.2.2.5: Reportes.

3.2.3 Descripción de las clases Modelos.

Nombre: m_demanda	
Tipo de clase: Modelo	
Responsabilidades:	
Nombre	m_demanda
Descripción	Constructor de la clase.
Nombre	BuscarInstitucion($id)
Descripción	Busca los datos de una institución de salud dada en el Registro de Unidades de Salud.
Nombre	NombreMunicipio($id_municipio)
Descripción	Busca el nombre de un municipio dado en el Registro de Ubicación.
Nombre	InsertarDemanda($demanda,$rb_origen,$rb_destino,$instituciones,$instituciones_des,direccion,$direccion_destino,$municipio_part,$municipio_part1)
Descripción	Inserta lo datos de una demanda en la Base de Datos.
Nombre	ListarDemanda($per_page,$offset)
Descripción	Crea un listado de demanda según un criterio de búsqueda.
Nombre	DireccionOrigen($id_demanda)
Descripción	Muestra la dirección de origen de una demanda dada.
Nombre	DireccionDestino($id_demanda)
Descripción	Muestra la dirección de destino de una demanda dada.
Nombre	DatosDemanda($id_demanda)
Descripción	Retorna los datos de una demanda dada.
Nombre	DatosClave($id_demanda)
Descripción	Retorna los datos de una demanda dada.
Nombre	ModificarDemanda($id_demanda,$id_ambulancia,$fecha,$hora)

Descripción	Modifica los datos de una demanda dada.
Nombre	ComboEstadoConciencia
Descripción	Crea un select con los estados de conciencia encontrados en la Base de Datos.
Nombre	IdDemanda
Descripción	Devuelve el siguiente id al ultimo existente en la tabla tb_demanda.
Nombre	Estado($id)
Descripción	Devuelve el estado de una demanda dada (Iniciada, Ejecución,Terminada)
Nombre	ModificarDemandaClave($demanda,$clave,$hora,$causa,$observacion)
Descripción	Modifica la Clave de una demanda dada.
Nombre	m_listar_demanda($fecha,$id_municipio)
Descripción	Recibe por parámetro una fecha y un id de un municipio, lo que hace es hacer una búsqueda de las demandas que se realizaron en esa fecha y en el municipio especificado.
Nombre	demanda($fecha)
Descripción	Recibe una fecha, lo que hace es una búsqueda por fecha de las demandas que se realizaron en ese día.

Tabla 3.2.3.1: Modelo demanda.

Nombre: M_Certificado	
Tipo de clase: Modelo	
Responsabilidades:	
Nombre	m_certificado
Descripción	Constructor de la clase.
Nombre	InsertarCertificado($certificado,$frecuencia,$rb_origen,$direccion,$municipio_part,$instituciones,$rb_destino,$direccion_destino,$municipio_part1,$instituciones_des)

Descripción	Inserta un certificado en la Base de Datos.
Nombre	bool_certificado($id_demandanu)
Descripción	Recibe el id de la demanda no urgente para verificar si esa demanda es un certificado.

Tabla 3.2.3.2: Modelo certificado.

Nombre: m_demandante	
Tipo de clase: Modelo.	
Responsabilidades:	
Nombre	m_demandante
Descripción	Constructor de la clase.
Nombre	InsertarDemandante($demandante)
Descripción	Inserta un demandante en la Base de Datos.
Nombre	demandante($id_demandante)
Descripción	Recibe el id del demandante y realiza una búsqueda en la base de datos dado el id para obtener todos los datos del mismo.

Tabla 3.2.3.3: Modelo demandante.

Nombre: m_demandanu	
Tipo de clase: Modelo	
Responsabilidades:	
Nombre	m_demandanu
Descripción	Constructor de la clase.
Nombre	DireccionOrigen($id_demanda)
Descripción	Retorna la dirección de origen de una demanda dada.
Nombre	DireccionDestino($id_demanda)
Descripción	Retorna la dirección de destino de una demanda dada.

Nombre	NombreMunicipio($id_municipio)
Descripción	Dado el identificador de un municipio retorna su nombre del servicio web Registro de Ubicación.
Nombre	BuscarInstitucion($id)
Descripción	Dado el identificador de una institución me devuelve los datos de la institución donde está.
Nombre	InsertarDemandaNU($demanda_nu, $rb_origen, $direccion, $municipio_part, $instituciones, $rb_destino, $direccion_destino, $municipio_part1, $instituciones_des)
Descripción	Inserta una Demanda No Urgente en la Base de Datos.
Nombre	InsertarCertificado($certificado, $frecuencia, $rb_origen, $direccion, $municipio_part, $instituciones, $rb_destino, $direccion_destino, $municipio_part1, $instituciones_des)
Descripción	Inserta un certificado en la Base de Datos.
Nombre	DiaSemana($fecha)
Descripción	Dado una fecha devuelve el día de la semana que pertenece.
Nombre	ListarCoordinadoraNU($municipio,$fecha)
Descripción	Lista todas las demanda de la coordinadora dada una base y una fecha.
Nombre	ListarControladorNU($municipio,$fecha)
Descripción	Lista todas las demandas del controlador, dada una base y una fecha.
Nombre	Visualizar($id_demanda)
Descripción	Carga los datos de una demanda dado su identificador en la Base de Datos.
Nombre	ComboSexo($id)

Descripción	Devuelve un select con femenino y masculino.
Nombre	ComboCarro($id)
Descripción	Devuelve un select con los tipos de autos existentes en la Base de Datos poniendo por defecto el que tiene un identificador dado.
Nombre	ActualizarDemandaNU($id_demanda, $demanda_nu, $rb_origen, $direccion, $municipio_part, $instituciones, $rb_destino, $direccion_destino, $municipio_part1, $instituciones_des)
Descripción	Actualiza en la Base de Datos los datos de una demanda dada.
Nombre	ActualizarCertificado($id_demanda, certificado, $frecuencia, $rb_origen, $direccion, $municipio_part, $instituciones, $rb_destino, $direccion_destino, $municipio_part1, $instituciones_des)
Descripción	Actualiza en la Base de Datos los datos de un certificado dado.
Nombre	SeleccionarOpciones($id_demanda)
Descripción	Retorna verdadero o false según si la demanda ya tiene un viaje asignado o no.
Nombre	ComboClave
Descripción	Retorna todas las claves existentes en la Base de Datos.
Nombre	ActualizarViaje($id_demanda,$viaje,$ambulancia)
Descripción	Actualiza los datos de una demanda asignándole un viaje.
Nombre	InsertarClave($id_demanda,$clave,$hora,$causa)
Descripción	Inserta una clave en la Base de Datos para una demanda dada.
Nombre	BuscarCertificadoNU($criterio)
Descripción	Busca los certificados en la Base de Datos que cumplen con un criterio dado.
Nombre	listar_demanda_nu($fecha)

Descripción	Recibe una fecha como parámetro y realiza una búsqueda de todas las demandas realizadas en esa fecha.

Tabla 3.2.3.4: Modelo demanda no urgente.

Nombre: m_direccion	
Tipo de clase: Modelo	
Responsabilidades:	
Nombre	m_direccion
Descripción	Constructor de la clase.
Nombre	InsertarDireccion($dir)
Descripción	Inserta una dirección en la Base de Datos.
Nombre	InsertarDemandaDireccion ($id_direccion, $id_demanda, $estado)
Descripción	Inserta el identificador de la demanda y de la dirección en la Base de Datos.
Nombre	InsertarDemandaDireccionNU ($id_direccion, $id_demanda, $estado)
Descripción	Inserta el identificador de la demanda no urgente y de la dirección en la Base de Datos.

Tabla 3.2.3.5: Modelo dirección.

Nombre: m_institucion	
Tipo de clase: Modelo	
Responsabilidades:	
Nombre	m_institucion
Descripción	Constructor de la clase.
Nombre	InsertarInstitucion($ins)
Descripción	Insertar una institución en la Base de Datos.

Nombre	InsertarDemandaInstitucion ($id_institucion, $id_demanda, $estado)
Descripción	Inserta el identificador de la demanda y de la institución en la Base de Datos
Nombre	InsertarDemandaInstitucionNU ($id_institucion, $id_demanda, $estado)
Descripción	Inserta el identificador de la demanda no urgente y de la institución en la Base de Datos.

Tabla 3.2.3.6: Modelo institución.

Nombre: m_nomencladores	
Tipo de clase: Modelo.	
Responsabilidades:	
Nombre	__construct
Descripción	Constructor de la clase.
Nombre	ListarClasif($per_page,$offset)
Descripción	Lista todas las clasificaciones del problema médico existentes en la Base de Datos.
Nombre	EliminarClasif($id)
Descripción	Elimina una clasificación de la Base de Datos dado su identificador.
Nombre	MostrarActualizar($id)
Descripción	Carga todos los datos de la Base de Datos de una clasificación dada.
Nombre	ActualizarClasificacion($id,$descripcion)
Descripción	Actualiza los datos de una clasificación.
Nombre	InsertarClasif($descripcion)
Descripción	Inserta una nueva clasificación del problema médico en la Base de Datos.
Nombre	InsertarClave($clave,$descripcion)
Descripción	Inserta una nueva clave en la Base de Datos

Nombre	ListarClave($per_page,$offset)
Descripción	Lista todas las claves existentes en la base de datos.
Nombre	ListarEstadoConciencia($per_page,$offset)
Descripción	Lista todos los estados de conciencia existentes en la Base de Datos
Nombre	EliminarClave($clave)
Descripción	Elimina una clave de la Base de Datos dado su identificador.
Nombre	MostrarDatosClave($clave)
Descripción	Carga los datos de la Base de Datos de una clave dada.
Nombre	ModificarClave($clave,$new_clave,$descripcion)
Descripción	Modifica los datos de una clave en la Base de Datos
Nombre	EliminarEstadoConciencia($id)
Descripción	Elimina un estado de conciencia de la Base de Datos.
Nombre	InsertarEstado($tipo)
Descripción	Insertar un nuevo estado de conciencia en la Base de Datos.
Nombre	DatosEstado($id_estado)
Descripción	Carga los datos de un estado de conciencia dado de la Base de Datos.
Nombre	ModificarEstado($codigo,$tipo)
Descripción	Modifica un estado de conciencia dado en la Base de Datos.
Nombre	ListarProblema($per_page,$offset)
Descripción	Lista todos los problemas médicos existentes en la Base de Datos.
Nombre	EliminarProblema($id)
Descripción	Elimina un problema médico dado de la Base de Datos.
Nombre	DatosProblema($id)
Descripción	Carga los datos de un problema médico dado de la Base de Datos.
Nombre	Clasificaciones

Descripción	Carga todas las clasificaciones existentes en la Base de Datos.
Nombre	ModificarProblema($codigo,$descripcion,$prioridad,$clasificacion)
Descripción	Modifica un problema médico en la Base de Datos.

Tabla 3.2.3.7: Modelo nomencladores.

Nombre: m_paciente	
Tipo de clase: Modelo.	
Responsabilidades:	
Nombre	m_paciente
Descripción	Constructor de la clase.
Nombre	InsertarPaciente($paciente)
Descripción	Inserta un paciente en la Base de Datos.

Tabla 3.2.3.8: Modelo paciente.

Nombre: m_problemamedico	
Tipo de clase: Modelo.	
Responsabilidades:	
Nombre	m_problemamedico
Descripción	Constructor de la clase.
Nombre	InsertarProblemaMedico($descripcion,$prioridad,$clasificacion)
Descripción	Insertar un problema médico en la Base de Datos.
Nombre	SelecionarClasificacionProblemasMedicos
Descripción	Selecciona todos los problemas médicos existentes en la Base de Datos.
Nombre	SeleccionarProblemaMedico($id_clasificacion)
Descripción	Carga todos los problemas médicos de la Base de Datos dada una clasificación.

Tabla 3.2.3.9: Modelo problema médico.

Nombre: serviciosweb	
Tipo de clase: Modelo	
Responsabilidades:	
Nombre	serviciosweb
Descripción	Constructor de la clase.
Nombre	ListarProvincias
Descripción	Listar todas las provincias del Registro de Ubicación.
Nombre	ListarMunicipios($id)
Descripción	Lista todos los municipios dado una provincia del Registro de Ubicación.
Nombre	ListarInstituciones($id)
Descripción	Lista todas las instituciones dado un municipio del Registro de Unidades de Salud.
Nombre	BuscarInstitucion($id)
Descripción	Busca los datos de una institución dado su identificador en el Registro de Unidades de Salud
Nombre	BuscarPersonalRegistro($registro)
Descripción	Buscar a una persona en el Registro de Personal de Salud dado su número de Registro Profesional.

Tabla 3.2.3.10: Modelo servicios web.

Nombre: m_signosvitales	
Tipo de clase: Modelo	
Responsabilidades:	
Nombre	m_signosvitales
Descripción	Constructor de la clase.

Nombre	InsertarSignosVitales($signos)
Descripción	Inserta un nuevo signo vital en la Base de Datos.

Tabla 3.2.3.11: Modelo signos vitales.

Nombre: m_usuario	
Tipo de clase: Modelo	
Responsabilidades:	
Nombre	m_usuario
Descripción	Constructor de la clase.
Nombre	Autenticar($usuario,$clave)
Descripción	Autentica a un usuario en el sistema.

Tabla 3.2.3.12: Modelo usuario.

3.3 Diseño de la BD.

Las bases de datos permiten el almacenamiento de información en maquinas accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información. Los cambios en datos no implican cambio en programas y viceversa, procurando un menor coste de mantenimiento. Reduce redundancia, mejora en la disponibilidad de datos.

A continuación se presenta una vista del modelo de datos.

CAPITULO III: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

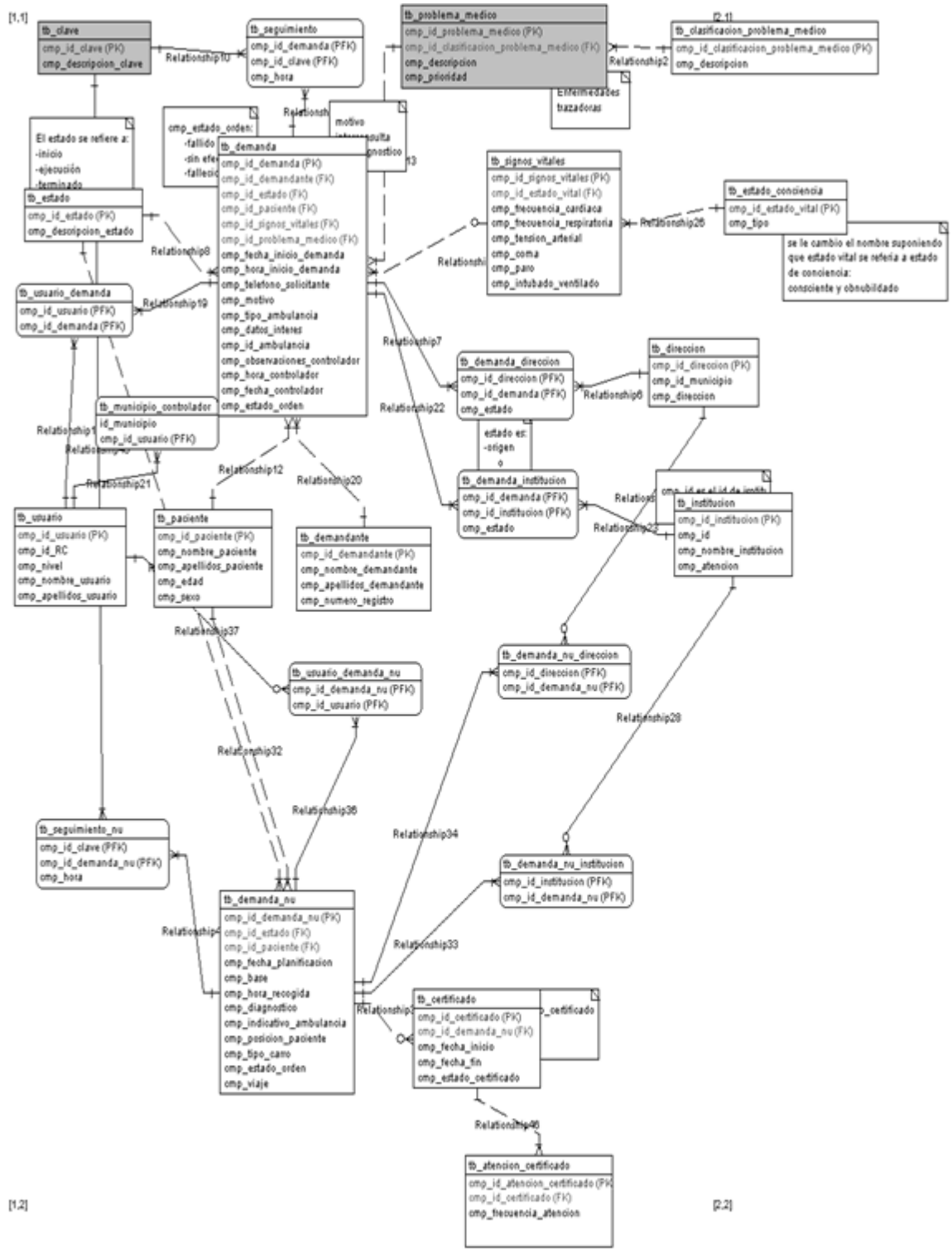


Figura 3.1: Modelo de datos

3.4 Descripción de las tablas de la base de datos.

Nombre: tb_demanda		
Descripción: Almacena los datos principales del paciente así como las decisiones de la enfermera coordinadora y el controlador.		
Atributo	**Tipo**	**Descripción**
cmp_id_demanda	Serial	Es la llave primaria de la tabla, la cual sirve para garantizar la unicidad en la Base de Datos y como es auto incremento manejados por la base de datos hace que la búsqueda sea más rápida.
cmp_id_demandante	Serial	Es llave extranjera de la tabla, esta garantiza que por cada demanda exista una persona que solicite el servicio.
cmp_id_estado	Serial	Es llave extranjera de la tabla, garantiza que cada demanda sepa el estado en que se encuentra (si ya fue atendida, si no ha sido atendida, o si se le está dando seguimiento).
cmp_id_paciente	Serial	Es llave extranjera de la tabla, garantiza que por cada demanda que se abra exista un paciente con todos sus datos.
cmp_id_signos_vitales	Serial	Es llave extranjera de la tabla, garantiza por cada demanda un grupo valoraciones médicas como: frecuencia cardiaca, frecuencia respiratoria, tensión arterial, coma, paro, intubado-ventilado.
cmp_id_problema_medico	Serial	Es llave extranjera de la tabla,

		garantiza que por cada demanda tenga un problema médico asociado.
cmp_fecha_inicio_demanda	Date	Fecha en que se abrió el caso de emergencia
cmp_hora_inicio_demanda	Time	Hora en que se abrió el caso de emergencia.
cmp_telefono_solicitante	Char(20)	Recoge el teléfono de la persona que solicita el servicio.
cmp_motivo	Varchar(20)	
cmp_tipo_ambulancia	Char(20)	La coordinadora asigna el tipo de ambulancia que requiere el caso
cmp_datos_interes	Varchar(300)	La coordinadora puede poner algunos datos significativos del caso.
cmp_id_ambulancia	Intiger	El controlador le asigna una ambulancia a la demanda quedando registrado el id en la base de datos.
cmp_observaciones_controlador	Varchar(300)	El controlador puede dar algunas observaciones del caso.
cmp_hora_controlador	Time	La hora cuando el controlador asigna la ambulancia.
cmp_fecha_controlador	Date	La fecha cuando el controlador asigna la ambulancia.
cmp_estado_orden	Char(20)	EL controlador especifica si la demanda fue sin efecto, fallida o tarjeta negra.

Tabla 3.4.1: tb_demanda

Nombre: tb_signos_vitales		
Descripción: Almacena un conjunto de valoraciones medicas la cual pertenece a una demanda.		
Atributo	**Tipo**	**Descripción**
cmp_id_signos_vitales	Serial	Es la llave primaria de la tabla, la cual sirve para garantizar la unicidad en la Base de Datos y como es auto incremento manejados por la base de datos hace que la búsqueda sea más rápida.
cmp_id_estado_vital		Es llave extranjera de la tabla, garantiza saber el estado de conciencia del paciente.
cmp_frecuencia_cardiaca	Intiger	Almacena la frecuencia cardiaca del paciente.
cmp_frecuencia_respiratoria	Intiger	Almacena la frecuencia respiratoria del paciente.
cmp_tension_arterial	Intiger	Almacena la tensión arterial del paciente.
cmp_coma	Intiger	Dice si el paciente está en coma.
cmp_paro	Intiger	Dice si el paciente está en paro.
cmp_intubado_ventilado	Char(1)	Dice si el paciente está intubado o ventilado.

Tabla 3.4.2: tb_signos_vitales.

Nombre: tb_estado_conciencia		
Descripción: Almacena el estado de conciencia del paciente en una demanda dada.		
Atributo	**Tipo**	**Descripción**
cmp_id_estado_vital	Serial	Llave primaria, la cual sirve para garantizar la unicidad en la Base de Datos y como es auto incremento manejados por la base de datos hace que la búsqueda sea más

		rápida.
cmp_tipo	Varchar(20)	Dice si está consiente o obnubilado.

Tabla 3.4.3: tb_estado_conciencia.

Nombre: tb_problema_medico		
Descripción: Almacena el problema médico que tiene el paciente dada una demanda, así como la prioridad que presenta.		
Atributo	**Tipo**	**Descripción**
cmp_id_problema_medico	Serial	Es llave primaria y es auto incremento.
cmp_id_clasificacion_problema_medico	Serial	Es llave extranjera y garantiza que por cada problema médico exista una clasificación.
cmp_descripcion	Varchar(20)	Almacena la descripción del problema médico que presenta el paciente.
cmp_prioridad	Intiger	Prioridad que presenta el problema médico

Tabla 3.4.4: tb_problema_medico.

Nombre: tb_demanda_direccion		
Descripción: Es una tabla que se generó por una relación de mucho a mucho entre demanda y dirección. Posibilita almacenar por cada demanda dos direcciones no institucionales como máximo.		
Atributo	**Tipo**	**Descripción**
cmp_id_direccion	Serial	Es llave primaria, almacena el id de la dirección de la demanda, puede existir dos id diferentes para una misma demanda y es auto incremento.
cmp_id_demanda	Serial	Es llave primaria, almacena el id de la demanda la cual puede aparecer dos veces con dos id dirección diferentes.
cmp_estado	Varchar(20)	Garantiza saber si la dirección de la

		demanda seleccionada es un origen o un destino.

Tabla 3.4.5: tb_demanda_direccion.

Nombre: tb_direccion		
Descripción: Almacena una dirección particular.		
Atributo	**Tipo**	**Descripción**
cmp_id_direccion	Serial	Es llave primaria y es auto incremento.
cmp_id_municipio	Varchar(20)	Almacena el id del municipio.
cmp_direccion	Varchar(100)	Almacena la descripción de la dirección.

Tabla 3.4.6: tb_direccion.

Nombre: tb_demanda_institucion		
Descripción: Es una tabla que se generó por una relación de mucho a mucho entre demanda e institución. Posibilita almacenar por cada demanda dos direcciones institucionales como máximo.		
Atributo	**Tipo**	**Descripción**
cmp_id_institucion	Serial	Es llave primaria, almacena el id de la dirección institucional de la demanda, puede existir dos id diferentes para una misma demanda y es auto incremento.
cmp_id_demanda	Serial	Es llave primaria, almacena el id de la demanda la cual puede aparecer dos veces con dos id institución diferentes.
cmp_estado	Varchar(20)	Garantiza saber si la dirección de la institución, de la demanda seleccionada es un origen o un destino.

Tabla 3.4.7: tb_demanda_institucion.

Nombre: tb_institucion		
Descripción: Almacena dirección de una institución.		
Atributo	**Tipo**	**Descripción**
cmp_id_institucion	Serial	Es llave primaria y es auto incremento.
cmp_id	Intiger	Almacena id del municipio donde se encuentra la institución.
cmp_nombre_institucion	Varchar(70)	Almacena el nombre de la institución.

Tabla 3.4.7: tb_institucion.

Nombre: tb_clave		
Descripción: Es un nomenclador que almacena las claves definidas por la que puede pasar una ambulancia (pero se toma como la demanda es quien cambia de claves).		
Atributo	**Tipo**	**Descripción**
cmp_id_clave	Serial	Llave primaria y es auto incremento.
cmp_descripcion_clave	Varchar(20)	Almacena tipos de claves definidas.

Tabla 3.4.8: tb_clave.

Nombre: tb_seguimiento		
Descripción: Es una tabla que se generó por la relación de muchos a muchos entre clave y demanda. Por cada demanda se tiene una cierta cantidad de claves.		
Atributo	**Tipo**	**Descripción**
cmp_id_demanda	Serial	Llave primaria, es el id de una demanda y es auto incremento.
cmp_id_clave	Serial	Llave primaria, representa al id de una clave.
cmp_hora	Time	Atributo que establece la hora en que una demanda entro en una clave determinada.

Tabla 3.4.9: tb_seguimiento.

Nombre: tb_estado		
Descripción: Almacena los diferentes estados por lo que pasa una demanda (inicio, ejecución y terminado).		
Atributo	**Tipo**	**Descripción**
cmp_id_estado	Serial	Llave primaria es auto incremento.
cmp_descripcion_estado	Varchar(20)	Tiene los tres estado posibles por los que puede pasar una demanda (inicio, ejecución y terminado).

Tabla 3.4.10: tb_estado.

Nombre: tb_usuario		
Descripción: Almacena la información imprescindible de un usuario del sistema.		
Atributo	**Tipo**	**Descripción**
cmp_id_usuario	Serial	Llave primaria y es auto incremento.
cmp_id_RC	Intiger	Almacena el id que tiene el usuario en el registro ciudadano.
cmp_nivel	Varchar(20)	Almacena el nivel que tiene el usuario.
cmp_nombre_usuario	Varchar(60)	Almacena el nombre del usuario.
cmp_apellidos_usuario	Varchar(60)	Almacena los apellidos del usuario.

Tabla 3.4.11: tb_usuario.

Nombre: tb_municipio_controlador		
Descripción: Tabla que identifica por cada controlador que exista el municipio al cual pertenece.		
Atributo	**Tipo**	**Descripción**
cmp_id_municipio_controlador	serial	Llave primaria y es auto incremento.
id_municipio	Intiger	Almacena el municipio del controlador.

cmp_id_usuario	Serial	Es el id del usuario al que se hace referencia.

Tabla 3.4.12: tb_municipio_controlador.

Nombre: tb_usuario_demanda		
Descripción: Es una tabla que se genera por la relación de mucho a mucho entre las tablas demanda y usuario. Una demanda puede atenderse por 2 usuarios (la enfermera coordinadora y el controlador).		
Atributo	**Tipo**	**Descripción**
cmp_id_usuario	Serial	Es llave primaria, dado una demanda relaciona los usuarios que la atendieron.
cmp_id_demanda	Serial	Es llave primaria, almacena el id de la demanda.

Tabla 3.4.13: tb_usuario_demanda.

Nombre: tb_demandante		
Descripción: Almacena los datos de la persona que solicita el servicio.		
Atributo	**Tipo**	**Descripción**
cmp_id_demandante	Serial	Es llave primaria y es auto incremento.
cmp_nombre_demandante	Char(20)	Almacena el nombre del demandante
cmp_apellidos_demandante	Char(20)	Almacena los apellidos del demandante
cmp_numero_registro	Intiger	Es un campo nulo, que puede entrarse datos o no, todo esta en dependencia de si el demandante es un profesional de la salud, si no lo es entonces se deja vacio, si lo es entonces se busca el número de registro en el Registro de Personal de Salud.

Tabla 3.4.14: tb_demandante.

Nombre: tb_paciente		
Descripción: Tabla que almacena los datos principales del paciente.		
Atributo	**Tipo**	**Descripción**
cmp_id_paciente	Serial	Es llave primaria y es auto incremento.
cmp_nombre_paciente	Varchar(20)	Almacena el nombre del paciente.
cmp_apellidos_paciente	Varchar(20)	Almacena los apellidos del paciente.
cmp_edad	Intiger	Almacena la edad del paciente.
cmp_sexo	Char(1)	Almacena el sexo del paciente.

Tabla 3.4.15: tb_paciente.

Nombre: tb_demanda_nu		
Descripción: Almacena los datos de los caso no urgente que se realiza en los centro coordinadores provinciales de emergencia médica.		
Atributo	**Tipo**	**Descripción**
cmp_id_demanda_nu	Serial	Llave primaria y es auto incremento.
cmp_id_estado	Serial	Es llave extranjera de la tabla, garantiza que cada demanda sepa el estado en que se encuentra (si ya fue atendida, si no ha sido atendida, o si se le está dando seguimiento).
cmp_id_paciente	Serial	Es llave extranjera de la tabla, garantiza que por cada demanda que se abra exista un paciente con todos sus datos.
cmp_fecha_planificacion	Date	Almacena las posibles planificaciones de los no urgencias
cmp_base	Intiger	Identifica el número de la base que va a atender el caso.

cmp_hora_recogida	Time	Señala la hora de recogida.
cmp_diagnostico	Varchar(150)	Es el diagnóstico del paciente.
cmp_indicativo_ambulancia	Varchar(20)	Almacena el indicativo de la ambulancia que atiende la demanda.
cmp_posicion_paciente	Char(1)	Almacena la posición que puede llevar el paciente en la ambulancia.
cmp_tipo_carro	Varchar(20)	El tipo de carro que va a atender el caso (Ambulancia o taxi).
cmp_estado_orden	Varchar(20)	Dice el estado de la demanda si fue (fallido, sin efecto, fallecido).
cmp_viaje	Intiger	Almacena el número del viaje para identificarlo de los demás viajes que se hagan.

Tabla 3.4.16: tb_demanda_nu.

Nombre: tb_seguimiento_nu		
Descripción: Es una tabla que se generó por la relación de muchos a muchos entre las tablas clave y tb_demanda no urgente. Por cada demanda no urgente hay una cierta cantidad de claves.		
Atributo	**Tipo**	**Descripción**
cmp_id_demanda_nu	Serial	Llave primaria, es el id de una demanda no urgente y es auto incremento.
cmp_id_clave	Serial	Llave primaria, representa al id de una clave.
cmp_hora	Time	Atributo que establece la hora en que una demanda entro en una clave determinada.

Tabla 3.4.17: tb_seguimiento_nu.

Nombre: tb_usuario_demanda_nu		
Descripción: Es una tabla que se genera por la relación de mucho a mucho entre la tabla demanda no urgente y usuario. Una demanda puede atenderse por 2 usuarios (la enfermera coordinadora y el controlador).		
Atributo	**Tipo**	**Descripción**
cmp_id_usuario	Serial	Es llave primaria, dado una demanda relaciona los usuarios que la atendieron.
cmp_id_demanda_nu	Serial	Es llave primaria, almacena el id de la demanda no urgente.

Tabla 3.4.18: tb_usuario_demanda_nu.

Nombre: tb_certificado		
Descripción: La demanda que se tramita puede tener un certificado por un período de tiempo al cual también se le da un seguimiento por el centro.		
Atributo	**Tipo**	**Descripción**
cmp_id_certificado	Serial	Llave primaria y es auto incremento.
cmp_id_demanda_nu	Serial	Es el id de la demanda no urgente, pues es una demanda no urgente con un certificado.
cmp_fecha_inicio	Date	Almacena la fecha de inicio del certificado
cmp_fecha_fin	Date	Almacena la fecha de vencimiento del certificado
cmp_estado_certificado	Char(20)	Dice si el certificado se encuentra activo o inactivo.

Tabla 3.4.19: tb_certificado.

Nombre: tb_atencion_certificado		
Descripción: Es una tabla donde brinda la información de cómo es que se atiende los certificados, las planificaciones según la frecuencia de atención con el especialista.		
Atributo	**Tipo**	**Descripción**

cmp_id_atencion_certificado	Serial	Llave primaria y es auto incremento.
cmp_id_certificado	Serial	Llave foránea que especifica que para un certificado contiene una planificación.
cmp_frecuencia_atencion	Varchar(20)	Es la planificación de la frecuencia con que debe atenderse el paciente.

Tabla 3.4.20: tb_atencion_certificado.

Nombre: tb_demanda_nu_direccion		
Descripción: Es una tabla que se generó por una relación de mucho a mucho entre demanda no urgente y dirección. Posibilita almacenar por cada demanda no urgente dos direcciones como máximo.		
Atributo	**Tipo**	**Descripción**
cmp_id_direccion	Serial	Es llave primaria, almacena el id de la dirección de la demanda no urgente, puede existir dos id diferentes para una misma demanda y es auto incremento.
cmp_id_demanda_nu	Serial	Es llave primaria, almacena el id de la demanda no urgente la cual puede aparecer dos veces con dos id dirección diferentes.

Tabla 3.4.21: tb_demanda_nu_direccion.

Nombre: tb_demanda_nu_institucion		
Descripción: Es una tabla que se generó por una relación de mucho a mucho entre demanda no urgente e institución. Posibilita almacenar por cada demanda no urgente dos direcciones institucionales como máximo.		
Atributo	**Tipo**	**Descripción**
cmp_id_institucion	Serial	Es llave primaria, almacena el id de la institución de la demanda no urgente, puede existir dos id diferentes para una misma demanda y es auto incremento.

cmp_id_demanda_nu	Serial	Es llave primaria, almacena el id de la demanda no urgente la cual puede aparecer dos veces con dos id dirección diferentes.

Tabla 3.4.22: tb_demanda_nu_institucion.

Nombre: tb_clasificacion_problema_medico		
Descripción: Es una tabla que guarda la clasificación de los problemas médicos. Esta tabla es un nomenclador. Los problemas médicos se clasifican en: - Aparato respiratorio. - Aparato cardiovascular. - Sistema Nervioso. - Abdomen y Pelvis. - Politrauma y traumatismos. - Piel y quemados. - Intoxicaciones aguda. - Genitourinario. - Pediatria. - Neonatologia. - Organos.		
Atributo	**Tipo**	**Descripción**
cmp_id_clasificacion_problema_medico	Serial	Es llave primaria, almacena el id de la tabla. Es auto incremento.
cmp_descripcion	Varchar(30)	Almacena la clasificación del problema médico.

Tabla 3.4.23: tb_clasificacion_problema_medico.

Nombre: tb_rol		
Descripción: Almacena los distintos roles que tiene la aplicación realizada.		
Atributo	**Tipo**	**Descripción**
cmp_id_rol	Serial	Es llave primaria, almacena el id de la tabla. Es auto incremento.
cmp_descripcion_rol	Varchar(20)	Campo que almacena los distintos tipos de roles que necesita la aplicación.

Tabla 3.4.24: tb_rol.

Nombre: tb_url		
Descripción: Almacena los distintos roles que tiene la aplicación realizada.		
Atributo	**Tipo**	**Descripción**
cmp_id_url	Serial	Es llave primaria, almacena el id de la tabla. Es auto incremento.
cmp_id_rol	serial	Es llave foránea, pertenece a la tabla rol, quiere decir que una dirección url pertenece a un rol especifico, se hace con el objetivo de especificar la entrada de usuarios a las paginas permitidas según su rol.
cmp_url	Varchar(100)	Almacena la dirección url a la que el rol puede acceder.

Tabla 3.4.25: tb_url.

En este capítulo se hizo un análisis de los componentes y módulos que son reutilizados así como las estrategias de integración, lo cual resultó de gran importancia ya que se economizó tiempo a la hora del desarrollo utilizando funcionalidades implementadas y probadas por otros desarrolladores. Se hizo una descripción de las clases más importantes como las clases controladoras, modelos y las vistas, así como una descripción de los métodos más importantes de cada una. Se muestra un diseño de la base de datos realizado en CASE Studio, y se describe cada tabla de la misma.

CONCLUSIONES

- Se realizó un estudio crítico a los modelos de análisis y diseño propuesto por los analistas.
- Se diseñó el Modelo de implementación y los artefactos que describen la base de datos.
- Se implementó el módulo Centro Coordinador Provincial de Emergencia Médica.
- El sistema permite consumir servicios de algunos registros pertenecientes al SISalud.
- El sistema permite consumir el servicio de disponibilidad de ambulancia del módulo Operaciones y le brinda información de las demandas.

RECOMENDACIONES

- Implementar las búsquedas en la Base de Datos con AJAX para lograr una mayor velocidad en las respuestas.
- Implementar métodos que permitan darle un orden lógico a la entrada de las claves.
- Implementar métodos que permitan visualizar dada una demanda, las claves por las que pasó, para poder dar una respuesta convincente en casos de quejas de la población, así como tomar medidas.
- Implementar una vía que minimice la dependencia con los servicios web del SISalud.

BIBLIOGRAFÍA

1. **Sosa Rosales, Dra. Maritza, Mojáiber de la Peña, Dr. Armando y Zacca Gonzáles, Dra. Grisel.** [En línea] agosto de 1999. [Citado el: 16 de febrero de 2008.] http://www.sld.cu/galerias/pdf/uvs/saludbucal/esbparte1.pdf.

2. **Álvarez Romírez, Dra. Ana Iris.** *Propuesta automatizada de control administrativo de la sección de transporte del Sistema Integrado de Urgencias Médicas.* 2007.

3. Professional Soft. *Professional Soft.* [En línea] [Citado el: 20 de febrero de 2008.] http://www.professional-soft.com/index.asp..

4. [En línea] [Citado el: 20 de octubre de 2007.] http://www.ambuwin.com/.

5. SEINCO. *SEINCO.* [En línea] [Citado el: 20 de febrero de 2008.] http://www.seinco.es/software/gestion-tts2000-extendido.asp#desc..

6. **Raul Rodas Hinostroza.** LinuxCentro.next. *LinuxCentro.next.* [En línea] [Citado el: 18 de noviembre de 2008.] http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.

7. Herramientas para desarrolladores : Cursos, manuales y guías de referencia : Cursos : Curso de JavaScript :. *Herramientas para desarrolladores : Cursos, manuales y guías de referencia : Cursos : Curso de JavaScript :.* [En línea] [Citado el: 16 de noviembre de 2007.] http://www.elcodigo.com/tutoriales/javascript/javascript1.html. 12.

8. Web experto. [En línea] 7 de junio de 2007. [Citado el: 18 de noviembre de 2007.] http://www.webexperto.com/articulos/articulo.php?cod=223.

9. El servidor de web Apache. *El servidor de web Apache.* [En línea] [Citado el: 16 de noviembre de 2007.] http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/c20.html.

10. Panorámica del sistema de gestión de base de datos MySQL. [En línea] [Citado el: 16 de febrero de 2008.] http://dev.mysql.com/doc/refman/5.0/es/what-is.html.

11. tufuncion. *tufuncion.* [En línea] [Citado el: 03 de junio de 2008.] http://www.tufuncion.com/codeigniter-php.

12. tufuncion. [En línea] [Citado el: 16 de febrero de 2008.] http://www.tufuncion.com/prototype.

13. Sentido Web. [En línea] [Citado el: 16 de febrero de 2008.] http://sentidoweb.com/2006/07/28/fpdf-libreria-php-para-crear-pdfs.php.

14. Services network. [En línea] [Citado el: 16 de junio de 2008.] http://www.neurored.com/empresa.asp?sec=45&nmsec=Los%20Servicios%20On-Demand%20y%20la%20Arquitectura%20SOA.

15. **Alfredo Barón Ocampo.** El Papel del DWH en una Arquitectura Orientada a Servicios. [En línea] [Citado el:] http://www2.inegi.gob.mx/dw/Ponencias/6%20de%20junio/INEGI%206Jun2008%20-%20Sun%20-%20DWH%20y%20SOA.ppt.

16. **Maldonado, Daniel M.** El CoDiGo K. [En línea] [Citado el: 09 de junio de 2008.] http://elcodigok.blogspot.com/2007/09/arquitectura-de-programacin-en-3-capas.html.

17. http://bibliodoc.uci.cu. *http://bibliodoc.uci.cu.* [En línea] [Citado el: 02 de 06 de 2008.] http://bibliodoc.uci.cu/pdf/reg00061.pdf.

18. SERVLETS Y PATRÓN MVC. *SERVLETS Y PATRÓN MVC.* [En línea] [Citado el: 12 de marzo de 2008.] http://www.dei.inf.uc3m.es/docencia/p_s_ciclo/pa4/practicas/mvc.pdf.

19. Que es Apache? [En línea] [Citado el: 16 de noviembre de 2007.] http://www.facilnet.net/matriz/web2/apache.html.

20. **Rivera, Volkan.** Tecnología y negocios. *Tecnología y negocios.* [En línea] 2 de agosto de 2007. [Citado el: 16 de novienbre de 2007.] http://www.volkanrivera.com/esp/?p=24.

21. **P.Letelier.** https://pid.dsic.upv.es. [En línea] [Citado el: 18 de noviembre de 2007.]

22. The world's most popular open source database. *The world's most popular open source database.* [En línea] [Citado el: 18 de noviembre de 2007.] http://dev.mysql.com/doc/refman/5.0/es/features.html.

23. The world's most popular open source database. *The world's most popular open source database.* [En línea] [Citado el: 18 de noviembre de 2007.] http://dev.mysql.com/doc/refman/5.0/es/what-is.html.

24. **Morán, Miguel Ángel.** Software Guru. *Software Guru.* [En línea] julio-agosto de 2007. [Citado el: 16 de noviembre de 2007.] http://www.sg.com.mx/content/view/340.

25. FPDF library. *FPDF library.* [En línea] [Citado el: 16 de noviembre de 2007.] http://www.fpdf.org/.

26. **Danysoft, Equipo.** Modelado de bases de datos. *Modelado de bases de datos.* [En línea] 25 de septiembre de 2006. [Citado el: 16 de noviembre de 2008.] http://www.danysoft.info/free/model2.pdf.

27. **Catalani, Exequiel.** ARQUITECTURA Modelo/Vista/Controlador. *ARQUITECTURA Modelo/Vista/Controlador.* [En línea] 20 de agosto de 2007. [Citado el: 16 de noviembre de 2007.] http://exequielc.wordpress.com/2007/08/20/arquitectura-modelovistacontrolador/.

28. **Rojas, Jose Fabricio.** Arquitectura en N Capas. [En línea] [Citado el: 16 de marzo de 2008.] http://devsoftx.wordpress.com/2008/04/24/arquitectura-en-n-capas/.

29. Arquitectura de tres capas. [En línea] [Citado el: 16 de marzo de 2008.] http://paolaqb.iespana.es/bd3c02.pps.

30. **Fanjul, Alejandro.** rquitectura de tres Capas. [En línea] [Citado el: 16 de marzo de 2008.] http://www.mhproject.org/media/blogs/mhpenlaces/Interno/Presentaciones/ATS-Interactiva/Arquitectura%20Tres%20Capas.ppt.

31. Generator FD. [En línea] [Citado el: 16 de marzo de 2008.] http://www.generatorfd.com/Arquitectura.aspx.

32. **Garrett, Jesse James.** adaptive path. *Ajax: A New Approach to Web Applications.* [En línea] [Citado el: 12 de marzo de 2008.] http://www.adaptivepath.com/ideas/essays/archives/000385.php.

33. Patrón "Modelo-Vista-Controlador". [En línea] [Citado el: 16 de febrero de 2008.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

34. El servidor de web Apache. [En línea] [Citado el: 16 de noviembre de 2007.] http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html.

35. LogicLibrary. [En línea] [Citado el: 09 de junio de 2008.] http://www.logiclibrary.com/.

36. desarrolloweb.com. [En línea] [Citado el: 9 de junio de 2008.] http://www.desarrolloweb.com/articulos/1622.php.

37. Desarrollo.net. *Arquitectura de Aplicaciones de 3 capas.* [En línea] http://www.dotnetjunkies.com/WebLog/desarrollonet/archive/2004/06/17/16855.aspx.

38. **Mónica Pinto Alarcón.** Proceso de Desarrollo de Aplicaciones Basadas en Componentes y Aspectos con MDA. [En línea] [Citado el: 16 de marzo de 2008.] http://www.ewh.ieee.org/reg/9/etrans/Marzo2005/paper55.pdf.

39. **Terreros, Julio Casal.** Desarrollo de Software basado en Componentes. [En línea] [Citado el: 16 de marzo de 2008.] http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3985/default.aspx.

40. raona software engineering. [En línea] [Citado el: 16 de junio de 2008.] http://www.raona.com/Not%C3%ADcies/Opini%C3%B3/SOA/tabid/150/Default.aspx.

41. IBM. *SOA (Arquitectura Orientada a Servicios).* [En línea] [Citado el: 16 de junio de 2008.] http://www-306.ibm.com/software/es/soa/launch/index.html.

42. clikear.com. *Desarrollo Orientado a Objetos con UML.* [En línea] [Citado el: 16 de febrero de 2008.] http://www.clikear.com/manuales/uml/.

43. LogicLibrary. [En línea] [Citado el: 09 de junio de 2008.] http://www.logiclibrary.com/.

44. LIBERIC. [En línea] [Citado el: 16 de junio de 2008.] http://www.liberic.com/liberic/portal?node_action=view&node=8c0d82c0023e9d6a44f627881dad1ce5.

45. mkm. [En línea] [Citado el: 16 de junio de 2008.] http://www.mkm-pi.com/mkmpi.php?article1220.

46. LIBERIC. [En línea] [Citado el: 16 de junio de 2008.] http://www.liberic.com/liberic/portal?node_action=view&node=8c0d82c0023e9d6a44f627881dad1ce5.

47. mkm. [En línea] [Citado el: 16 de junio de 2008.] http://www.mkm-pi.com/mkmpi.php?article1220.

DOCUMENTOS CITADOS

[1] **Sosa Rosales, Dra. Maritza, Mojáiber de la Peña, Dr. Armando and Zacca Gonzáles, Dra. Grisel.** [Online] agosto 1999. [Cited: febrero 16, 2008.] http://www.sld.cu/galerias/pdf/uvs/saludbucal/esbparte1.pdf

[2] **Álvarez Romírez, Dra. Ana Iris.** *Propuesta automatizada de control administrativo de la sección de transporte del Sistema Integrado de Urgencias Médicas.* 2007.

[3] Ídem a la Referencia 2.

[4] Ídem a la Referencia 2.

[5] Ídem a la Referencia 2.

[6] Ídem a la Referencia 2.

[7] Ídem a la Referencia 2.

[8] Ídem a la Referencia 2.

[9] Ídem a la Referencia 2.

[10] Ídem a la Referencia 2.

[11] Professional Soft. *Professional Soft.* [Online] [Cited: febrero 20, 2008.] http://www.professional-soft.com/index.asp.

[12] Ídem a la Referencia 11.

[13] Ídem a la Referencia 11.

[14] Ídem a la Referencia 11.

[15] Ídem a la Referencia 11.

[16] Ídem a la Referencia 11.

[17] [Online] [Cited: octubre 20, 2007.] http://www.ambuwin.com/.

[18] Ídem a la Referencia 17

[19] Ídem a la Referencia 17

[20] SEINCO. *SEINCO.* [Online] [Cited: febrero 20, 2008.] http://www.seinco.es/software/gestion-tts2000-extendido.asp#desc.

[21] Ídem a la Referencia 20.

[22] Ídem a la Referencia 20.

[23] **Raul Rodas Hinostroza.** LinuxCentro.next. *LinuxCentro.next.* [Online] [Cited: noviembre 18, 2008.] http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.

[24] Ídem a la Referencia 23.

[25] Ídem a la Referencia 23.

[26] Ídem a la Referencia 23.

[27] Ídem a la Referencia 23.

[28] Ídem a la Referencia 23.

[29] Herramientas para desarrolladores : Cursos, manuales y guías de referencia : Cursos : Curso de JavaScript :. *Herramientas para desarrolladores : Cursos, manuales y guías de referencia : Cursos : Curso de JavaScript :.* [Online] [Cited: noviembre 16, 2007.] http://www.elcodigo.com/tutoriales/javascript/javascript1.html. 12.

[30] Ídem a la Referencia 29.

[31] Ídem a la Referencia 29.

[32] Web experto. [Online] junio 7, 2007. [Cited: noviembre 18, 2007.] http://www.webexperto.com/articulos/articulo.php?cod=223.

[33] Ídem a la Referencia 32.

[34] Ídem a la Referencia 32.

[35] Ídem a la Referencia 32.

[36] Ídem a la Referencia 32.

[37] Ídem a la Referencia 32.

[38] Ídem a la Referencia 32.

[39] Ídem a la Referencia 32.

[40] Ídem a la Referencia 32.

[41] Ídem a la Referencia 32.

[42] El servidor de web Apache. *El servidor de web Apache.* [Online] [Cited: noviembre 16, 2007.] http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/c20.html.

[43] Ídem a la Referencia 42.

[44] Panorámica del sistema de gestión de base de datos MySQL. [Online] [Cited: febrero 16, 2008.] http://dev.mysql.com/doc/refman/5.0/es/what-is.html.

[45] Ídem a la Referencia 44.

[46] Ídem a la Referencia 44.

[47] Ídem a la Referencia 44.

[48] tufuncion. *tufuncion.* [Online] [Cited: junio 03, 2008.] http://www.tufuncion.com/codeigniter-php.

[49] Ídem a la Referencia 48.

[50] tufuncion. [Online] [Cited: febrero 16, 2008.] http://www.tufuncion.com/prototype.

[51] Ídem a la Referencia 50.

[52] Sentido Web. [Online] [Cited: febrero 16, 2008.] http://sentidoweb.com/2006/07/28/fpdf-libreria-php-para-crear-pdfs.php.

[53] Ídem a la Referencia 52.

[54] **Maldonado, Daniel M.** El CoDiGo K. [Online] [Cited: junio 09, 2008.] http://elcodigok.blogspot.com/2007/09/arquitectura-de-programacin-en-3-capas.html.

[55] Ídem a la Referencia 54.

[56] http://bibliodoc.uci.cu. *http://bibliodoc.uci.cu.* [Online] [Cited: 06 02, 2008.] http://bibliodoc.uci.cu/pdf/reg00061.pdf.

[57] Ídem a la Referencia 56.

[58] Ídem a la Referencia 56.

[59] Ídem a la Referencia 56.

[60] Ídem a la Referencia 56.

[61] Ídem a la Referencia 56.

[62] Ídem a la Referencia 56.

[63] Ídem a la Referencia 56.

[64] Ídem a la Referencia 56.

[65] Ídem a la Referencia 56.

[66] Ídem a la Referencia 56.

[67] Ídem a la Referencia 56.

[68] Ídem a la Referencia 56.

[69] Ídem a la Referencia 56.

[70] Ídem a la Referencia 56.

[71] Ídem a la Referencia 56.

[72] SERVLETS Y PATRÓN MVC. *SERVLETS Y PATRÓN MVC.* [Online] [Cited: marzo 12, 2008.] http://www.dei.inf.uc3m.es/docencia/p_s_ciclo/pa4/practicas/mvc.pdf.

[73] Ídem a la Referencia 72.

[74] Ídem a la Referencia 72.

[75] Ídem a la Referencia 72.

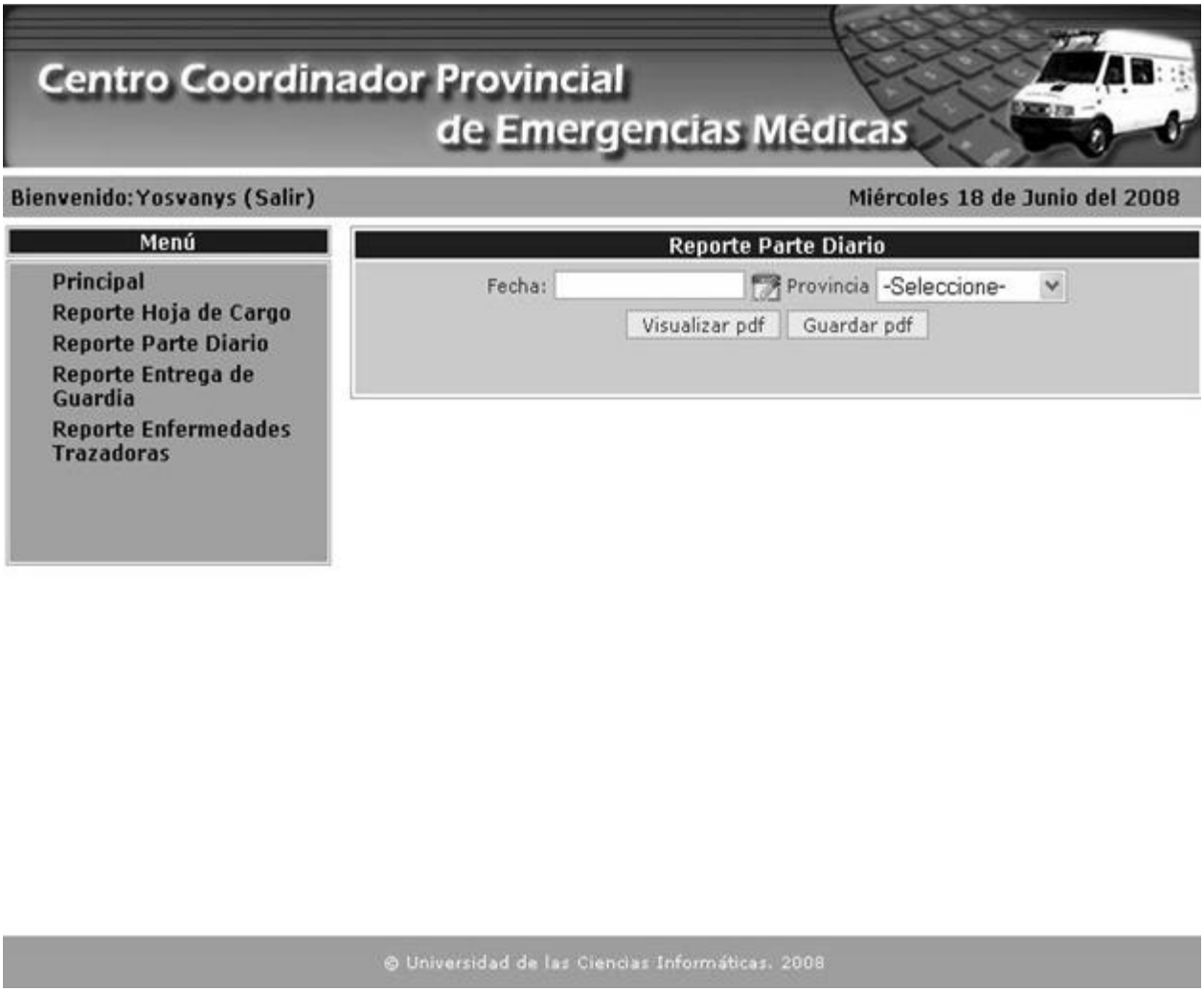
ANEXOS

Anexo 1: Página de autenticación del módulo Centro Coordinador Provincial de Emergencia Médica.

Anexo 2: Página de bienvenida del módulo Centro Coordinador Provincial de Emergencia Médica.

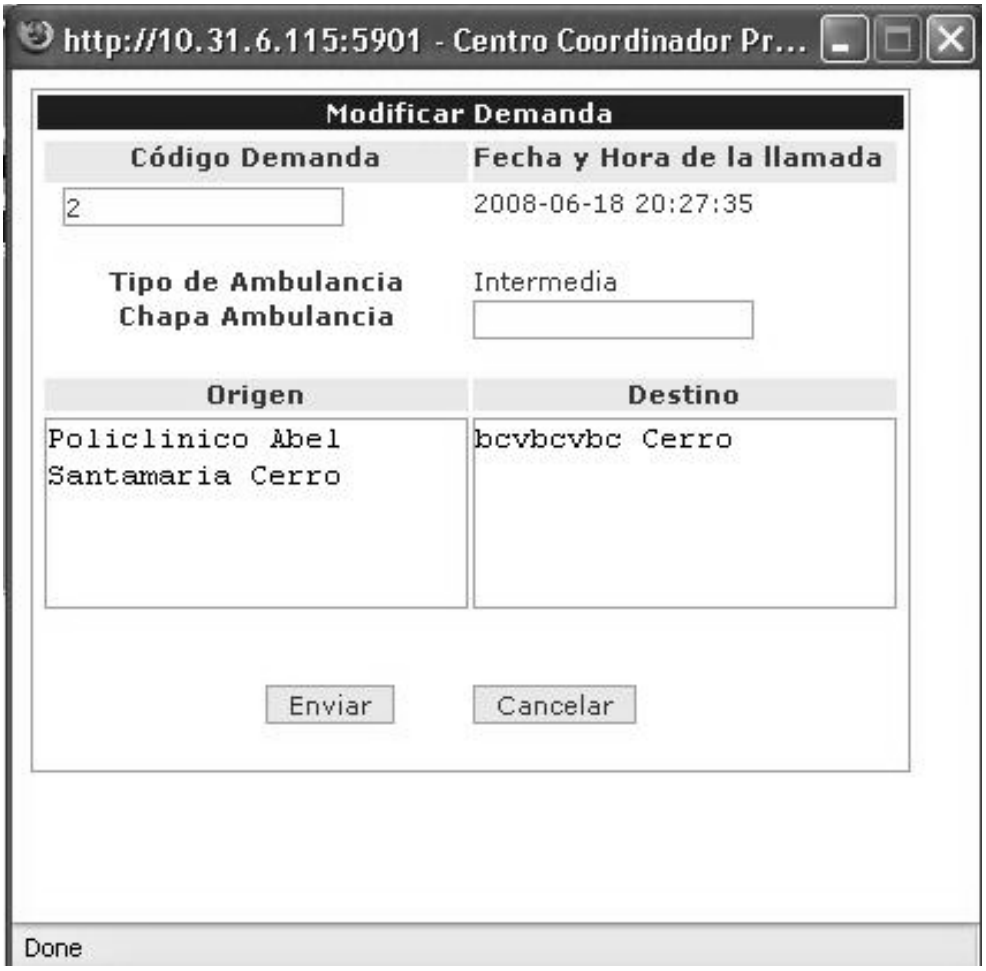
Anexo 3: Página de la coordinadora donde inserta los datos principales de la demanda.

Anexo 4: Página del jefe de guardia, donde tiene acceso a los reportes.



Anexo 5: Página del controlador donde recibe las demandas para darle seguimiento.

Anexo 6: Página del controlador donde asigna el tipo de ambulancia que necesita la demanda .



Anexo 7: Página del controlador donde le da seguimiento a la demanda asignando las claves por la cual pasa la ambulancia.

GLOSARIO DE TÉRMINOS

Aplicación (Sistema): Sistema que ofrece a un usuario final un conjunto coherente de casos de uso.

Aplicación Web: Es una aplicación informática que los usuarios utilizan accediendo a un servidor Web a través de un navegador o browser.

Ambulancia

Es un vehículo que los servicios médicos utilizan para responder las llamadas de emergencia.

Ambulancias grandes o de emergencia médica

Son las Ambulancias Intensivas o de Apoyo Vital Avanzado, estas son Ambulancias ubicadas en las Bases de Ambulancia de cada Municipio y al servicio de este, y Estratégicamente existirá alguna desplegada en un Área Intensiva Municipal, Policlínico de Urgencias, Hospitales y en la Unidad Neonatal para las Ambulancias Intensivas Neonatales, el objetivo es disminuir tiempo de atención emergente.

En todos los casos la regulación de las Emergencias se realizara por los Puestos de Dirección Provinciales de Ambulancias para cada Base de Ambulancias Municipal, y nunca podrán actuar sin esta coordinación. El Puesto de Dirección de Ambulancias de cada Provincia tomará decisiones para movimiento dentro de la provincia, y por situaciones excepcionales autorizara el servicio y movimiento de un Municipio a otro si fuera necesario, así como a una provincia colindante, y para otras provincias debe autorizarse por el Puesto de Dirección de La Emergencia Nacional.

Ambulancias de urgencias

Son para el traslado de pacientes urgentes con requerimientos de ambulancias que no constituyen una Emergencia Médica, excepto cuando por situaciones especiales se autoriza por el Puesto de Dirección de Ambulancias Provincial.

Estas son Ambulancias ubicadas en las Bases de Ambulancia de cada Municipio y al servicio de este, y Estratégicamente existirá alguna desplegada en un Área Intensiva Municipal, Policlínico de Urgencias, Hospitales u otra área distante, el objetivo es disminuir tiempo de atención para la urgencia.

En todos los casos la regulación de las Urgencias se realizara por los Puestos de Dirección Provinciales de Ambulancias para cada Base de Ambulancias Municipal, y nunca podrán actuar sin esta coordinación. El Puesto de Dirección de Ambulancias de cada Provincia tomará decisiones para movimiento dentro de la provincia, y por situaciones excepcionales autorizara el servicio y movimiento de un Municipio a otro si fuera necesario, así como a una provincia colindante.

Ambulancias de transporte sanitario no urgente

Son para el traslado de pacientes con patologías que no son Urgencias y que por sus características específicas necesitan ser transportados en Ambulancia. Están ubicadas en las Bases de Ambulancia de cada Municipio, y podrán ser usadas en función de urgencias menos importantes que requieran traslados.

Controlador

Persona que se encarga de la asignación de ambulancias y darle seguimiento a la demanda en el Centro Coordinador Provincial de Emergencia Médica.

Coordinadora

Persona que se encarga de atender las llamadas y crear la demanda en el Centro Coordinador Provincial de Emergencia Médica.

Demanda

Es una planilla donde se llenan los datos más importantes del caso a atender.

Emergencia hospitalaria

Cuerpos de guardia, terapias intensivas, intermedias, unidades de quemados, que están situados dentro de las unidades hospitalarias.

Consultas que se realizan entre profesionales. Por lo general se realiza desde un médico general a un especialista, o un especialista determinado a otro de una especialidad diferente, para comprobar diagnósticos o decidir en casos dudosos.

Emergencia móvil

Es toda la red de vehículos destinados a socorrer en caso de emergencias como ambulancias o cualquier otro tipo de transporte sanitario.

Nivel de atención primaria

Se refiere a la atención médica prestada en el nivel primario de salud que incluye consultorios médicos de la familia, puestos médicos de urgencia, policlínicos, salas de observación de policlínicos, hogares y terapias municipales.

Nivel de atención secundaria

Se refiere a la atención médica prestada en el nivel secundario de salud que incluyen hospitales generales, pediátricos y ginecobstétricos.

Nivel de atención terciaria

Se refiere a la atención médica prestada en el nivel terciario de salud que incluyen los institutos especializados, como pueden ser Instituto Cardiovascular, Instituto de Nefrología, Instituto de Angióloga, etcétera, que aunque se encuentren insertado dentro de los hospitales funcionan como ente aparte.

Nivel de emergencias y urgencias

Es un nivel intermedio para las urgencias y emergencia entre el nivel de atención primaria y secundaria o terciaria. Es un sistema asistencial que comienza en el escenario y continúa con el transporte asistido de urgencia al hospital apropiado para la patología del paciente, proporcionando sostén vital básico y avanzado continuo en el camino y-o en el servicio de urgencia de una primera institución, hasta poder poner en marcha un proceso organizado en una UCI (Unidad de Cuidados Intensivos).

No urgencias ambulanciables

En este grupo, entran todas las situaciones de transporte de pacientes ambulanciables que necesitan un traslado hacia una institución médica, o desde una institución médica a su hogar, pero que se

puede realizar en el transcurso del día sin que represente este traslado peligro para su vida o secuelas posteriores. Ejemplo altas de hospital, consultas médicas, traslado entre hospitales, turnos de exámenes médicos especiales, y otros. En muchas ocasiones se incluyen en este grupo las urgencias sociales.

Número de Indicativo

Número con carácter de código que se asigna a cada ambulancia, que es único y coincide con el número de localización por planta radial. Es independiente del número de la chapa que porte.

Pacientes ambulanciables

Se llama así a los pacientes que por sus patologías o secuelas no puedan deambular o ser trasladados sentados en un transporte convencional.

Primera prioridad

Urgencias con afectación vital o peligro vital potencial que generan un peligro vital inmediato. Requieren de atención inmediata.

Red de urgencia primaria

Es todo un sistema establecido en el país donde se atienden las urgencias fuera de los hospitales o institutos, como cuerpos de guardias de policlínicos, salas de observación, terapias intensivas municipales.

Segunda prioridad

Urgencia verdadera con requerimientos hospitalarios: paciente con enfermedad o lesiones que de no atenderse en un tiempo mediato puede generar complicaciones fatales.

Servicios web

Es un conjunto 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.

Soap

Simple Object Access Protocol es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. También es un conjunto de normas que define como deben ser las solicitudes y respuestas entre un usuario Web y un Servicio Web.

Tercera prioridad

Urgencia banal o sin riesgos: pacientes que requieren ser vistos de urgencia pero sin peligro vital mediato o tardío.

Test Diagnósticos

Término utilizado para referirse a todo tipo de pruebas cuyo objetivo es efectuar diagnóstico. Pueden ser de laboratorio (hematológicas, químicas, serológicas), radiográficas (rayos x, ultrasonidos, Tomografías, Resonancias Magnéticas) broncoscopías.

Transporte Sanitario

Es aquel que se realiza para el desplazamiento de personas enfermas, accidentadas o por otra razón sanitaria, en vehículos especialmente acondicionados al efecto.

UDDI (Universal Description Discovery and Integration): es un registro que permite al proveedor de un Servicio Web, anunciarlo y describirlo para que otros usuarios lo puedan encontrar y hacer uso de él.

WSDL (Web Services Description Language): describe pormenorizadamente la funcionalidad de un Servicio Web y los protocolos necesarios, tales como SOA, para interactuar con el mismo.

XML (Extensible Markup Language): es un estándar para definir los nombres y propiedades de los ítems de datos que se intercambian entre el servicio solicitante y el servicio proveedor.