Universidad de las Ciencias Informáticas

Facultad 3

Título: Sistema de Gestión de Mantenimiento Vehicular v1.0: Diseño e Implementación del proceso de Administración de Inspecciones Técnicas y Accidentes.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Lisvan Porro Pimienta

Tutor(es): Ing. Mailyn Hernández Gómez

Ing. Pedro Manuel Alas Verdecia.

"Año 54 de la Revolución"

La Habana, Cuba, Mayo 2012

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los días del mes de junio del año 2011.

Lisvan Porro Pimienta

_____
Firma del Autor

Ing. Mailyn Hernández Gómez.

_____
Firma del Tutor

Ing. Pedro Manuel Alas Verdecia.

_____
Firma del Tutor

AGRADECIMIENTOS

Quiero agradecer:

A mis padres por su apoyo incondicional, su amor, su confianza, su comprensión, su dedicación y orgullo hacia mi persona.

A mi abuela por todo el cariño y amor que me ha brindado.

A mi tía y mi prima por el cariño y la confianza que siempre han depositado en mí.

A mi tutora Mailyn por la ayuda y la dedicación que me ha brindado.

A mis amigos y amigas, por su ayuda y apoyo incondicional.

A la revolución que ha hecho posible la realización de un sueño.

A todos aquellos que han aportado un granito de arena en mi formación.

A todos en general muchas gracias.

RESUMEN

En la Universidad de las Ciencias Informáticas, específicamente en el Departamento de Soluciones Empresariales se desarrolla el Sistema de Gestión de Mantenimiento Vehicular para el Cuerpo de la Policía Nacional Bolivariana. Este sistema incluye entre los procesos que automatiza la Administración de Inspecciones Técnicas y Accidentes.

El objetivo del presente trabajo de diploma es presentar el diseño e implementación de las funcionalidades desarrolladas para estos procesos, a través de las cuales se llevará el control de los accidentes en los que incurren las unidades policiales y las inspecciones técnicas que se le realizan a las mismas.

Para la realización de este trabajo se utilizó la metodología, herramientas y lenguajes definidos por el proyecto, los cuales son explicados durante el desarrollo del mismo.

El diseño de la propuesta de solución realizada para los procesos Administración de Inspecciones Técnicas y Accidentes fue validado mediante el uso de métricas que demostraron que el diseño realizado es simple y con calidad aceptable. También se aplicaron pruebas de caja blanca y caja negra a la implementación realizada demostrando el buen funcionamiento del sistema. Además fue revisado por Calisoft y por el cliente otorgando avales que evidencian el grado de satisfacción con la solución propuesta.

Palabras claves: Accidentes, Inspecciones Técnicas, unidades policiales.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), es uno de los elementos característicos de la sociedad actual; lo que ha provocado una explosión en la capacidad de procesar y almacenar grandes volúmenes de información. Con el desarrollo de estas tecnologías las empresas han optado por la informatización de sus procesos en aras de lograr ser más competitivas en el mundo globalizado de hoy, ya que los sistemas informáticos tienen la capacidad de almacenar grandes cantidades de información, permitiendo esto una mejor planificación y control en la gestión de los procesos que realizan.

Las empresas de Transporte actualmente se han enfocado en la utilización de sistemas de flotas de vehículos para gestionar sus procesos ya que estos ofrecen las siguientes ventajas:

- Mejora de la productividad
- Aumento de la vida útil de los vehículos
- Reducción de los tiempos de paros.

El área de Transporte del Cuerpo de la Policía Nacional Bolivariana no encontrándose ajeno a estos cambios ha decidido realizar la informatización de todos los procesos relacionados con la gestión de sus unidades policiales. Dentro de estos procesos se encuentra la Administración de Inspecciones Técnicas y Accidentes.

Las inspecciones técnicas se realizan con el objetivo de verificar el estado técnico de cada una de las partes de las unidades policiales para facilitar la toma de decisiones a partir de esta información para la realización de trabajos de mantenimientos ya sean preventivos planificados o correctivos.

La gestión de los accidentes se realiza con el propósito de contar con un registro que permita conocer datos generales del accidente como la fecha en la que ocurrió, número de expediente que tiene asociado , daños sufridos, involucrados internos y externos, vehículos involucrados, comisión investigadora y datos del expediente generado en fiscalía.

En el área de Transporte del Cuerpo de la Policía Nacional Bolivariana estos procesos se realizan de forma manual lo cual trae como consecuencia que el mantenimiento tanto preventivo planificado como correctivo de estas unidades no se realice de forma eficiente ya que la información necesaria para la realización de estos procesos no se encuentra centralizada, lo cual provoca pérdida de información,

desorden en cuanto a la secuencia de pasos a seguir para la realización de aquellos procesos que dependen unos de los otros, ejemplo Ejecución del mantenimiento depende del proceso Inspecciones técnicas, pues no se puede generar una Orden de trabajo sin antes haber realizado una inspección técnica.

Existen algunos conceptos que se manejan como parte del mantenimiento de las unidades policiales como por ejemplo las causas de fallas, unidades de medida de los medidores asociados a las unidades policiales, tipos de mantenimiento preventivo planificado, documentos técnicos, accidentes e inspecciones técnicas que se gestionan actualmente de forma independiente no existiendo un registro de estos que pueda ser consultado y utilizado para integrarse con aquellos procesos que dependen de ellos para su funcionamiento. No existe actualmente un registro de accidentes por dependencias en un periodo determinado y la única forma de obtener esta información tan necesaria para el área de Transporte es revisando cada uno de los expedientes de las unidades policiales los cuales se archivan en papel duro, resultando esto un proceso engorroso para el personal de esta área del CPNB (Cuerpo de la Policía Nacional Bolivariana).

La problemática planteada permitió definir el siguiente **problema a resolver**: ¿Cuál es la especificación de los artefactos necesarios para la informatización de los requisitos identificados en los procesos Administración de Inspecciones Técnicas y Accidentes del Sistema de Gestión de Mantenimiento Vehicular del área de Transporte del CPNB?

Para la solución de este problema se plantea como **Objetivo General:** Realizar el diseño e implementación de los procesos Administración de Inspecciones Técnicas y Accidentes del Sistema de Gestión de Mantenimiento Vehicular del área de Transporte del CPNB.

Se plantea como **objeto de estudio**: Procesos de desarrollo de software para sistemas de gestión.

Y el **campo de acción** se enmarca en el: Diseño e implementación de los procesos Administración de Inspecciones Técnicas y Accidentes del Sistema de Gestión de Mantenimiento Vehicular del área de Transporte del CPNB.

Para dar cumplimiento al objetivo general planteado se han definido los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación.
- Realizar el diseño de la solución.

- Realizar la implementación de la solución.
- Validar los resultados obtenidos.

Se define como **idea a defender:** La realización del diseño e implementación de los procesos Administración de Inspecciones Técnicas y Accidentes permitirá obtener los artefactos necesarios para la informatización de los requisitos identificados para este proceso.

Para el desarrollo del presente trabajo se utilizan los siguientes métodos de la investigación científica:

- Analítico sintético: El uso de este método se utiliza para realizar un análisis de los procesos Administración de Inspecciones Técnicas y Accidentes de los sistemas de gestión de flotas vehiculares existentes en Cuba y el mundo. De esta forma se logra entender el tema, facilitando su estudio y logrando definir una estrategia para llegar al resultado final con más facilidad. Este método permite la extracción de los elementos más importantes de cada documento analizado.
- Histórico lógico: El uso de este método permite estudiar cómo se realizaban los procesos Administración de Inspecciones Técnicas y Accidentes desde que surgió el Cuerpo de la Policía Nacional Bolivariana, para comprender su funcionamiento.

El presente trabajo de diploma se encuentra estructurado de la siguiente forma: una introducción, tres capítulos, anexos y la bibliografía utilizada.

Capítulo 1 Fundamentación teórica.

En este capítulo se realiza un estudio de los sistemas de gestión de flotas vehiculares existentes en Cuba y el mundo, haciendo énfasis en los procesos Administración de Inspecciones Técnicas y Accidentes. Se exponen los elementos que justifican la selección de las herramientas, lenguaje de programación, servidores de base de datos y aplicaciones, frameworks y metodología, utilizados para la realización del diseño e implementación de los procesos Administración de Inspecciones Técnicas y Accidentes del CPNB.

Capítulo 2 Diseño e implementación.

En este capítulo se realizan y describen las clases del diseño, diagrama de componentes, modelos de datos, artefactos estos que posteriormente se utilizarán para la realización de la implementación de los procesos Administración de Inspecciones Técnicas y Accidentes del CPNB. Se abordarán los patrones de

diseño utilizados, las integraciones entre los componentes definidos para el sistema y los estándares de codificación que se van a utilizar para la implementación.

Capítulo 3 Validación y Pruebas

Se realiza la validación del diseño propuesto empleando métricas y la implementación a través de pruebas de caja blanca y caja negra.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

Este capítulo tiene como objetivo mostrar un estudio del estado del arte entorno al objeto de estudio y campo de acción definido para el trabajo de diploma. Se realizará un análisis crítico de varios sistemas de mantenimiento vehicular existentes en Cuba y el mundo. Además se describen las tecnologías, lenguajes, metodologías de desarrollo y herramientas que pudieran ser utilizadas para dar solución a la problemática existente.

1.2. Procesos Gestionar Accidentes y Administración de Inspecciones Técnicas.

- **Proceso Gestionar Accidentes:** Este proceso tiene como objetivo registrar los accidentes de las unidades policiales del Cuerpo de la Policía Nacional Bolivariana, registrando de estos datos generales como: la fecha y hora en la que ocurrió el accidente, el lugar y la causa del mismo. Se registran los datos de los involucrados internos y externos, comisión investigadora del accidente y los datos del expediente legal creado en fiscalía. Luego se realiza una inspección técnica con el objetivo de verificar el estado técnico de cada una de las partes de la unidad policial.
- **Proceso Administración de Inspecciones técnicas:** Este proceso tiene como objetivo inspeccionar el estado técnico de cada una de las partes de las unidades policiales. Este proceso puede realizarse por varios motivos: accidente, mantenimiento preventivo planificado de una unidad, producto de un robo, hurto, proceso legal o por la generación de una Orden de trabajo. Luego de haberse realizado la inspección técnica se realiza un Informe de resultado a partir de la información registrada en esta, con el objetivo de notificar a los directivos sobre las decisiones tomadas a partir de los resultados obtenidos en la inspección técnica.

1.3. Sistemas existentes. Estado del arte.

En el mundo existen varios sistemas de gestión de flotas de vehículo, los cuales brindan facilidades en el control y administración de todas las tareas que están relacionadas a los vehículos que posea una determinada empresa. Dentro de estos se encuentran:

Administración de flotas- Gestión de Vehículos

El sistema de administración de flotas, es un sistema online que cuenta entre sus funcionalidades

- **Módulo de Vehículos:** Permite llevar los datos del vehículo, del propietario y de dos conductores distintos con sus respectivos turnos, esto para facilitar el control de la información de los taxis o vehículos que trabajan 24 horas al día.
- **Movimiento diario:** Se registra el movimiento de cada vehículo, ingresos, egresos y novedades.
- **Despachos – Viajes:** Permite llevar un control de los despachos de trabajos o viajes. Este módulo es especial para camiones, buses, vehículos inter municipales y servicios especiales.
- **Control combustible:** Permite de forma automática producir reportes de eficiencia de consumo de combustible.
- **Accidentes:** Lleva un historial de todos los datos pertenecientes al accidente de cualquier vehículo, también permite crear nuevos registros de acuerdo a la necesidad.
- **Cuentas de Conductores:** Permite generar reportes del estado de la cuenta de un conductor por fechas o de las cuentas de todos los conductores a la vez.
- **GPS - rastreo satelital:** Permite realizar un rastreo satelital de cada vehículo o a la cuenta de la flota que se administra.
- **Reportes –Informes:** Permite realizar monitoreo de todas las actividades del vehículo tanto económicas como en el área de mantenimiento.
- **Inspecciones, chequeos:** Permite registrar los chequeos o inspecciones que se le realizan a los vehículos.
- **Equipos – Inventario:** Permite llevar el inventario, la programación por fechas del mantenimiento de equipos y también produce reportes con la localidad y otra información útil (1).

SoftFlot (Solución integral para la administración de una flotilla de vehículos).

SoftFlot es un sistema que fue diseñado para controlar una flotilla de vehículos. Este sistema fue realizado por una empresa mexicana llamada Comsun que se especializa en la Administración y Seguridad de los vehículos para el negocio (2). Entre las funcionalidades que contiene este software se encunetran:

- Registro de su flotilla.
- Registro de empleados.
- Asignación de conductores.
- Almacén de recursos materiales.
- Gastos de viaje.
- Pool vehicular.
- Costos y presupuestos.

- Bitacora de combustibles, talachas y siniestros.
- Asignar paquetes de tareas de mantenimiento a los vehículos mediante el armado de paquetes de tareas mantenimiento.
- Registro de últimos mantenimientos.
- Ordenes de servicio programables, correctivas y libres.
- Reporte de fallas.
- Calendario de mantenimiento.
- Historiales de mantenimiento y consumos.
- Configurar SoftFlot a medida de sus necesidades (2).

QMaint

QMaint es otro software de gestión de mantenimiento, perteneciente a una empresa española, incorpora la experiencia de 12 años en consultoría de mantenimiento y las especificaciones recogidas en más de 200 implantaciones. Gestiona y optimiza la actividad de mantenimiento, aumentando la productividad y vida útil de los equipos e instalaciones. Este sistema presenta un diseño flexible y modular, debido a esta característica el software se adapta a cualquier sector y entorno de trabajo. (3)

QMaint está compuesto por módulos funcionales integrados. Entre ellos se encuentran los siguientes (3).

- **Gestión de activos:** Organiza los activos críticos, gestiona toda la información relacionada y controla al detalle los costos de mantenimiento. Esta sección incluye las siguientes funcionalidades:
    - Datos técnicos.
    - Contratos.
    - Alarmas y averías asociadas.
    - Repuestos y ubicaciones.
    - Historial de Órdenes de Trabajo.
    - Costos directos e indirectos.
    - Calibración y mediciones.
    - Planos y documentos.

- **Planificación de mantenimiento:** Diseña los planes de mantenimiento a realizar, se le asignan a los activos y se decide la frecuencia de ejecución. Esta sección brinda las siguientes funcionalidades:
  - Mantenimiento Preventivo y Predictivo.
  - Planes de trabajo.
  - Previsión de recursos.
  - Rutas de mantenimiento.
  - Mantenimiento cíclico.
- **Órdenes de Trabajo:** Genera automáticamente las órdenes de trabajo a realizar, adjunta la documentación técnica y planos necesarios y distribuye la carga de trabajo entre los recursos disponibles.
- **Almacén y compras:** Lleva un control efectivo de Stocks y automatiza todos los procedimientos de compra. Esta sección brinda las siguientes funcionalidades:
  - Gestión de múltiples almacenes.
  - Gestión de ubicaciones.
  - Ficha de repuesto.
  - Impresión de etiquetas.
  - Inventario y gestión de Stocks.
  - Órdenes de compra.
  - Seguimiento de recepciones.
  - Gestión de contratos.
  - Gestión y Control de Facturas.
  - Petición y análisis de ofertas.
- **Gestión de Recursos:** Optimiza la gestión de los recursos internos y externos, conociendo en todo momento su disponibilidad. Esta sección brinda las siguientes funcionalidades:
  - Personal propio y externo.
  - Turnos de trabajo.
  - Calendario laboral.
  - Vehículos y herramientas.
  - Materiales y repuestos.
  - Contratos y facturas externas.

El MP versión 9

El MP es un software profesional para el control y administración del mantenimiento que ayuda a mantener organizada toda la información que requiere un departamento de Mantenimiento. (4)Esta solución constituye una poderosa herramienta que brinda las siguientes funcionalidades:

- Documentar información de equipos y localizaciones.
- Documentar planes y rutinas de mantenimiento rutinario.
- Organizar y programar trabajos de mantenimiento.
- Organizar historiales referentes a trabajos realizados y recursos utilizados.
- Generar una gran cantidad de consultas gráficas y reportes relacionados con la gestión del mantenimiento.

El MP 9 está dividido en distintos módulos interconectados entre sí. A continuación una breve explicación de los módulos principales de esta herramienta (4):

- **Catálogo de equipos y localizaciones:** Al registrar en el sistema el catálogo de equipos y localizaciones, se pueden asignar trabajos de mantenimiento tanto a equipos como a localizaciones, así como documentar la localización de cada uno de los mismos.
- **Planes de mantenimiento:** Este módulo permite la formación de planes o rutinas de mantenimiento donde el usuario establece las partes de sus equipos, las actividades de mantenimiento a realizar para cada una y la frecuencia.
- **Control de lecturas:** La frecuencia con que se realizan las actividades rutinarias las puede establecer el usuario en base a tiempo o a lectura, como por ejemplo: kilómetros y horas de uso. Incluso es posible establecer frecuencias combinadas en fechas y lecturas
- **Calendarios:** El MP calcula de forma automática los calendarios de mantenimiento, una vez que se tienen los equipos con sus respectivos planes o rutinas de mantenimiento. En los calendarios se muestran las fechas de cuándo deben realizarse cada una de las actividades.
- **Mantenimiento no rutinario:** Además de los trabajos rutinarios definidos en los planes de mantenimiento, el MP permite registrar trabajos no rutinarios, es decir, trabajos que se realizan una vez.
- **Mantenimiento predictivo:** El MP grafica dichos valores y alerta sobre todos aquellos equipos que tienen alguna medición fuera o cercana a los límites, permitiendo generar oportunamente las órdenes de trabajo correspondientes de revisión o reparación.

- **Recursos:** El MP permite al personal de mantenimiento consultar la existencia de repuestos y disponibilidad de las herramientas necesarias antes de iniciar un trabajo, evitando demoras y pérdidas de tiempo por no contar con los repuestos necesarios.
- **Órdenes de trabajo:** El módulo de generación de órdenes de trabajo del MP presenta una lista actualizada con todos aquellos trabajos programados para realizarse en un período determinado. Partiendo de esta lista de trabajos que deben realizarse, el usuario genera las órdenes de trabajo agrupando los trabajos por especialidades o cualquier otro criterio. Conforme se van realizando los trabajos contemplados en las órdenes de trabajo, los encargados deberán reportar al MP, sobre los trabajos realizados, de modo que en base a dicha información el MP calculará las fechas próximas para cuando deban volver a realizarse.
- **Vales y consumos:** El MP genera de forma automática los vales de almacén. Bastará con mencionar al almacenero el número de vale de almacén generado por el MP, para que desde el programa de inventario se genere el movimiento de salida sin necesidad de volver a teclear toda la información contenida en el vale.
- **Análisis de información:** El MP cuenta con un módulo que permite graficar costos, fallas, paros, cantidad de actividades realizadas mensualmente, así como la generación de gráficas comparativas entre diferentes equipos. Incluye también un módulo denominado historia gráfica que presenta en forma gráfica los mantenimientos realizados a cada equipo, mostrando los períodos protegidos y desprotegidos. Este módulo constituye un indicador para evaluar la vulnerabilidad de nuestros equipos y permite evaluar en forma práctica el cumplimiento de los programas de mantenimiento establecidos para cada equipo.
- **Garantías, documentos y ligas a páginas de Internet:** El MP cuenta también con un módulo para el control de garantías que permite consultar las garantías vigentes por equipo, indicando las vigentes y las vencidas. También sirve como un organizador de documentos, en donde el usuario podrá consultar tablas y especificaciones técnicas, planos y manuales. Los documentos que el usuario registre en el MP pueden ser cualquier tipo de archivo (Word, Excel, Auto Cad y Acrobat Reader).

Eugcom

Eugcom es un sistema computacional diseñado para la sólida y correcta administración de flotas de vehículos. Dentro de sus características principales se encuentran (5):

- Fácil manejo para el usuario.
- Rápido acceso a los datos y consultas.
- Máxima Seguridad, restringiendo a cada usuario, dependiendo del acceso que disponga.
- Amigable Entorno Gráfico.
- Maneja claves de seguridad por usuario.
- Sin límites de usuarios.

Eugcom está compuesto por módulos funcionales integrados. Entre ellos se encuentran los siguientes (5):

- **Vehículos:** Módulo de Mantención (Permite Crear, Modificar, Eliminar e Imprimir) de todos los datos de los vehículos del sistema.
- **Proveedores:** Módulo de ingreso y mantención de todos los proveedores de insumos y servicios del sistema.
- **Combustibles:** Módulo de ingreso y mantención de los datos que componen el movimiento de cargas de combustible efectuados a los vehículos.
- **Mantención de Servicios:** Módulo de mantención de los servicios que componen los movimientos relacionados a los servicios realizados al vehículo.
- **Orden de Trabajo:** Módulo de ingreso y mantención de las órdenes de trabajo emitidas a los vehículos para efectuar una cierta cantidad de servicios en él.
- **Cargas de Combustible:** Módulo de búsqueda de datos que permite ejecutar consultas y obtener resultados acerca de las cargas de combustible registradas en el sistema. Genera un informe sumamente personalizado según los requerimientos del usuario y permite calcular los datos más importantes por cada vehículo.
- **Viajes Realizados:** Módulo de búsqueda de datos de los movimientos de viajes realizados por los vehículos en un lapso de tiempo y según las especificaciones establecidas por el usuario.
- **Reprocesamiento de Cargas de Combustible:** Módulo que permite reprocesar los datos automáticos generados en el movimiento de cargas de combustible como distancia y rendimiento, y genera un informe de errores generados por los usuarios en las cargas de combustible registradas en el sistema, y es capaz de reparar automáticamente los errores más usuales. Este módulo evita que los informes de combustible contengan información no válida después de editar o eliminar un movimiento.

SGestMan

Es un sistema informático desarrollado en Cuba por la empresa GAMMA S.A. para la gestión integral del mantenimiento. Su arquitectura de trabajo es cliente/servidor sobre plataforma Oracle o SQL Server. El sistema permite una rápida adaptación a nuevos requisitos de los usuarios gracias a la flexibilidad y potencia de la herramienta utilizada para el desarrollo. También brinda al cliente la posibilidad de personalizar el SGestMan e integrarlo a otro software. El sistema cuenta con 10 módulos de trabajo, los cuales recogen y procesan la información especializada de cada área de la empresa relacionada con la gestión de mantenimiento. Estos módulos son los siguientes (6):

- **Informes:** Este módulo está previsto para obtener toda la información introducida en el sistema en forma impresa, permitiendo realizar análisis estadísticos técnicos y económicos de la información procesada.
- **Producción:** Este módulo está previsto para controlar los servicios que se realizan a clientes externos mediante órdenes de producción e indicadores, así como incidencias productivas que ocurren en una empresa de producción o de servicio.
- **Logística:** Este módulo está previsto para enlazarse con nomencladores de productos y necesidades de compras, lo cual posibilita la inclusión de todos aquellos productos y/o materiales necesarios para la realización de los servicios de mantenimiento que se realizan en la instalación.
- **Administración:** Este módulo está previsto para llevar todo el control de los accesos de los usuarios del sistema, así como los permisos a las opciones de cada uno de los módulos. Garantiza la integridad y fiabilidad de la información que se incluye en el resto del sistema.
- **Patrimonio:** Este módulo está previsto para estructurar toda la información de los diferentes elementos que forman parte de cualquier empresa, sea de producción o de servicios, lo cual garantiza poder tener muy bien definida todas las necesidades que en materia de mantenimiento precisa una empresa para contar con un Patrimonio con un alto nivel de disponibilidad y utilización técnica.
- **Recursos Humanos:** Este módulo está previsto para estructurar toda la información de los empleados que forman parte de la organización del mantenimiento, y con los cuales se garantiza la disponibilidad de los objetos que conforman el Patrimonio de una empresa.
- **Órdenes de servicio:** Este módulo está previsto para llevar todo el control técnico y económico de las órdenes de servicio que se realizan por los diferentes ejecutores de mantenimiento, tanto internos como externos, de manera que se puedan tener todos los históricos de los objetos del Patrimonio con un alto nivel de detalle.

- **Contratos:** Este módulo está previsto para llevar toda la información técnica y económica de las contrataciones que hace el departamento de mantenimiento para garantizar la disponibilidad de los objetos del Patrimonio. De esta forma podrán ser incluidos todos aquellos prestadores de servicios que reparen equipamientos de las empresas.
- **Mantenimiento Preventivo:** Este módulo está previsto para preparar toda la estrategia de proyección, programación y planificación de acciones de Mantenimiento Preventivo, dirigido a garantizar el óptimo desempeño del equipamiento, la máxima disponibilidad, y la reducción de costos por concepto de reparaciones y mantenimientos no previstos.
- **Economía:** Este módulo está previsto para controlar de forma dinámica y en todos los niveles la actividad económica de mantenimiento, a partir de la valoración de todos los costos incurridos en las órdenes de servicios desarrolladas a los clientes.

El SGestMan ha sido implantado en varias empresas de manera exitosa, tales como BrasCuba Cigarrillos S.A., la empresa de níquel Ernesto Ché Guevara, en el Central Azucarero de alcohol y azúcar DIANA (Brasil), en el Hospital Dr. Juan Graham Casasús (México), en la Central de Azúcar Sanagro SP (Brasil) y en la Unidad Básica Puerto Moa(Holguín). (6)

OffiMant

Es otra herramienta informática desarrollada en Cuba por la empresa DESOFT S.A, en Camagüey y ha sido generalizado en 132 empresas nacionales. Es un software que facilita una adecuada gestión de la información del mantenimiento a equipos e instalaciones en un entorno visual de sencillo manejo.

Este software se caracteriza por ser una herramienta multiusuario, presenta gran flexibilidad al cambio y es muy fácil de usar. También proporciona una amplia versatilidad en informes, brinda integración y enlaces y es de alta productividad para las empresas que lo utilizan. (7)

Dentro de las funciones principales que realiza OffiMant se encuentran las siguientes:

- Crea carpeta técnica de equipos.
- Planifica tareas.
- Controla órdenes de trabajos.
- Fiscaliza presupuesto de gasto.
- Gestiona productos en almacén.
- Gestiona solicitudes de trabajos.

Un adecuado uso de este sistema garantiza ahorros por conceptos como: mano de obra, equipos fuera de servicio, reducción de inventarios, paradas no programadas y toma de decisiones.

OffiMant es una aplicación modular, o sea, está compuesto por módulos funcionales, tales como (7):

- **Módulo de Administración:** Registra la licencia de usuarios que permite posteriormente trabajar en el módulo de Mantenimiento. Registra la base de datos y logra la seguridad de toda la información.
- **Módulo de Mantenimiento:** Permite definir y mantener toda la información relacionada con los activos, establecer y planificar tareas, generar solicitudes y órdenes de trabajo.
- **Activos:** Registra una amplia diversidad de información técnico-económica relacionada con los activos existentes.
- **Planificación:** Confecciona de forma rápida y sencilla un plan de mantenimiento por cada activo, a partir de las tareas de mantenimiento definidas.
- **Solicitudes de trabajo:** Procesa cualquier incidencia reportada por los usuarios del sistema, permitiendo asignarlas a una orden de trabajo según convenga.
- **Órdenes de trabajo:** Genera órdenes de trabajo según necesidad de mantenimiento preventivo o imprevisto, facilitando el seguimiento y asignación de los recursos usados. En la vista Órdenes de trabajo podrá crearse una orden de trabajo imprevista y darle seguimiento a las generadas anteriormente.
- **Presupuesto:** Define un completo esquema presupuestario para el mantenimiento según proyecciones de costos, a corto, mediano y largo plazo.
- **Reportes:** Permite analizar y evaluar el mantenimiento a través de variados informes. Además posibilita su pre-visualización y/o impresión.
- **Módulo de Solicitud de Trabajo:** Emite solicitudes desde diferentes estaciones de trabajo al departamento de mantenimiento.

Valoraciones sobre los sistemas estudiados

En la tabla 1 se muestra una comparación sobre los sistemas analizados. Los indicadores que se utilizan para comparar son:

- Gestión de Inspecciones técnicas.
- Gestión de Accidentes.

- Tipo de licenciamiento.
- Tipo de sistema.

Tabla 1 Análisis comparativo de los sistemas estudiados.

Sistemas analizados	Gestión de Inspecciones técnicas	Gestión de Accidentes	Tipo de licenciamiento	Tipo de sistema
Administración de flotas-Gestión de Vehículos	Si	Si	Propietario	Web
SoftFlot	No	No	Propietario	Escritorio
QMaint	No	No	Propietario	Escritorio
MP versión 9	No	No	Propietario	Escritorio
Eugcom	No	No	Propietario	Escritorio
SGestMan	No	No	Propietario	Escritorio
OffiMant	No	No	Propietario	Escritorio

Luego de haber realizado un análisis comparativo referente a los sistemas estudiados se llega a la conclusión de que ninguno puede ser utilizado como parte del Sistema de Gestión de Mantenimiento Vehicular, específicamente para la gestión de los procesos Administración de Inspecciones Técnicas y Accidentes por las siguientes razones:

- Son aplicaciones de escritorio, por tanto, no se pueden utilizar, ya que en el CPNB todos los sistemas que se desarrollan se montan sobre un servidor central para que todas las dependencias policiales conectadas a internet y mediante un navegador se conecten al sistema.
- Algunos de estos sistemas no permiten gestionar el mantenimiento de vehículos, por tanto, tampoco controlan la gestión de los accidentes e inspecciones técnicas de las unidades policiales.
- Fueron desarrollados sobre tecnologías propietarias por cuanto no pueden ser utilizados ya que el CPNB tiene como política la utilización de tecnologías libres en todas sus áreas, además de que tienen que pagar precios elevados por las licencias para su utilización y de que no se pueden

añadir nuevas funcionalidades que necesita el área de Transporte en cuanto a la Administración de Inspecciones Técnicas y Accidentes.

- El sistema Administración de flotas - Gestión de Vehículos realiza la gestión de los accidentes e inspecciones técnicas, pero no solucionan de forma completa las necesidades del área de Transporte del CPNB. En la gestión de las inspecciones técnicas no permite obtener informaciones necesarias como: el estado de la unidad, el motivo por el cual se le realiza la inspección técnica a esa unidad, el responsable directo, el lugar donde se realiza, y el estado de las partes de la unidad a la cual se le va a realizar la inspección técnica, un informe donde se recojan los resultados obtenidos en la inspección técnica realizada, además de no contar con información propia del área de Transporte del CPNB. En la gestión de los accidentes no permite registrar información importante como: las causas que provocaron el accidente, los datos de los involucrados externos del accidente, los datos de la comisión investigadora y del proceso legal realizado en fiscalía, además de no llevar el control de información propia del área de Transporte del CPNB.

1.4. Modelo de desarrollo

Modelo de desarrollo orientado a componentes.

Para el desarrollo de los sistemas informáticos del centro CEIGE[1], se definió como marco de trabajo a utilizar el sistema Sauxe. Para el desarrollo de Sauxe es preciso que cada uno de los equipos de desarrollo posean un modelo estándar, así como una definición clara de las responsabilidades de cada uno de los roles que se ven involucrados en el desarrollo de la solución. Debido a esto se utiliza el modelo de desarrollo orientado a componentes fruto de la colaboración entre las líneas de desarrollo del centro CEIGE, de acuerdo a las necesidades que presentan y teniendo en cuenta los principales riesgos con los que se cuentan en el proyecto.

El modelo de desarrollo orientado a componentes se caracteriza principalmente por ser iterativo e incremental, ágil, adaptable al cambio y centrado en la arquitectura. Este modelo tiene como objetivo, reducir el tiempo de trabajo, el esfuerzo que requiere implementar una aplicación y los costos del proyecto, y de esta forma, incrementar el nivel de productividad de los grupos de desarrolladores y minimizar los riesgos globales sin incurrir en grandes gastos. De esta manera, las pequeñas empresas pueden tener mayor confiabilidad a la hora de realizar una inversión tecnológica. (8)

[1] Centro de Informatización de la Gestión de Entidades.

Otra de las ventajas de este modelo es que pertenece al paradigma de programación de sistemas abiertos, los cuales son extensibles y tienen una interacción con componentes heterogéneos que ingresan o abandonan el sistema de forma dinámica, es decir, que los componentes pueden ser reemplazados, por otros independientemente de su arquitectura y desarrollo. (8)

Una característica muy importante a tener en cuenta para la selección de este modelo es que usándola se puede integrar lo mejor de las tecnologías para desarrollar una aplicación de manera personalizada, a la medida de las necesidades de los clientes. Esto permite a los desarrolladores y a la empresa adquirir las tecnologías que más se adapten a sus necesidades, y no incurrir en gastos de licenciamiento o soporte y actualización de las grandes soluciones, ya que muchas de estas tecnologías son gratis y existen bajo la premisa de Software Libre, lo cual añade otra gran ventaja. (8)

1.5. Arquitectura de software.

La arquitectura de software integra componentes, conectores, configuraciones y restricciones. Estos elementos son la base de los modelos o vistas que describen la arquitectura, mediante su identificación es posible establecer bajo qué estilos, patrones y condiciones tendrá lugar la dinámica del sistema.

El equipo central de arquitectura del centro CEIGE ha definido que al sistema Sauxe le sean aplicados los fundamentos del estilo de Arquitectura basada en componentes, donde el sistema será descompuesto en componentes, promoviendo el desarrollo y utilización de componentes reutilizables. (9)Cada componente definido contará con una interfaz que determina tanto las operaciones que el componente implementa como las que precisa utilizar de otros componentes durante su ejecución.

El patrón conocido como Modelo-Vista-Controlador separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes: El modelo en el que estará la lógica de negocio de nuestra aplicación, la vista que agrupa las clases y ficheros que tengan que ver con la presentación y el controlador que al recibir los eventos de la interfaz, se encarga de llamar en el modelo al experto del negocio que sabe que es lo que hay que hacer con la petición del usuario. Este patrón arquitectónico será empleado para el desarrollo del Sistema de Gestión de Mantenimiento Vehicular debido a que su utilización fue definida por el CEIGE, además de estar implementado dentro del marco de trabajo Sauxe, sobre el cual será desarrollado el sistema.

1.6. Tecnología

Para el desarrollo de los procesos propuestos se utilizaron varias tecnologías seleccionadas por el Departamento de Tecnología del CEIGE.

AJAX

El término AJAX es un acrónimo de Asynchronous JavaScript[2] + XML[3]. Es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. Permite la comunicación entre la capa de la vista y la capa de la controladora. (10)

AJAX presenta numerosas ventajas, entre las que se encuentran:

- Permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.
- Soportada por la mayoría de los navegadores modernos.
- Portabilidad.
- La página se asemeja a una aplicación de escritorio.

Las tecnologías que forman AJAX son (10):

- XHTML[4] y CSS[5], para crear una presentación basada en estándares.
- DOM[6], para la interacción y manipulación dinámica de la presentación.
- XML, XSLT[7] y JSON[8], para el intercambio y la manipulación de información.
- XMLHttpRequest[9], para el intercambio asíncrono de información.

[2] Lenguaje de programación orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

[3] Lenguaje de Marcas Extensible del inglés Extensible Markup Language.

[4] Lenguaje Extensible de Marcado de Hipertexto del inglés Extensible Hypertext Markup Language.

[5] Hojas de estilo en cascada del inglés Cascading Style Sheets.

[6] Modelo de Objetos del Documento del inglés Document Object Model.

[7] Lenguaje de Hojas Extensibles de Transformación del inglés Extensible Stylesheet Language.

[8] Notación de Objetos de JavaScript del inglés JavaScript Object Notation.

- Java Script, para unir todas las demás tecnologías.

1.7. Lenguajes de programación.

PHP

PHP (acrónimo de Hipertexto Preprocessor) es un lenguaje "del lado del servidor" creado para el desarrollo de páginas Web dinámicas, puede ser incluido con facilidad dentro del código HTML[10]. (11)

Características de PHP (11):

- **Gratuito**. Al tratarse de software libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre.
- **Versatilidad**. PHP puede usarse con la mayoría de los sistemas operativos.
- **Eficiente** Con escaso mantenimiento y un servidor gratuito (en este caso, Apache), puede soportar sin problema millones de visitas diarias.
- **Integración con múltiples bases de datos**. Esencial para una página Web verdaderamente dinámica. Aunque MySQL es la base de datos que mejor trabaja con PHP, puede conectarse también a PostgreSQL y Oracle.
- **Capacidad de expandir su potencial** utilizando una enorme cantidad de módulos.
- **Posee una amplia documentación en su página oficial**, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- **No requiere definición de tipos de variables**.
- **Tiene manejo de excepciones**.

Debido a todas estas características y por ser el lenguaje definido por el Marco de Trabajo Sauxe se decidió utilizar como lenguaje de programación en su versión 5.2.

JavaScript

JavaScript es un lenguaje de script multiplataforma orientado a objetos, está diseñado para una fácil incrustación en otros productos y aplicaciones, tales como los navegadores Web. Dentro de un entorno

[9] Lenguaje de Marcas Extensible/Protocolo de Transferencia de Hipertextos del inglés Extensible Markup Language/ Hypertext Transfer Protocol.

[10] Lenguaje de Marcas Hipertextuales del inglés HyperText Markup Language.

anfitrión, JavaScript puede ser conectado a los objetos de su entorno para proveer un control programable sobre éstos. (12)

El núcleo de JavaScript contiene un conjunto central de objetos, tales como Array (arreglos), Date (fechas) y Math (objetos matemáticos), además de un conjunto central de elementos del lenguaje tales como los operadores, estructuras de control y sentencias. El núcleo de JavaScript puede ser extendido para una variedad de propósitos complementándolo con objetos adicionales; por ejemplo (12):

- **JavaScript del lado Cliente** extiende el núcleo del lenguaje proporcionando objetos para el control del navegador (Navigator o cualquier Web browser) y su Modelo Objeto Documento (DOM). Por ejemplo, las extensiones del lado del cliente permiten a una aplicación ubicar elementos en un formulario HTML y responder a los eventos de usuario tales como los clics del mouse, entradas del formulario y navegación de páginas.
- **JavaScript del lado Servidor** extiende el núcleo del lenguaje proporcionando objetos relevantes para la ejecución de JavaScript en un servidor. Por ejemplo, las extensiones del lado del servidor permiten que una aplicación se comunique con una base de datos relacional, proporcionar continuidad de la información desde una invocación de la aplicación a otra o efectuar la manipulación de archivos en un servidor.

1.8. Frameworks

Un framework, en el desarrollo de un software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado para ayudar a desarrollar y unir los diferentes componentes de un proyecto. (13)

El marco de trabajo Sauxe definido por el centro CEIGE se ha estructurado de manera tal que facilite la reutilización de los diferentes componentes y que sea de fácil entendimiento para todos los programadores que desarrollen sobre el mismo. Sauxe utiliza ExtJS para implementar la capa de presentación, se apoya en Zend, una extensión de Zend Framework para el desarrollo de la lógica del negocio y para la gestión de los datos utiliza Doctrine.

EXTJS

ExtJS es una biblioteca de Javascript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML[11] y DOM. Es neutral al lenguaje que se use en el servidor. Siempre que el resultado se envíe a la página en el formato adecuado, ExtJS no se preocupará de lo que pase en el servidor. (14)

ExtJS es simplemente un framework escrito en Javascript que permite desarrollar aplicaciones web complejas de una forma cómoda sin tener que lidiar con los típicos problemas de la programación. Soporte para construir interfaces gráficas complejas y dinámicas, comunicar datos de forma asíncrona con el servidor y manejar datos de distinta índole de una manera simple. ExtJS se ha basado en otros framewroks previamente construidos y los han unificado obteniendo una mayor potencialidad. (14)Se estará haciendo uso de ExtJS en su versión 3.4.

Zend Framework

Es un framework de alta calidad y de código abierto para el desarrollo de aplicaciones y servicios web con PHP. Zend Framework brinda facilidades de uso y poderosas funcionalidades. Proporciona soluciones para construir modernos, robustos y seguros sitios web, está diseñado para php 5 y posee buenas capacidades de ampliación. Presenta las siguientes características (15):

- Proporciona un sistema de caché de forma que se puedan almacenar diferentes datos.
- Proporcionan los componentes que forman la infraestructura del patrón Modelo-Vista-Controlador.
- Proporciona una capa de acceso a base de datos, construida sobre PDO10 pero ampliándola con diferentes características.
- Proporciona mecanismos de filtrado y validación de entradas de datos.
- Permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX.
- Proporciona capacidades de búsqueda sobre documentos y contenidos.

Se estará haciendo uso de Zend Framework en su versión 1.5.

Doctrine Framework

Doctrine es un potente y completo sistema de mapeado de objetos relacionales (ORM siglas en inglés) para PHP 5.2 o superior, que posee una formidable capa de abstracción a base de datos. Entre sus

[11] HTML dinámico del inglés Dynamic HyperText Markup Language.

principales características se encuentra la posibilidad de escribir de manera opcional las consultas a la base de datos Esto les proporciona una alternativa poderosa a diseñadores de SQL manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Permite también exportar una base de datos existente a sus clases correspondientes y también convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos (16). Se estará haciendo uso de Doctrine Framework en su versión 0.11.

1.9. Herramientas

Para el desarrollo de los procesos propuestos se utilizaron varias herramientas seleccionadas por el Departamento de Tecnología del CEIGE.

Visual paradigm

Herramienta CASE[12] Visual Paradigm.

Visual Paradigm para UML (Lenguaje unificado de modelado) 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 costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y documentación. Proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos UML (17). Se estará haciendo uso de Visual Paradigm en su versión 5.0.

Características de Visual Paradigm (17):

- Es profesional.
- Es amigable.
- Contiene facilidades para redactar especificaciones de casos de uso del sistema.
- Sincronización entre diagramas de entidad-relación y diagramas de clases.
- Generación de código e Ingeniería inversa.
- Soporte de UML versión 2.1.
- Interoperabilidad con otras aplicaciones.

[12] Ingeniería de Software Asistida por Computadora del inglés Computer Aided Software Engineering.

Debido a las ventajas que ofrece Visual Paradigm y por ser una herramienta multiplataforma se decide utilizarlo como herramienta CASE. Además permite la modelación de los procesos de negocio y la Universidad de las Ciencias Informáticas cuenta con la licencia de esta herramienta.

Servidor de aplicaciones. Servidor web Apache 2.0.

Es una tecnología de código de fuente abierta, puede ser usado en varios sistemas operativos, lo que lo hace prácticamente universal. Es un servidor altamente configurable de diseño modular, trabaja con gran cantidad de lenguajes como por ejemplo: Perl, PHP y otros lenguajes de script. Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Características (18):

- Multiplataforma.
- Apache es una tecnología gratuita de código de fuente abierta.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que se instalen cuando lo necesiten.
- Apache trabaja con Perl, PHP y otros lenguajes de script. Perl destaca en el mundo del script y Apache utiliza su parte del pastel de Perl tanto con soporte CGI[13] como con soporte mod_perl[14]. También trabaja con Java y páginas JSP[15]. Teniendo todo el soporte que se necesita para tener páginas dinámicas.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Tiene una alta configurabilidad en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor.

[13] Interfaz de entrada común del inglés Common Gateway Interface.

[14] Módulo opcional para el servidor Apache

[15] Páginas de Servidor Java del inglés Java Server Programming.

Servidores de base de datos

Se denomina Sistema Gestor de Base de Datos (siglas: SGBD) al conjunto de programas que permiten definir, construir y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad.

El servidor de base de datos que se va a utilizar para la implementación de los procesos Administración de Inspecciones Técnicas y Accidentes es PostgreSQL, ya que es un servidor de base de datos relacional, libre. (19) Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins de gran tamaño. Se estará haciendo uso de PostgreSQL en su versión 8.3 ó superior e inferior a 8.4

Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas (19):

- Cuenta con una gran comunidad de desarrollo en Internet.
- Su código fuente está disponible sin costo alguno.
- Es multiplataforma.
- Fue diseñado para ambientes de alto volumen.
- Tiene mejor soporte para vistas, procedimientos almacenados en el servidor, transacciones, almacenamiento de objetos de gran tamaño y además tiene ciertas características orientadas a objetos.

Navegadores

El navegador que se va a utilizar es Mozilla Firefox porque ofrece las siguientes ventajas (20):

- Es de software libre, ágil, práctico y en renovación constante.
- Se puede modificar según las necesidades del usuario.
- Presenta navegación por pestañas.
- Está diseñado para realizar un bajo consumo de recursos.

Se usará Mozilla Firefox en su versión 2.2 o superior.

Eclipse

Eclipse es un entorno integrado de desarrollo de código abierto, portable y multiplataforma. Fue creado inicialmente por IBM[16] y en la actualidad es mantenido por la Fundación Eclipse, la cual lo define como “una especie de herramienta universal, un IDE abierto y extensible para todo y nada en particular”. Tiene

[16] Empresa Internacional de Negocio de Máquinas del inglés International Business Machines.

una arquitectura basada en plug-ins y posibilita integrar diversos lenguajes de programación, así como introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces y ayuda en línea para librerías. Se puede integrar con Visual Paradigm usando el Entorno de Desarrollo Inteligente (Smart Development Environment), provocando un entendimiento superior entre analistas y desarrolladores. Eclipse permite el trabajo en equipo mediante el empleo de SubVersion y del Sistema de Control de Versiones (CVS), los cuales constituyen una combinación de vistas y editores que muestran los diversos aspectos de los recursos del proyecto organizados por el rol o la tarea del desarrollador. (21) Se estará haciendo uso de Eclipse en su versión 3.3.

1.10. Conclusiones parciales

Al culminar el presente capítulo se puede concluir que los sistemas existentes en Cuba y el mundo poseen una serie de deficiencias las cuales impiden que sean utilizados para realizar la informatización de los procesos del área de Transporte del CPNB, por lo que han quedado sentadas las bases para la implementación de un sistema informático capaz de satisfacer las necesidades que actualmente presenta esta área en cada uno de los procesos que realiza, específicamente la Administración de Inspecciones Técnicas y Accidentes del CPNB. En este trabajo se realizará el diseño e implementación de estos procesos utilizando el modelo de desarrollo, lenguajes de programación, herramientas y las tecnologías antes mencionadas.

CAPÍTULO 2: DISEÑO E IMPLEMENTACIÓN

2.1. Introducción

En este capítulo se exponen los resultados del diseño e implementación realizados para los procesos Administración de Inspecciones Técnicas y Accidentes. Se parte del análisis de los artefactos generados por los analistas como elementos de entrada para una mejor comprensión del negocio. Se presentan los mecanismos y patrones que se utilizaron para el desarrollo del diseño, los diagramas de clases conformados y las descripciones de las clases del diseño. El capítulo comprenderá además elementos fundamentales de la implementación de los procesos Administración de Inspecciones Técnicas y Accidentes, en ese caso se presentan: la definición de los estándares de codificación, los diagramas de componente y despliegue, la representación de la estrategia de integración y la descripción de las funcionalidades.

2.2. Análisis de los artefactos generados por los analistas

El punto de partida para diseñar la propuesta de solución es realizar un análisis de los artefactos entregados por los analistas del Sistema de Gestión de Mantenimiento Vehicular, estos artefactos son: Modelo Conceptual, Descripción de los procesos de negocio Administración de Inspecciones Técnicas y Accidentes y las Especificaciones de los requisitos funcionales identificados para ambos procesos.

Se analizó el Modelo Conceptual ya que en este artefacto se identificaron los conceptos que se manejaban en el negocio y era necesario dominarlos ya que los analistas en el artefacto Especificación de requisitos funcionales hacían referencia a los mismos.

El artefacto Descripción de los procesos de negocio Administración de Inspecciones Técnicas y Accidentes se estudió en aras de comprender cómo funcionaban estos procesos dentro del área de Transporte del CPNB para entender mejor qué es lo que se quería informatizar.

El artefacto Especificación de requisitos funcionales se analizó porque a partir de la información que contiene se va a realizar el diseño de la propuesta de solución planteada por los analistas. Luego del análisis realizado se comprobó que la especificación de los requisitos funcionales identificados estaba correctamente redactada, eran claros, posibles de probar y cubrían todas las funcionalidades necesarias, además de que fueron validados y liberados por el cliente y por Calisoft. Al no detectarse errores en dichos artefactos no se propusieron cambios y se concluyó que la solución propuesta era correcta, clara, completa y consistente, por cuanto se podía proceder al diseño de la misma.

2.3. Requisitos funcionales

Los requisitos funcionales identificados por los analistas del Sistema de Gestión de Mantenimiento Vehicular que van a ser diseñados e implementados como parte de la propuesta de solución son los siguientes:

1. Agrupación de requisitos Gestionar inspecciones técnicas.
2. Agrupación de requisitos Gestionar inspecciones técnicas de una unidad.
3. Agrupación de requisitos Gestionar accidentes.
4. Agrupación de requisitos Gestionar accidentes de una unidad policial.
5. Agrupación de requisitos Gestionar tipo de unidad (Nomencladores).
6. Agrupación de requisitos Gestionar partes.
7. Asociar partes a tipo de unidad.
8. Quitar partes a tipo de unidad.
9. Cambio de estado de la unidad.
10. Relación de inspecciones técnicas en un período.
11. Relación de accidentes en un período.
12. Agrupación de requisitos Gestionar informes de resultados.
13. Reinicio del medidor de una unidad policial.
14. Eliminar última lectura del medidor de una unidad policial.
15. Agrupación de requisitos Gestionar lectura de los medidores de las unidades policiales.

2.4. Diseño de la solución

El diseño del software es un proceso iterativo mediante el cual los requisitos se traducen en un plano para construir el software. El proceso del diseño traduce requisitos en una representación del software donde se pueda evaluar su calidad antes de que comience la implementación (22).

Objetivos del diseño.

- Transformar los requisitos en clases del diseño para la futura implementación del sistema.
- Elaborar una arquitectura para el sistema.
- Adaptar el diseño para que concuerde con el ambiente de implementación, diseñando el sistema para el rendimiento.

El Sistema de Gestión de Mantenimiento Vehicular se va a realizar a través de una aplicación Web por lo que se decidió que para el desarrollo de las clases del diseño se utilizarán los siguientes estereotipos web (23):

- Client Page (Página cliente): Una instancia de página cliente es una página Web, con formato HTML. Mezcla de datos, presentación y lógica. Son interpretadas por el navegador. Sus atributos son las variables declaradas dentro del script.
- Build (Construye): Representa una asociación especial que relaciona las páginas cliente con las páginas servidor. Las páginas que se encuentran en el servidor construyen las páginas en el cliente. Debe ser una relación direccional, donde una página servidor puede construir una o más páginas cliente.
- Submit (Envía): Es la relación que se crea siempre entre una página servidor y un formulario, a través de esta relación el formulario manda los valores de sus campos al servidor, para ser procesados por la página servidor.
- Include (Inclusión): Especifica una situación en la que un caso de uso tiene lugar dentro de otro caso de uso.
- Instantiate (Instancia): Es una relación de uso, es decir, una clase usa a otra, porque la necesita para su cometido. Indica que el origen crea instancias del destino.

2.5. Mecanismos de diseño

Los mecanismos de diseño se utilizan con el objetivo de abreviar los diagramas de clases. Cada diseñador establece sus propios mecanismos de diseño, teniendo siempre en cuenta los patrones y estilos seleccionados (24).

Para la realización del diseño de los procesos Administración de Inspecciones Técnicas y Accidentes del Sistema de Gestión de Mantenimiento Vehicular se definieron los siguientes mecanismos de diseño:

Mecanismo de diseño para las páginas clientes.

Imagen 1 Mecanismos de diseño para las páginas clientes

Todas las páginas clientes del diseño incluyen las siguientes clases:

- Ext-all.js: Es la encargada de la creación de los componentes visuales de la vista. Está incluida dentro de las clases que trae EXT JS.
- Ext-base.js: Encargada del manejo de las solicitudes y respuestas, manejo de componentes de EXT.
- Ucid-all.js: Encargada de mostrar la interfaz estándar.

Mecanismo de diseño para los nombres de las clases.

Los diagramas de clases del diseño del Sistema de Gestión de Mantenimiento Vehicular se realizaron por agrupaciones de requisitos funcionales que tuvieran cierto grado de dependencia entre sí.

En el caso de las agrupaciones de requisitos conformadas por adicionar, modificar, eliminar, listar y buscar esta agrupación se divide en dos clases, una que agrupa los requisitos adicionar, modificar, y eliminar a la cual se le denomina AME_Nombre de la clase, y otra clase que agrupa los requisitos listar y buscar a la cual se le pone el nombre LB_Nombre de la clase.

Mecanismo de diseño para las clases controladoras.

Imagen 2 Mecanismo de diseño para las clases controladoras

Todas las clases controladoras definidas en el diseño propuesto heredan de la clase ZendExt_Controller_Secure, ya que en ella se incluyen numerosas funcionalidades comunes.

Mecanismo de diseño para las clases modelos.

Imagen 3 Mecanismo de diseño para las clases modelo

Todas las clases modelos o model definidas en el diseño heredan de la clase ZendExt_Model, ya que esta incluye las principales funciones para el manejo de los datos.

2.6. Diagramas de clases del diseño.

Los diagramas de clases según la clasificación UML son diagramas de estructura estática donde la representación de los requerimientos se lleva a cabo a través de las clases del sistema y sus interrelaciones. Representan una abstracción del dominio de modo que es formalizado el análisis de conceptos y constituyen el pilar básico del modelado, mostrando en términos generales qué debe hacer el sistema (25).

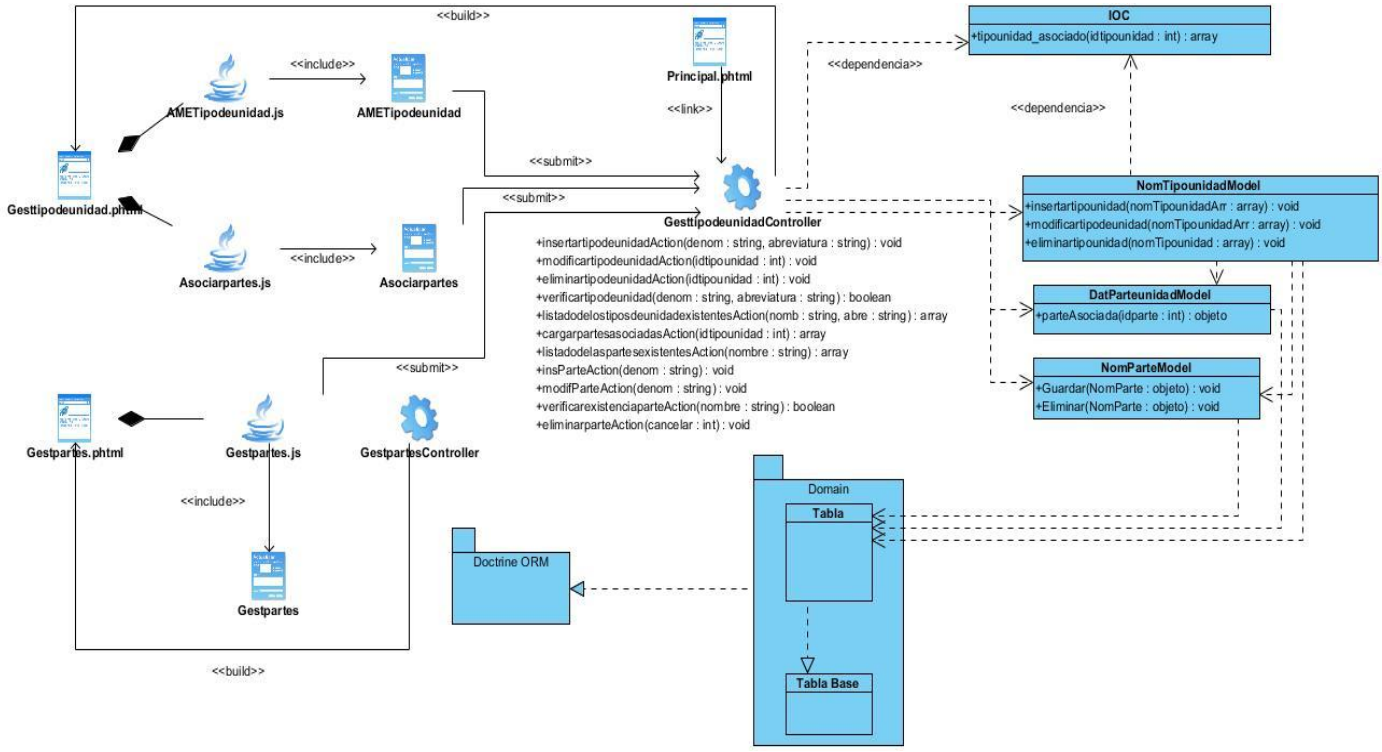

Imagen 4 Diagrama de clases Gestionar tipo de unidad

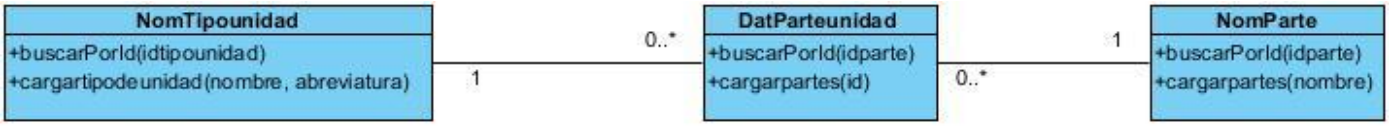

Imagen 5 Diagrama de clases del paquete de dominio Gestionar tipo de unidad

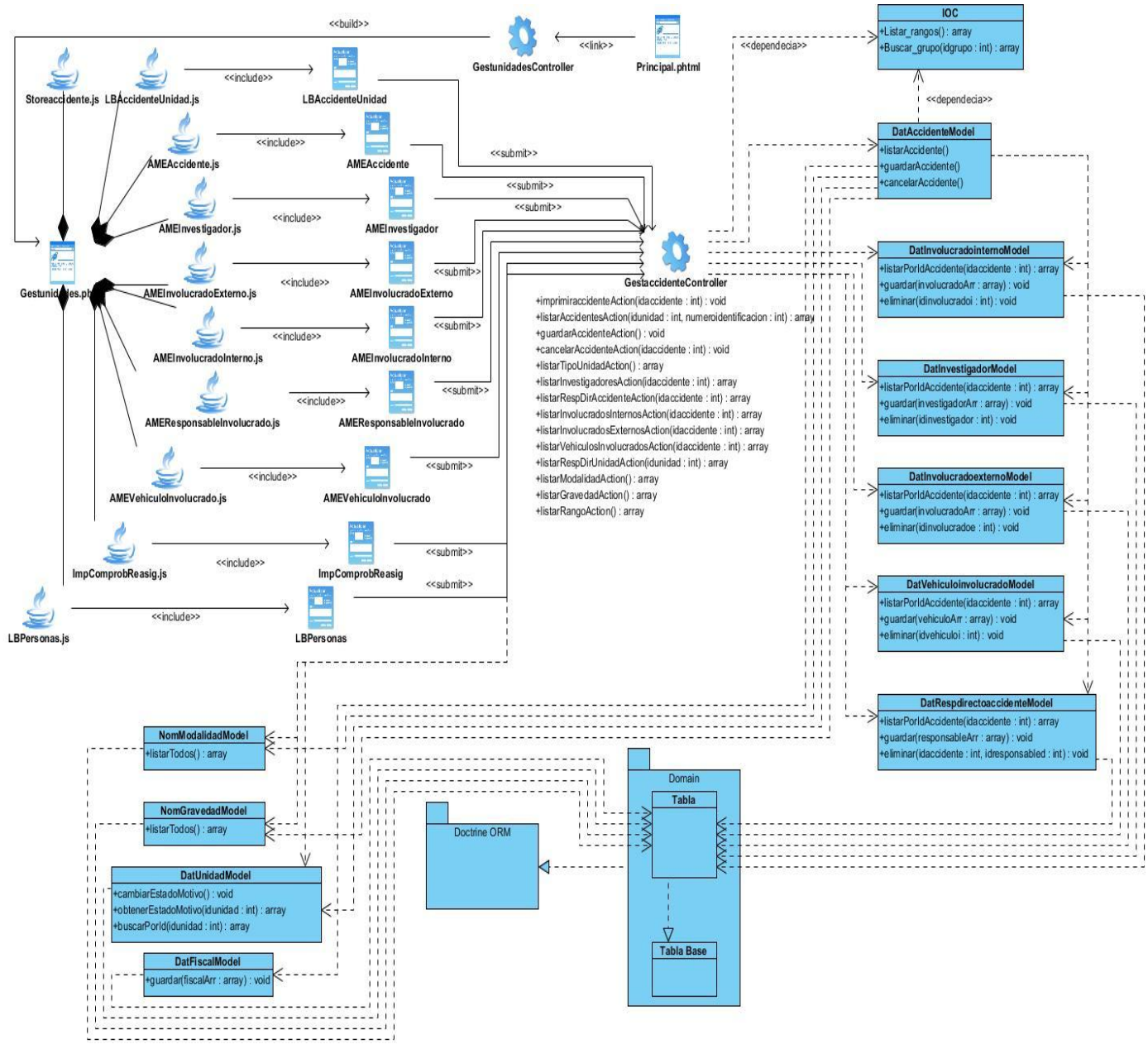

Imagen 6 Diagrama de clases Gestionar Accidentes de una unidad

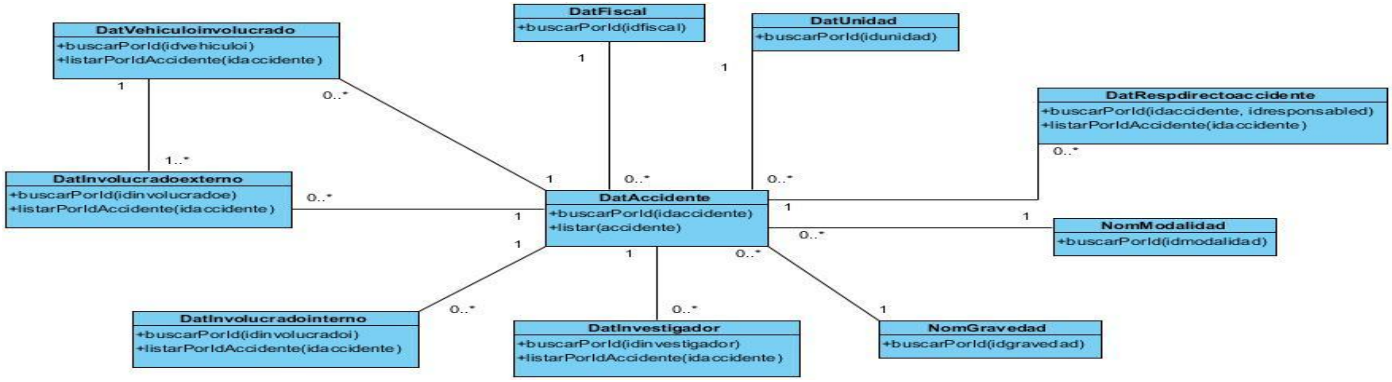

Imagen 7 Diagrama de clases del paquete de dominio Gestionar Accidentes de una unidad

Los diagramas de clases restantes se encuentran en el **Anexo 1**.

2.7. Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores y debe ser aplicable a diferentes problemas de diseño en distintas circunstancias (24).

Patrones de diseño aplicados:

Modelo-Vista-Controlador: Es un patrón de diseño que plantea la separación de diferentes clases en dependencia de la función que realizan con el propósito de que sea posible manejar dinámicamente la forma en que se procesan solicitudes y cómo se mostrarán los resultados al usuario final (26). En fin se puede decir que separa la lógica de negocio de la presentación o interfaz.

- **Modelo:** El Modelo se encuentra dividido en los paquetes bussines, el cual encierra las clases donde se realiza la lógica del negocio, y domain, donde se encuentran las clases del dominio, las que se encargan de leer y/o escribir datos en las tablas de la base de datos.
- **Vista:** Maneja la visualización de la información. Le brinda al usuario la posibilidad de interactuar con el Controlador mediante los mensajes de eventos y le permite visualizar las respuestas a sus peticiones. Se encarga de presentar la interfaz al usuario.
- **Controlador:** Representa el gestor de eventos, el cual está compuesto por las clases controladoras. Este atiende los pedidos que llegan desde la Vista accediendo al paquete models.

Dentro de los patrones utilizados en la aplicación, se encuentran los GRASP[17], los cuales describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades, de ellos se utilizaron:

Experto: Se evidencia cuando las clases controladoras delegan la responsabilidad de ejecutar determinadas acciones a las clases del modelo cuya información es más adecuada para la resolución del problema en cuestión.

Creador: Es adaptable a las clases del paquete Domain, quienes son las encargadas de crear los objetos de tipo Doctrine_Query, para permitir el acceso a la información almacenada a nivel de datos.

Controlador: Se pone de manifiesto a través del uso de una clase controladora que es la que administra el flujo de acciones invocadas desde la interfaz de usuario y en cada una de ellas asocia la lógica del negocio al modelo responsable. En el modelo, donde está implementada la lógica del negocio, las clases del modelo pueden instanciar otras clases en dependencia de la información que se requiera.

Bajo acoplamiento: En el Modelo de Datos se definieron un conjunto de clases persistentes, entre las cuales se establecieron las relaciones necesarias de manera que fueran más independientes y reutilizables para reducir el impacto de los cambios y acrecentar la oportunidad de una mayor productividad.

Alta cohesión: Su empleo está dado en que fueron asignadas responsabilidades a las clases de forma tal que la cohesión siguiera siendo alta, o sea, cada clase se encargará de realizar solamente las funciones que estén en correspondencia con la responsabilidad que posea.

Durante el diseño de la aplicación se emplearon patrones GOF[18], específicamente:

Cadena de Responsabilidad: Está concebido que ante la ocurrencia de un error al realizarse una determinada consulta a la base de datos el mismo sea manejado por el Modelo, creando una nueva excepción de tipo ZendExt_Exception. Dicha excepción debe ser propagada al Controlador, el cual será el encargado de capturarla y enviarla a la Vista ya traducida, esta última por su parte mostrará un mensaje al

[17] Patrones de Software para la Asignación General de Responsabilidades del inglés General Responsibility Assignment Software Patterns.

[18] Pandilla de los Cuatro del inglés Gang of Four.

usuario en un lenguaje entendible notificando el error y sin especificar detalles del mismo. De esta manera se distribuyen las responsabilidades entre los diferentes componentes, evidenciándose por lo tanto el empleo de este patrón.

Fachada: La utilización de este patrón está reflejada en la creación y empleo de la clase IOC[19], ya que por la necesidad de integración entre los módulos del Sistema de Gestión de Mantenimiento Vehicular, era necesaria la implementación de una clase fachada donde fueran publicados los servicios necesarios por estos, facilitando su interacción.

Solitario/Singleton: Este patrón fue utilizado en las clases del dominio, donde los métodos contenidos en estas se hicieron de forma estática, con lo cual se garantizaba que pudieran ser accedidos sin crear una instancia de las mismas. Por ejemplo: para acceder al método listarTodos() de la clase del domain DatAccidente, desde la clase DatAccidenteModel, del modelo, solo era necesario poner DatAccidente::listarTodos().

2.8. Descripción de las clases del diseño

A continuación se presentan las descripciones de las clases del diseño identificadas.

Tabla 2 Descripción de la clase GestaccidenteController

Nombre: GestaccidenteController	
Tipos de clase: Controladora	
Para cada responsabilidad:	
Nombre	**listarAccidentesAction()**
Descripción	Permite listar todos los accidentes de las unidades policiales.
Nombre	**guardarAccidenteAction()**
Descripción	Permite adicionar o modificar los datos de un accidente.
Nombre	**imprimiraccidenteAction()**
Descripción	Esta responsabilidad, a través de las librerías TCPDF, muestra en formato PDF[20] los datos del accidente.
Nombre	**cancelarAccidenteAction()**
Descripción	Permite cancelar un accidente determinado.
Nombre	**listarTipoUnidadAction()**
Descripción	Permite obtener mediante un servicio todos los tipos de unidad.
Nombre	**listarInvestigadoresAction()**
Descripción	Permite listar todos los investigadores de un accidente determinado.

[19] Inversión de Control del inglés Inversion of Control

[20] Formato de Documento Portable del inglés Portable Document Format.

Nombre	**listarRespDirAccidenteAction()**
Descripción	Permite listar los datos de los responsables directos de un accidente determinado.
Nombre	**listarInvolucradosInternosAction()**
Descripción	Permite listar los datos de los involucrados internos de un accidente determinado.
Nombre	**listarInvolucradosExternosAction()**
Descripción	Permite listar los datos de los involucrados externos de un accidente determinado.
Nombre	**listarVehiculosInvolucradosAction()**
Descripción	Permite listar los datos de los vehículos involucrados de un accidente determinado.
Nombre	**listarRespDirUnidadAction()**
Descripción	Permite obtener los datos del responsable directo de una unidad.
Nombre	**listarModalidadAction()**
Descripción	Permite obtener un listado de las modalidades.
Nombre	**listarGravedadAction()**
Descripción	Permite obtener un listado de las gravedades del accidente.
Nombre	**listarRangoAction()**
Descripción	Este método permite obtener mediante un servicio cada unos de los datos del rango.

Tabla 3 Descripción de la clase DatAccidenteModel

Nombre: DatAccidenteModel	
Tipos de clase: Modelo	
Para cada responsabilidad:	
Nombre	**guardar($accidenteArr)**
Descripción	A partir de un arreglo pasado con todos los datos obtenidos de un accidente, permite adicionar o modificar un accidente al sistema.
Nombre	**cancelar($idaccidente)**
Descripción	A partir de un id de un accidente permite cancelar el mismo, es decir, el accidente pasará a un estado inactivo pero no se elimina de la base de datos.
Nombre	**listar($accidente, $limit, $start)**
Descripción	Permite listar los datos de todos los accidentes que se encuentran en el sistema.
Nombre	**buscarPorId($idaccidente)**
Descripción	A partir del id de un accidente permite obtener los datos del mismo.
Nombre	**buscarPorIdExt($idaccidente)**
Descripción	Permite obtener los datos de un accidente a partir de su identificador.

La descripción de las restantes clases del diseño definidas, se encuentra en el **Anexo 2**.

2.9. Modelo de Datos

Un modelo de datos es una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo (27).

El modelo propuesto consta de 24 tablas, para su elaboración se tuvo en cuenta la reducción a la mínima expresión de los campos nulos.

En este modelo se representan los datos y las relaciones entre estos, a través de una colección de tablas, en las cuales los renglones (tuplas) equivalen a cada uno de los registros que contendrá la base de datos y las columnas corresponden a las características (atributos) de cada registro localizado en la tupla. El mismo se presenta a continuación.

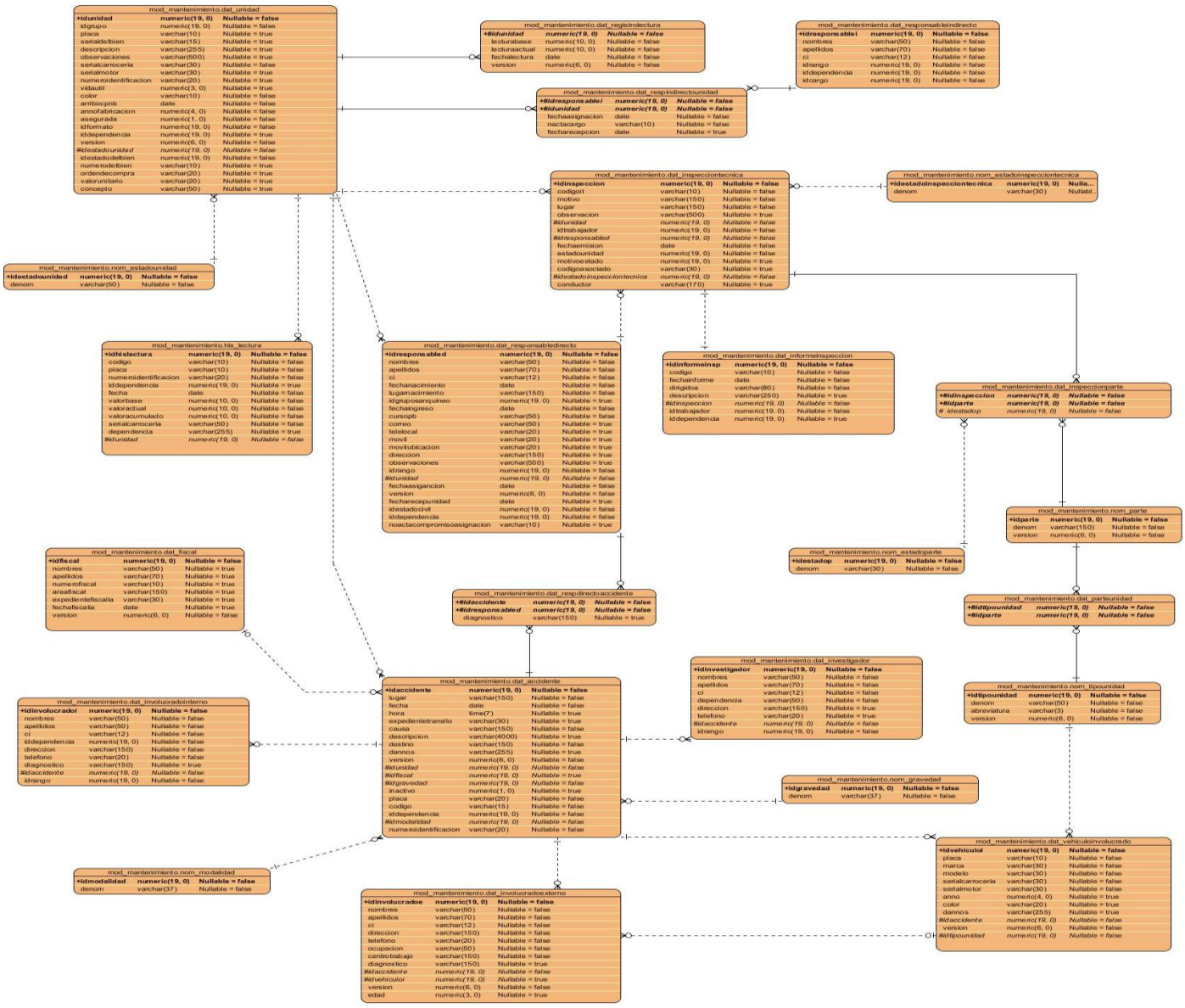

Imagen 4 Modelo de datos

2.10. Implementación de la solución

Diagrama de componentes

Un diagrama de componentes es modelado por el arquitecto del sistema y representa gráficamente cómo un sistema de software es dividido en componentes mostrando las dependencias existentes entre estos. El diagrama contiene componentes, interfaces, relaciones entre ellos y puede contener paquetes utilizados para agrupar elementos del modelo; mostrando las dependencias lógicas entre componentes de software (28). A continuación se presenta el diagrama de componentes elaborado desde la perspectiva del componente Vehículo.

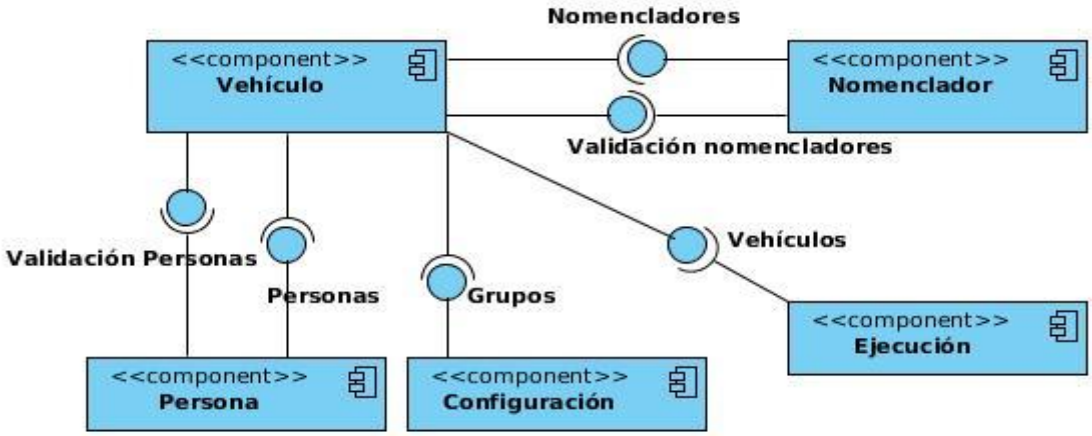

Imagen 5 Diagrama de componentes: Perspectiva Vehículo

Descripción de cada unos de los componentes que integran el diagrama:

Nomencladores: Permite la gestión de los nomencladores de propiedades, causas de fallas, documentos técnicos, tipos de mantenimiento, tipos de unidades, partes, accesorios, unidades de medida, herramientas y repuestos del sistema de Mantenimiento.

Configuración: Permite la gestión del grupo y de los clientes a los cuales se les van a gestionar los servicios de mantenimiento.

Persona: Se encarga de lo relativo a la gestión del personal de mantenimiento del CPNB.

Vehículo: Se encarga de gestionar todas las funcionalidades relacionadas con el registro de unidades y de las incidencias de estas (robo, hurto o proceso legal), también maneja las funcionalidades relacionadas con la gestión de los accidentes de las unidades policiales y el registro de las inspecciones técnicas.
Ejecución: Se encarga de generar las órdenes de trabajo producto de un mantenimiento preventivo

planificado o correctivo, registrando los gastos incurridos tanto humanos como materiales de la ejecución del mantenimiento realizado para cada unidad policial.

2.11. Diagrama de despliegue

El diagrame de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. Está compuesto por nodos, dispositivos y conectores. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema. Permite el mapeo de procesos dentro de los nodos, asegurando la distribución del comportamiento a través de aquellos nodos que son representados (29).

Se definirá una arquitectura cliente-servidor detallada de la siguiente manera. **Ver Imagen 6**.

Cliente: Computador, la aplicación se ejecuta a través de un navegador internet instalado, en este caso se debe usar el Mozilla Firefox sobre cualquier sistema operativo (se sugiere GNU/Linux, en específico la distribución de Ubuntu para estaciones donde se conectarán dispositivos externos: Impresoras).

Servidor Web: Radica la lógica de negocio de la aplicación. Servidor Web Apache2. Utilizando el lenguaje PHP.

Servidor de Base de datos: Servidor de Datos PostgreSQL 8.3

A continuación se muestra el Diagrama de despliegue del sistema:

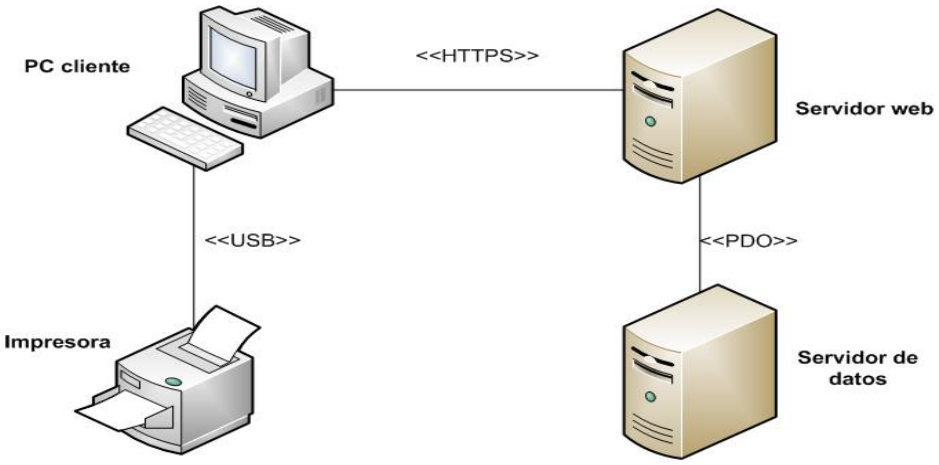

Imagen 6 Diagrama de despliegue

2.12. Estándares de codificación

Un estándar de codificación son reglas que se siguen para la escritura del código fuente. De tal manera que a otros programadores se les facilite entender tu código (como identificar las variables, las funciones o métodos, etc.). Aseguran que todos los programadores trabajen de forma coordinada y mantengan un código de calidad. Si se aplica de forma continuada un estándar de codificación bien definido caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener (30). Dentro de los estándares utilizados se encuentran los siguientes:

Nomenclatura de las clases

El nombre de las clases se escribe la primera letra con mayúsculas y las demás con minúsculas, en caso de ser un nombre compuesto, el segundo nombre se escribe seguido del primero y con minúsculas. El nombre estará dado según la función que realiza. Ejemplo: Gestionaraccidente.

Nomenclatura de las clases según el tipo

- El nombre de las clases controladoras se escribe según su nomenclatura y se le adicionará al final “Controller”. Ejemplo: GestionarinspeccionesController.
- El nombre de las clases del Modelo que se encuentran dentro de la carpeta “bussines” después de escribir el nombre según su nomenclatura se le añade “Model” y al inicio se le colocará “Nom”, en caso de que sea un nomenclador, de lo contrario será Dat. Ejemplo: DatInformeinspeccionModel.
- Las clases del dominio que su encuentran dentro de la carpeta “domain”, son generadas mediante la herramienta Doctrine Generator 3.0 desarrollada por el CEIGE, y son nombradas igual que las tablas de la base de datos. Ejemplo: DatInspeccionparte.
- Las clases bases que se encuentran dentro de la carpeta “generated” son generadas mediante la herramienta Doctrine Generator 3.0 desarrollada por el CEIGE y delante del nombre de la clase se le añade “Base”. Ejemplo: BaseDatAccidente.
- El nombre de las clases de la vista comenzará con la primera letra en mayúscula y el resto en minúscula, en caso de que este sea compuesto, se utilizará la notación PascalCasing. A este nombre se le agregarán al principio, las letras en mayúscula de las funciones que se realizarán a través de estas clases. Ejemplo: AMEAccidente(adicionar, modificar y eliminar), LBInspeccionesTecnicas(listar,buscar).

Nomenclatura de los métodos o funciones

El nombre de los métodos de una clase comienzan con minúsculas, en caso de que sea compuesto se empleará notación CamelCasing, seguido del primer nombre comienza el segundo con la primera letra en mayúscula. Deben describir el propósito del mismo. Ejemplos: guardar(), listarInvolucradoInterno(). Si es una función de una clase controladora después del nombre se le agrega la palabra “Action”. Ejemplo: listarResponsablesDirectosAction().

Nomenclatura de las variables

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing. En la capa de la Vista, las variables que contengan componentes que formen parte de la interfaz, se escriben comenzando con un prefijo que indique el tipo de componente que contiene, en los demás casos el prefijo indicará el tipo de datos. Ejemplo: storeAccidentes, btnModificarInvestigador, gridInvolucradosExternos, arrAccidente.

2.13. Integración entre componentes

En los componentes se aplica la integración vertical que consiste en el flujo de los datos desde la vista hacia la capa de datos y viceversa, pasando por los diferentes elementos que componen la arquitectura. Esta consta de cuatros nodos de integración, la vista y el controlador, el controlador y el modelo, el que vincula el modelo con el framework doctrine y el que se encuentra entre el doctrine y la base de datos. Todo el código dentro un mismo componente utiliza llamadas a métodos o eventos de forma directa. La comunicación entre diferentes módulos y componentes se realiza mediante llamadas a la inversión de control. El IOC especifica respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que otro módulo o componente lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.

Cada componente tiene su registro de los datos de los módulos en un fichero xml con el nombre IOC donde se publican los métodos o servicios que se brindan a los demás componentes del subsistema (servicios internos) y/o a otros subsistemas (servicios externos), que será mapeado por el framework para el funcionamiento del mismo, dicho fichero tiene por nombre IOC y registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema. La base de datos es accedida de forma directa mediante controladoras y los componentes rehusados son integrados mediantes interfaces sencillas. Cada componente tiene una clase Service que se encarga de publicar los métodos o

servicios que se le puede brindar a otras líneas o componentes. Para acceder a los servicios externos se llama al objeto integrator, el nombre del subsistema y el nombre del método publicado en el IOC externo (que se encuentra en el paquete comun del sistema). Para acceder a los servicios internos se llama al objeto pIntegrator, el nombre del componente y el nombre del método publicado en el IOC interno (que se encuentra en el paquete comun del propio subsistema) (30).

Publicación de servicios

Para cada uno de los componentes fueron publicados en sus clases Services, los mismos métodos que son utilizados en estos para la recuperación de los datos en las clases del modelo y del dominio, realizándose de esta manera con el objetivo de reutilizar código ya existente. Los servicios fueron implementados de modo que reciban muchos o ningún parámetro, siempre que fuera posible, disminuyendo con esto la cantidad de servicios a publicar.

Los procesos Administración de Inspecciones Técnicas y Accidentes se encuentran dentro del componente Vehículo, el cual cuenta con las clases Services, DatInspeccionTecnicaService, DatResponsabledirectoService y DatRegistrolecturaService, las cuales contienen todos los servicios que son consumidos por los restantes componentes del Sistema de Gestión de Mantenimiento Vehicular, con los que se relacionan.

A continuación se muestran los servicios que son brindados por el componente Vehículo dentro de los procesos Administración de Inspecciones Técnicas y Accidentes:

Tabla 4 Servicios que brinda el componente Vehículo dentro de los procesos Administración de Inspecciones Técnicas y Accidentes.

Servicios que brinda	Componentes que lo utilizan	Código	Descripción
Obtener_lectura	**Ejecución**	**MTTO-Vehic-1**	**Se obtienen a partir del identificador de una unidad todas las lecturas de los medidores de la misma.**
modificarRegistroLectura	**Ejecución**	**MTTO-Vehic-2**	**Permite modificar el**

			registro de lectura de una unidad dado fechalectura, idunidad y lecturaactual.
Reiniciar_medidor	**Ejecución**	**MTTO-Vehic-3**	**Permite reiniciar medidor dado idunidad, fecha,antes_reinicio y valor_reinicio.**
Codigo_dado_id_it	**Ejecución**	**MTTO-Vehic-6**	**Permite obtener todos los códigos de las unidades mediante el idinspeccion.**
guardarInspeccionTecnica	**Ejecución**	**MTTO-Vehic-7**	**Permite guardar los datos de una InspeccionTecnica dado idinspeccion.**
buscar_ResponsableDirecto	**Ejecución**	**MTTO-Vehic-14**	**Permite obtener todos los datos del responsable directo pasado por parámetro mediante el idresponsable.**

Servicios que consume el componente Vehículo para la implementación de los procesos Administración de Inspecciones Técnicas y Accidentes como parte de la integración:

Tabla 5 Servicios que consume el componente Vehículo dentro de los procesos Administración de Inspecciones Técnicas y Accidentes.

Servicios que consume	**Componentes que lo brindan**	**Código**	**Descripción**
Buscar_unidadmedida	**Nomencladores**	**MTTO_Nom_6**	**Permite obtener los datos de la unidad de medida según id.**

Buscar_tipounidad	**Nomencladores**	**MTTO_Nom_8**	**Buscar tipo de unidad según id.**
Buscar_parte	**Nomencladores**	**MTTO_Nom_15**	**Buscar parte según id.**
Listar_rangos	**Nomencladores**	**MTTO_Nom_23**	**Permite obtener un listado de todos los rangos.**
lista_Parte_DadoId	**Nomencladores**	**MTTO_Nom_27**	**Permite obtener los datos de una parte a partir de su identificador.**
Buscar_grupo	**Configuración**	**MTTO_Conf_4**	**Buscar grupo según id.**
Tipounidad_asociado	**Configuración**	**MTTO_Conf_6**	**A partir del identificador de un tipo de unidad devuelve verdadero o falso si este está asociado o no a un grupo.**
listar_RecursosHumanos	**Persona**	**MTTO_Pers_4**	**Muestra un listado de todas las personas que se encuentran registradas en el sistema.**
buscar_Trabajador	**Persona**	**MTTO_Pers_5**	**A partir del identificador de una persona, se obtienen todos los datos de la misma.**

2.14. Descripción de la implementación por funcionalidades

A continuación se realizará una breve descripción de las funcionalidades implementadas.

Gestionar accidentes

En este componente se registran los accidentes de las unidades policiales del CPNB. La siguiente imagen muestra la vista de cuando se accede al módulo Vehículos, esta vista mostrará los accidentes registrados en el sistema, con los siguientes atributos: Código, Número de identificación, Placa, Serial de carrocería, Fecha, Lugar y Hora. Brinda las opciones de Adicionar, Modificar, Cancelar e Imprimir. Además posee una búsqueda, la cual se realiza por los campos Número de identificación, Placa, Serial de carrocería y Fecha.

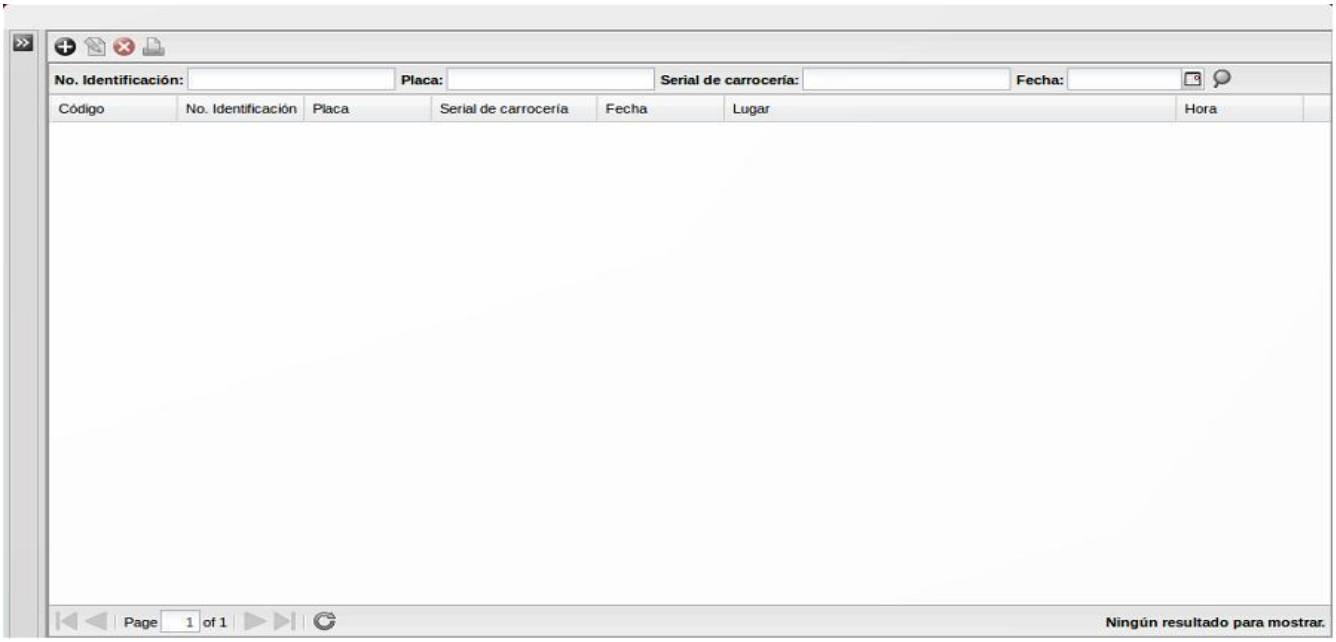

Imagen 7 Gestionar accidentes

Adicionar accidente

Al seleccionar la opción Adicionar se muestra una interfaz a través de la cual pueden ser insertados todos los datos que son necesarios para registrar el accidente. La interfaz está compuesta por diferentes pestañas, en las cuales se registran los datos del accidente. Se registran los datos generales del accidente, datos de la unidad, los datos de los responsables directos, los involucrados internos y externos, vehículos involucrados, comisión investigadora del accidente y los datos del expediente legal. Al dar clic en el botón Aplicar, se añade al sistema el accidente registrado y se limpian los campos de la interfaz para ingresar uno nuevo, en caso de haberle dado clic al botón Aceptar, de igual manera se ingresan los datos del accidente en el sistema y se muestra un mensaje de información, pero seguidamente se vuelve a la vista principal. En el **Anexo 3** se muestran las diferentes vistas de la interfaz para adicionar un accidente.

Modificar accidente

Si se quiere modificar un accidente de los existentes, se selecciona el mismo y posteriormente se va a la opción Modificar que activará una interfaz similar a la de Adicionar pero con todos los campos del elemento seleccionado cargados. Se modifican los campos que se quieren cambiar y se presiona Aceptar para confirmar la operación. Se mostrará un mensaje de información.

Cancelar accidente

Para cancelar un accidente que haya sido registrado en el sistema, se selecciona el mismo y luego se va a la opción Cancelar. Cuando se haya terminado de realizar la operación se mostrará un mensaje de confirmación. Si se selecciona Aceptar, será cancelado el accidente, es decir, el accidente pasará a un estado inactivo pero no se elimina de la base de datos y se mostrará un mensaje de información.

Imprimir accidente

Para imprimir un accidente, primeramente se selecciona este y luego la opción imprimir, luego se mostrará en formato PDF una vista previa de cómo quedará conformado el expediente del accidente.

Gestionar accidente de una unidad

En este componente se registran los accidentes de una unidad específica, seleccionada en la interfaz Unidades policiales y posteriormente se selecciona el botón Accidentes, accediendo a la interfaz donde se registran los accidentes de esa unidad y además se muestran los siguientes atributos: Código, Número de identificación, Placa, Serial de carrocería, Fecha, Lugar y Hora. Brinda las opciones de Adicionar, Modificar, Cancelar e Imprimir. Además posee una búsqueda, la cual se realiza por los campos Fecha y Serial de carrocería.

Adicionar accidente a una unidad

Para adicionar un accidente hay que seleccionar una unidad que se encuentre en estado operativa y luego se selecciona la opción Adicionar donde se muestra una interfaz similar a la de la imagen (de Adicionar accidente) a través de la cual pueden ser insertados todos los datos que son necesarios para registrar el accidente.

Modificar accidente a una unidad

Si se quiere modificar un accidente de los existentes, se selecciona el mismo y posteriormente se va a la opción Modificar que activará una interfaz similar a la de Adicionar pero con todos los campos del elemento cargados. Se modifican los campos que se quieren cambiar y se presiona Aceptar para confirmar la operación. Se mostrará un mensaje de información.

Cancelar accidente a una unidad

Para cancelar un accidente a una unidad, se selecciona el mismo y luego se va a la opción Cancelar. Cuando se haya terminado de realizar la operación se mostrará un mensaje de confirmación. Si se

selecciona Aceptar, será cancelado el accidente, es decir, el accidente pasará a un estado inactivo pero no se elimina de la base de datos y se mostrará un mensaje de información.

Imprimir accidente a una unidad

Para imprimir un accidente a una unidad, primeramente se selecciona este y luego la opción imprimir, una vez realizado esto se mostrará en formato PDF una vista previa de cómo quedará conformado el expediente del accidente.

Registrar inspecciones técnicas

En este componente se registran las inspecciones técnicas que se realizan a las unidades policiales del CPNB. La siguiente imagen muestra la vista de cuando se accede al módulo Taller, esta vista mostrará las inspecciones técnicas registradas en el sistema, con los siguientes atributos: Número de inspección, Fecha de emisión, Serial de carrocería, Grupo de unidad, Responsable, Realizador por y el Estado de la inspección. Brinda las opciones de Adicionar, Modificar, Cancelar, Imprimir y realizar un Informe de resultado. Además posee una búsqueda, la cual se realiza por los campos Número de inspección, Placa y Serial de carrocería.

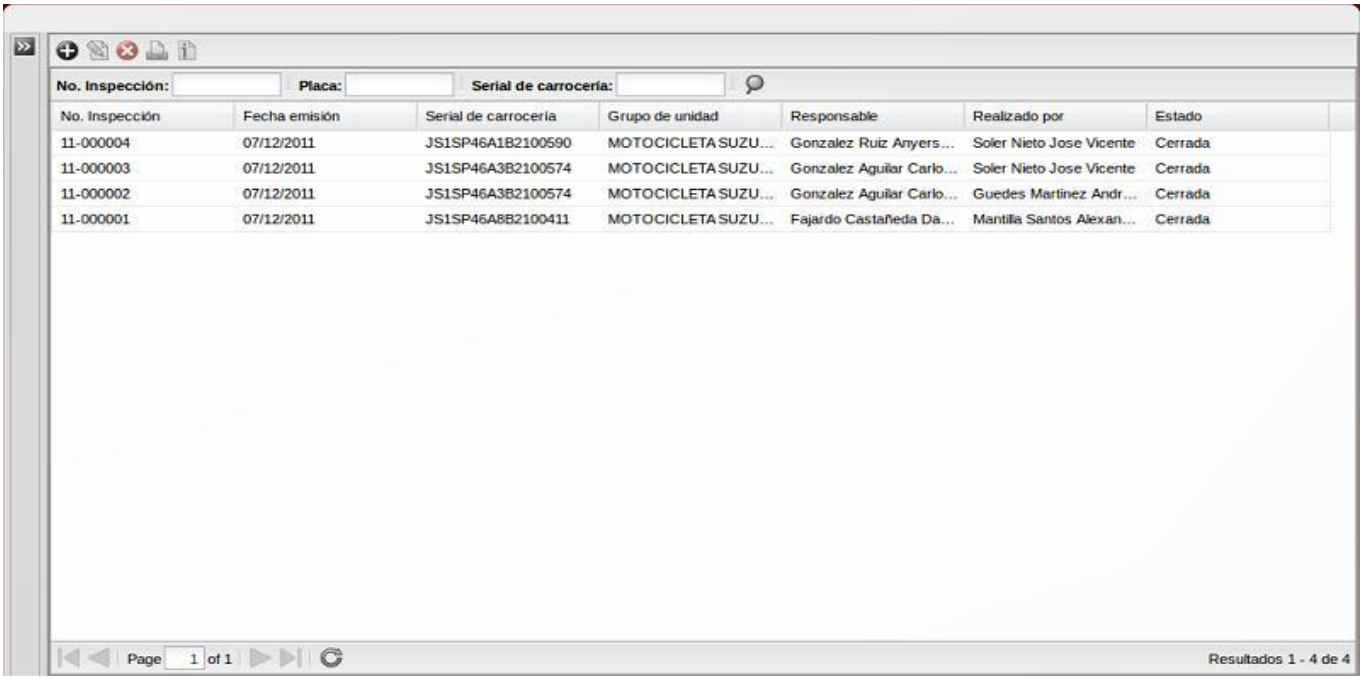

Imagen 8 Registrar inspecciones técnicas

Adicionar Inspección técnica

Al seleccionar la opción Adicionar se muestra una interfaz a través de la cual pueden ser insertados todos los datos que son necesarios para registrar la inspección técnica. Al dar clic en el botón Aplicar, se añade al sistema la inspección técnica registrada y se limpian los campos de la interfaz para ingresar una nueva, en caso de haberle dado clic al botón Aceptar, de igual manera se ingresan los datos de la inspección en el sistema y se muestra un mensaje de confirmación, pero seguidamente se vuelve a la vista principal. En el **Anexo 4** se muestra la interfaz para adicionar una inspección técnica.

Modificar Inspección técnica

Si se quiere modificar una inspección técnica de las existentes, se selecciona la misma, la cual tiene que encontrarse en estado abierta y posteriormente se va a la opción Modificar que activará una interfaz similar a la de Adicionar pero con la diferencia de que se le añade el campo Estado de la unidad y que los demás campos vienen cargados. Se modifican los campos que se quieren cambiar y se presiona Aceptar para confirmar la operación. Se mostrará un mensaje de información. En el **Anexo 5** se muestra la interfaz para modificar una inspección técnica.

Cancelar Inspección técnica

Para cancelar una inspección técnica que haya sido registrada en el sistema, se selecciona la misma, la cual tiene que estar en estado abierta y luego se va a la opción Cancelar, seguidamente se visualiza un mensaje de confirmación. Si se selecciona Aceptar, será cancelada la inspección técnica, es decir, la inspección pasará a inactiva pero no se elimina de la base de datos y se mostrará un mensaje de información.

Imprimir Inspección técnica

Para imprimir una inspección técnica, primeramente se selecciona esta y luego la opción imprimir, una vez realizado esto se mostrará en formato PDF una vista previa de cómo quedará conformado el expediente del accidente.

Realizar Informe resultado

Para generar un informe resultado de la inspección técnica, primeramente se selecciona esta y luego la opción Informe resultado. Si a dicha inspección no se le ha generado un informe resultado aún, entonces

se activará una interfaz que permitirá crear el informe resultado con los resultados de la inspección técnica. En el **Anexo 6** se muestra la interfaz para crear un informe resultado.

Registrar Inspecciones técnicas a una unidad

En este componente se registran las inspecciones técnicas de una unidad específica, seleccionada en la interfaz Unidades policiales y posteriormente se selecciona el botón Inspección, accediendo a la interfaz donde se registran las inspecciones técnicas de esa unidad. En la vista interfaz se muestran las inspecciones registradas por esa unidad, además de los siguientes atributos: Número de inspección, Fecha de emisión, Responsable, Realizador por y el Estado de la inspección. Brinda las opciones de Adicionar, Modificar, Cancelar, Imprimir y realizar un Informe de resultado. Además posee una búsqueda, la cual se realiza por el campo Número de inspección.

Adicionar Inspección técnica a una unidad

Al seleccionar la opción Adicionar se muestra una interfaz a través de la cual pueden ser insertados todos los datos que son necesarios para registrar la inspección técnica. Al dar clic en el botón Aceptar, se añade al sistema la inspección técnica registrada y se muestra un mensaje de confirmación, y seguidamente se vuelve a la vista principal. En el **Anexo 7** se muestra la interfaz para adicionar una inspección técnica a una unidad policial.

Modificar Inspección técnica a una unidad

Si se quiere modificar una inspección técnica de las existentes, se selecciona la misma, la cual tiene que encontrarse en estado abierta y posteriormente se va a la opción Modificar que activará una interfaz con los campos cargados. Se modifican los campos que se quieren cambiar y se presiona Aceptar para confirmar la operación. Se mostrará un mensaje de información. En el **Anexo 8** se muestra la interfaz para modificar una inspección técnica a una unidad policial.

Cancelar Inspección técnica a una unidad

Para cancelar una inspección técnica a una unidad, se selecciona la misma, la cual tiene que estar en estado abierta y luego se va a la opción Cancelar. Cuando se haya terminado de realizar la operación se mostrará un mensaje de confirmación. Si se selecciona Aceptar, será cancelada la inspección, es decir, la inspección pasará a inactiva pero no se elimina de la base de datos y se mostrará un mensaje de información.

Imprimir Inspección técnica a una unidad

Para imprimir una inspección técnica a una unidad, primeramente se selecciona la inspección y luego la opción imprimir, una vez realizado esto se mostrará en formato PDF una vista previa de cómo quedará conformado el expediente del accidente.

Registro de lectura

En este componente se registran las lecturas de los medidores de las unidades policiales del CPNB. La interfaz mostrará las lecturas registradas, con los siguientes atributos: Número de identificación, Serial de carrocería, Placa, Unidad de medida, Valor base, Valor actual, Valor acumulado y Fecha de lectura. Brinda las opciones de Registrar una nueva lectura, Eliminar última lectura, Reiniciar medidor y crear un historial de lectura. Además posee una búsqueda, la cual se realiza por los campos Número de identificación, Placa y Serial de carrocería.

Gestionar tipo de unidad

En este componente se registran los tipos de unidad, donde se muestran los siguientes atributos: Nombre y Abreviatura. Brinda las opciones de Adicionar, Modificar y Eliminar. Además posee una búsqueda, la cual se realiza por los campos Nombre y Abreviatura. En el **Anexo 9** se muestra la interfaz para gestionar tipos de unidad.

2.15. Conclusiones parciales

En este capítulo se mostró la solución que se propuso, tanto para el diseño, como para la implementación de los procesos Administración de Inspecciones Técnicas y Accidentes, quedando evidenciado en los artefactos construidos en cada una de las etapas desarrolladas. Se expusieron los mecanismos y patrones de diseño utilizados, también se elaboraron artefactos como los diagramas de clases de diseño, que permitieron desarrollar de manera más eficiente la implementación de los procesos Administración de Inspecciones Técnicas y Accidentes y el diagrama de componentes desde la perspectiva del módulo Vehículo que permitió una visión más clara de la implementación del sistema, se conformó el modelo de datos, las integraciones entre los componentes que están relacionados con los procesos Administración de Inspecciones Técnicas y Accidentes y fueron explicadas de forma breve las funcionalidades de los mismo.

CAPÍTULO 3: VALIDACIÓN Y PRUEBAS

3.1. Introducción

En el presente Capítulo se muestran algunas métricas que se aplican actualmente para validar la calidad en el diseño de software y se definen cuáles se aplicaron al diseño de la solución propuesta. Estas proporcionan una medida de cuán evolucionado se encuentra el desarrollo de la aplicación informática desde la visión interna que proporcionan los parámetros que estas definen. Además se valida mediante pruebas el software realizado, haciendo uso específico de las pruebas de caja blanca y caja negra.

3.2. Métricas para evaluar el diseño propuesto

Una métrica es un instrumento que cuantifica un criterio y persigue comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado al nivel del proyecto (31).

Para la evaluación de la calidad del diseño propuesto para los procesos Administración de Inspecciones Técnicas y Accidentes del Sistema de Gestión de Mantenimiento Vehicular se hizo un estudio de la creación de métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto; en el mismo se abarcan atributos de calidad que permiten medir la calidad del diseño propuesto. Dentro de los que se encuentran:

- **Responsabilidad:** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación:** Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización:** Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento:** Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- **Complejidad del mantenimiento:** Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costos y la planificación del proyecto.

- **Cantidad de pruebas:** Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto diseñado.

Las métricas concebidas como instrumento para evaluar la calidad del diseño y su relación con los atributos de calidad definidos son las siguientes:

- Tamaño Operacional de Clase (TOC).
- Relaciones entre Clases (RC).

Las métricas escogidas son bastante eficientes ya que dan una medida de la calidad del diseño del componente y su utilización es sencilla y fácil.

Tamaño Operacional de Clase (TOC)

El tamaño operacional de clase (TOC), está dado por el número de métodos asignados a una clase.

Tabla 6 Tamaño operacional de clase (TOC)

Atributo que afecta	Modo en que lo enfoca
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Instrumento de medición de la métrica Tamaño operacional de clase (TOC).

Tabla 7 Rango de valores de para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC.

Atributos de calidad	Categorías	Criterio
Responsabilidad	Baja	<= Prom.(4.2)
	Media	Entre Prom. y 2 * Prom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	<= Prom.
	Media	Entre Prom. y 2 * Prom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2* Prom.
	Media	Entre Prom. y 2 * Prom.
	Alta	<= Prom.

Para la evaluación de las clases fueron utilizados además los umbrales para el tamaño general de las clases.

Tabla 8 Umbrales para el TOC

Tamaño operacional de la clase	Criterio
Pequeña	<= Prom.(4.2)
Media	Entre Prom. y 2 * Prom.
Grande	> 2* Prom.

Tabla 9 Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

No	Clase	Cantidad de operaciones	Responsabilidad	Complejidad de implementación	Reutilización	Tamaño de la clase
1	GestaccidenteController	14	Alta	Alta	Baja	Grande
2	GestionarinspeccionesController	15	Alta	Alta	Baja	Grande
3	GestpartesController	1	Baja	Baja	Alta	Pequeña
4	GestinspeccionTecnicaunidadController	11	Alta	Alta	Baja	Grande
5	GestregistrolecturaController	5	Media	Media	Media	Media
6	GesttipodeunidadController	11	Alta	Alta	Baja	Grande
7	DatAccidenteModel	5	Media	Media	Media	Media
8	DatInspeccionTecnicaModel	7	Media	Media	Media	Media
9	DatInvolucradoexternoModel	3	Baja	Baja	Alta	Pequeña
10	DatInformeinspeccionModel	3	Baja	Baja	Alta	Pequeña
11	DatFiscalModel	2	Baja	Baja	Alta	Pequeña
12	DatRespdirectoaccidenteModel	3	Baja	Baja	Alta	Pequeña
13	DatUnidadModel	3	Baja	Baja	Alta	Pequeña

14	DatVehiculoinvolucradoModel	4	Baja	Baja	Alta	Pequeña
15	HislecturaModel	2	Baja	Baja	Alta	Pequeña
16	NomEstadoparteModel	1	Baja	Baja	Alta	Pequeña
17	NomGravedadModel	2	Baja	Baja	Alta	Pequeña
18	NomModalidadModel	2	Baja	Baja	Alta	Pequeña
19	DatInspeccionparteModel	2	Baja	Baja	Alta	Pequeña
20	DatInvestigadorModel	3	Baja	Baja	Alta	Pequeña
21	DatInvolucradointernoModel	3	Baja	Baja	Alta	Pequeña
22	DatRegistrolecturaModel	3	Baja	Baja	Alta	Pequeña
23	DatResponsabledirectoModel	1	Baja	Baja	Alta	Pequeña
24	NomEstadounidadModel	1	Baja	Baja	Alta	Pequeña
25	NomTipounidadModel	3	Baja	Baja	Alta	Pequeña
26	NomParteModel	3	Baja	Baja	Alta	Pequeña
27	DatParteunidadModel	1	Baja	Baja	Alta	Pequeña

Resultados de la aplicación de la métrica

Analizando los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, teniendo en cuenta que el 74% del las clases pertenecientes a los procesos Administración de Inspecciones Técnicas y Accidentes se encuentran dentro de la categoría de pequeñas, y que el 85% de las clases de dichos procesos poseen evaluaciones positivas en los atributos de calidad (Responsabilidad,

Complejidad de Implementación y Reutilización), se puede concluir que el diseño realizado es simple, además de que tiene una calidad aceptable.

Relaciones entre Clases (RC)

Las relaciones entre las clases (RC), está dado por el número de relaciones de uso de una clase con otras.

Tabla 10 Relaciones entre clases (RC)

Atributo que afecta	Modo en que lo enfoca
Acoplamiento	Un aumento del RC implica un aumento del acoplamiento de la clase.
Complejidad del mantenimiento	Un aumento del RC implica un aumento de la complejidad de mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 11 Rango de valores para la evaluación técnica de los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas) relacionados con la métrica RC.

Atributos de calidad	Categorías	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2

	Alto	>2
Complejidad de mantenimiento	Baja	<= Prom.
	Media	Entre Prom. y 2 * Prom.
	Alta	> 2*Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2 * Prom.
	Alta	<= Prom.
Cantidad de pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2 * Prom.
	Alta	> 2*Prom.

Tabla 12 Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas).

No	Clase	Cantidad de relaciones	Acoplamiento	Complejidad del mantenimiento	Reutilización	Cantidad de pruebas

1	DatAccidenteModel	10	Alto	Alta	Baja	Alta
2	DatInspeccionTecnicaModel	6	Alto	Alta	Baja	Alta
3	DatInvolucradoexternoModel	1	Bajo	Baja	Alta	Baja
4	DatInformeinspeccionModel	1	Bajo	Baja	Alta	Baja
5	DatFiscalModel	1	Bajo	Baja	Alta	Baja
6	DatRespdirectoaccidenteModel	1	Bajo	Baja	Alta	Baja
7	DatUnidadModel	1	Bajo	Baja	Alta	Baja
8	DatVehiculoinvolucradoModel	1	Bajo	Baja	Alta	Baja
9	HislecturaModel	2	Medio	Media	Media	Media
10	NomEstadoparteModel	1	Bajo	Baja	Alta	Baja
11	NomGravedadModel	1	Bajo	Baja	Alta	Baja
12	NomModalidadModel	1	Bajo	Baja	Alta	Baja
13	DatInspeccionparteModel	2	Medio	Media	Media	Media
14	DatInvestigadorModel	1	Bajo	Baja	Alta	Baja
15	DatInvolucradointernoModel	1	Bajo	Baja	Alta	Baja
16	DatRegistrolecturaModel	3	Alto	Alta	Baja	Alta
17	DatResponsabledirectoModel	1	Bajo	Baja	Alta	Baja
18	NomEstadounidadModel	1	Bajo	Baja	Alta	Baja

19	NomTipounidadModel	4	Alto	Alta	Baja	Alta
20	NomParteModel	1	Bajo	Baja	Alta	Baja
21	DatParteunidadModel	1	Bajo	Baja	Alta	Baja
22	DatInspeccionparte	0	Ninguno	Baja	Alta	Baja
23	DatInvestigador	0	Ninguno	Baja	Alta	Baja
24	DatInvolucradointerno	0	Ninguno	Baja	Alta	Baja
25	DatRegistrolectura	0	Ninguno	Baja	Alta	Baja
26	DatResponsabledirecto	0	Ninguno	Baja	Alta	Baja
27	NomEstadounidad	0	Ninguno	Baja	Alta	Baja
28	NomModalidad	0	Ninguno	Baja	Alta	Baja
29	DatAccidente	0	Ninguno	Baja	Alta	Baja
30	DatFiscal	0	Ninguno	Baja	Alta	Baja
31	DatInformeinspeccion	0	Ninguno	Baja	Alta	Baja
32	DatInspeccionTecnica	0	Ninguno	Baja	Alta	Baja
33	DatInvolucradoexterno	0	Ninguno	Baja	Alta	Baja
34	DatRespdirectoaccidente	0	Ninguno	Baja	Alta	Baja
35	DatUnidad	0	Ninguno	Baja	Alta	Baja
36	DatVehiculoinvolucrado	0	Ninguno	Baja	Alta	Baja

37	Hislectura	0	Ninguno	Baja	Alta	Baja
38	NomEstadoparte	0	Ninguno	Baja	Alta	Baja
39	NomGravedad	0	Ninguno	Baja	Alta	Baja
40	DatParteunidad	0	Ninguno	Baja	Alta	Baja
41	NomParte	0	Ninguno	Baja	Alta	Baja
42	NomTipounidad	0	Ninguno	Baja	Alta	Baja

Resultados de la aplicación de la métrica

Al analizar los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño de los procesos Administración de Inspecciones Técnicas y Accidentes es simple y tiene una calidad aceptable para realizar la implementación, teniendo en cuenta que el 90% de las clases incluidas en dichos procesos poseen menos de 3 dependencias de otras clases. Además de que el 85% no tiene o es bajo el acoplamiento entre clases. Por último los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización poseen niveles aceptables en un 85 % de las clases.

3.3. Pruebas de caja blanca

La Prueba de Caja Blanca también se conoce como Prueba de Caja Transparente o de Cristal. Esta prueba consiste específicamente en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento (31). Dentro de la prueba de caja blanca se incluyen las Técnicas de Pruebas que serán descritas a continuación:

- **Prueba del Camino Básico:** Permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos.
- **Prueba de Condición:** Ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método.

- **Prueba de Flujo de Datos:** Se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.
- **Prueba de Bucles:** Se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución de todos los bucles en sus límites operacionales.

La prueba de caja blanca empleada en la solución desarrollada fue la prueba del camino básico, la cual permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjuto básico de caminos de ejecución, garatizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa (31).

Para realizar esta técnica es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se enumeran las sentencias de código del procedimiento realizado sobre el método guardarInspeccionTecnica ($arrinspeccion).

```
public function guardarInspeccionTecnica(&$arrinspeccion){
    $this->conn->beginTransaction();//1
        if(!empty($arrinspeccion['idinspeccion'])){//2
            $inspeccion = DatInspeccionTecnica::buscarPorId($arrinspeccion['idinspeccion']);//3
            $motivo=0;//3
            if($arrinspeccion['idEstadounidadMod']==3){//4
                $motivo=6;//5
            }//6
            $modeloUnidad= new DatUnidadModel();//7
            if($arrinspeccion['idEstadounidadMod']){//8
              $arrUnidad= $modeloUnidad->cambiarEstadoMotivo($arrinspeccion['idunidad']);//9
              $inspeccion->idestadoinspecciontecnica = 2;//9
            }//10
              $codigoInspeccion=$inspeccion->codigoit;//11
        }//12
        else{//13
            $respuesta=$this->ComprobarExistenciaInspeccionUnidad( $arrinspeccion['idunidad'] );//14
            if($respuesta==1){//15
                throw new ZendExt_Exception('MTTOGUOO2');//16
            }//17
            else{//18
                $inspeccion = new DatInspeccionTecnica();//19
                $inspeccion->idestadoinspecciontecnica = 1;//19
                $numero=DatInspeccionTecnica::obtenerNoInspeccion();//19
                $modeloUnidad= new DatUnidadModel();//19
                $arrUnidad= $modeloUnidad->cambiarEstadoMotivo($arrinspeccion['idunidad']);//19
                $inspeccion->motivo = $arrinspeccion['motivoInspeccion'];//19
                $inspeccion->lugar = $arrinspeccion['lugar'];//19
                $inspeccion->idunidad = $arrinspeccion['idunidad'];//19
                $inspeccion->idtrabajador = $arrinspeccion['idtrabajador'];//19
                $inspeccion->idresponsabled = empty($arrinspeccion['idresponsabled'])?Null:$arrinspeccion
                $inspeccion->fechaemision = $arrinspeccion['fechaemision'];//19
```

```
                $inspeccion->codigoit =$numero;//19
                $inspeccion->estadounidad =empty($arrUnidad['idestadounidad'])?Null:$arrUnidad['idestadou
                $inspeccion->motivoestado = empty($arrUnidad['idmotivoestadounidad'])?Null:$arrUnidad['id
                $inspeccion->codigoasociado =empty($arrinspeccion['codigoasociado'])?Null:$arrinspeccion[
                $codigoInspeccion=$numero;//19
            }//20
        }//21
            $inspeccion->observacion = $arrinspeccion['observaciones']; //22
            $inspeccion->save();//22
            $arrinspeccion['idInspeccion']= $inspeccion->idinspeccion;//22
            $arrayPartes = new DatInspeccionparteModel();//22
                if (isset($arrinspeccion['estadoPartes'])){//23
                $arrayPartes->guardar($arrinspeccion['estadoPartes'],$inspeccion->idinspeccion);//24
                }//25
    $this->conn->commit();//26
    $arrinspeccion['action']=$action;//26
    $arrinspeccion['codigoInspeccion']=$codigoInspeccion;//26
    $arrinspeccion['idinspeccion']=$inspeccion->idinspeccion;//26
}//27
```

Imagen 9 Código del método guardarInspeccionTecnica($arrinspeccion)

A continuación se muestra el grafo de flujo asociado a la funcionalidad.

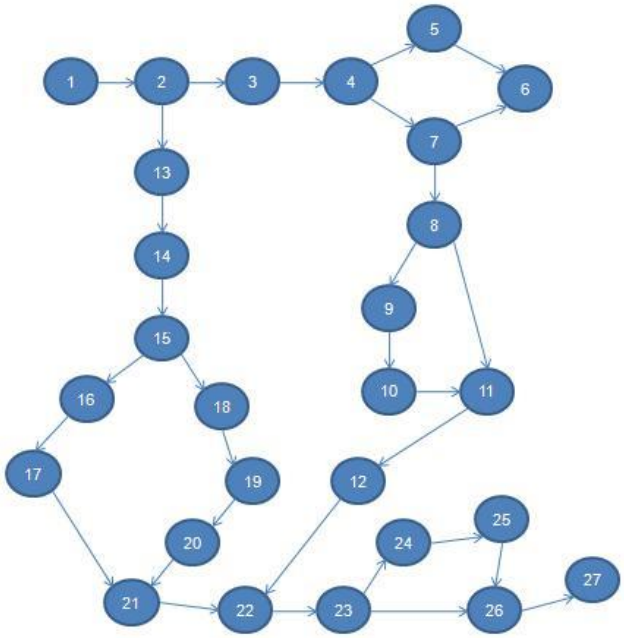

Imagen 10 Grafo de flujo asociado al algoritmo guardarInspeccionTecnica($arrinspeccion)

Cálculo de la complejidad ciclomática a partir de un segmento de código

Luego de haber construido el grafo, se realiza el cálculo de la complejidad ciclomática, mediante las tres fórmulas utilizadas para el análisis de complejidad del algoritmo, las cuales tienen que arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

Fórmulas para calcular complejidad ciclomática:

1. V (G) = (A - N) + 2

Donde **"A"** es la cantidad de Aristas y **"N"** la cantidad de Nodos.

V (G) = (31 - 27) + 2

V (G) = 6

2. V (G) = P + 1

Siendo **"P"** la cantidad de Nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 5 + 1

V (G) = 6

3. V (G) = R

Donde **"R"** representa la cantidad de regiones en el grafo.

V (G) = 6

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 6, lo que significa que existen esa cantidad posible de caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo.

Tabla 13 Caminos básicos del flujo

Número	Camino básico
1	**1-2-3-4-5-6-7-8-9-10-11-12-22-23-24-25-26-27**
2	**1-2-3-4-7-8-9-10-11-12-22-23-24-25-26-27**

3	**1-2-3-4-7-8-11-12-22-23-24-25-26-27**
4	**1-2-3-4-7-8-11-12-22-23-26-27**
5	**1-2-13-14-15-16-17-21-22-23-24-25-26-27**
6	**1-2-13-14-15-18-19-20-21-22-23-24-25-26-27**

Luego se procede a ejecutar los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo.

Caso de prueba para el camino básico 1

Camino: 1-2-3-4-5-6-7-8-9-10-11-12-22-23-24-25-26-27

Descripción: El dato de entrada cumplirá con el siguiente requisito:

El parámetro $arrinspeccion no está vacío, contiene el identificador de una inspección técnica determinada y el estado de la unidad a la cual se le va a realizar la inspección técnica es inoperativa.

Entrada: $arrinspeccion = Array ([estadoPartes] => Array([0] => Array([idparte] => 9000000, [denomParte] => ruedas, [idestadop] => 2, [denomEstado] => Bueno)) [idEstadounidadMod] => 3, [idinspeccion] => 90000000022, [idtrabajador] =>,[fechaemision] =>, [idresponsabled] =>, [lugar] =>, [motivoInspeccion] =>, [conductor] =>, [idunidad] => 90000000001, [observaciones] =>)

Resultados esperados: Se espera que sean modificados los datos de la inspección técnica entrados por parámetro y el estado de la unidad pasará a ser inoperativa con motivo taller.

Caso de prueba para el camino básico 2

Camino: 1-2-3-4-7-8-9-10-11-12-22-23-24-25-26-27

Descripción: El parámetro $arrinspeccion no está vacío, contiene el identificador de una inspección técnica determinada y el estado de la unidad a la cual se le va a realizar la inspección técnica no es inoperativa.

Entrada: $arrinspeccion = Array ([estadoPartes] => Array([0] => Array([idparte] => 9000000, [denomParte] => ruedas, [idestadop] => 2, [denomEstado] => Bueno)) [idEstadounidadMod] => 1, [idinspeccion] =>

90000000022, [idtrabajador] =>,[fechaemision] =>, [idresponsabled] =>, [lugar] =>, [motivoInspeccion] =>, [conductor] =>, [idunidad] => 90000000001, [observaciones] =>)

Resultados esperados: Se espera que sean modificados los datos de la inspección técnica entrados por parámetro, la unidad pasará a estado operativa y la inspección técnica será cerrada.

Caso de prueba para el camino básico 3

Camino: 1-2-3-4-7-8-11-12-22-23-24-25-26-27

Descripción: El parámetro $arrinspeccion no está vacío, contiene el identificador de una inspección técnica determinada y no posee el estado de la unidad.

Entrada: $arrinspeccion = Array ([estadoPartes] => Array([0] => Array([idparte] => 9000000, [denomParte] => ruedas, [idestadop] => 2, [denomEstado] => Bueno)) [idEstadounidadMod] =>, [idinspeccion] => 90000000022, [idtrabajador] =>,[fechaemision] =>, [idresponsabled] =>, [lugar] =>, [motivoInspeccion] =>, [conductor] =>, [idunidad] => 90000000001, [observaciones] =>)

Resultados esperados: Se espera que sean modificados los datos de la inspección técnica entrados por parámetro.

Caso de prueba para el camino básico 4

Camino: 1-2-3-4-7-8-11-12-22-23-26-27

Descripción: El parámetro $arrinspeccion no está vacío, contiene el identificador de una inspección técnica determinada, no posee el estado de la unidad y el estado de las partes de la unidad a la cual se le está realizando la inspección técnica.

Entrada: $arrinspeccion = Array ([estadoPartes] => Array()) [idEstadounidadMod] =>, [idinspeccion] => 90000000022, [idtrabajador] =>,[fechaemision] =>, [idresponsabled] =>, [lugar] =>, [motivoInspeccion] =>, [conductor] =>, [idunidad] => 90000000001, [observaciones] =>)

Resultados esperados: Se espera que sean modificados los datos de la inspección técnica entrados por parámetro y que no sea guardado el estado de las partes de la unidad a la cual se le está realizando la inspección técnica.

Caso de prueba para el camino básico 5

Camino: 1-2-13-14-15-16-17-21-22-23-24-25-26-27

Descripción: El parámetro $arrinspeccion no está vacío, contiene los datos necesarios para adicionar una nueva inspección técnica, aunque no posee idinspeccion.

Entrada: $arrinspeccion = Array ([estadoPartes] => Array()) [idEstadounidadMod] =>, [idinspeccion] =>, [idtrabajador] =>,[fechaemision] =>, [idresponsabled] =>, [lugar] =>, [motivoInspeccion] =>, [conductor] =>, [idunidad] => 90000000001, [observaciones] =>)

Resultados esperados: Se espera que muestre un mensaje informando que la unidad a la cual se le quiere realizar una inspección técnica, ya posee una inspección, por cuanto no puede ser adicionada.

Caso de prueba para el camino básico 6

Camino: 1-2-13-14-15-18-19-20-21-22-23-24-25-26-27

Descripción: El parámetro $arrinspeccion no está vacío, contiene los datos necesarios para adicionar una nueva inspección técnica.

Entrada: Array([estadoPartes] => Array([0] => Array([idparte] => 9000004, [denom] => ARRANQUE, [idestadop] =>)[1] => Array([idparte] => 9000005, [denom] => ASIENTO, [idestadop] =>))[idtrabajador] => 9000003, [fechaemision] => 09/05/2012, [idresponsabled] =>, [lugar] => Transporte, [motivoInspeccion] => accidente, [conductor] => Pedro, [idunidad] => 90000000093, [observaciones] =>)

Resultados esperados: Se espera que sean adicionados los datos de la inspección técnica entrados por parámetro.

Con la aplicación de los casos de prueba expuestos anteriormente se comprobó que el flujo de trabajo de las funcionalidades es correcto, ya que se probó que cada sentencia es ejecutada al menos una vez, cumpliéndose así las condiciones de la prueba.

3.4. Pruebas de caja negra

A este tipo de prueba también se le conoce como Prueba de Caja Opaca o Inducida por los Datos. Se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación, esta prueba se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo internamente, es decir, solo trabaja sobre su interfaz externa (31).

En esencia permite 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.

Dentro de la prueba de caja negra se incluyen las Técnicas de Pruebas que serán descritas a continuación (31):

Partición equivalente: Es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

Análisis de valores límite: Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites como técnica de prueba. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el análisis de valores límite lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el análisis de valores límite obtiene casos de prueba también para el campo de salida.

Para cada uno de los requisitos funcionales fueron definidos casos de pruebas, los cuales son los siguientes:

Caso de prueba para el requisito Adicionar accidente

Condiciones de ejecución

- Se debe identificar y autenticar ante el sistema, además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar el subsistema **Mantenimiento/Vehículos/Accidentes**.

Tabla 14 Descripción del caso de prueba para el requisito Adicionar accidente

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1:Adicionar accidente.	El sistema debe permitir	EP 1.1: Adicionar accidente	– Se introducen los

	adicionar accidentes.	introduciendo datos válidos.	datos del accidente correctamente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
		EP 1.2: Adicionar accidente introduciendo datos válidos presionando el botón **Aplicar**.	– Se introducen los datos del accidente correctamente. – Se presiona el botón **Aplicar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
		EP 1.3: Adicionar accidente introduciendo datos inválidos.	– Se introducen los datos inválidos del accidente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de error.
		EP 1.4: Adicionar accidente dejando campos vacíos.	– Se introducen los datos dejando algún campo en blanco. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando

		del error. – Se presiona el botón **Aceptar**.
	EP 1.5: Adicionar accidente dejando la pestaña Involucrado interno, Involucrado externo, Vehículo involucrado, Comisión investigadora y Proceso legal vacíos.	– Se introducen los datos dejando estos campos en blanco. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
	EP 1.6: Adicionar accidente dejando un vehículo involucrado sin al menos un involucrado externo asociado.	– Se introducen los datos del accidente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de error. – Se presiona el botón **Aceptar**.
	EP 1.7: Adicionar accidente a una unidad policial en el sistema con el mismo No. Expediente de tránsito de un accidente a una unidad policial ya registrado en el sistema.	– Se introducen los datos. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error. – Se presiona el botón **Aceptar**.
	EP 1.8: Adicionar accidente a una unidad policial en el sistema con el mismo No. Fiscal de un accidente a una unidad policial ya registrado	– Se introducen los datos. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.

	en el sistema.	– Se presiona el botón **Aceptar**.
	EP 1.9: Adicionar accidente a una unidad policial en el sistema con el mismo No. Expediente Fiscalía de un accidente a una unidad policial ya registrado en el sistema.	– Se introducen o no los datos del accidente. – Se presiona el botón **Cancelar**.
	EP 1.10: Cancelar.	Se introducen o no los datos del accidente. Se presiona el botón **Cancelar**.

Para más información sobre el caso de prueba para el requisito Adicionar accidente, consultar el documento CG-SM-DR-337. Los casos de prueba elaborados para los restantes requisitos funcionales se encuentran en los documentos: CG-SM-DR-342, CG-SM-DR-340, CG-SM-DR-338, CG-SM-DR-339, CG-SM-DR-341, CG-SM-DR-237, CG-SM-DR-236, CG-SM-DR-234, CG-SM-DR-233, CG-SM-DR-235, CG-SM-DR-238, CG-SM-DR-251, CG-SM-DR-250, CG-SM-DR-249, CG-SM-DR-252, CG-SM-DR-298, CG-SM-DR-296, CG-SM-DR-297, CG-SM-DR-292, CG-SM-DR-294, CG-SM-DR-290, CG-SM-DR-293, CG-SM-DR-295, CG-SM-DR-291, CG-SM-DR-346, CG-SM-DR-348, CG-SM-DR-344, CG-SM-DR-343, CG-SM-DR-345, CG-SM-DR-347, CG-SM-DR-243, CG-SM-DR-243, CG-SM-DR-245, CG-SM-DR-248, CG-SM-DR-243, CG-SM-DR-242, CG-SM-DR-241, CG-SM-DR-244, CG-SM-DR-287, CG-SM-DR-285, CG-SM-DR-286, CG-SM-DR-288, CG-SM-DR-260, CG-SM-DR-257, CG-SM-DR-258, CG-SM-DR-255, CG-SM-DR-254, CG-SM-DR-253, CG-SM-DR-256, CG-SM-DR-240, CG-SM-DR-289, CG-SM-DR-378, CG-SM-DR-207, CG-SM-DR-208, CG-SM-DR-204, CG-SM-DR-206, CG-SM-DR-205, CG-SM-DR-217, CG-SM-DR-218, CG-SM-DR-214, CG-SM-DR-216, CG-SM-DR-215, CG-SM-DR-192, CG-SM-DR-184.

La aplicación desarrollada fue probada por la entidad Calisoft, donde fue comprobado el correcto funcionamiento de la misma a partir de los diseños de los casos de prueba. Como constancia de los satisfactorios resultados obtenidos ante esta evaluación, esta entidad emitió un acta de liberación de software (ver **Anexo 10**).

Una vez liberada la aplicación fue presentada ante el cliente, el cual luego de haberla probado por su parte estuvo satisfecho con el producto entregado; prueba de esto lo constituyen los avales otorgados por el mismo y por el gerente general del proyecto por parte de Cuba, en los cuales consta que el sistema realizado cumple con las condiciones de calidad necesarias y satisface las necesidades plasmadas por los clientes, contribuye a mejorar la gestión de los procesos que se realizan en el área de Transporte del CPNB y les ofrece ciertas ventajas su empleo. Para más información acerca de los avales emitidos consultar **Anexo 10**.

3.5. Conclusiones parciales

En el capítulo se hace una valoración crítica del diseño propuesto mediante el uso de las métricas (TOC y RC) obteniéndose como resultado que el diseño estaba realizado de forma simple y con una calidad aceptable. Se aborda acerca las pruebas de caja blanca y caja negra y las particularidades de cada una de ellas. Se hace una validación del sistema realizado utilizándose la prueba del camino básico, en la cual se obtuvieron resultados favorables, demostrándose con ello la calidad y eficiencia del sistema.

CONCLUSIONES GENERALES

Una vez terminado el presente trabajo de diploma se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, para esto:

- Se analizaron ventajas y deficiencias de sistemas informáticos tanto nacionales como internacionales vinculados a la gestión de flotas; evidenciándose de esta manera la no existencia de una solución informática capaz de ejecutar las funcionalidades necesarias para el área de Transporte del Cuerpo de la Policía Nacional Bolivariana.
- Se realizó el diseño y la implementación de los procesos Administración de Inspecciones Técnicas y Accidentes del Sistema de Gestión de Mantenimiento Vehicular del área de Transporte del CPNB, bajo los estándares definidos en el CEIGE, probado y validado mediante métricas y pruebas de software.

La solución propuesta es novedosa, su importancia radica en la realización de los procesos Administración de Inspecciones Técnicas y Accidentes del Sistema de Gestión de Mantenimiento Vehicular del área de Transporte del CPNB, permitiendo llevar un control de los accidentes y las inspecciones técnicas en esta área. Se logró humanizar el trabajo a los clientes, pues anteriormente todo este trabajo era de forma manual y se tornaba engorroso. Se tributa a la toma de decisiones de la jefatura a través de los reportes que ofrecen consolidados de información acerca de la situación actual de las unidades policiales.

RECOMENDACIONES

Se recomienda que en el desarrollo de futuros sistemas para la gestión de mantenimiento sea utilizado el presente trabajo como referencia en cuanto a gestión de los accidentes y las inspecciones técnicas de vehículos de un parque automotor.

Se recomienda que no sea publicada la información que es tratada en el presente trabajo en eventos internacionales pues se maneja información confidencial del Cuerpo de la Policía Nacional Bolivariana.

BIBLIOGRAFÍA

1. Gestion de flotas Software. [En línea] 2008-2011. [Citado el: 17 de octubre de 2011.] http://www.administraciontaxis.com.

2. Desarrollo de Software InteraSystem, S.A. de C.V. SoftFlot 3.0 Administración de Flotillas. [En línea] 2012. [Citado el: 10 de febrero de 2012.] http://www.usa.interasystem.com.

3. Web Corporativa de Advantur. [En línea] [Citado el: 12 de Noviembre de 2011.] http://www.advantur.com/Empresa.htm.

4. MP9 Software de Gestión de Mantenimiento CMMS/EAM - El más vendido en América Latina. [En línea] 2008 - 2011. [Citado el: 17 de Noviembre de 2011.] http://www.mpsystemsa.com.

5. Eugcom - Software Gestión de Flotas - Administración de Vehículos. [En línea] [Citado el: 18 de Noviembre de 2011.] http://www.eugcom.cl/flotas.html.

6. Sgestman. [En línea] 2010. [Citado el: 20 de Noviembre de 2011.] http://www.sgestman.cu.

7. Matanzas,Cuba:DESOFT,Offimant. [En línea] 2011. [Citado el: 23 de Noviembre de 2011.]

8. Ariza Rojas, Maribel y Molina García, Juan Carlos. Introducción y principios básicos del desarrollo de software basado en componentes. 2004.

9. Lidia Fuentes, J. M. Desarrollo de Software Basado en Componentes. 2003.

10. Abartia Team. Uso de la tecnología Ajax en el desarrollo web | Abartia Team. [En línea] 2008. [Citado el: 18 de Noviembre de 2011.] http://www.abartiateam.com/desarrollo-web/200602_uso-de-la-tecnologia-ajax-en-el-desarrollo-web.

11. PÉREZ, M. D. Software para el tratamiento inteligente de datos sobre patentes. [En línea] 2008. http://bvs.sld.cu/revistas/aci/v17_5_08/aci06508.htm..

12. Centro de Desarrollo Mozilla. JavaScript - MDN. [En línea] 2011. [Citado el: 22 de Noviembre de 2011.]

13. Framework - EcuRed. [En línea] [Citado el: 12 de Enero de 2012.] http://www.ecured.cu/index.php/Framework.

14. ExtJs. [En línea] 2008. [Citado el: 16 de Noviembre de 2011.] http://extjs.com/products/license.php..

15. ZEND FRAMEWORK. [En línea] [Citado el: 21 de Noviembre de 2011.] http://framework.zend.com/manual/en/.

16. Doctrine - PHP Object Persistence Libraries and More. [En línea] [Citado el: 19 de Noviembre de 2011.] http://www.doctrine-project.org..

17. UML CASE tool for software development. [En línea] 2006. [Citado el: 23 de Noviembre de 2011.] http://www.visual-paradigm.com/product/vpuml.

18. Ciberaula. Una Introducción a Apache. [En línea] 2010. [Citado el: 14 de Noviembre de 2011.] http://linux.ciberaula.com/articulo/linux_apache_intro.

19. PostgreSQL Global Development Group. PostgreSQL. [En línea] 1996-2011. [Citado el: 16 de Noviembre de 2011.] http://www.postgresql.org.

20. H8Red. Ventajas de usar Mozilla Firefox. [En línea] 2005-2011. [Citado el: 13 de Noviembre de 2011.] http://h8red.cl/2006/ventajas-de-usar-mozilla-firefox.

21. Ide Eclipse, Breve Guía. [En línea] [Citado el: 23 de Noviembre de 2011.] http://www.slideshare.net/Benedeti/ide-eclipse-breve-gua-201399.

22. Pressman, Roger S. Ingeniería del Software: Un enfoque práctico. 2005.

23. Cruz, Marietta Quevedo. Análisis y diseño del componente Matriz y Proyección del subsistema Mantenimiento de Cedrux. La Habana : s.n., 2011.

24. Ivisate, Annerys Armenteros. Análisis y diseño del componente Ejecución del subsistema Mantenimiento de Cedrux. La Habana : s.n., 2011.

25. Visconti, Marcelo y Astudillo, Hernán. Fundamento de Ingeniería de Software.

26. Pavón Mestras, Juan. El patrón Modelo-Vista-Controlador (MVC). Madrid : Universidad Complutense Madrid : s.n., 2008-2009.

27. Modelo de datos. [En línea] [Citado el: 15 de Enero de 2012.] http://aurea.es/wp-content/uploads/modelodedatos.pdf..

28. Sparx Systems - Tutorial UML 2 - Diagrama de Componentes. [En línea] 2000-2007. [Citado el: 5 de febrero de 2012.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html.

29. EcuRed:Enciclopedia cubana. [En línea] [Citado el: 12 de Febrero de 2012.] http://www.ecured.cu/index.php/Diagrama_de_despliegue.

30. Lázaro Pérez Bajuelo, Danier Estévez Laborí. Diseño e Implementación del Componente Nómina del subsistema Capital Humano del Sistema Integral de Gestión CEDRUX. La Habana : s.n., 2010.

31. Pressman, Roger S. Ingenieria de Software, Un enfoque práctico. 2005.