Universidad de las Ciencias Informáticas

Facultad 7

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Implementación del Registro de Operaciones del Sistema Informatizado para el Centro Nacional de Urgencias Médicas.

Autor: Dennys Lázaro Hernández Quintana

Tutor: Ing. Jose Alejandro Segura Roque

Ing. Maidelys Pulido Morera

Ciudad de La Habana, Junio de 2008.

"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 7 de La Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los 27 días del mes de Junio del año 2008.

Tutor: Ing. Maidelys Pulido Morera

Tutor: Ing. Jose Alejandro Segura Roque

Autor: Dennys Lázaro Hernández Quintana

DATOS DE CONTACTO

Ing. Jose Alejandro Segura Roque: Correo electrónico: jsegura@uci.cu

Graduado en la especialidad de Ingeniería en Ciencias Informáticas en el año 2007. Imparte la asignatura de Práctica Profesional. Ha presentado ponencias y trabajos en eventos científicos obteniendo diferentes reconocimientos y premios. Es jefe del Área Temática Sistemas Especializados en la Facultad 7 de la Universidad de las Ciencias Informáticas.

Ing. Maidelys Pulido Morera: Correo electrónico: mpulido@uci.cu

Profesor recién graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI). Imparte las asignaturas Ingeniería de Software I y II en la Facultad 7. Ha presentado ponencias y trabajos en eventos científicos obteniendo diferentes reconocimientos y premios. Responsable del Grupo de Trabajo de la Comisión de Carrera y del perfil Informática para la Salud de la Facultad 7.

AGRADECIMIENTOS

"El agradecimiento es la memoria del corazón"

A La Revolución y a Fidel por ser el máximo creador de esta nuestra casa de altos estudios.

A mis padres, por el apoyo incondicional que me han dado durante toda mi vida.

A La Universidad y su colectivo de profesores que me han formado durante estos cinco años.

A todos mis compañeros, que de una forma u otra han contribuida en la realización de esta tesis. En especial a mis amigos Yanosky y Daymara quienes me ayudaron mucho en la realización de la misma.

A mis tutores por la ayuda brindada en el desarrollo de esta tesis.

Dedico este trabajo a:

A mi madre, por ser lo más grande que tengo en este mundo. Mi fuente de inspiración diaria para ser cada vez mejor, esa luz que me ilumina a seguir adelante.

A mi familia, por su dedicación y apoyo en todo momento. Especialmente a mi padre y mi hermana que siempre han sido para mi ejemplo a seguir.

A todos mis compañeros y amigos que en todo este tiempo me han acompañado y apoyado en todas mis decisiones.

RESUMEN

En la actualidad, la gestión de la información y de los servicios que prestan las Bases de Ambulancias de Cuba, se realiza de forma manual y archivando la información en formato duro (papel). Esto provoca demora en los servicios, pérdida de la información y que los reportes estadísticos del todo confiable. Para darle solución a esta problemática, se desarrolló el presente trabajo que tiene como objetivo implementar una aplicación Web que facilite la gestión de la información relacionada con las operaciones de ambulancias en las bases de Cuba.

Para desarrollar el sistema se utilizaron como lenguaje de programación del lado del servidor, PHP, y del lado del cliente, JavaScript. Además la metodología de desarrollo RUP, con UML para preparar todos los artefactos, siendo elaborados en la herramienta de modelado Visual Paradigm. También, se utilizaron otras herramientas como Zend Studio y Dreamweaver, la técnica de desarrollo AJAX y MySQL como gestor de base de datos.

Como resultado de este trabajo se obtuvo una aplicación Web capaz de gestionar la información y con su puesta en marcha permitirá cambiar la situación existente en las Bases de Ambulancias. Logrando una mayor confidencialidad, integridad y disponibilidad de la información manejada. Así como mejoras en los servicios, disminución de los riesgos de pérdida de datos y mejoras en los reportes estadísticos. Además, se brindará información básica a otros componentes del Sistema Informatizado para el SIUM.

Índice.

INTRODUCCIÓN

A lo largo de la historia, el hombre siempre ha desarrollado instrumentos y herramientas que lo ayudan a mejorar y perfeccionar sus actividades productivas, influyendo en su modo de vida. A mediados del siglo XX el desarrollo científico y tecnológico hace posible que una nueva rama, la informática, comience un ascendente camino que llega hasta la actualidad, donde se ha generalizado en todos los sectores de la sociedad.

En Cuba se trabaja intensamente con el objetivo de utilizar las Tecnologías de la Información y las Comunicaciones (TIC) para apoyar la salud pública. Las acciones que se han emprendido en este sentido parten de reconocer la importancia crucial de la revolución científico-técnica que se vive. Pero sobre todo hacen énfasis en la necesidad de priorizar el factor humano y adecuar estos avances a los problemas reales del país. La sociedad cubana continuamente se nutre de los nuevos avances en las TIC.

La Revolución lleva a cabo un fuerte programa para desarrollar la infraestructura tecnológica en el sector de la salud, con el objetivo de facilitar el aprendizaje y el intercambio de información de alta calidad científica y profesional. Permitiendo dar total apoyo a las especialidades donde se desenvuelven los técnicos y profesionales de la salud.

Se han diseñado estrategias de bajo costo para hacer uso de la información y las comunicaciones como instrumentos y ponerlas a disposición del avance socialista. Se impone trabajar con mayor capacidad de investigación y desarrollo en estas áreas del conocimiento y adaptarlas a las condiciones de la sociedad donde existen potencialidades, fruto de su adecuada política educacional.

En Cuba se está llevando a cabo un proyecto de informatización de la sociedad, comprometidos con este amplio programa de informatización participan ampliamente varios sectores, todos con un objetivo común: promover la importancia de la informatización en la creación de una nueva sociedad para mejorar los servicios al ciudadano, incorporar información para el provecho de todas las esferas de desarrollo del país, educar y preparar las nuevas generaciones universalizando la cultura científica. Entre los sectores priorizados para la informatización están la educación y la salud, que son dos pilares importantes de la sociedad cubana. En el proceso de informatización social el MINSAP no está ajeno al desarrollo de las TIC, para realizar de forma óptima todos los servicios que le brinda a la población.

En 1997 con el objetivo de garantizar en la atención primaria de salud, la asistencia de urgencia mediante servicios de emergencia médica móvil, en un esfuerzo por reducir los índices de mortalidad se consolidó el Centro Nacional de Urgencias Médicas (SIUM).

El SIUM 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 emergencia, mediante un proceso de evaluación y decisión médica, a través de los diferentes eslabones del SNS.

El SIUM surge como una entidad presupuestada, con un funcionamiento que desde el punto de vista estructural, está organizada por bases, cada una de las cuales tiene muy bien definida su área de responsabilidad, según el territorio que les corresponda. El objetivo es acercar a la población lo más posible, a este esencial servicio, el cual desde el punto de vista humano, en la agilidad, rapidez y eficiencia, juega un papel determinante en la salud del pueblo y en su bienestar general.

Los subsistemas que se enmarcan dentro del SIUM son los programas de socorrismo, la red de urgencia primaria, la emergencia móvil y hospitalaria, y las terapias intensivas e intermedias, cada una de ellas con sus subsistemas docentes.

En el proceso de atención a la urgencia médica en toda situación que requiera atención inmediata ejerce un esencial papel las Bases de Ambulancias. Dichos centros tienen como premisa atender al paciente óptimamente con un mínimo tiempo de respuesta hacia la demanda de urgencia.

Con tres tipos de ambulancias cuenta el SIUM: las intensivas, con infraestructura de última tecnología y un médico, una enfermera y un paramédico; las intermedias, más convencionales con enfermera y paramédico, y las básicas, utilizadas para el traslado de pacientes que no pueden hacer uso del transporte urbano por sus afecciones.

Las Bases de Ambulancias constituyen centros donde se encuentran ambulancias destinadas al traslado de enfermos y heridos que requieren actuación médica inmediata. Su principal objetivo dentro del área de atención a la urgencia es procesar el flujo de demanda de emergencia o urgencia médica que comienza con una llamada telefónica de cualquier ciudadano o Institución de Salud al Centro Coordinador de Emergencia Médica.

Se realiza la solicitud de una ambulancia por parte del Centro Coordinador de Emergencia Médica a la Base de Ambulancia del territorio correspondiente a la petición donde se le trasmite la información de la demanda al expedidor en la Base de Ambulancia, dándole salida a una ambulancia para atender la solicitud firmando una hoja de ruta que contiene el origen y destino de la urgencia. En este proceso hay que dejar plasmado en una historia clínica ambulatoria del paciente; la actividad del control del trabajo médico de la ambulancia que primeramente se llena con los datos de la solicitud de demanda de urgencia médica.

En las Bases de Ambulancias se realiza un control diario de las operaciones con el objetivo de garantizar una disponibilidad técnica eficiente para cada ambulancia y garantizar un mayor coeficiente de disponibilidad técnica en la Base, contando así con mayor número de ambulancias disponibles ante cualquier situación de urgencia médica.

Diariamente, la Base de Ambulancia debe mantener actualizado al Centro Coordinador de su coeficiente de disponibilidad técnica, se deben controlar los servicios que se brindan (si son de emergencias o urgencias médicas). Así como del consumo de combustible, kilómetros recorridos por la ambulancia, estado de los neumáticos, de las baterías, explotación y mantenimiento de las ambulancias. Debe existir un análisis estadístico de su accidentabilidad a efectos de tomar medidas proactivas sobre seguridad vial y selección de tipo de vehículos. Los informes sobre las ambulancias en cada base deben enviarse al Centro de Operaciones correspondiente.

El SIUM en cada provincia, se encuentra en un pleno proceso de renovación en su infraestructura que dará un giro al tratamiento de las urgencias y emergencias en Cuba y en este cambio juega un papel fundamental la informatización del mismo ya que la información referente a la gestión de operaciones se realiza de forma manual.

Actualmente, el país enfrenta la informatización de la sociedad apoyado en instituciones como la Universidad de las Ciencias Informáticas (UCI). Esta le ha asignado a la Facultad 7 algunas tareas encaminadas al logro de ese objetivo. En ella, la producción se encuentra dividida en áreas temáticas, una de la cuales es Sistemas Especializados, cuya tarea es desarrollar sistemas que permitan la gestión de la información en los diferentes sectores de la salud, entre los que se encuentran las Bases de Ambulancias con los servicios que la conforman.

Ante la situación antes planteada surge lo que constituye el **Problema a Resolver** en esta investigación.

¿Cómo facilitar la gestión de la información relacionada con las operaciones de ambulancias en las bases de Cuba?

Como **Objeto de Estudio** se ha identificado el Proceso de gestión de la información del Centro Nacional de Urgencias Médicas y como **Campo de Acción** el Proceso de gestión de la información en las Bases de Ambulancias.

Se propone como **Objetivo General**

Diseñar una base de datos e implementar una aplicación Web que facilite la gestión de la información relacionada con las operaciones de ambulancias en las bases de Cuba.

Se plantean entonces un grupo de **Tareas** que permitirán satisfacer el objetivo general y que se pueden resumir en las siguientes:

1) Valorar las nuevas tendencias de las tecnologías de la información relacionadas con las aplicaciones Web.
2) Realizar un análisis crítico de la selección de la tecnología, metodología de desarrollo de software, lenguaje de programación, gestor de base de datos y herramientas a utilizar.
3) Analizar las necesidades de funcionamiento del sistema informático.
4) Valorar la integración con otros componentes ya existentes en SISalud, así como con otras partes del Sistema Informatizado para el Centro Nacional de Urgencias Médicas (SIUM).
5) Definir los patrones de arquitectura a utilizar.
6) Obtener el Modelo de Implementación y los artefactos necesarios que describan la Base de Datos del Módulo Operaciones.
7) Realizar la implementación del Módulo Operaciones, basándose en la propuesta de arquitectura definida por el MINSAP (Orientada a Servicios y Basada en Componentes).
8) Realizar la implementación del Módulo Operaciones utilizando los patrones de diseño establecidos en el Análisis y Diseño, así como la implementación de la capa de acceso a datos y procedimientos almacenados.
9) Realizar las pruebas necesarias para garantizar el correcto funcionamiento de la aplicación implementada.

Como principal resultado esperado del presente trabajo de diploma se obtendrá el diseño de la base de datos e implementación del Módulo Operaciones del Sistema Informatizado para el Centro Nacional de Urgencias Médicas. El sistema será una aplicación Web capaz de gestionar la información y con su puesta en marcha permitirá cambiar la situación existente en las Bases de Ambulancias. Logrando una mayor confidencialidad, integridad y disponibilidad de la información manejada. Así como mejoras en los servicios, disminución de los riesgos de pérdida de datos y mejoras en los reportes estadísticos. Además, se brindará información básica a otros componentes del Sistema Informatizado para el SIUM.

El contenido de este trabajo se estructura en tres capítulos:

El Capítulo 1. Fundamentación teórica: Se realiza un estudio del estado del arte donde se brinda información referente a los softwares existentes vinculados al campo de acción y se describen los principales aspectos de las herramientas utilizadas en la producción del sistema informático, siendo estas las elegidas por el proyecto.

El Capítulo 2. Descripción y análisis de la solución propuesta. Se realiza una valoración crítica del diseño propuesto por los analistas, un análisis sobre componentes, módulos o implementaciones que podrían ser reutilizadas para dar solución al problema propuesto, así como la descripción de las clases implementadas para dar solución al problema y las tablas de la base de datos generada.

El Capítulo 3. Validación de la solución propuesta. En este capítulo se realizara un análisis y descripción de los posibles "Test de unidades" para validar la solución propuesta. Se realiza una descripción de los valores utilizados para los test, así como una evaluación de la ejecución y los resultados obtenidos.

CAPÍTULO1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se realiza un análisis detallado del estado del arte de las distintas técnicas de programación que existen a nivel internacional, nacional y de la Universidad. Se hace un estudio de los lenguajes de programación, tecnologías y metodologías relacionadas con dichas técnicas, así como las plataformas de desarrollo que la soportan. Se realiza un estudio crítico y valorativo de la técnica de programación, plataforma y librerías usadas para el desarrollo del sistema, teniendo en cuenta las necesidades y las características del entorno donde se aplicará la solución propuesta.

1.1 Sistemas existentes vinculados al Campo de Acción

En la actualidad, Cuba no cuenta con un sistema informático que le proporcione un control de las operaciones que se realizan en una base de ambulancias. Existen varias soluciones informáticas a nivel internacional que gestionan la información de flotas de vehículos, en su esencia enfocan a los talleres de vehículos como sistemas independientes, la esencia de cómo enfocar el sistema depende en gran medida de las necesidades reales del cliente no siendo estos de gran ayuda para el desarrollo de el software.

- **VehiculosPro 4.20**

  Esta es una aplicación desarrollada en España. Se puede usar tanto a nivel particular como talleres, agencias de alquiler de vehículos, empresas de ambulancias, empresas de transporte, etc. Permite añadir nuevos vehículos, modificar, eliminar los datos de los mismos. Imprime una relación de todos los registros dados de alta, y los intervalos de mantenimiento de cada uno de ellos. Permite controlar por kilómetros meses, días y horas las tareas de mantenimiento que el usuario defina, controla y avisa de cuando hay que hacer el siguiente mantenimiento. Controla los neumáticos que usa el vehículo, kilómetros de uso, etc.

  Permite anotar todos aquellos datos tales como combustible y número de neumáticos. Permite definir el intervalo en kilómetros, meses, días y horas entre un mantenimiento y otro. Permite hacer un seguimiento sobre el consumo de su vehículo litros por 100 Km. y litros por hora si el control se hace por horas de trabajo. Permite un control global sobre los kilómetros recorridos por cada vehículo, repostajes, consumo por 100 Km. Permite llevar un control sobre seguro de accidentes, tarjeta de sanidad (en el caso de ambulancias) la inspección técnica de vehículos (I.T.V.), tarjeta de transporte.

Permite llevar un control de los accidentes sufridos por cada vehículo, datos del vehículo contrario, descripción, daños ocasionados, etc.

- **Sistema de gestión de flota de vehículos de supplest software solutions**

  Esta es una aplicación desarrollada en Argentina. Controla todo el proceso (ingreso, mantenimiento y egreso) de una flota de vehículos o un conjunto de maquinarias, abarcando entre otros los aspectos de consumo de combustible y lubricantes, mantenimiento preventivo, correctivo y mayor, impuestos y seguros, requerimientos de repuestos, etc. Consta de varios módulos que desarrollan las siguientes funciones: Administración de usuarios y permisos, Gestión de vehículos, Gestión de proveedores, Consumo y mantenimiento, Registro de talonarios de consumo.

- **Anago – Mantenimiento de Flota**

  Esta es una aplicación desarrollada en Argentina Se compone de cuatro áreas de trabajo: Reparaciones, Abastecimiento de Combustible, Mantenimiento de Neumáticos, Vencimientos y Revisiones.

  El módulo de Reparaciones permite consultar: El historial de todas las reparaciones efectuadas en la flota y controlar detalladamente los costos ocasionados en concepto de materiales, repuestos y mano de obra así como las facturas de los talleres.

  El módulo de Abastecimiento de Combustible: Proporciona un detallado historial de abastecimientos de combustibles de cada uno de los vehículos. Calcula el kilometraje recorrido, el consumo y controla las tarjetas de combustibles de los proveedores. Permite controlar las cisternas de combustible de la empresa, advirtiendo el nivel de reposición y gestionado las compras automáticamente. Proporciona un control detallado de consumo medio cada vehículo, kilómetros recorridos, gastos efectuados con las tarjetas de un proveedor.

  El módulo de Mantenimiento de Neumáticos: Administra la información sobre proveedores, fabricantes, modelos, costos, kilometraje, posición de montaje y otros datos de todos los neumáticos de la empresa.

  La ventana de la gestión de neumáticos tiene los siguientes componentes:

  ✓ Lista de stock: muestra todos los neumáticos actualmente almacenados.
  ✓ Lista de vehículos: muestra la información de la flota.
  ✓ Caja de herramientas: permite gestionar los neumáticos mediante drag&drop.

  La ventana de ejes: permite visualizar la configuración del vehículo.

  La ventana de detalle: muestra los costos de cada neumático y otras informaciones.

Módulo de Vencimientos y Revisiones: Controla vencimientos de mantenimiento preventivo y revisiones técnicas de los vehículos para evitar tiempos de parada en la flota. Se realiza a través de un sistema gráfico de avisos y alarmas. Los vencimientos pueden definirse por: kilómetros recorridos, fecha, horas de uso.

- **Mantenimiento de Flotas v1.4**

  Programa de gestión y mantenimiento de flota de vehículos. Funcionalidades que permite: Consta de un fichero de vehículos donde se darán de alta los vehículos con control de seguro, consumo combustible, consumo de ruedas, viajes, y además podrás mantener el control de:(Siniestros, gastos e ingresos varios, mantenimiento preventivo, por Km. o por fechas, reparaciones)

  También ofrece: Archivo documental donde podrás introducir todos los documentos recibidos para ese vehículo, fichero de empleados con control de pago a empleados, consumo de combustible por empleado siniestros por empleado, viajes por empleado, gastos e ingresos por empleado y equipo de protección individual, fichero de proveedores con consumo de combustible por proveedor, siniestros, ruedas, mantenimiento preventivo, facturas de compra, fichero de compras control de compras por proveedor, fotos de vehículos en la ficha ,fotos de empleados en la ficha de empleados ,control de compras a proveedores ,listados por pantalla o impresora con selección o filtro a crear por el usuario.

- **Vivid Workshop Data**

  Este software presenta particularidades que lo hacen especial como: La calidad y la cantidad de datos que pueden incluirse (en formato electrónico) en WorkshopData ATI (Advanced Technology and Interface Tecnología y Interfaz avanzado) es insuperable, los datos se basan estrictamente en datos originales OEM, ofrece la mejor relación calidad/precio, cobertura de todos los automóviles disponibles en el mercado europeo, se conecta fácilmente a otras bases de datos que ofrece posibilidades ilimitadas de intercambio de datos por Internet. Además ofrece casi 7.000 tipos de motores, más de 50.000 planos técnicos, más de 6.000 sistemas de gestión de motor, miles de diagramas eléctricos, millones de tiempos de reparación especificados.

- **Softpicks**

  Gestiona la reparación de vehículos, automóviles, motos, camiones, etc. y mantiene al día y organizados todos los movimientos. Esta aplicación permite, entre otras cosas: Controlar las reparaciones hechas a un vehículo con detalle de repuestos, mano de obra, importe y fecha. Imprimir una orden de reparación como: factura, presupuesto con o sin precios, albarán con o sin precios, encargos del cliente, imprimir todos los albaranes de una matrícula ,control de almacén de repuestos

,crear una orden con clientes, vehículos, repuestos dados de alta en el programa o sin dar de alta previamente ,control de facturas de proveedores ,rendimiento de operarios ,etiquetas de clientes para mailing ,imprimir recibos de facturas/presupuestos, formato de impresión de ordenes, facturas, albaranes, etc. Modificable por el usuario, listados por pantalla y por impresora, generar listados, informes.

- **AutoSoft Taller**

  AutoSoft Taller 2.50 Básica es la más avanzada versión del mejor programa de gestión especialmente diseñado para talleres mecánicos, de chapa y pintura, totalmente adaptable a cualquier país de habla hispana. Es una poderosa herramienta tecnológica adaptada a cualquier necesidad e imprescindible en todo taller mecánico y de servicios automotrices, sin importar su tamaño. Permite organizar un taller y controlar todos los procesos administrativos de manera eficaz, utilizando un entorno gráfico que facilita su uso, aunque nunca se haya utilizado una computadora, además es de rápida instalación, de uso intuitivo y capaz de funcionar de manera segura y estable en redes locales bajo ambiente Windows.

  Este software permite entre otras cosas: elaborar presupuestos, órdenes de compras, control de inventario, manejo de compañías de seguro, asignación de manos de obras, facturación, historial de vehículos, cuentas por pagar, cuentas por cobrar, reporte de ganancias por trabajo, etc. Es adaptable a talleres de cualquier tamaño que deseen aprovechar las ventajas de la informática. Es un programa que permite llevar control de la gestión administrativa y optimizar la eficiencia de su empresa, a un precio razonable y con garantía de servicio de atención al cliente totalmente gratis.

- **Sistema de Gestión de Flota Vehicular(URUMAN 2005)**

  Aplicación desarrollada en Montevideo – Uruguay el cual consiste básicamente en tres módulos: Información Técnica y Mantenimiento (ITM),Operación de Flota, Control de Gestión y Consultas.

  Permite las siguientes funcionalidades: Registro y análisis de accidentes vehiculares, Controla la autorización del chofer para la utilización del vehículo y registra fecha, hora, marca de odómetro y marca de hidrómetro en el encendido y apagado de motor y detenciones, permitiendo el cálculo de kilómetros recorridos, velocidad, y horas ociosas, realiza controles previos a la carga de combustible y transfiere la información almacenada en el vehículo hacia la unidad de la estación durante la recarga, quién remite diariamente la información al centro de procesamiento de datos (CPD) del administrador, permite la facturación de los consumos de los organismos, sustituyendo la utilización de los vales de combustible.

1.1.1 Valoración de los sistemas vinculados al Campo de Acción

Estos sistemas no son factibles para solucionar el problema a pesar de tener muchas de las funcionalidades que se requieren puesto que se necesita desarrollar un sistema basado en plataforma libre. Todas las aplicaciones referentes al Sistema Nacional de Salud deben converger en una única red por lo que se hace necesario que todos los sistemas informáticos estén desarrollados en entornos web, usando de esta manera herramientas de desarrollo fundamentalmente libres u OpenSource.
Esto traería consigo grandes ventajas ya que el uso de las tecnologías libres es económico, el bajo coste o nulo de los productos libres permite ampliar sus infraestructuras sin que se vean mermados sus intentos de crecimiento por no poder hacer frente al pago de grandes cantidades dado el costo de las licencias.

Todos estos softwares tienen como tipo de licencia propietaria que representan una gran barrera ya que:

- Es ilegal extender una pieza de software propietario para adaptarla a las necesidades particulares de un problema específico. En caso de que sea vitalmente necesaria tal modificación, es necesario pagar una elevada suma de dinero a la compañía fabricante, para que sea ésta quien lleve a cabo la modificación a su propio ritmo de trabajo y sujeto a su calendario de proyectos.
- Es ilegal hacer copias del software propietario sin antes haber contratado las licencias necesarias.
- Si la compañía fabricante del software propietario se va a la banca rota el soporte técnico desaparece.
- Los clientes que contrataron licencias para el uso de ese software quedan completamente abandonados a su propia suerte.

Otra desventaja que presentan estos softwares es que son aplicaciones de escritorio. En las aplicaciones de escritorio cada cliente tiene que poseer dicha aplicación en su computadora, ocupando un espacio que puede ser usado para otros servicios. No siendo así con una aplicación Web ya que al utilizar la misma solo es necesario tener la aplicación en un servidor y los usuarios pueden acceder a ella sin tenerla en su máquina, resolviendo así el problema de la distancia, ya que el cliente solo tiene que conectarse al servidor, lo que no ocupa espacio en su máquina.

Teniendo en cuenta las características y facilidades del sitio Web, se plantea la implementación de una aplicación Web que facilite la gestión de la información relacionada con las operaciones de ambulancias en las bases de Cuba, brindándole a los usuarios eficiencias en las respuestas a sus solicitudes.

1.2 Tecnologías, metodología y herramientas utilizadas

Constituye un objetivo fundamental alcanzar y mantener un nivel técnico acorde con el desarrollo actual en la informatización de la información para la gestión de cualquier proceso a desarrollar, para lo cual es necesario hacer un estudio detallado da las tecnologías a utilizar y las posibilidades de desarrollo que estas brindan, así como los conceptos ligados a estas.

A continuación en esta sección se describe los principales conceptos, tecnologías y herramientas propuestas para el desarrollo de la solución tratada en el trabajo. Para dar solución al problema y luego de haber analizado un grupo de tecnologías y lenguajes candidatos, se decidió que la aplicación se realizara usando los lenguajes de programación HTML, Java Script y PHP 5.2.3, como tecnología a Ajax, los datos se almacenarán en el gestor de bases de datos MySQL 5.0.36, como herramientas al App Server 2.2.4, Zend Studio 5.5.0, Code Igniter 1.6.0, Dreamweaver 8.0 , Visual Paradigm 6.0 y Case Studio 2, como metodología RUP y como lenguaje de modelado UML,todo esto debido a que el proyecto y la facultad definió trabajar con los mismos debido a sus características, por lo que se dará una pequeña información sobre estos programas.

1.2.1 Lenguajes de programación

Los lenguajes de programación son creados por el hombre para poder comunicarse con las computadoras. De esta forma un lenguaje de programación consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.

HTML, o HyperText Markup Language (lenguaje de marcas hipertextuales), es el formato con el que se le dice a un explorador de Web cómo mostrar los documentos multimedia. Los documentos mismos son archivos de texto simple (ASCII) con "etiquetas" (tags) especiales, o códigos, que el explorador del Web sabe cómo interpretar y visualizar en la pantalla. Con un simple editor de textos, usted puede

crear su propias páginas Web, o centros de información que se conecten a Internet. Es un lenguaje del lado del cliente, de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. Gracias a Internet y a los navegadores del tipo *Internet Explorer*, Opera, Firefox o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos, es una aplicación de SGML conforme al estándar internacional ISO 8879. (1)

JavaScript es un lenguaje del lado del cliente de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.
Dirigido por eventos, estará listo para actuar en cuanto un evento (un clic en un botón, por ejemplo) sea ejecutado, implementa una sencilla interfaz de objetos/propiedades/métodos. Se integra dentro del código HTML de las páginas Web, se ejecuta en el navegador al mismo tiempo que las sentencias van descargándose junto con el código HTML. Brinda rapidez a la aplicación web ha la hora de las validaciones de los formularios. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado. (2)

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje del lado del servidor, interpretado de alto nivel embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl, con solamente un par de características PHP específicas. La meta de este lenguaje es permitir escribir a los creadores de páginas Web, páginas dinámicas de una manera rápida y fácil. (3)

Tiene una comunidad muy grande de desarrolladores, existen miles de lugares donde se pueden encontrar: documentación, tutoriales, ejemplos de código, foros. Si se tiene un problema con PHP se puede encontrar la respuesta en muchos sitios en donde los usuarios comparten el conocimiento adquirido en el proceso de desarrollo. Su rendimiento es muy bueno y verdaderamente eficiente, utilizando un servidor modesto se pueden atender millones de peticiones al día. Además de ello si se

necesita mejorar este rendimiento Zend Technologies ha desarrollado versiones especiales para incrementarlo.

Diseñada para trabajar sobre la web, por ello, trae un conjunto muy amplio de funciones para ser utilizadas en diferentes tareas relacionadas con la web. Se puede conectar con bases de datos, conectar a *web services*, parsear XML, enviar email, generar PDFs, generar imágenes, etc. Basadas en estas librerías existes clases implementadas para facilitar el trabajo de los desarrolladores. Otro punto es que hay desarrolladores que agregan librerías especializadas para extender las funcionalidades de PHP. Este lenguaje está disponible para la mayoría de sistemas operativos existentes. Desde Unix, Linux, Microsoft Windows, MAC, entre otros. Una vez desarrollada, la aplicación PHP puede funcionar en cualquiera de estos sistemas operativos sin necesidad de modificar el código.

1.2.2 Tecnología AJAX

AJAX, acrónimo de *Asynchronous JavaScript And XML* (JavaScript y XML asíncronos, donde XML es un acrónimo de Extensible Markup Language), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, 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 en la misma. (4)

1.2.3 Herramientas

MYSQL: Hasta el momento es el gestor de bases de datos muy rápido de los que el autor ha visto. Sólo es más lento creando tablas y cambiando la estructura de estas, creando y eliminando índices principalmente. Está soportado por la gran mayoría de los SO como Solarix, Linux, Windows, Unix SCO, SGI Iris, Power PC, Mac OS X Server, IBM AIX Para su conexión se utiliza ODBC, JDBC, API nativa. No implementa todo el estándar ANSI SQL 92. No soporta disparadores ni procedimientos almacenados.

Servidores de Aplicaciones (Application Servers): Designados a veces como un tipo de middleware (software que conecta dos aplicaciones), los servidores de aplicaciones ocupan una gran parte del territorio entre los servidores de bases de datos y el usuario, y a menudo los conectan.

Zend Studio

Son muchos los desarrolladores que trabajan con Zend Studio, es posiblemente uno de los mejores IDE del momento. Se trata de un programa de la casa Zend, uno de los mayores impulsores de PHP, orientada a desarrollar aplicaciones web, como no, en PHP. Zend Studio es un editor de texto para páginas PHP que proporciona un buen número de ayudas desde la creación y gestión de proyectos hasta la depuración del código.

CodeIgniter es un framework para PHP .Está pensado para ofrecer un alto rendimiento, ser ligero y fácilmente instalable (puede usarse en un alojamiento compartido y no es necesario tener acceso a la línea de comando). Este permite trabajar con librerías como FPDF, SESSION y otras que son de gran ayuda. Además, gracias a la documentación y los foros (en inglés ambos), el aprendizaje es muy rápido y en un par de horas se puede empezar a trabajar con él. Por supuesto, será necesario más tiempo para conocer su funcionamiento completo, pero enseguida pueden empezar a hacerse cosas muy interesantes. Viene con varias librerías para gestionar el acceso a datos, sesiones de usuarios, formularios, la seguridad, etc.... Además la comunidad de usuarios ha creado una serie de plugins, clases y librerías para extenderlo que lo hacer aún más interesante.

Dreamweaver es la herramienta de diseño de páginas web más avanzada, tal como se ha afirmado en muchos medios. Aunque sea un experto programador de HTML el usuario que lo maneje, siempre se encontrarán en este programa razones para utilizarlo, sobretodo en lo que a productividad se refiere. Cumple perfectamente el objetivo de diseñar páginas con aspecto profesional, y soporta gran cantidad de tecnologías, además muy fáciles de usar.

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

Case Studio 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 necesita sistema operativo: Win95/98/98SE/Me/2000/NT/XP.

1.2.4 Metodología de desarrollo de software

RUP es una de estas metodologías, que al desarrollar un software se encarga muy bien de ordenar el flujo de trabajo, porque es un proceso que integra las múltiples facetas del desarrollo (tiene 9 flujos de trabajo, de los cuales 6 son ingenieriles y los demás de gestión, a su vez está integrado por 4 fases en la cuales se distribuyen las actividades a realizar, dando como resultado los artefactos de cada uno). Además:

- Proporciona una guía para ordenar las actividades de un equipo.
- Ayuda a dirigir las tareas de cada desarrollador por separado y del equipo como un todo.
- Especifica los artefactos que deben desarrollarse.
- Brinda criterios para el control y la medición de los productos y actividades de proyectos.

Los verdaderos aspectos definitorios del Proceso Unificado se resumen en tres fases claves, dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Esto es lo que hace único al Proceso Unificado. (5)

1.2.5 Lenguaje de modelado

UML

El Lenguaje Unificado de Modelado (UML –Unified Modeling Language–) ayuda a especificar, visualizar y documentar las partes del desarrollo de un software, por lo que se define como un lenguaje gráfico. Además, facilita y permite modelar desde el punto de vista conceptual, tal es el caso del proceso de negocio y las funciones de sistema, de igual forma modelar las cosas concretas como definir y escribir las clases en un lenguaje determinado, los esquemas de base de datos. Permite modelar, construir y fundamentar los elementos que forman un sistema software orientado a objetos.

Conclusiones

Las bases de ambulancias en Cuba no cuentan con ningún sistema automatizado, ni siquiera una página Web publicada con informaciones. Después de hacer un análisis de las posibilidades y los beneficios que traería la implantación de una aplicación se decide que es de vital importancia que se lleve a cabo el desarrollo de la misma.

En este capítulo se dieron a conocer conceptos, características, ventajas y la importancia que representa una aplicación Web para las bases de ambulancias. En resumen, una serie de detalles que hacen de la creación del módulo la vía fundamental para la solución al problema del SIUM en las Bases de Ambulancias.

CAPÍTULO 2. DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

Introducción

Este capítulo contiene el resultado de la búsqueda y análisis de la información vinculada al objeto de estudio, procesos a automatizar y conceptos asociados al domino del sistema. Además, se define el diseño de la solución propuesta, los principales elementos tenidos en cuenta a la hora de hacer cada una de las implementaciones para lograr un producto con la calidad y la eficiencia requerida. Se fundamenta la integración del Módulo Operaciones con otros componentes existentes en el Sistema de Información para la Salud (SISalud).

El desarrollo de este capítulo permitió conocer más el trabajo de los procesos vinculados con las operaciones realizadas en las Bases de Ambulancias de Cuba por lo que fue necesaria la colaboración realizada por las analistas del sistema. Así como la búsqueda de información referente a dicho tema para lograr que una vez implementado el sistema, este satisfaga las necesidades del usuario final.

2.1 Valoración crítica del diseño propuesto por el analista

El diseño propuesto, se valora como bueno, ya que en el se puede obtener las clases fundamentales que deben ser definidas para que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, dando una idea clara de lo que se debe implementar. Unido a esto, se encuentran los diagramas de interacción, que explican la colaboración que existe entre las clases y cómo son llamados los métodos y sentencias dentro de cada una, de los cuales se obtiene la información necesaria para conocer el orden de las acciones a implementar.

El diseño propuesto fue creado de manera tal que permitió llevar a cabo la implementación del módulo de forma clara y limpia. En esta aplicación jugó un papel fundamental el patrón de diseño Modelo-Vista-Controlador (MVC), por lo que se dará una breve descripción acerca de los patrones de diseño y en especial del MVC y del patrón de software Active Record.

Patrones de diseño.

Un patrón es un modelo que se puede seguir para realizar algo. Los patrones surgen de la experiencia de seres humanos de tratar de lograr ciertos objetivos. Los patrones capturan la experiencia existente y probada para promover buenas prácticas.

Los Patrones de Diseño (Design Patterns) son la base para la búsqueda de soluciones a problemas

comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Los patrones de diseño han contribuido a dar flexibilidad y extensibilidad a los diseños. Pero en adición, han demostrado ser una forma muy útil (exitosa) de reutilizar diseño, ya que ellos no sólo nombran, abstraen e identifican aspectos claves de estructuras comunes de diseño, sino que generalmente son descritos en una forma específica documental, haciendo su comprensión y aplicación fácil para el conjunto de desarrolladores.

Los beneficios que un patrón produce pueden ser medidos en varios sentidos:

- Contribuyen a reutilizar diseño, identificando aspectos claves de la estructura de un diseño que:
- Puede ser aplicado en una gran cantidad de situaciones.
- No es despreciable, ya que ésta provee numerosas ventajas: reduce los esfuerzos de desarrollo.
- Y mantenimiento, mejora la seguridad, eficiencia y consistencia de los diseños, y proporciona un considerable ahorro en la inversión.
- Mejoran (aumentan, elevan) la flexibilidad, modularidad y extensibilidad, factores internos e íntimamente relacionados con la calidad percibida por el usuario.
- Incrementan el vocabulario de diseño, ayudando a diseñar desde un mayor nivel de abstracción.

Modelo Vista Controlador.

En ocasiones se lo define más bien como un patrón de diseño o como práctica recurrente.

Un propósito común en numerosos sistemas es el de tomar datos de un almacenamiento y mostrarlos al usuario. Luego que el usuario introduce modificaciones, las mismas se reflejan en el almacenamiento. Dado que el flujo de información ocurre entre el almacenamiento y la interfaz, una tentación común, un impulso espontáneo (hoy se llamaría un anti-patrón) es unir ambas piezas para reducir la cantidad de código y optimizar la performance.

Sin embargo, esta idea es antagónica al hecho de que la interfaz suele cambiar, o acostumbra depender de distintas clases de dispositivos (clientes ricos, browsers, PDAs); la programación de

interfaces de HTML, además, requiere habilidades muy distintas de la programación de lógica de negocios. Otro problema es que las aplicaciones tienden a incorporar lógica de negocios que van más allá de la transmisión de datos. El patrón conocido como Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes:

-Modelo. El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

-Vista. Maneja la visualización de la información.

-Controlador. Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

- El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace).
- El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos handler o callback.
- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los

cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice.

Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.

✓ La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual. La separación entre vista y controlador puede ser secundaria en aplicaciones de clientes ricos y, de hecho, muchos frameworks de interfaz implementan ambos roles en un solo objeto. En aplicaciones de Web, por otra parte, la separación entre la vista (el browser) y el controlador (los componentes del lado del servidor que manejan los requerimientos de HTTP) está mucho más taxativamente definida.

Entre las ventajas del estilo señaladas están:

- Soporte de vistas múltiples. Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación de Web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes.

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

Entre las desventajas, se han señalado:

- Complejidad. El patrón introduce nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad de la solución. También se profundiza la orientación a eventos del código de la interfaz de usuario, que puede llegar a ser difícil de depurar. En rigor, la configuración basada en eventos de dicha interfaz corresponde a un estilo particular (arquitectura basada en eventos) que aquí se examina por

separado.

- Costo de actualizaciones frecuentes. Desacoplar el modelo de la vista no significa que los desarrolladores del modelo puedan ignorar la naturaleza de las vistas. Si el modelo experimenta cambios frecuentes, por ejemplo, podrían desbordar las vistas con una lluvia de requerimientos de actualización. Hace pocos años sucedía que algunas vistas, tales como las pantallas gráficas, involucraban más tiempo para plasmar el dibujo que el que demandaban los nuevos requerimientos de actualización. (6)

Active Record.

Patrón de software utilizado en aplicaciones robustas, que permite trabajar los registros de una tabla en una base de datos como instancias de una clase, por ejemplo Clientes ó Productos en los cuales se puede aplicar métodos Buscar, Guardar y Borrar sin necesidad de utilizar sentencias SQL. (7)

2.2 Análisis de posibles implementaciones, componentes o módulos que puedan ser rehusados.

Estrategias de integración

La utilización de una arquitectura basada en componentes y orientada a servicios, donde cada módulo es un componente con funcionalidad propia y que utiliza los servicios brindados por otros, hacen que el funcionamiento de Módulo Operaciones haga uso de otros componentes de SISalud, los cuales se detallan a continuación. La posibilidad de utilizar los servicios Web brindados por otros componentes, garantiza la reutilización de los datos y evita la duplicación de la información.

El Módulo Operaciones se integra con el Módulo de Centro Coordinador de Emergencia Médica Provincial (CCEMP) a través de servicios web, en este caso el Módulo Operaciones le brinda al CCEMP la disponibilidad de las ambulancias de las diferentes bases y permite la inserción de los datos de demandas y modificar las claves que describen el estado en que se encentran las ambulancias. La base de datos es accedida de forma directa mediante clases modelos y los componentes rehusados son integrados mediante interfaces sencillas.

2.2.1 Componente de Seguridad (SAAA)

Este componente tiene implementado un mecanismo de seguridad basado en el modelo de Autenticación, Autorización y Auditoría. La autenticación es la primera acción del usuario en el sistema y consiste en suministrar un nombre de usuario único y una contraseña. Si el usuario no se encuentra registrado se reportará un error de acceso. Si el usuario autenticado se encuentra registrado se autoriza su acceso y se crea un certificado digital que contiene un identificador único (token) de 32 caracteres, que tiene como información: el identificador de usuario, el nivel de acceso (Nacional, Provincial, Municipal o Unidad de Salud), el identificador de nivel de acceso, un listado de los módulos a los que el usuario tiene acceso y el tipo de acceso en cada uno de ellos (Editor o Visualizador).

2.2.2 Registro de Ubicación (RU)

El Registro de Ubicación es uno de los componentes no médicos del Registro Informatizado de Salud. El mismo gestiona la información de las Provincias, Municipios, Localidades, Calles y Manzanas del país. Los servicios del mismo se utilizan por el Módulo Operaciones para solicitar los criterios de búsqueda en las diferentes opciones relacionadas con las informaciones que se brindan en el RU.

2.3 Descripción de clases

Nombre: C_Ambulancia	
Tipo de Clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Insertar_Ambulancia**
Descripción:	Se insertan los datos de una ambulancia.
Nombre:	**Lista**

Descripción:	Dado los parámetros de búsqueda escogidos, elige a que método entrar. (**Listar_Ambulancias, Listar_Ambulancias1, Listar_Ambulancias2, Listar_Ambulancias3, Listar_Ambulancias4**).
Nombre:	**Listar_Ambulancias**
Descripción:	Muestra las ambulancias con algunos de sus datos dado los parámetros de búsqueda categoría, observación y motivo de paralizada.
Nombre:	**Listar_Ambulancias1**
Descripción:	Muestra las ambulancias con algunos de su dato dado los parámetros de búsqueda categoría y observación.
Nombre:	**Listar_Ambulancias2**
Descripción:	Muestra las ambulancias con algunos de su dato dado los parámetros de búsqueda observación y motivo.
Nombre:	**Listar_Ambulancias3**
Descripción	Muestra las ambulancias con algunos de sus datos dado el parámetro de búsqueda categoría.
Nombre:	**Listar_Ambulancias4**
Descripción:	Muestra las ambulancias con algunos de sus datos dado el parámetro de búsqueda observación.
Nombre:	**CargarDatos**
Descripción:	Muestra los datos de la ambulancia escogida para ver sus datos (se le pasa como parámetro el identificador de ambulancia).

Nombre:	**CargarDatosModificar**
Descripción:	Muestra los datos de la ambulancia escogida a modificar (se le pasa como parámetro el identificador de ambulancia).
Nombre:	**ModificarAmbulancia**
Descripción:	Modifica los datos de la ambulancia escogida.
Nombre:	**Disponibilidad**
Descripción:	Muestra la disponibilidad de ambulancias, o sea las que se encuentran trabajando.

Nombre: C_Accidente	
Tipo de Clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Insertar_Accidente**
Descripción:	Se insertan los datos de un accidente
Nombre:	**Lista**
Descripción:	Dado los parámetros de búsqueda escogidos, elige a que método entrar (**Listar_Accidentes, Listar_Accidentes1, Listar_Accidentes2**).
Nombre:	**Listar_Accidentes**
Descripción:	Muestra los accidentes con algunos de su dato dado los parámetros de

	búsqueda fecha y categoría de accidente.
Nombre:	**Listar_Accidentes1**
Descripción:	Muestra los accidentes con alguno de sus datos dado el parámetro de búsqueda fecha.
Nombre:	**Listar_Accidentes2**
Descripción:	Muestra los accidentes con algunos de sus datos dado el parámetro de búsqueda categoría de accidente.
Nombre:	**CargarDatos**
Descripción:	Muestra los datos del accidente escogido para ver sus datos (se le pasa como parámetro el identificador de accidente).
Nombre:	**CargarDatosModificar**
Descripción:	Muestra los datos del accidente escogido a modificar (se le pasa como parámetro el identificador de accidente).
Nombre:	**ModificarAccidente**
Descripción:	Modifica los datos del accidente escogido.

Nombre: C_Bateria
Tipo de Clase: Controladora
Para cada responsabilidad:

Nombre:	**Insertar_Bateria**
Descripción:	Se insertan los datos de una batería.
Nombre:	**Lista**
Descripción:	Dado el parámetro de búsqueda estado escogido te llevaría al método Listar_Baterias.
Nombre:	**Listar_Baterias**
Descripción:	Muestra las baterías con algunos de sus datos dado el parámetro de búsqueda estado.
Nombre:	**CargarDatos**
Descripción:	Muestra los datos de la batería escogida para ver sus datos (se le pasa como parámetro el identificador de la batería).
Nombre:	**CargarDatosModificar**
Descripción:	Muestra los datos de la batería escogida a modificar (se le pasa como parámetro el identificador de la batería).
Nombre:	**ModificarBateria**
Descripción:	Modifica los datos de la batería escogida.

Nombre: C_Neumatico
Tipo de Clase: Controladora

Para cada responsabilidad:	
Nombre:	**Insertar_Neumatico**
Descripción:	Se insertan los datos de un neumático.
Nombre:	**Lista**
Descripción:	Dado el parámetro de búsqueda estado escogido te llevaría al método Listar_Neumaticos.
Nombre:	**Listar_Neumaticos**
Descripción:	Muestra los neumáticos con algunos de sus datos dado el parámetro de búsqueda estado.
Nombre:	**CargarDatos**
Descripción:	Muestra los datos del neumático escogido para ver sus datos (se le pasa como parámetro el identificador del neumático).
Nombre:	**CargarDatosModificar**
Descripción:	Muestra los datos del neumático escogido a modificar (se le pasa como parámetro el identificador del neumático).
Nombre:	**Modificar_Neumatico**
Descripción:	Modifica los datos del neumático escogido.

Nombre: Entidad_Ambulancia	
Tipo de Clase: Entidad	
Atributo	**Tipo**
op_tipo	int
op_motivo_paralizada	int
op_tipo_combustible	int
op_capacidad	float
op_norma_consumo	float
op_observacion	int
op_estado	varchar
op_anno	int
op_no_motor	int
op_no_chasis	int
op_no_serie	int
op_no_ambulancia	int
op_chapa	varchar
op_categoria	int

op_fecha	date
op_plan	float
op_id_marca	int
op_modelo	int
op_atencion	int
op_base	int

Nombre: Entidad_Accidente	
Tipo de Clase: Entidad	
Atributo	**Tipo**
op_fecha	date
op_categoria	int
op_responsable	varchar
op_direccion	varchar
op_valor	double
op_paramedico	varchar
op_cant_heridos	int

op_cant_fallecidos	int
op_noaccidente	int

Nombre: Entidad_Bateria	
Tipo de Clase: Entidad	
Atributo	**Tipo**
op_motivo_retiro	int
op_estado	varchar
op_marca	int
op_Code	int
op_fecha_instalacion	date
op_fecha_retiro	date
op_situacion	int
op_id_ambulancia	int
op_numero	int

Nombre: Entidad_Neumatico	
Tipo de Clase: Entidad	
Atributo	**Tipo**
op_numero_capas	int
op_estado	varchar
op_km_recorrido	float
op_marca	int
op_medida	double
op_tipo	int
op_pais	int
op_fecha_insertado	date
op_no_neumatico	int
op_situacion	int
op_posicion	varchar
op_chapa	int
op_fecha_retiro	date
op_motivo_retiro	int

Nombre: V_Insertar_Ambulancia	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Ambulancia	
Descripción General: Permite al usuario insertar los datos de una ambulancia	
Parámetros de Salida	
Descripción	**Tipo**
El id del tipo de ambulancia (ejemplo: Panel)	int
El id del motivo de paralizada en caso de estarlo	int
El id del tipo de combustible (mostrando los tipos de combustible)	int
El id del tipo de observación (mostrando las observaciones:Trabajando o Paralizada)	int
El id de la categoría de la ambulancia (mostrando las categorías:Intensiva, Intermedia o Básica)	int
El id de la marca de la ambulancia (ejemplo: Mercedes Benz, Citroen ...etc)	int
El id del modelo de la ambulancia(mostrando los modelos, ejemplo: 308, Berlingo)	int

El id de atención (mostrando por quien es atendida, talleres del MINSAP o otros)	int
Parámetros de Entrada	
Descripción	**Tipo**
El id del tipo de ambulancia (mostrando los tipos de ambulancias, ejemplo: Panel)	int
El id del motivo de paralizada en caso de estarlo, mostrando los motivos de paralizada	int
El id del tipo de combustible (mostrando los tipos de combustible)	int
La capacidad	float
La norma consumo	float
El id del tipo de observación (mostrando las observaciones: Trabajando o Paralizada)	int
El estado (Disponible o de Baja)	varchar
El año de la ambulancia	int
El número de motor	int
El número de chasis	int
El número de serie	int
El número de ambulancia	int

La chapa de la ambulancia	varchar
El id de la categoría de la ambulancia (mostrando las categorías: Intensiva, Intermedia o Básica)	int
La fecha de inserción	date
El plan de mantenimiento	float
El id de la marca de la ambulancia (mostrando las marcas, ejemplo: Mercedes Benz, Citroen ...etc)	int
El id del modelo de la ambulancia(mostrando los modelos, ejemplo: 308, Berlingo)	int
El id de atención (mostrando por quien es atendida, talleres del MINSAP o otros)	int

Nombre: V_Lista_Buscar_Ambulancia
Tipo de Clase: Interfaz
Caso de Uso: Gestionar Ambulancia
Descripción General: Permite al usuario buscar alguna información referente a las ambulancias dado los criterios de búsqueda categoría, observación y motivo de paralizada en caso de tener como observación que su estado es paralizada
Parámetros de Entrada

Descripción	Tipo
El id del tipo de observación (mostrando los tipos de observación Trabajando o Paralizada)	int
El id de la categoría de la ambulancia (mostrando las categorías Intensiva, Intermedia o Básica)	int
El id del motivo de paralizada en caso de estarlo, mostrando los motivos de paralizada	int

Nombre: V_MostrarDatosAmbulancia	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Ambulancia	
Descripción General: Permite al usuario ver los datos de la ambulancia deseada	
Parámetros de salida	
Descripción	**Tipo**
Tipo de ambulancia (ejemplo: Panel)	varchar
Motivo de paralizada en caso de estarlo	varchar
Tipo de combustible	varchar
La capacidad	float

La norma consumo	float
Tipo de observación (Trabajando o Paralizada)	varchar
El estado (Disponible o de Baja)	varchar
El año de la ambulancia	int
El número de motor	int
El número de chasis	int
El número de serie	int
El número de ambulancia	int
La chapa de la ambulancia	varchar
La categoría de la ambulancia (Intensiva, Intermedia o Básica)	varchar
La fecha de inserción	date
El plan de mantenimiento	float
La marca de la ambulancia (ejemplo: Mercedes Benz, Citroen ...etc)	varchar
El modelo de la ambulancia(ejemplo: 308, Berlingo)	varchar
Por quien es atendida, ejemplo: talleres del MINSAP o otros)	varchar

La base de la ambulancia	varchar

Nombre: V_Modificar_Ambulancia	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Ambulancia	
Descripción General: Permite al usuario modificar los datos de la ambulancia deseada y la vez le muestra los datos que esta posee	
Parámetros de salida	
Descripción	**Tipo**
Tipo de ambulancia (ejemplo: Panel)	varchar
Motivo de paralizada en caso de estarlo	varchar
Tipo de combustible	varchar
La capacidad	float
La norma consumo	float
Tipo de observación (Trabajando o Paralizada)	varchar
El estado (Disponible o de Baja)	varchar
El año de la ambulancia	int

El número de motor	int
El número de chasis	int
El número de serie	int
El número de ambulancia	int
La chapa de la ambulancia	varchar
La categoría de la ambulancia (Intensiva, Intermedia o Básica)	varchar
La fecha de inserción	date
El plan de mantenimiento	float
La marca de la ambulancia (ejemplo: Mercedes Benz, Citroen ...etc)	varchar
El modelo de la ambulancia(ejemplo: 308, Berlingo)	varchar
Por quien es atendida, ejemplo: talleres del MINSAP o otros)	varchar
La base de la ambulancia	varchar
Parámetros de Entrada	
Descripción	**Tipo**
El id del tipo de ambulancia (mostrando los tipos de ambulancia, ejemplo: Panel)	int

El id del motivo de paralizada en caso de estarlo (mostrando los motivos de paralizada)	int
El id del tipo de combustible (mostrando los tipos de combustible)	int
La capacidad	float
La norma consumo	float
El id del tipo de observación (mostrando las observaciones, Trabajando o Paralizada)	int
El estado (Disponible o de Baja)	varchar
El año de la ambulancia	int
El número de motor	int
El número de chasis	int
El número de serie	int
El número de ambulancia	int
La chapa de la ambulancia	varchar
La categoría de la ambulancia (Intensiva, Intermedia o Básica)	int
La fecha de inserción	date
El plan de mantenimiento	float

El id de la marca de la ambulancia (ejemplo: Mercedes Benz, Citroen ...etc)	int
El id del modelo de la ambulancia(ejemplo: 308, Berlingo)	int
El id de atención (por quien es atendida, talleres del MINSAP o otros)	int
El id de la Base de la ambulancia	int

Nombre: V_Listar_Ambulancias	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Ambulancia	
Descripción General: Le muestra al usuario algunos datos de las ambulancias que el mismo seleccionó dado alguno de los criterios de búsqueda	
Parámetros de Salida	
Descripción	**Tipo**
El número de las ambulancias	int
La chapa de las ambulancias	varchar
La marca de las ambulancias	varchar
El modelo de las ambulancias	varchar

El tipo de combustible de las ambulancias	varchar

Nombre: V_Insertar_Accidente	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Accidente	
Descripción General: Permite al usuario insertar los datos de un accidente	
Parámetros de Salida	
Descripción	**Tipo**
El id de ambulancia (mostrando las chapas)	int
El id de categoría de accidente (mostrando las categorías)	int
Parámetros de Entrada	
Descripción	**Tipo**
La fecha del accidente	date
El id de la categoría de accidente (mostrando las categorías)	int
Responsabilidad (si o no)	varchar

Dirección del accidente	varchar
Valor en $pesos del accidente	double
Nombre del paramédico	varchar
La cantidad de heridos	int
La cantidad de fallecidos	int
El número de accidente	int
El id de ambulancia (mostrando las chapas de ambulancias)	int

Nombre: V_Lista_Buscar_Accidente	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Accidente	
Descripción General: Permite al usuario buscar alguna información referente a los accidentes dado los criterios de búsqueda fecha y categoría	
Parámetros de Entrada	
Descripción	**Tipo**
Las fechas de los accidentes ocurridos	date
El id de las categoría de los accidentes (mostrando las categorías)	int

Nombre: V_Listar_Accidente	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Accidente	
Descripción General: Le muestra al usuario algunos datos de los accidentes que el mismo seleccionó dado alguno de los criterios de búsqueda	
Parámetros de Salida	
Descripción	**Tipo**
Fechas de los accidentes	date
Chapas de las ambulancias	varchar
Categorías de los accidentes	varchar
Responsabilidades de los accidentes (si o no)	varchar
Paramédico del Accidente	varchar

Nombre: V_MostrarDatosAccidente
Tipo de Clase: Interfaz
Caso de Uso: Gestionar Accidente

Descripción General: Permite al usuario ver los datos del accidente seleccionado	
Parámetros de salida	
Descripción	**Tipo**
Chapa de la ambulancia del accidente	varchar
Dirección del accidente	varchar
Nombre del paramédico que estuvo en el accidente	varchar
Valor en $pesos del accidente	float
Categoría del accidente	varchar
Fecha del accidente	date
Responsabilidad del accidente (si o no)	varchar
Cantidad de heridos en el accidente	int
Cantidad de fallecidos en el accidente	int
Número de accidente	int

Nombre: V_Modificar_Accidente
Tipo de Clase: Interfaz
Caso de Uso: Gestionar Accidente

Descripción General: Permite al usuario modificar los datos del accidente seleccionado	
Parámetros de salida	
Descripción	**Tipo**
Chapa de la ambulancia del accidente	varchar
Dirección del accidente	varchar
Nombre del paramédico que estuvo en el accidente	varchar
Valor en $pesos del accidente	float
Categoría del accidente	varchar
Fecha del accidente	date
Responsabilidad del accidente (si o no)	varchar
Cantidad de heridos en el accidente	int
Cantidad de fallecidos en el accidente	int
Número de accidente	int
Parámetros de Entrada	
Descripción	**Tipo**
La fecha del accidente	date
El id de la categoría de accidente	int

(mostrando las categorías)	
Responsabilidad (si o no)	varchar
Dirección del accidente	varchar
Valor en $pesos del accidente	double
Nombre del paramédico	varchar
La cantidad de heridos	int
La cantidad de fallecidos	int
El número de accidente	int
El id de ambulancia (mostrando las chapas de ambulancias)	int

Nombre: V_Insertar_Bateria	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Batería	
Descripción General: Permite al usuario insertar los datos de una batería	
Parámetros de Salida	
Descripción	**Tipo**
El id de ambulancia (mostrando las	int

chapas)	
El id de situación de la batería (mostrando las situaciones: Nueva, Reparada...etc.)	int
El id de marca de batería (mostrando las marcas)	int
El id del país de la batería	int
Parámetros de Entrada	
Descripción	**Tipo**
El id de ambulancia (mostrando las chapas)	int
El id de situación de la batería (mostrando las situaciones: Nueva, Reparada...etc.)	int
El id de marca de batería (mostrando las marcas)	int
El id del país de la batería	int
La fecha de instalada la batería	date
El número de la batería	int

Nombre: V_Lista_Buscar_Bateria
Tipo de Clase: Interfaz

Caso de Uso: Gestionar Batería	
Descripción General: Permite al usuario buscar alguna información referente a las baterías dado el criterio de búsqueda seleccionado	
Parámetros de Entrada	
Descripción	**Tipo**
El estado de la batería (Disponible o de Baja)	varchar

Nombre: V_Listar_Baterias	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Batería	
Descripción General: Le muestra al usuario algunos datos de las baterías que el mismo seleccionó dado el criterio de búsqueda	
Parámetros de Salida	
Descripción	**Tipo**
Números de las baterías	varchar
Chapas de las ambulancias a las que pertenecen las baterías	varchar
Marcas de las baterías	varchar

Situación de las baterías	varchar
Tiempo de trabajo de las baterías	int

Nombre: V_MostrarDatosBateria	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Batería	
Descripción General: Permite al usuario ver los datos de la batería seleccionada	
Parámetros de salida	
Descripción	**Tipo**
Chapa de la ambulancia de la batería	varchar
Situación de la batería	varchar
Número de la batería	int
Motivo de retiro de la batería	varchar
País de la batería	varchar
Fecha de instalada la batería	date
Fecha de retirada la batería	date
Estado de la batería	varchar

Marca de la batería	varchar

Nombre: V_Modificar_Bateria	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Batería	
Descripción General: Permite al usuario ver los datos de la batería seleccionada	
Parámetros de salida	
Descripción	**Tipo**
Chapa de la ambulancia de la batería	varchar
Situación de la batería	varchar
Número de la batería	int
Motivo de retiro de la batería	varchar
País de la batería	varchar
Fecha de instalada la batería	date
Fecha de retirada la batería	date
Estado de la batería	varchar
Marca de la batería	varchar

Parámetros de Entrada	
Descripción	**Tipo**
El id de ambulancia (mostrando las chapas)	int
El id de situación de la batería (mostrando las situaciones: Nueva, Reparada...etc.)	int
El id de marca de batería (mostrando las marcas)	int
El id del país de la batería	int
La fecha de instalada la batería	date
El número de la batería	int
El estado de la batería (Disponible o de Baja)	varchar
El id de motivo de retiro en caso que este de baja, mostrando los motivos de retiro	int
La fecha de retirado en caso que se le vaya a dar baja	date

Nombre: V_Insertar_Neumático
Tipo de Clase: Interfaz

Caso de Uso: Gestionar Neumático	
Descripción General: Permite al usuario insertar los datos de un neumático	
Parámetros de Salida	
Descripción	**Tipo**
El id de ambulancia (mostrando las chapas)	int
El id de situación del neumático (mostrando las situaciones: Nuevo, Recapado...etc.)	int
El id de marca del neumático (mostrando las marcas)	int
El id del país del neumático	int
El id del tipo de neumático	int
Parámetros de Entrada	
Descripción	**Tipo**
El id de ambulancia (mostrando las chapas)	int
El id de situación del neumático (mostrando las situaciones: Nuevo, Recapado...etc.)	int
El id de marca del neumático (mostrando	int

las marcas)	
El id del país del neumático	int
El id del tipo de neumático	int
La posición del neumático	int
El número de neumático	int
Los km recorridos del neumático	float
El número de capas del neumático	int
La fecha de insertado el neumático	date
La medida del neumático	float

Nombre: V_Lista_Buscar_Neumatico	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Neumático	
Descripción General: Permite al usuario buscar alguna información referente a los neumáticos dado el criterio de búsqueda seleccionado	
Parámetros de Entrada	
Descripción	**Tipo**

El estado del neumático (Disponible o de Baja)	varchar

Nombre: V_Listar_Neumaticos	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Neumático	
Descripción General: Le muestra al usuario algunos datos de los neumáticos que el mismo seleccionó dado el criterio de búsqueda	
Parámetros de Salida	
Descripción	**Tipo**
Números de los neumáticos	int
Números de las ambulancias a las que pertenecen las baterías	int
Marcas de los neumáticos	varchar
Situación de los neumáticos	varchar
Medida de los neumáticos	float

Nombre: V_MostrarDatosNeumático
Tipo de Clase: Interfaz

Caso de Uso: Gestionar Neumático	
Descripción General: Permite al usuario ver los datos del neumático seleccionado	
Parámetros de salida	
Descripción	**Tipo**
Chapa de la ambulancia del neumático	varchar
Situación del neumático	varchar
Número del neumático	int
Motivo de retiro del neumático	varchar
País del neumático	varchar
Fecha de instalado el neumático	date
Fecha de retirado el neumático	date
Estado del neumático	varchar
Marca del neumático	varchar
Tipo de neumático (Convencional o Radial... etc.)	varchar
Posición del neumático	varchar
Medida del neumático	float
Número de capas del neumático	int

Km recorridos del neumático	float

Nombre: V_Modificar_Neumático	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Neumático	
Descripción General: Permite al usuario modificar los datos del neumático seleccionado	
Parámetros de salida	
Descripción	**Tipo**
Chapa de la ambulancia del neumático	varchar
Situación del neumático	varchar
Número del neumático	int
Motivo de retiro del neumático	varchar
País del neumático	varchar
Fecha de instalado el neumático	date
Fecha de retirado el neumático	date
Estado del neumático	varchar
Marca del neumático	varchar

Tipo de neumático (Convencional o Radial... etc.)	varchar
Posición del neumático	varchar
Medida del neumático	float
Número de capas del neumático	int
Km recorridos del neumático	float
Parámetros de Entrada	
Descripción	**Tipo**
El id de ambulancia (mostrando las chapas)	int
El id de situación del neumático (mostrando las situaciones: Nuevo, Recapado...etc.)	int
El id de marca del neumático (mostrando las marcas)	int
El id del país del neumático	int
El id del tipo de neumático	int
La posición del neumático	int
El número de neumático	int
Los km recorridos del neumático	float

El número de capas del neumático	int
La fecha de insertado el neumático	date
La medida del neumático	float

Nombre: V_InsertarNomencladorMarca	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Marca	
Descripción General: Permite al usuario insertar, modificar y eliminar marcas de ambulancias	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Marca de Ambulancia	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Marca de Ambulancia	varchar

Nombre: V_InsertarModeloAmbulancia

Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Modelo	
Descripción General: Permite al usuario insertar, modificar y eliminar modelos de ambulancias	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Modelo de Ambulancia	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Modelo de Ambulancia	varchar

Nombre: V_InsertarAtencion
Tipo de Clase: Interfaz
Caso de Uso: Gestionar Nomenclador Atendida
Descripción General: Permite al usuario insertar, modificar y eliminar quien taller atiende a las ambulancias

Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Atención	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Atención	varchar

Nombre: V_Insertar_Categoria_Ambulancia	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Categoría de Ambulancia	
Descripción General: Permite al usuario insertar, modificar y eliminar categorías de ambulancias	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Categoría de ambulancia	varchar
Parámetros de Entrada	

Descripción	**Tipo**
Categoría de ambulancia	varchar

Nombre: V_Insertar_Categoria_Accidente	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Categoría de Accidente	
Descripción General: Permite al usuario insertar, modificar y eliminar las categorías de accidente	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Categoría de accidente	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Categoría de accidente	varchar

Nombre: V_Insertar_Combustible	
Tipo de Clase: Interfaz	

Caso de Uso: Gestionar Nomenclador Tipo de Combustible	
Descripción General: Permite al usuario insertar, modificar y eliminar tipos de combustible	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Tipo de combustible	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Tipo de combustible	varchar

Nombre: V_Insertar_Marca_Bateria	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Marca de Batería	
Descripción General: Permite al usuario insertar, modificar y eliminar las marcas de baterías	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int

Marca de batería	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Marca de batería	varchar

Nombre: V_Insertar_Marca_Neumatico	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Marca de Neumático	
Descripción General: Permite al usuario insertar, modificar y eliminar las marcas de neumáticos	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Marca de neumático	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Marca de neumático	varchar

Nombre: V_Insertar_MotivoParalizada_Ambulancia	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Motivo de Paralizada la Ambulancia	
Descripción General: Permite al usuario insertar, modificar y eliminar los motivos de paralizada la ambulancia	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Motivo de paralizada	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Motivo de paralizada	varchar

Nombre: V_Insertar_MotivoRetiro_Bateria
Tipo de Clase: Interfaz
Caso de Uso: Gestionar Nomenclador Motivo de Retiro de la Batería
Descripción General: Permite al usuario insertar, modificar y eliminar los motivos de retiro

de la batería	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Motivo de retiro	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Motivo de retiro	varchar

Nombre: V_Insertar_MotivoRetiro_Neumatico	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Motivo de Retiro de los Neumáticos	
Descripción General: Permite al usuario insertar, modificar y eliminar los motivos de retiro de los neumáticos	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Motivo de retiro	varchar

Parámetros de Entrada	
Descripción	**Tipo**
Motivo de retiro	varchar

Nombre: V_Insertar_Observacion	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Observación	
Descripción General: Permite al usuario insertar, modificar y eliminar el tipo de observación (Trabajando o Paralizada)	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Observación	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Observación	varchar

Nombre: V_Insertar_Situacion_Bateria

Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Situación de la Batería	
Descripción General: Permite al usuario insertar, modificar y eliminar las situaciones de la batería	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Situación	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Situación	varchar

Nombre: V_Insertar_Situacion_Neumatico
Tipo de Clase: Interfaz
Caso de Uso: Gestionar Nomenclador Situación de los Neumático
Descripción General: Permite al usuario insertar, modificar y eliminar las situaciones de los neumáticos
Parámetros de salida

Descripción	**Tipo**
Identificador	int
Situación	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Situación	varchar

Nombre: V_Insertar_Tipo_Ambulancia	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Tipo de Ambulancia	
Descripción General: Permite al usuario insertar, modificar y eliminar los tipos de ambulancias	
Parámetros de salida	
Descripción	**Tipo**
Identificador	int
Tipo	varchar
Parámetros de Entrada	
Descripción	**Tipo**

Tipo	varchar

Nombre: V_Insertar_Tipo_Neumatico	
Tipo de Clase: Interfaz	
Caso de Uso: Gestionar Nomenclador Tipo de Neumático	
Descripción General: Permite al usuario insertar, modificar y eliminar los tipos de neumáticos	
Parámetros de salida	
Descripción	**Tipo**
Identificador	Int
Tipo	Varchar
Parámetros de Entrada	
Descripción	**Tipo**
Tipo	Varchar

Nombre: V_Insertar_Base
Tipo de Clase: Interfaz
Caso de Uso: Gestionar Nomenclador Base

Descripción General: Permite al usuario insertar, modificar y eliminar las bases	
Parámetros de salida	
Descripción	**Tipo**
Identificador	Int
Base	varchar
Parámetros de Entrada	
Descripción	**Tipo**
Base	varchar

2.4 Diseño de la base de datos

Una base de datos o banco de datos es un conjunto de datos pertenecientes al un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.

En la actualidad y debido al desarrollo tecnológico de campos como la informática y la electrónica la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos. En este epígrafe se muestra el modelo de datos y las descripciones de las tablas de la base de datos generada.

2.4.1 Modelo de datos.

Un modelo de datos es aquel que describe de una forma abstracta cómo se representan los datos, sea

en una empresa, en un sistema de información o en un sistema de gestión de base datos **(Ver Anexo no. 1).** Básicamente consiste en una descripción de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores.

2.4.2 Descripción de las tablas de la base de datos.

Nombre: 'marca'		
Descripción: Es un nomenclador para la marca de las ambulancias		
Atributo	**Tipo**	**Descripción**
cmp_id_marca	Integer	Identificador de la tabla
cmp_marca	Varchar (20)	Marca de la ambulancia

Nombre: 'modelo'		
Descripción: Es un nomenclador para el modelo de las ambulancias		
Atributo	**Tipo**	**Descripción**
cmp_id_modelo	Integer	Identificador de la tabla
cmp_modelo	Varchar (10)	Modelo de la ambulancia

Nombre: 'tb_accidente'

Descripción: Recoge los datos generales de los accidentes		
Atributo	**Tipo**	**Descripción**
cmp_id_accidente	Integer	Identificador de la tabla
cmp_fechaaccidente	Date	Fecha del accidente
cmp_responsable	Varchar (20)	Si es responsible o no
cmp_direccion	Varchar (30)	Dirección del accidente
cmp_valor	Double	Costo del accidente
cmp_paramedico	Varchar (50)	Nombre del paramédico
cmp_cant_heridos	Integer	Cantidad de heridos
cmp_cant_fallecidos	Integer	Cantidad de fallecidos
cmp_noaccidente	Integer	Número de accidente
cmp_id_categoria	Integer	Identificador de la tabla 'tb_categoriaaccidente'

Nombre: 'tb_accidente_tb_ambulancia'		
Descripción: Es la tabla que relaciona la tabla 'tb_accidente' con la tabla 'tb_ambulancia'		
Atributo	**Tipo**	**Descripción**
cmp_id_ambulancia	Integer	Identificador de la tabla

		'tb_ambulancia'
cmp_id_accidente	Integer	Identificador de la tabla 'tb_accidente'

Nombre: 'tb_ambulancia'		
Descripción: Recoge los datos generales de las ambulancias		
Atributo	**Tipo**	**Descripción**
cmp_id_ambulancia	Integer	Identificador de la tabla 'tb_ambulancia'
cmp_id_marca	Integer	Identificador de la tabla 'marca'
cmp_id_categoria	Integer	Identificador de la tabla 'marca'
cmp_id_tipoambulancia	Integer	Identificador de la tabla 'categoria'
cmp_id_motpa	Integer	Identificador de la tabla 'tb_motivopalizada'
cmp_id_combustible	Integer	Identificador de la tabla 'tb_combustible'
cmp_id_modelo	Integer	Identificador de la tabla 'modelo'

cmp_id_atencion	Integer	Identificador de la tabla 'tb_atencion'
cmp_id_base	Integer	Identificador de la tabla 'tb_bases'
cmp_id_observacion	Integer	Identificador de la tabla 'tb_observacion'
cmp_capacidad	Float	Capacidad en kg
cmp_norma_consumo	Double	Norma de consumo por Km
cmp_estado	Varchar (20)	Estado (Disponible o Baja)
cmp_anno	Integer	Año de la ambulancia
cmp_no_motor	Integer	Número de motor
cmp_no_chasis	Integer	Número de chasis
cmp_no_serie	Varchar (20)	Número de serie
cmp_no_ambulancia	Integer	Número de ambulancia
cmp_chapa	Varchar (10)	Chapa
cmp_fecha	Date	Fecha de instalación
cmp_plan_mantenimiento	Float	Plan de mantenimiento
cmp_clave	Varchar (10)	Clave, indica el estado en que

		está el vehículo

Nombre: 'tb_atencion'		
Descripción: Es un nomenclador que recoge los datos de que taller atiende las ambulancias		
Atributo	**Tipo**	**Descripción**
cmp_id_atencion	Integer	Identificador de la tabla
cmp_atencion	Varchar (30)	Describe que taller atiende la ambulancia

Nombre: 'tb_bases'		
Descripción: Es un nomenclador que recoge los datos de la base a la que pertenecen las ambulancias		
Atributo	**Tipo**	**Descripción**
cmp_id_base	Integer	Identificador de la tabla
cmp_base	Varchar (60)	Nombre de la base 'tb_accidente'

Nombre: 'tb_bateria'

Descripción: Recoge los datos generales de las baterías		
Atributo	**Tipo**	**Descripción**
cmp_id_bateria	Integer	Identificador de la tabla
Code	Char (3)	Identificador de la tabla 'tb_pais'
cmp_id_situacion	Integer	Identificador de la tabla 'tb_situacion_bateria'
cmp_id_ambulancia	Integer	Identificador de la tabla 'tb_ambulancia'
cmp_id_marca	Integer	Identificador de la tabla 'tb_marcabateria'
cmp_id_motivoretiro	Integer	Identificador de la tabla 'tb_motivoretiro_bateria'
cmp_estadobateria	Varchar (20)	Estado de la batería (Disponible o de Baja)
cmp_fecha_inst	Date	Fecha de instalación
cmp_fecha_ret	Date	Fecha de retirada
cmp_numero	Integer	Número de la batería

Nombre: 'tb_categoria'

Descripción: Es un nomenclador que recoge los datos de las categorías de las ambulancias		
Atributo	**Tipo**	**Descripción**
cmp_id_categoria	Integer	Identificador de la tabla
cmp_categoria	Varchar (30)	Categoría de la ambulancia

Nombre: 'tb_categoriaaccidente'		
Descripción: Es un nomenclador que recoge los datos de las categorías de los accidentes		
Atributo	**Tipo**	**Descripción**
cmp_id_categoria	Integer	Identificador de la tabla
cmp_categoria_accidente	Varchar (30)	Categoría del accidente

Nombre: 'tb_combustible'		
Descripción: Es un nomenclador que recoge los tipos de combustible de las ambulancias		
Atributo	**Tipo**	**Descripción**
cmp_id_combustible	Integer	Identificador de la tabla
cmp_combustible	Varchar (30)	Tipo de combustible

Nombre: 'tb_demanda'		
Descripción: Recoge los datos de una demanda		
Atributo	**Tipo**	**Descripción**
cmp_id_demanda	Integer	Identificador de la tabla
cmp_paciente	Varchar (40)	Nombre del paciente
cmp_fecha_demanda	Date	Fecha de la demanda
cmp_tipo_ambulancia	Varchar (20)	Tipo de ambulancia
cmp_chapa	Varchar (20)	Chapa de la ambulancia
cmp_origen	Varchar (40)	Lugar de salida
cmp_destino	Varchar (40)	Lugar de llegada
cmp_edad	Integer	Edad del paciente

Nombre: 'tb_marca_neumatico'		
Descripción: Es un nomenclador que recoge datos sobre las marcas de neumáticos		
Atributo	**Tipo**	**Descripción**
cmp_id_marcaneumatico	Integer	Identificador de la tabla
cmp_marca_neumatico	Varchar (30)	Marca de neumático

Nombre: 'tb_marcabateria'		
Descripción: Es un nomenclador que recoge datos sobre las marcas de baterías		
Atributo	**Tipo**	**Descripción**
cmp_id_marca	Integer	Identificador de la tabla
cmp_marca_bateria	Varchar (30)	Marca de batería

Nombre: 'tb_motivoparalizada'		
Descripción: Es un nomenclador que recoge datos sobre de motivos de paralizadas las ambulancias		
Atributo	**Tipo**	**Descripción**
cmp_id_motpa	Integer	Identificador de la tabla
cmp_motivoparalizada	Varchar (80)	Motivo de paralizada

Nombre: 'tb_motivoretiro_bateria'		
Descripción: Es un nomenclador que recoge datos sobre de motivos de retiro de las baterías		
Atributo	**Tipo**	**Descripción**
cmp_id_motivoretiro	Integer	Identificador de la tabla

cmp_motivo_retiro_bateria	Varchar (30)	Motivo de retiro

Nombre: 'tb_motivoretiro_neumatico'		
Descripción: Es un nomenclador que recoge datos sobre de motivos de retiro de los neumáticos		
Atributo	**Tipo**	**Descripción**
cmp_id_motivoretironeumatico	Integer	Identificador de la tabla
cmp_motivoretiro_neumatico	Varchar (60)	Motivo de retiro

Nombre: 'tb_ neumatico'		
Descripción: Es un nomenclador que recoge datos sobre de motivos de retiro de los neumáticos		
Atributo	**Tipo**	**Descripción**
cmp_id_neumatico	Integer	Identificador de la tabla
cmp_id_motivoretironeumatico	Integer	Motivo de retiro
Code	Char (3)	Identificador de la tabla 'tb_pais'
cmp_id_tiponeumatico	Integer	Identificador de la tabla

		'tb_tipo_neumatico'
cmp_id_situacionneumatico	Integer	Identificador de la tabla 'tb_situacion_neumatico'
cmp_id_ambulancia	Integer	Identificador de la tabla 'tb_ambulancia'
cmp_id_marcaneumatico	Integer	Identificador de la tabla 'tb_marca_neumatico'
cmp_no_capas	Integer	Número de capas
cmp_estadoneumatico	Varchar (20)	Estado del neumático (Disponible o de Baja)
cmp_medida	Double	Medida
cmp_fecha	Date	Fecha de instalado
cmp_no_neumatico	Integer	Número de neumático
cmp_km_recorridos	Float	Km recorridos
cmp_posicion	Varchar (30)	Posición
cmp_fecha_retiro	Date	Fecha de retirado

Nombre: 'tb_observacion'
Descripción: Es un nomenclador que recoge datos sobre los tipos de observación (Trabajando o Paralizada)

Atributo	Tipo	Descripción
cmp_id_observacion	Integer	Identificador de la tabla
cmp_observacion	Varchar (30)	Observación

Nombre: 'tb_pais'		
Descripción: Recoge los nombres de los países existente		
Atributo	**Tipo**	**Descripción**
Code	Char (3)	Identificador de la tabla
Name	Varchar (52)	Nombre del país
LocalName	Varchar (45)	Local de un país

Nombre: 'tb_situacion_bateria'		
Descripción: Es un nomenclador que recoge datos sobre las situación en que se encuentran las baterías		
Atributo	**Tipo**	**Descripción**
cmp_id_situacion	Integer	Identificador de la tabla
cmp_situacion_bateria	Varchar (30)	Situación

Nombre: 'tb_situacion_neumatico'		
Descripción: Es un nomenclador que recoge datos sobre las situación en que se encuentran las neumáticos		
Atributo	**Tipo**	**Descripción**
cmp_id_situacionneumatico	Integer	Identificador de la tabla
cmp_situacion_neumatico	Varchar (30)	Situación

Nombre: 'tb_tipo_neumatico'		
Descripción: Es un nomenclador que recoge datos sobre los tipos de neumáticos		
Atributo	**Tipo**	**Descripción**
cmp_id_tiponeumatico	Integer	Identificador de la tabla
cmp_tipo_neumatico	Varchar (30)	Tipo de neumático

Nombre: 'tb_tipoambulancia'		
Descripción: Es un nomenclador que recoge datos sobre los tipos de ambulancias		
Atributo	**Tipo**	**Descripción**
cmp_id_tipoambulancia	Integer	Identificador de la tabla
cmp_tipoambulancia	Varchar (50)	Tipo de ambulancia

2.5 Generalidades de la implementación

2.5.1 Modelo de Despliegue

El diagrama de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de como se distribuye la funcionalidad entre los nodos de cómputo. Es una colección de nodos y arcos; donde cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar.

Muestra la configuración de los componentes hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución. En este tipo de diagramas intervienen nodos, asociaciones de comunicación, componentes dentro de los nodos y objetos que se encuentran a su vez dentro de los componentes. Un nodo es un objeto físico en tiempo de ejecución, es decir una máquina que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento, a su vez puede estar formada por otros componentes.

El diagrama de despliegue muestra la topología del hardware sobre el que se ejecuta el sistema.

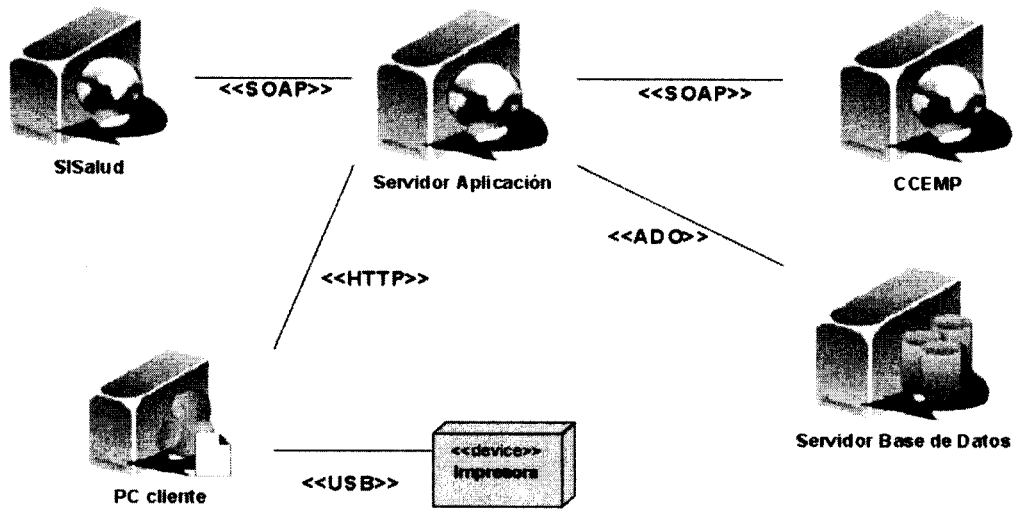


Fig. Diagrama de Despliegue

Conclusiones

En este capitulo se describe como está implementado el sistema y se realiza una valoración crítica del diseño propuesto por el analista. También se analizan los módulos que se necesitan para funcionar. Se brinda una descripción sobre las tablas de la base de datos y sobre las clases más importantes, ya sean clases controladoras, clases de interfaz o clases de entidades que conforman la solución propuesta, permitiendo conocer la distribución de las mismas, y facilitando su entendimiento para futuras actualizaciones.

CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

En este capítulo se realizara un análisis y descripción de los posibles "Test de unidades" para validar la solución propuesta. Se realiza una descripción de los valores utilizados para los test, así como una evaluación de la ejecución y los resultados obtenidos.

3.1 Búsqueda o diseño de los test de unidades que permitan validar la solución propuesta

3.1.1 Niveles de Prueba

Existen diferentes niveles de pruebas, cada una de ellas se realiza en determinados momentos del ciclo de vida del software, la siguiente tabla resume las mismas:

- Unidad.
- Integración.
- Sistema.
- Aceptación.

Este modelo describe a un nivel alto de abstracción las fases del ciclo de desarrollo en las que se involucran las pruebas y los niveles de las mismas, la figura a continuación muestra este modelo.

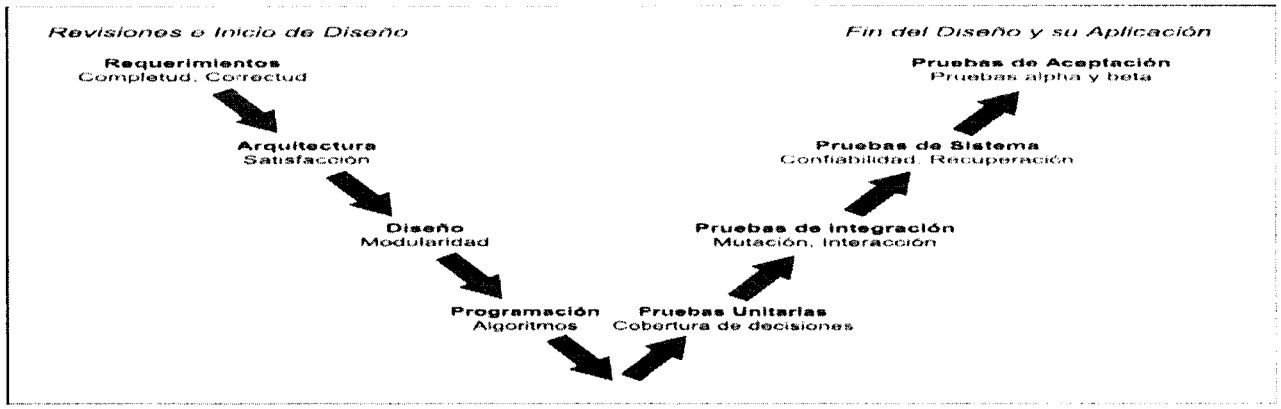


Fig. Niveles de prueba con las fases de desarrollo de software

Nivel de Unidad

En este nivel de prueba fundamentalmente se ejecutan casos de pruebas de caja blanca diseñados para:

✓ Probar las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente, conservan su integridad durante todos los pasos de ejecución del algoritmo.

✓ Probar las condiciones límites para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento.

✓ Ejercitar todos los caminos independientes (caminos básicos) de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez. Y, finalmente, se prueban todos los caminos de manejo de errores.

Los casos de pruebas diseñados para ejecutar a nivel de unidad deben descubrir errores como:

- ✓ Comparaciones entre tipos de datos distintos.
- ✓ Operadores lógicos o de procedencia incorrectos.
- ✓ Igualdad esperada cuando los errores de precisión la hacen poco probable.
- ✓ Variables o comparaciones incorrectas.
- ✓ Terminación de bucles inapropiada o inexistente.
- ✓ Fallo de salida cuando se encuentra una iteración divergente.
- ✓ Variables de bucles modificadas de forma inapropiada.

La prueba de límites es probablemente la más importante, es una tarea del paso de la prueba de unidad. El software falla con frecuencia en sus condiciones límites. Las pruebas que ejercitan las

estructuras de datos, el flujo de control y los valores de los datos por debajo y por encima de los máximos y los mínimos son muy apropiadas para descubrir estos errores. (8)

Nivel de Integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño. En el proceso de integración puede derivarse la realización de los casos de uso del sistema, estos describen como interactúan las clases y los objetos y por lo tanto la interacción de los componentes. (9)

Los encargados de diseñar los casos de pruebas de integración consideran como entrada a los diagramas de secuencia de las realizaciones de los casos de uso, pues ahí buscan las combinaciones de entradas, salidas y estado inicial del sistema que dan lugar a escenarios que usan las clases y por lo tanto los componentes que participan en los diagramas.

Un caso de prueba se deriva de un diagrama de secuencia, este describe una o varias secuencias del negocio. Después, se procede a ejecutar el caso de prueba de integración diseñado, creando trazas de ejecución o ejecutándolo paso a paso, a continuación se compara las interacciones actuales con los diagramas de secuencia y de no ser igual se trata de un defecto que puede dar al traste con un error.

Nivel de Sistema

Las pruebas de sistemas caen fuera del ámbito del proceso de software y no las realiza únicamente el desarrollador del software. Sin embargo los pasos durante el diseño del software y durante la prueba pueden mejorar enormemente la posibilidad de de éxito de las pruebas de sistema. Las pruebas de sistema principalmente se centran en verificar la interacción de los actores con el sistema, por lo que a menudo los casos de pruebas se obtienen a partir de las descripciones de los casos de uso. Aunque también se le aplican prueba al sistema como un todo. (10)

Nivel de Aceptación

La mayoría de los desarrolladores de productos de software llevan a cabo un proceso denominado pruebas Alfa o Beta para descubrir errores considerando que solo el usuario final puede descubrir.

La prueba Alfa es llevada a cabo por el cliente en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador de observador registrando los errores y los problemas de uso. Estas pruebas se llevan a acabo en un entorno controlado.
Las pruebas Betas se llevan a acabo por los usuarios finales en sus puestos de trabajo, y a diferencia de la prueba Alfa el desarrollador no está presente, así que esta prueba no puede ser controlada por el desarrollador, por lo que el cliente debe registrar todas las inconformidades y tramitárselas al desarrollador. (11)

3.1.2 Métodos de Pruebas

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

La prueba de caja blanca del software se comprueba los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado.

3.1.3 Tipos de Pruebas

Caja blanca

Consiste en realizar pruebas para verificar que líneas específicas de código funcionan tal como está definido. También se le conoce como prueba de caja-transparente. Esta se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado o afirmado. Esto genera gran cantidad de caminos posibles por lo que hay que dedicar esfuerzos a la determinación de las condiciones de prueba que se van a verificar. (12)

Estas tienen como objetivos:

- ✓ Garantizar que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo,

programa o método.

- ✓ Ejercitar todas las decisiones lógicas en las vertientes verdadera y falsa.
- ✓ Ejecutar todos los bucles en sus límites operacionales.
- ✓ Ejercitar las estructuras internas de datos para asegurar su validez.

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

Existen varias técnicas para la aplicación de las pruebas de Caja Blanca, en este trabajo se utilizará la técnica del camino básico. Los pasos que se siguen para aplicar esta técnica son:

1. Se numeran las instrucciones del algoritmo y se construye el grafo de flujo. Las instrucciones simples se agrupan en un mismo nodo. Las condiciones compuestas se separan en varios nodos.

2. Calcular la complejidad ciclomática del grafo. Ésta indica el número máximo de caminos linealmente independientes dentro del grafo.

V (G) = Nº de regiones del grafo

V (G) = Nº de Aristas - Nº de Nodos + 2

V (G) = Nº de Nodos Predicado + 1

3. Diseñar los caminos linealmente independientes de menor a mayor de manera que exista como mucho una arista de diferencia entre ellos.

4. Elaborar los casos de prueba (datos de entrada al algoritmo y resultados esperados) para cada uno de los caminos.

Caja negra

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose

en los requisitos funcionales del sistema y ejercitándolos. (13)

La misma no es una alternativa a las técnicas de prueba anteriormente vistas, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la Caja Blanca.

Estas pruebas permiten encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

Para preparar los casos de pruebas hacen falta un número de datos que ayuden a la ejecución de los estos casos y que permitan que el sistema se ejecute en todas sus variantes, pueden ser datos válidos o inválidos para el programa según si lo que se desea es hallar un error o probar una funcionalidad. Los datos se escogen atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa corra bien.

Para esta prueba, existen varias técnicas, entre ellas están:

1. Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

2. Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases

de prueba que hay que desarrollar. En este apartado se hará uso de esta técnica para aplicarle a cierto caso de uso del sistema desarrollado el método de la caja negra.

3.2 Descripción de los test de caja negra

El objetivo del test es validar la totalidad de los casos de usos implementados a través de casos de pruebas de caja negra.

Alcance

Pruebas de Sistema: Consiste en realizarle pruebas al sistema cuando ya está integrado y consiste en probar los requisitos funcionales definidos por el cliente, en este nivel se realizan principalmente pruebas por el método de prueba de caja negra.

3.3 Evaluación de la ejecución del test y de los resultados obtenidos

Teniendo en cuenta el caso de uso Gestionar Ambulancia

Las pruebas realizadas a este caso de uso son:

- ✓ **Rf1** Insertar los datos ambulancias.
- ✓ **Rf2** Buscar ambulancias.
- ✓ **Rf3** Visualizar los datos de ambulancias.
- ✓ **Rf4** Modificar los datos de ambulancias.

Caso de Prueba del Rf1: Insertar los datos Ambulancia

Descripción

El Caso de Uso se inicia cuando el usuario necesita insertar una ambulancia. Selecciona la opción de deseada en el menú principal de trabajo y habiendo sido localizada esta, llena los datos de la misma, finalizando el caso de uso.

Flujo central

- ✓ El sistema muestra la interfaz que permite registrase al usuario.
- ✓ Si el usuario se registra correctamente le permite acceder a la aplicación a las páginas a las

que el mismo tiene acceso dentro de las cual se va encontrar el menú de trabajo.

- ✓ El usuario selecciona la opción "Insertar Ambulancia" del menú de trabajo.
- ✓ El usuario llena los datos para insertar ambulancia.
- ✓ El usuario pulsa el botón "Aceptar".

Condiciones de ejecución

- ✓ Usuario autentificado satisfactoriamente por el sistema.

Tienen que estar presente en la base de datos

- ✓ Tipos de combustibles.
- ✓ Marcas de Ambulancias.
- ✓ Modelos de Ambulancias.
- ✓ Categorías de Ambulancias.
- ✓ Tipos de Vehículos.
- ✓ Atenciones.
- ✓ Observaciones.

Clases válidas	**Clases no válidas**	**Resultados de la prueba**	**Resultados de la prueba**	**Observaciones**	**Cumplimiento**
El usuario una vez utenticado selecciona la opción Insertar Ambulancia" en el menú de trabajo	No se muestra la interfaz de insertar ambulancia	Muestra la interfaz de insertar ambulancia	Satisfactorias	La operación se realizó correctamente	100 %
El usuario inserta los datos de la ambulancia	No se insertan los datos de la ambulancia	Si se insertan los datos de la ambulancia	Satisfactorias	La operación se realizó correctamente	100 %

Caso de Prueba del Rf2: Buscar ambulancias

Descripción

El Caso de Uso se inicia cuando el usuario necesita buscar una o varias ambulancias. Selecciona la opción "Buscar Ambulancias" en el menú principal de trabajo y habiendo entrada a la página correspondiente, elige el criterio de búsqueda y selecciona el botón "Buscar", finalizando el caso de uso.

Flujo central

- ✓ El sistema muestra la interfaz que permite registrase al usuario.
- ✓ Si el usuario se registra correctamente le permite acceder a la aplicación a las páginas a las que el mismo tiene acceso dentro de las cual se va encontrar el menú de trabajo.
- ✓ El usuario selecciona la opción "Buscar Ambulancias" del menú de trabajo.
- ✓ El usuario busca por algún criterio de búsqueda.
- ✓ El usuario pulsa el botón "Aceptar".
- ✓ El usuario obtiene parte de la información de la o las ambulancias buscadas.

Condiciones de ejecución

- ✓ Usuario autentificado satisfactoriamente por el sistema.
- ✓ Categorías de ambulancias (en caso de que existan ambulancias insertadas previamente y se elija este criterio).
- ✓ Las observaciones (en caso de que existan ambulancias insertadas previamente y se elija este criterio).
- ✓ Motivos de paralizadas (en caso de que existan ambulancias paralizadas y se elija este criterio).

Clases válidas	Clases no válidas	Resultados de la prueba	Resultados de la prueba	Observaciones	Cumplimiento

El usuario una vez autenticado selecciona la opción "Buscar Ambulancias" en el menú de trabajo	No se muestra la interfaz de insertar ambulancia	Muestra la interfaz de buscar ambulancia	Satisfactorias	La operación se realizó correctamente	100 %
El usuario elige los criterios de búsqueda y selecciona el botón "Buscar"	El sistema no muestra ningún resultado	El sistema muestra como resultados las ambulancias que coinciden con los criterios de búsqueda entrados	Satisfactorias	La operación se realizó correctamente	100 %

Caso de Prueba del Rf3: Visualizar los datos de ambulancia

Descripción

El Caso de Uso se inicia cuando el usuario necesita ver alguna información de una o varias ambulancias. Selecciona la opción "Buscar Ambulancias" en el menú principal de trabajo y habiendo entrada a la página correspondiente, elige el criterio de búsqueda y selecciona el botón "Buscar", aparecerán las ambulancias según el criterio en otra página en la cual escogerá la ambulancia de la cual desea ver sus datos, al dar clic sobre la imagen de visualizar se le mostrarán los datos correspondientes, finalizando el caso de uso.

Flujo central

✓ El sistema muestra la interfaz que permite registrase al usuario.

✓ Si el usuario se registra correctamente le permite acceder a la aplicación a las páginas a las que el mismo tiene acceso dentro de las cual se va encontrar el menú de trabajo.

✓ El usuario selecciona la opción "Buscar Ambulancias" del menú de trabajo.

✓ El usuario busca por algún criterio de búsqueda.

✓ El usuario pulsa el botón "Aceptar".

✓ El usuario obtiene parte de la información de la o las ambulancias buscadas.

✓ El usuario pulsa en el ícono de visualizar.

✓ El usuario obtiene la información de la ambulancia de la cual desea ver sus datos.

Condiciones de ejecución

✓ Usuario autentificado satisfactoriamente por el sistema.

✓ Categorías de ambulancias (en caso de que existan ambulancias insertadas previamente y se elija este criterio).

✓ Las observaciones (en caso de que existan ambulancias insertadas previamente y se elija este criterio).

✓ Motivos de paralizadas (en caso de que existan ambulancias paralizadas y se elija este criterio).

Clases válidas	**Clases no válidas**	**Resultados de la prueba**	**Resultados de la prueba**	**Observaciones**	**Cumplimiento**
El usuario una vez autenticado selecciona la opción "Buscar Ambulancias" en el menú de trabajo	No se muestra la interfaz de insertar ambulancia	Muestra la interfaz de buscar ambulancia	Satisfactorias	La operación se realizó correctamente	100 %

El usuario elige los criterios de búsqueda y selecciona el botón "Buscar"	El sistema no muestra ningún resultado	El sistema muestra como resultados las ambulancias que coinciden con los criterios de búsqueda entrados	Satisfactorias	La operación se realizó correctamente	100 %
El usuario elige en el ícono visualizar la ambulancia deseada	El sistema no muestra ningún resultado	El sistema muestra los datos de la ambulancia deseada	Satisfactorias	La operación se realizó correctamente	100 %

Caso de Prueba del Rf4: Modificar los datos de ambulancias

Descripción

El Caso de Uso se inicia cuando el usuario necesita modificar alguna información de una o varias ambulancias. Selecciona la opción "Buscar Ambulancias" en el menú principal de trabajo y habiendo entrada a la página correspondiente, elige el criterio de búsqueda y selecciona el botón "Buscar", aparecerán las ambulancias según el criterio en otra página en la cual escogerá la ambulancia a la cual desea modificar sus datos, al dar clic sobre la imagen de modificar se le mostrarán los datos correspondientes a la misma y le entrará los nuevos datos, finalizando el caso de uso.

Flujo central

- ✓ El sistema muestra la interfaz que permite registrase al usuario.
- ✓ Si el usuario se registra correctamente le permite acceder a la aplicación a las páginas a las

que el mismo tiene acceso dentro de las cual se va encontrar el menú de trabajo.

- ✓ El usuario selecciona la opción "Buscar Ambulancias" del menú de trabajo.
- ✓ El usuario busca por algún criterio de búsqueda.
- ✓ El usuario pulsa el botón "Aceptar".
- ✓ El usuario obtiene parte de la información de la o las ambulancias buscadas.
- ✓ El usuario pulsa en el ícono de modificar ambulancia.
- ✓ El usuario obtiene la información de la ambulancia que quiere modificar.
- ✓ El usuario modifica la información de la ambulancia.

Condiciones de ejecución

- ✓ Usuario autentificado satisfactoriamente por el sistema.
- ✓ Categorías de ambulancias (en caso de que existan ambulancias insertadas previamente y se elija este criterio).
- ✓ Las observaciones (en caso de que existan ambulancias insertadas previamente y se elija este criterio).
- ✓ Motivos de paralizadas (en caso de que existan ambulancias paralizadas y se elija este criterio).

Clases válidas	**Clases no válidas**	**Resultados de la prueba**	**Resultados de la prueba**	**Observaciones**	**Cumplimiento**
El usuario una vez autenticado selecciona la opción "Buscar Ambulancias" en el menú de trabajo	No se muestra la interfaz de insertar ambulancia	Muestra la interfaz de buscar ambulancia	Satisfactorias	La operación se realizó correctamente	100 %

El usuario elige los criterios de búsqueda y selecciona el botón "Buscar"	El sistema no muestra ningún resultado	El sistema muestra como resultados las ambulancias que coinciden con los criterios de búsqueda entrados	Satisfactorias	La operación se realizó correctamente	100 %
El usuario elige en el ícono modificar ambulancia	El sistema no muestra ningún resultado	El sistema muestra los datos de la ambulancia que desea modificar	Satisfactorias	La operación se realizó correctamente	100 %
El usuario inserta nuevos datos y pulsa el botón "Modificar"	El sistema no modifica los datos	El sistema modifica los datos	Satisfactorias	La operación se realizó correctamente	100 %

Una de las últimas fases del ciclo de vida antes de entregar un programa para su explotación, es la fase de pruebas. Esta fase del desarrollo de un software es la que mayor cantidad de tiempo y de esfuerzo requiere, se estima que la mitad del esfuerzo de desarrollo de un programa tanto en tiempo como en gastos se invierte en esta.

Esta fase añade valor al producto, todos los programas poseen errores y la fase de pruebas los descubre, siendo este el valor que le añade. Mientras más errores se encuentren al software en esta fase mejor. Una prueba del software es un conjunto de actividades que se lleva a cabo sistemáticamente, que puede planificarse por adelantado y ejecutar una vez construido el código para

la revisión final de las especificaciones, del diseño y de la codificación del software.

Conclusiones

En este capítulo se realizó un estudio de los principales niveles, métodos y tipo de prueba que se llevan a cabo durante el ciclo de vida del software. Se describieron los test de caja negra y los valores utilizados, así como, la ejecución de los mismos y algunos resultados obtenidos. Al caso de uso Gestionar Ambulancia se le realizaron las siguientes pruebas: Insertar datos de ambulancias, Buscar ambulancias, Visualizar datos de ambulancias y Modificar ambulancias, en los cuales no se detectaron errores.

CONCLUSIONES

En el presente trabajo, se llegaron a las siguientes conclusiones:

- Se estudiaron de forma satisfactoria, sistemas informáticos vinculados con la gestión de la información de las bases de ambulancias.
- Se realizó un estudio de las principales herramientas y tecnologías usadas en el proyecto, lo cual permitió desarrollar la aplicación.
- Se estudió el diseño propuesto por las analistas para tener una visión más amplia del mismo, resultando más fácil la comprensión del trabajo para poder avanzar más rápido en la implementación.
- Se ha cumplido el objetivo de la investigación, pues se realizó la implementación del sistema que permite controlar la gestión de la información relacionada con las operaciones en las Bases de Ambulancias de Cuba.

RECOMENDACIONES

Se recomienda a la Facultad y a los desarrolladores del Área Temática Sistemas Especializados:

- Desplegar la aplicación en las Bases de Ambulancias para la realización de las pruebas piloto.
- Desarrollar nuevas funcionalidades para el sistema.
- La implementación de una versión que utilice como gestor de base de datos PostgresSQL por ser un gestor libre ya que se utilizó MySQL que actualmente es software propietario.
- Definir un plan de capacitación que permita el adiestramiento del personal de las Bases de Ambulancias.

REFERENCIA BIBLIOGRÁFICA

1. WebEstilo. [En línea] Febrero de 2008. http://www.webestilo.com/html/.

2. Herramientas para desarrolladores. [En línea] Febrero de 2008. http://www.elcodigo.net/tutoriales/javascript/javascript1.html.

3. Manual de PHP. [En línea] enero de 2008. http://es.php.net/manual/es/introduction.php.

4. Maestros del Web. [En línea] marzo de 2008. http://www.maestrosdelweb.com/editorial/ajax.

5. Informatizate. [En línea] Marzo de 2008. www.informatizate.net/articulos/pdfs/metodologias_de_desarrollo_de_software_07062004.pdf.

6. Modelo Vista Controlador. [En línea] Marzo de 2008. http://es.wikipedia.org/wiki/Modelo_Vista_Controlador.

7. Patrón ActiveRecord. [En línea] Marzo de 2008. http://es.wikipedia.org/wiki/Patr%C3%B3n_ActiveRecord.

8. PRESSMAN, R. *Ingeniería del Software: Un enfoque práctico.* Madrid: McGraw Hill Prentice-Hall.

9. Ídem a referencia 8.

10. Ídem a referencia 8.

11. Ídem a referencia 8.

12. Ídem a referencia 8.

13. Ídem a referencia 8.

BIBLIOGRAFÍA

1. ANAGO-Mantenimiento de Flota [En línea]. - 28 de Enero de 2008. - http://www.anago.com.ar.

2. Auto Soft Taller [En línea]. - Noviembre de 2008. - http://www.autosofttaller.com.

3. Ciberaula [En línea]. - Mayo de 2008. - http://linux.ciberaula.com/articulo/linux_apache_intro.html.

4. cuervachoo [En línea]. - Febrero de 2008. - http://cuervachoo.bloringa.net/post-899560.html.

5. desarrolloweb.com [En línea]. - Abril de 2008. - http://www.desarrolloweb.com/articulos/332.php.

6. error500 [En línea]. - Mayo de 2008. - http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_bas.

7. Free Download Manager [En línea]. - Abril de 2008. - http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5Bcuenta_de_Linux_14716_p.

8. Herramientas para desarrolladores. [En línea] Febrero de 2008.
http://www.elcodigo.net/tutoriales/javascript/javascript1.html.

9. Informatizate. [En línea] Marzo de 2008.
www.informatizate.net/articulos/pdfs/metodologias_de_desarrollo_de_software_07062004.pdf.

11. Jacobson, I.; Booch, G. y Rumbaugh, J. El Proceso Unificado de Desarrollo de software. Addison-Wesley

12. Java Script [En línea]. - Marzo de 2008. - http://es.wikipedia.org/wiki/JavaScript.

13. Lenguajes del lado servidor o cliente [En línea]. - Mayo de 2008. - http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

14. Maestros del Web. [En línea] marzo de 2008. http://www.maestrosdelweb.com/editorial/ajax.

15. Mantenimiento de Flota v1.4 [En línea]. - Enero de 2008. - http://www.zeroprogramas.com/programas/mantenimiento-de-flotas-v1-4.asp.

16. masadelante.com [En línea]. - Marzo de 2008. - http://www.masadelante.com/faq-tipos-de-servidores.htm.

17. Manual de PHP. [En línea] enero de 2008. http://es.php.net/manual/es/introduction.php.

18. Modelo Vista Controlador. [En línea] Marzo de 2008.
http://es.wikipedia.org/wiki/Modelo_Vista_Controlador.

19. MySQL con Clase [En línea]. - Mayo de 2008. - http://mysql.conclase.net/curso/index.php.

20. Nideaderedes [En línea]. - Marzo de 2008. - http://nideaderedes.urlansoft.com/2007/02/27/code-igniter-framework-php/.

21. Patrón ActiveRecord. [En línea] Marzo de 2008.-
http://es.wikipedia.org/wiki/Patr%C3%B3n_ActiveRecord.

22. PRESSMAN, R. *Ingeniería del Software: Un enfoque práctico.* Madrid: McGraw Hill Prentice-Hall.

23. pixelco.us blog [En línea]. - Mayo de 2008. - http://pixelco.us/blog/codeigniter-framework-php/.

24. Red de Software [En línea]. - Noviembre de 2007. - http://espanol.softpicks.net/software/Taller-de-reparaci-oacute-n-de-autom-oacute-viles_es-32005.htm.

25. servicios globales de BT [En línea]. - Mayo de 2008. -
http://www.bt.es/perspectives/serviciosweb.html.

26. Supplest [En línea]. - 26 de Enero de 2008. - www.supplest.com/sp/soluciones/sgfv.htm.

27. tufunción [En línea]. - Febrero de 2008. - http://www.tufuncion.com/zend-studio.

28. Uruman [En línea]. - Febrero de 2008. -
http://www.uruman.org/material_tecnico/X%20Autores/GSilva.pdf.

29. Vehículos Pro [En línea]. - Enero de 2008. - http://www.tinitasoft.com/.

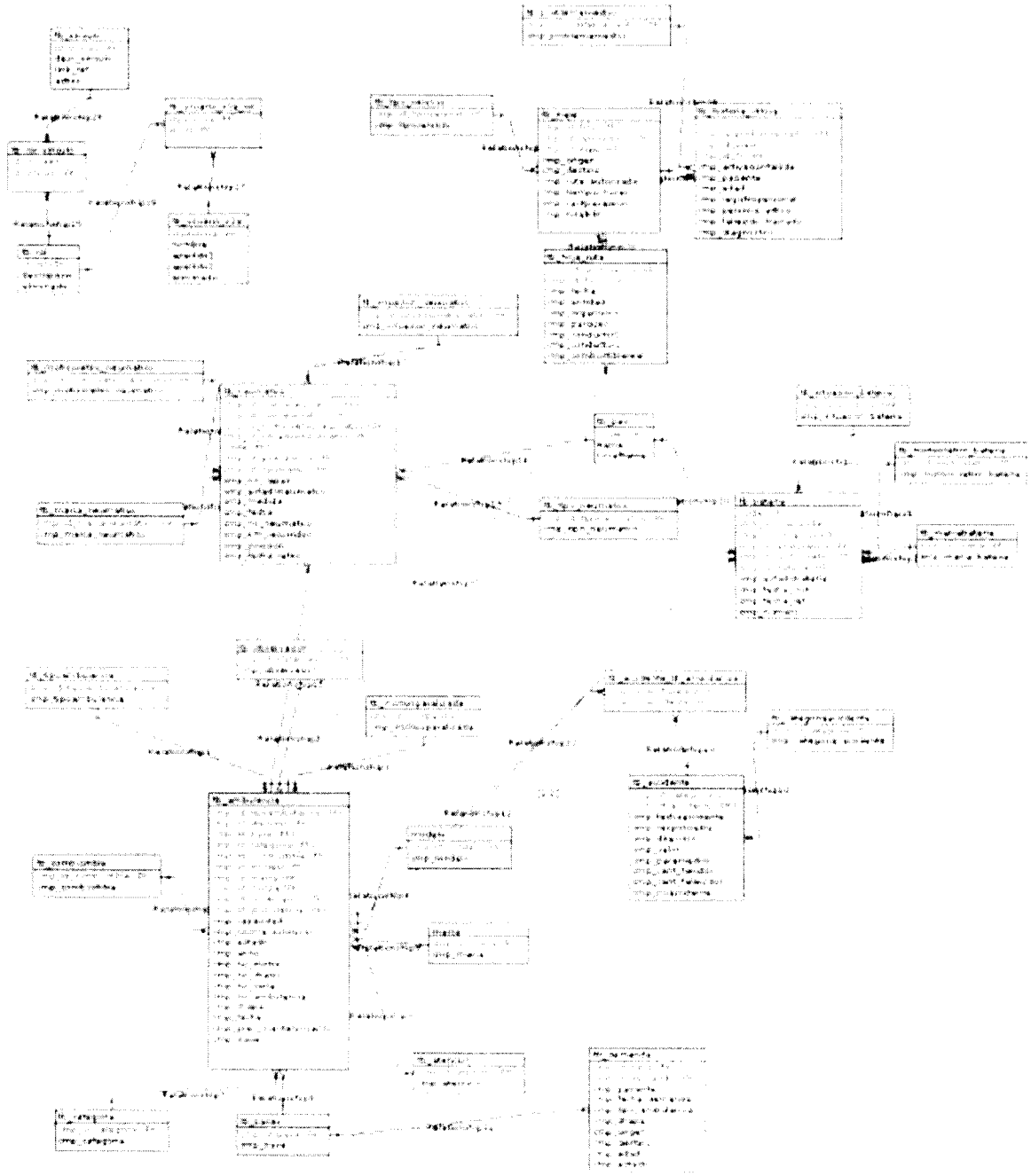
30. vico.org [En línea]. - Mayo de 2008. - http://www.vico.org/TRAD_obert/TRAD_WAE_abierto.html.

31. WebEstilo. [En línea] Febrero de 2008. http://www.webestilo.com/html/.

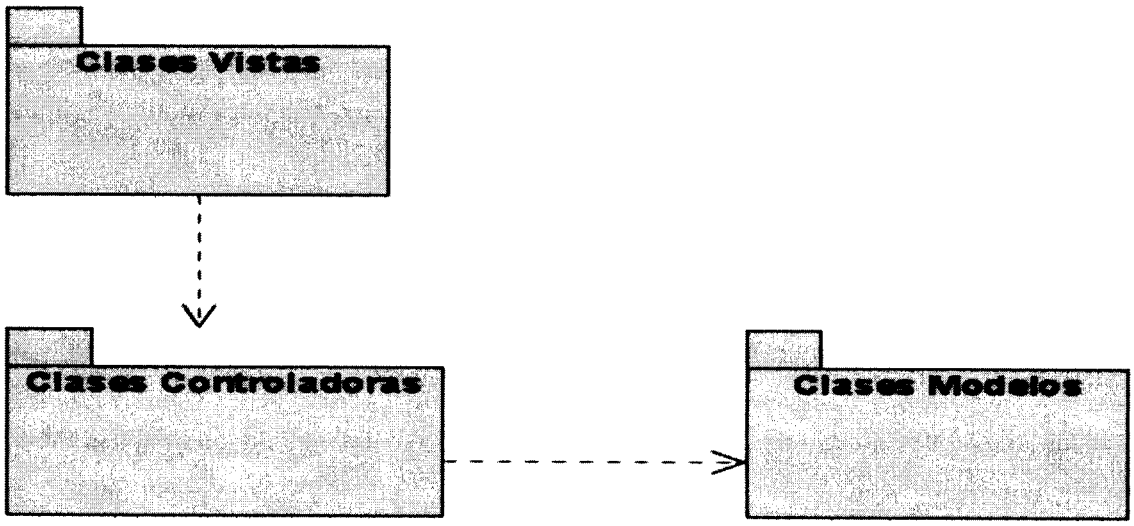
32. Wikipedia [En línea]. - Mayo de 2008. - http://es.wikipedia.org/wiki/MySQL.

ANEXOS

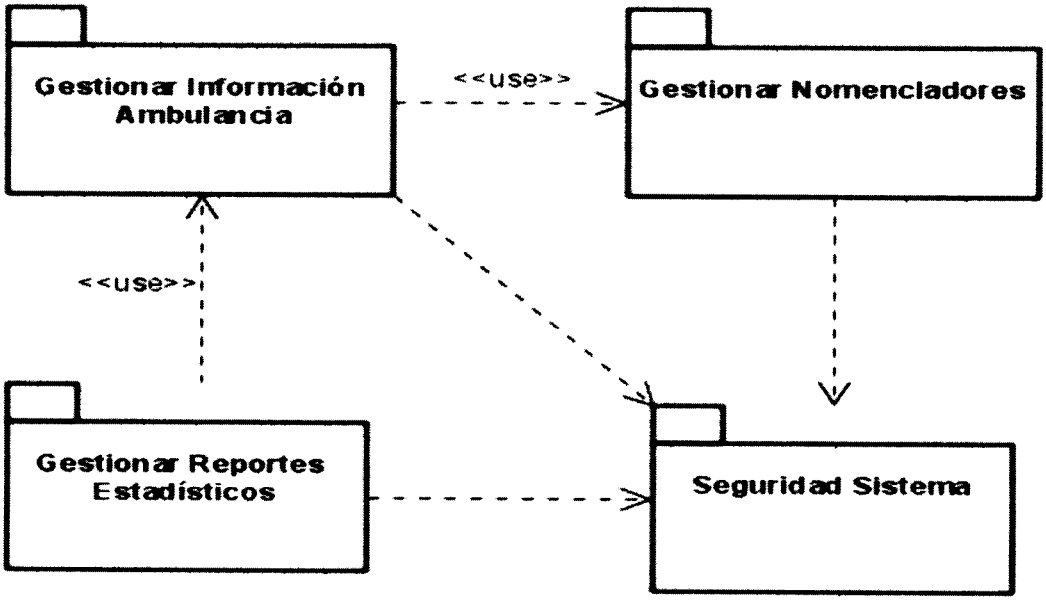
Anexo No. 1 Modelo de Datos



Anexo No. 2 Organización del sistema según su arquitectura



Anexo No. 3 Diagrama de paquetes del sistema



GLOSARIO DE TÉRMINOS:

Accidente del tránsito: Hecho que ocurre en la vía, donde intervienen por lo menos un vehículo en movimiento y que produce como resultado del mismo la muerte, lesiones de personas o daños materiales.

Batería: Es un recipiente capaz de almacenar la energía de la corriente producida por el alternador.

Coeficiente de Disponibilidad Técnica (CDT): Es el indicador fundamental mediante el cual se evalúa el estado técnico del vehículo.

Complejidad Ciclomática: Es una métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa, da el límite superior del número de pruebas que se deben realizar.

Demanda: Solicitud de atención especializada a un caso de emergencia o urgencia médica.

Disponibilidad Técnica: Seguridad que una ambulancia sea operable en un tiempo dado.

Electrónica: Es la rama de la física y fundamentalmente una especialización de la ingeniería que estudia y emplea sistemas cuyo funcionamiento se basa en la conducción y el control de flujo microscópico de los electrones u otras partículas cargadas eléctricamente.

Emergencia: Son las urgencias de primera prioridad: enfermos o lesionados con peligro vital inmediato real o potencial. Su traslado requiere de Apoyo Vital Avanzado, después de las primeras medidas de estabilización.

Expedir Ambulancia: Dar curso a la demanda de emergencia o urgencia médica enviando una ambulancia para atenderla.

FIREWALL: Programa designado para prevenir el acceso no autorizado desde y hacia una red privada.

HTML: Lenguaje de marcado de hipertexto, es el lenguaje autoritario para crear documentos en la World Wide Web. Define la estructura de un documento web usando etiquetas y atributos.

Informática: Es la disciplina que estudia el tratamiento automático de la información utilizando dispositivos electrónicos y sistemas computacionales. También es definida como el procesamiento de información en forma automática. Para esto los sistemas informáticos deben realizar las siguientes tareas básicas: Entrada: Captación de información., Procesamiento o tratamiento de dicha información, Salida: Transmisión de resultados binarios.

Internet: Sistema de redes de computación ligadas entre sí, con alcance mundial, que facilita servicios de comunicación de datos como registro remoto, transferencia de archivos, correo electrónico y grupos de noticias. Internet es una forma de conectar las redes de computación existentes que amplía en gran medida el alcance de cada sistema participante.

ISO: Organización internacional de estandarización. Ha definido un número importante de estándares computacionales.

Mantenimiento técnico: Es el conjunto de operaciones que se le realiza a un vehículo con el objetivo de restablecer y mantener su capacidad de trabajo.

MVC: Modelo Vista Controlador.

Neumático: Es un recipiente de aire que actúa de enlace entre el vehículo y la superficie de contacto por lo que rueda.

Norma de Mantenimiento: Es la cantidad de kilómetros que deben consumir los vehículos para realizar las operaciones de mantenimientos.

ODBC: Es un estándar de acceso a Bases de Datos desarrollado por Microsoft Corporation, el objetivo de ODBC es hacer posible acceder a cualquier dato desde cualquier aplicación, sin importar, qué Sistema Gestor de Bases de Datos (DBMS por sus siglas en inglés) almacene los datos, ODBC logra esto al insertar una capa intermedia llamada manejador de Bases de Datos, entre la aplicación y el DBMS, el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el DBMS entienda.

Operaciones: Conjunto de acciones realizadas por la Base de Ambulancia respecto a sus vehículos como consumo de combustible, viajes realizados, pacientes trasladados, mantenimiento etc.

PostgreSQL: Es un gestor de base de datos muy usado en la actualidad debido a que pertenece a la familia de software libre.

Requerimiento: Funcionalidad requerida por un usuario para resolver un problema o satisfacer uno o varios objetivos.

RUP: Metodología de desarrollo de software basada en UML. Organiza el desarrollo de software en 4 fases.

SGML: Consiste en un sistema para la organización y etiquetado de documentos. La Organización Internacional de Estándares (ISO) ha normalizado este lenguaje en 1986.

Sistemas de gestión de base de datos (SGBD): (en inglés: Database management system, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Sitio Web: Es un conjunto de páginas web, típicamente comunes a un dominio de Internet o subdominio en la World Wide Web en Internet.

Usuario autenticado: Es aquel que ha proporcionado información mediante la cual el mecanismo de seguridad garantiza su identificación al intentar acceder a los componentes del sistema.