Universidad de las Ciencias Informáticas

Facultad 3

Título: Diseño e Implementación del proceso Ejecución del sistema Mantenimiento Vehicular v 1.0

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

Autor: Ariel Portal Hernández.

Tutores: Ing. Mailyn Hernández Gómez.

Ing. Dasiel Otero Dartayet.

"Año 54 de la Revolución"

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 31 días del mes de Mayo del año 2012.

Ariel Portal Hernández

_____

Firma del Autor

Ing. Mailyn Hernández Gómez.

_____

Firma del Tutor

Ing. Dasiel Otero Dartayet.

_____

Firma del Tutor

AGRADECIMIENTOS

Agradezco a todos los que supieron brindarme su ayuda, a mi familia por siempre estar presente, en especial a mis padres que siempre me apoyaron. Quisiera agradecer a todas las personas que de una forma u otra han contribuido al desarrollo de este trabajo, en especial a mis tutores que me ayudaron en los momentos necesarios, a mis compañeros de proyecto que estuvieron presentes en cada momento para brindarme su ayuda. A mis amistades por el apoyo brindado durante estos cinco años, a todos les entrego toda mi gratitud.

RESUMEN

En el área de Transporte del Cuerpo de la Policía Nacional Bolivariana (CPNB) de Venezuela, los procesos relacionados con la ejecución de los trabajos de mantenimiento se realizan de forma manual, por lo que no se tiene un control en las Órdenes de trabajo generadas a partir de inspecciones técnicas realizadas, ni en los recursos utilizados en las mismas, provocando así considerables pérdidas de recursos e información. A pesar de la existencia de varios sistemas que garantizan una buena gestión del proceso Ejecución del mantenimiento, la necesidad de mejorar este proceso en esta área según las obligaciones existentes ha impulsado la adopción de un nuevo sistema de gestión, que una vez instalado posibilite brindar a las dependencias policiales una aplicación que gestione la ejecución del mantenimiento vehicular de acuerdo a las necesidades reales, que facilite y mejore el trabajo diario.

El presente trabajo comprende el Diseño y la Implementación del Componente Ejecución del mantenimiento del sistema Mantenimiento Vehicular de manera tal que se logre la integración de este proceso con el resto de los procesos que se realizan en esta área.

El diseño realizado fue validado mediante métricas que demostraron que el diseño era simple, con alta reutilización y de poca complejidad. La implementación realizada fue validada mediante la aplicación de pruebas de caja negra y caja blanca evidenciando su correcto funcionamiento. Además fue revisada y liberada por Calisoft y por el cliente, generando avales que demuestran el grado de satisfacción del cliente con la solución propuesta.

PALABRAS CLAVE: Ejecución, mantenimiento, Órdenes de trabajo.

Tabla de Contenidos

Tabla de Contenidos

INTRODUCCIÓN

El desarrollo de la informática y las nuevas tecnologías en el mundo está en constante crecimiento, lo cual ha posibilitado que muchas empresas y organismos opten por la informatización de los procesos que realizan, en aras de optimizar los gastos incurridos producto de su ejecución. Dentro de estos procesos se encuentran los relacionados con el mantenimiento de equipos e instalaciones, pues la informatización de estos proporciona un adecuado control en toda la ejecución del mantenimiento y un mejor desempeño por parte de las empresas previniendo futuros problemas funcionales.

Anteriormente las empresas solo aplicaban el mantenimiento cuando estaban en presencia de fallas que no permitían el funcionamiento continuo de sus equipos, esto traía como consecuencia pérdida de recursos y gastos muy elevados al tener que esperar por la presencia de la avería para actuar, o sea solo aplicaban el mantenimiento correctivo. Por este motivo las empresas que manejan flotas de vehículos han optado por aplicar otros tipos de mantenimiento como el preventivo planificado, ya que este no tiene la obligación de esperar por que ocurran las averías para intervenir en la reparación, lo cual previene futuras fallas que pueden atrasar el funcionamiento de la empresa.

Los procesos relacionados con la ejecución del mantenimiento de los vehículos en las empresas que gestionan flotas de vehículos se enfocan en la aplicación del mantenimiento preventivo planificado y correctivo, partiendo de la generación de una Orden de trabajo[1] y registrando los recursos tanto humanos como materiales utilizados en la realización de estos trabajos ya sean preventivos o correctivos.

En el área de Transporte del CPNB el proceso Ejecución del mantenimiento se realiza de forma manual, lo cual trae como consecuencia la pérdida de información al no tener un control eficiente sobre las Órdenes de trabajo generadas para las unidades policiales. Además no se lleva un registro de los repuestos, herramientas y recursos humanos utilizados para la ejecución de los trabajos de mantenimiento, lo cual constituye un riesgo para el correcto desempeño del área de Transporte de la policía pues no existe un control sobre los recursos utilizados en los trabajos preventivos planificados o correctivos realizados a las unidades policiales en un período determinado, llevando esto a posibles delitos y una mala planificación presupuestaria por parte de la institución. Además no existe una

[1] Una Orden de trabajo es un documento que recoge los trabajos de mantenimiento que se le van a realizar a una o varias unidades policiales, los materiales que se consumieron en cada uno de los trabajos realizados y los recursos humanos.

integración entre este proceso y el resto de los procesos que se realizan en esta área, ejecutándose los mismos de forma independiente, por ejemplo antes de generarse una Orden de trabajo debe realizarse una inspección técnica, esto actualmente no se realiza ya que no se encuentran integrados los procesos Ejecución del mantenimiento y Administración de inspecciones técnicas, lo cual provoca que no se puedan detectar posibles averías de las unidades policiales durante la realización de las inspecciones técnicas y corregirlas a tiempo.

La situación problemática planteada permitió definir como **Problema a resolver**: ¿Cuál es la especificación de los artefactos necesarios para la informatización de los requisitos identificados en el proceso Ejecución del mantenimiento 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 del proceso Ejecución del mantenimiento del Sistema de Gestión de Mantenimiento Vehicular del CPNB.

Se plantean como **Objetivos específicos:**

1- Elaborar el marco teórico de la investigación.
2- Realizar el diseño de la propuesta de solución.
3- Realizar la implementación de la propuesta de solución.
4- Validar los resultados obtenidos.

Se define como **Objeto de Estudio**: Procesos de desarrollo de software de gestión.

El **Campo de Acción** se enmarca en: Diseño e implementación del proceso Ejecución del mantenimiento del Sistema de Gestión de Mantenimiento Vehicular del CPNB.

Para desarrollar el presente trabajo se llevaron a cabo las siguientes **tareas**:

- Realizar análisis crítico de los sistemas de flota vehicular existentes en Cuba, Venezuela y el mundo partiendo de la forma en la que gestionan el proceso Ejecución del mantenimiento.
- Análisis de los artefactos entregados por el equipo de análisis.
- Realización de las clases del diseño del proceso Ejecución del mantenimiento de forma tal que se integre con el resto de los procesos implementados del área de Transporte del CPNB.
- Implementación de las funcionalidades identificadas.
- Diseño del modelo de datos teniendo en cuenta el análisis realizado.

- Validación del diseño propuesto y la implementación realizada para el proceso ejecución del mantenimiento.

Se define como **idea a defender**:

La realización del diseño e implementación del Proceso Ejecución del mantenimiento vehicular permitirá obtener los artefactos necesarios para la informatización de los requisitos identificados para este proceso.

Los métodos de la investigación científica que se utilizaron para el desarrollo del presente trabajo son:

- **Analítico sintético:** Mediante el uso de este método se realizó un análisis del proceso Ejecución del mantenimiento del CPNB de Venezuela. De esta forma se logró una adecuada comprensión del tema a desarrollar, haciendo fácil su estudio y logrando la identificación de los elementos más importantes de estos procesos.
- **Histórico lógico:** El uso de este método permitió conocer y comprender el funcionamiento de algunos sistemas de mantenimiento de flotas vehiculares existentes en Cuba, Venezuela y el mundo. Además se realiza un estudio del proceso de Ejecución del mantenimiento del área de Transporte desde que surge el CPNB.

El presente trabajo se encuentra estructurado de la siguiente forma: una introducción, tres capítulos, glosario, anexos, y la bibliografía consultada para el completamiento del marco teórico de la investigación.

Capítulo 1. Fundamentación teórica: En este capítulo se realiza una profunda investigación sobre el proceso Ejecución del mantenimiento abordando su importancia dentro del CPNB de Venezuela. También se realiza un análisis crítico de varios sistemas que gestionen el mantenimiento de flotas existentes en Venezuela, Cuba y el mundo. Se especifica sobre la utilización de las herramientas, tecnologías y lenguajes de programación que se van a utilizar para la implementación del proceso Ejecución del mantenimiento.

Capítulo 2. Diseño e implementación: En este capítulo se realiza un análisis de los artefactos generados durante los flujos de trabajo Modelado de negocios y Requerimientos para el proceso Ejecución del mantenimiento. Se especifican los patrones utilizados para la realización de las clases del diseño de la propuesta de solución. Se realiza la descripción de los estándares de codificación y de implementación por funcionalidades, así como de la publicación de los servicios necesarios para la integración de este proceso con el resto de los procesos del área de Transporte del CPNB.

Capítulo 3. Validación y Pruebas: En este capítulo se describe cómo se realizó la validación de la solución propuesta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El presente capítulo trata temas relacionados con el proceso Ejecución del mantenimiento del área de Trasporte del CPNB. Se expone un análisis crítico de sistemas de gestión de flotas de vehículos existentes en Cuba, Venezuela y el mundo, que gestionen el mantenimiento de flotas de vehículos. Se describen además las características de lenguajes de programación, herramientas, tecnologías y arquitectura de software que se van a emplear para el desarrollo del sistema.

1.2 Proceso Ejecución del mantenimiento.

El proceso Ejecución del mantenimiento del área de Transporte del CPNB comienza con la generación de Órdenes de trabajo producto de un mantenimiento preventivo planificado o correctivo. Antes de generarse una Orden de trabajo debe realizarse una inspección técnica con el objetivo de determinar trabajos de mantenimiento preventivo que pueden ser detectados, esta actividad actualmente no se realiza debido a que no existe una integración entre el proceso Ejecución del mantenimiento y el proceso Administración de inspecciones técnicas. En las Órdenes de trabajo se registran los recursos utilizados en la realización de un mantenimiento determinado, dígase repuestos, herramientas y recursos humanos. En caso de que las Órdenes de trabajo se generen para mantenimientos correctivos se registran las causas y tipos de fallas asociados, en aras de realizar análisis que permitan tomar decisiones respecto al mantenimiento preventivo de las unidades policiales.

1.3 Sistemas de gestión del mantenimiento vehicular.

1.3.1 Sistemas de gestión de mantenimiento vehicular existentes en el mundo.

En la actualidad existen algunos sistemas que gestionan procesos relacionados con la ejecución del mantenimiento vehicular con el objetivo de garantizar la durabilidad y continua función de los vehículos. Dentro de estos sistemas se encuentra **SoftFlot**, el cual fue creado por la empresa mexicana InteraSystem. Actualmente cuenta con una red de distribuidores en diferentes países de Latinoamérica como Perú, Chile, Colombia, República Dominicana y Venezuela (InterSystem, 2011).

Este sistema fue diseñado para administrar cualquier tipo de flota de vehículos. Trabaja sobre plataforma Windows y comprende las siguientes áreas de procesos (InteraSystem, 2011):

1. **Vehículos** permite guardar toda la información detallada de la flota de vehículos como: Placa, Número de Motor, Número de Serie, Llantas, Tipo de Combustible, Unidades de medida para combustible, Tipo de medidor, Seguros asociados y capacidades de carga.
2. **Empleados** permite registrar la información de las personas de la empresa (mecánicos, choferes, ayudantes), o sea todos los involucrados en la administración de la flota de vehículos.
3. **Empresas** registra la información de las Empresas externas con las cuales existe una relación comercial o convenio para la compra de refacciones o para la realización de algún servicio a los vehículos como: afinaciones, engrasados, alineaciones y llantas.
4. **Mantenimiento** es el encargado de llevar una correcta planificación de las tareas de mantenimiento, preventivas o correctivas.
5. **Logística** es donde los usuarios registran todo lo relacionado con la transportación de cualquier tipo de producto de un lugar a otro, cobrando un flete al cliente por este servicio.
6. **Llantas** registra todo lo relacionado con los neumáticos y kilometraje de los vehículos.
7. **Administración** procesa los módulos que tienen que ver con costos, presupuestos, o sea todo lo relacionado con inversión económica.
8. **Operación** registra los procesos relacionados con la asignación de conductores a los vehículos y el control del grupo vehicular (InterSystem, 2011).

Este sistema ofrece la ventaja de almacenar datos que permiten realizar el mantenimiento preventivo y correctivo de las flotas de vehículos de manera eficaz y confiable, garantizando la optimización de los costos de mantenimiento.

Otro sistema que gestiona el mantenimiento de flotas de vehículos es **ANAGO**. Es un sistema de gestión eficiente de mantenimiento preventivo y correctivo para asegurar el nivel de servicio al cliente y para que los costos de operación sean lo más bajo posible, compuesto de los siguientes módulos (Systeam - Logística Empresaria, 2010):

1. El módulo de **reparaciones** permite consultar el historial de todas las reparaciones efectuadas en la flota y controlar detalladamente los costos ocasionados en concepto de materiales, repuestos y mano de obra así como las facturas de los talleres.
2. El módulo **consumo de combustible** permite un control detallado de abastecimientos lo que proporciona valiosa información como: consumo promedio por cada vehículo, kilómetros recorridos, gastos efectuados con las tarjetas de un proveedor, etc.
3. El módulo de **mantenimiento de neumáticos** permite conocer detalladamente las características de cada uno de los neumáticos; como el costo por kilómetro, la distancia recorrida, modelo, fabricante, proveedor y otras informaciones.
4. El imprescindible control de vencimientos de mantenimiento preventivo y revisiones técnicas de los vehículos se realiza con suma facilidad en el módulo de **vencimientos** a través de un sistema gráfico de avisos y alarmas (Systeam - Logística Empresaria, 2010).

1.3.2 Sistemas de gestión de mantenimiento vehicular existentes en Venezuela.

Otro sistema distribuido en Venezuela y que gestiona el mantenimiento de flotas de vehículos es **Transportex**. Este sistema es una aplicación informática completa y sencilla para la gestión de empresas de transporte de mercancía vía terrestre. Es una aplicación de escritorio que necesita sistema operativo Windows (Windows 7, Vista, XP, 2003 o 2000). Su arquitectura de trabajo es cliente/servidor que trabaja con Microsoft SQL Server 2005 Edición Express (Grupo Máximo, 2011).

Está compuesta por tres módulos principales:

1. **Gestión de Neumáticos:** Este componente permite conocer características de los neumáticos, distancia que recorren, espesor, ubicación, y desgaste.
2. **Actividades de Mantenimiento:** Este componente permite la programación de las tareas de mantenimiento preventivo o correctivo para los vehículos.
3. **Control de Viajes:** Este módulo establece el registro de viajes de despachos de mercancía realizados por los vehículos, controlando los adelantos y montos a pagar a cada conductor como el salario y días no laborables.

Este sistema permite el manejo de datos precisos, mejor control de los gastos**,** mayor control en la ejecución de actividades de mantenimiento, mejor análisis técnico y económico y un control detallado de las flotas de vehículos (Grupo Máximo, 2011).

Los sistemas mencionados anteriormente no pueden ser utilizados en el área de Transporte del CPNB debido a que presentan los siguientes inconvenientes:

- No realizan inspecciones técnicas antes de generar una Orden de trabajo, proceso este necesario en el área de Transporte del CPNB.
- No registran en las Órdenes de trabajo los recursos (repuestos, herramientas) utilizados en la ejecución del mantenimiento sino en otro documento, afectando esto el control de estos recursos por parte de la institución al no tener toda la información relacionada con la ejecución del mantenimiento unificada en un documento que sea manejado directamente por los implicados en este proceso.
- No permiten obtener información sobre :
    1. Mantenimientos realizados por período.
    2. Mantenimientos realizados por grupo de unidades.
    3. Historial de mantenimiento de una unidad.
    4. Tipos de mantenimiento por tipo de unidad.

    Información esta necesaria para el área de Transporte del CPNB.
- Son aplicaciones de escritorio por cuanto no pueden ser utilizadas en el CPNB, pues este centro tiene como política que todos los sistemas de gestión que utilicen deben ser aplicaciones web que puedan ser instaladas en un servidor central, al cual todas las Dependencias policiales puedan acceder usando un navegador web.
- No permiten la gestión de inspecciones técnicas y accidentes, procesos estos claves que se realizan en el área de Transporte del CPNB. Son software propietario, o sea que su código fuente no puede ser modificado para agregar las nuevas funcionalidades que necesita el área de Transporte de la CPNB para gestionar sus procesos, además de que Venezuela debe pagar

precios elevados por su uso. También están desarrollados utilizando tecnologías de software propietarios y el CPNB tiene como política el uso de tecnologías de software libre.

1.3.3 Sistemas de gestión de mantenimiento vehicular existentes en Cuba.

En Cuba existen algunos sistemas que gestionan los procesos relacionados con la Ejecución del mantenimiento, entre estos se encuentra **Offimant,** desarrollado por la empresa DESOFT S.A para la gestión integral del mantenimiento preventivo, correctivo y predictivo de equipos e inmuebles. Cuenta con tres módulos que brindan facilidades para la planificación, gestión y control del mantenimiento, permitiendo la inspección, revisión y monitoreo del estado técnico de los equipos e inmuebles registrando la información obtenida en una carpeta técnica. Permite la generación y control de Órdenes de trabajo, con un seguimiento de los gastos de materiales, piezas y recursos humanos (Empresa Nacional de Software, División Matanzas, 2011).

1. **Módulo de Administración:** Registra la licencia de usuarios que permitirá posteriormente trabajar en el Módulo de Mantenimiento, registra la Base de Datos y logra la seguridad de toda la información.
2. **Módulo de Mantenimiento:** Permite definir y mantener toda la información relacionada con los activos, establecer y planificar tareas, generar solicitudes y Órdenes de trabajo.
3. **Módulo de Solicitud de Trabajo:** Emite solicitudes desde diferentes estaciones de trabajo al departamento de mantenimiento.

Otro sistema de origen cubano para la gestión del mantenimiento es **SgestMan**, desarrollado con GeneXus V.8.0, generado para Visual Basic y tecnología .NET por la empresa GAMMA S.A. Su arquitectura de trabajo es cliente / servidor sobre plataforma Oracle o SQL Server (Citmatel, 2006).

El sistema cuenta con 10 módulos de trabajo, los cuales procesan la información especializada de cada área de la empresa relacionada con la gestión del mantenimiento.

1. **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.

2. **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 en materia de mantenimiento que precisa una empresa para contar con un Patrimonio con un alto nivel de disponibilidad y utilización técnica.
3. **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.
4. **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.
5. **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.
6. **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.
7. **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.
8. **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.
9. **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.

10. **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 (Citmatel, 2006).

Estos sistemas nacionales presentan una serie de inconvenientes que imposibilitan que puedan ser utilizados para la gestión de los procesos que se desarrollan en el área de Transporte del CPNB, dentro de los que se encuentra:

- No permiten la gestión del mantenimiento de vehículos.
- No gestionan inspecciones técnicas y accidentes, procesos estos que se realizan en el área de Transporte del CPNB.
- No gestionan el mantenimiento correctivo a partir de Memorándum de Solicitud de Mantenimiento, pues este es un documento oficial que sólo se maneja en el área de Transporte del CPNB.
- Son aplicaciones de escritorio que requieren ser instaladas en cada puesto de trabajo para su correcto funcionamiento, lo cual viola la política impuesta por este centro al no contar con un servidor central de aplicaciones el cual permita acceder al sistema a través de navegación web.
- No cumplen con la política del uso de Software libre que existe en el CPNB.

1.4 Modelo de desarrollo orientado a componentes.

Para lograr la realización del Sistema de Gestión de Mantenimiento Vehicular de forma que se garantice una integración entre sus componentes se hace necesaria la utilización de un modelo estándar, por lo cual se emplea el modelo de desarrollo orientado a componentes. Este modelo muestra una clara definición de las responsabilidades de los roles involucrados en el desarrollo. Permite el desarrollo de software basado en componentes, el cual se ha transformado en uno de los métodos más seguros tanto para la construcción de grandes sistemas como para pequeñas aplicaciones, ya que el mismo permite el incremento de la calidad del software producido, el aumento de la productividad de los grupos de desarrollo y la reducción del riesgo global del proyecto (Desarrollo de Software Basado en Componentes, 2003), también permite la reutilización de software que posibilita realizar software complejo en cortos períodos de tiempo, simplifica las pruebas de calidad del software mejorando la calidad del software y logra ciclos de desarrollo más cortos (Microsoft, 2011). Al usar este modelo 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 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. Por su gran importancia y las ventajas que ofrece para el desarrollo de software este modelo de desarrollo es el utilizado por el CEIGE[2] para el desarrollo de los sistemas que se realizan en cada uno de sus departamentos, específicamente en el departamento SOLEM[3].

1.5 Arquitectura de software basada en Componentes.

El término 'arquitectura' es heredado de otras disciplinas de la ciencia. Se entiende por arquitectura a un conjunto de piezas de distintos tipos, que encajan entre sí y cumplen una función determinada. La arquitectura presenta además el impacto del cambio de una de las piezas. Dentro del paradigma de componentes, las piezas son los componentes. La arquitectura de componentes dirá con qué tipos de componentes y en qué relación de dependencia se encuentran (Andrés Vignaga, 2003).

Para el desarrollo del Sistema de Gestión de Mantenimiento Vehicular se adopta esta arquitectura ya que es utilizada en el CEIGE para el desarrollo de sus software ya que cubre aspectos únicamente lógicos y es totalmente independiente de la tecnología con la cual se implementarán los componentes y sobre la cual se hará el despliegue del sistema. Esta vista lógica permite medir el nivel de acoplamiento del sistema y razonar sobre los efectos de modificar o reemplazar un componente (Andrés Vignaga, 2003).

1.6 Tecnologías.

1.6.1 AJAX.

El término AJAX es un acrónimo de Asynchronous JavaScript + XML[4], que se traduce como "JavaScript asíncrono + XML". En realidad AJAX no es una tecnología, se trata realmente de muchas tecnologías, cada una floreciendo por su propio mérito, uniéndose en poderosas nuevas formas (Garrett, 2005).

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

[3]Departamento de Soluciones Empresariales.

[4]Lenguaje de marcas extensible (XML) es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

AJAX incorpora:

- Presentación basada en estándares usando **XHTML**[5]y **CSS**[6],
- Exhibición e interacción dinámicas usando el **DOM**[7],
- Intercambio y manipulación de datos usando **XML**, **XSLT**[8] y **JSON**[9],
- Recuperación de datos asincrónica usando **XMLHttpRequest**[10],
- JavaScript unificando las tecnologías antes mencionadas.

[5] Acrónimo en inglés de extensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web.

[6] Hojas de estilo en cascada viene del inglés Cascading Style Sheets, del que toma sus siglas. CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en **HTML** o **XML** (y por extensión en **XHTML**).

[7] (Modelo de Objetos del Documento o Modelo en Objetos para la representación de Documentos), es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML

[8] XSLT o Transformaciones XSL es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML.

[9]Acrónimo de JavaScript Object Notation, que se traduce como notación de objetos de JavaScript. Es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

[10]XMLHttpRequestes una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores web.

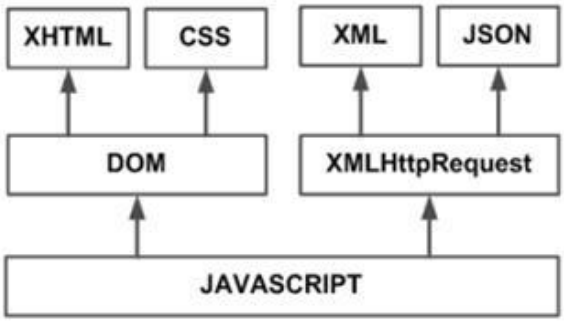


Imagen 1. Tecnologías agrupadas bajo el concepto AJAX.

En las aplicaciones web tradicionales, las acciones del usuario en la página (dar clic en un botón o seleccionar un valor de una lista) desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario. En el siguiente esquema, la parte izquierda muestra el modelo tradicional de las aplicaciones web. La parte derecha muestra el modelo propuesto por AJAX (Pérez, 2009):

Imagen 2. Comparación gráfica del modelo tradicional de aplicación web y del nuevo modelo propuesto por AJAX.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

1.6.2 Zend Framework.

Zend Framework es un marco de trabajo para el desarrollo de aplicaciones web y servicios web con PHP. Entre sus principales características se encuentran que implementa el patrón MVC (Modelo Vista Controlador) y cuenta con módulos para manejar archivos PDF[11], canales RSS[12], Servicios web. La estructura de los componentes de Zend Framework permite a cada componente estar construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado, aunque se pueden utilizar de forma individual, los componentes de la librería estándar de Zend Framework conforman un potente y extensible marco de trabajo de aplicaciones web al combinarse (Morera, 2011). Ofrece una abstracción de base de datos fácil de usar, lo que simplifica las consultas a su base de datos, sin la necesidad de realizar consultas SQL, además de contar con amplia documentación (Zend Technologies Ltd, 2006 - 2011). Está definido en el marco de trabajo que se va a utilizar.

Zend Framework brinda ventajas como:

- Facilita el mantenimiento de las aplicaciones.
- Es posible utilizarlo en modo "desacoplado", es decir, aquellas clases o componentes que sean necesarios en cada proyecto, sin arrastrar todo el marco de trabajo para cualquier pequeña necesidad.
- Tiene el respaldo de la compañía ZEND, creadora de PHP, lo que asegura su continuidad futura, tanto como la del propio lenguaje PHP (Business Development Software S.L, 2011).

[11] Formato de documento portátil, de sus siglas del inglés portable document format.

[12] **RSS** son las siglas de Rich Site Summary (Resumen enriquecido del sitio), un formato XML para sindicar o compartir contenido en la web. Se utiliza para difundir información actualizada frecuentemente a usuarios que se han suscrito a la fuente de contenidos.

1.6.3 ExtJS.

Es una plataforma de JavaScript que se usa para el desarrollo de aplicaciones web usando tecnologías como DHTML[13] y DOM siendo compatible con la mayoría de los navegadores actuales, por ejemplo Firefox, Internet Explorer y otros. Esta plataforma es utilizada ya que se encuentra integrada al marco de trabajo que se va a utilizar.

Entre sus ventajas se encuentran:

- ExtJS ayuda a construir aplicaciones sostenibles más rápido.
- Integra la tecnología Ajax de una forma transparente al usuario.
- Dispone de una alta documentación que le permite a los desarrolladores nutrirse de las librerías que posee el marco de trabajo, posee una comunidad de desarrollo extensa, en el foro oficial se obtienen rápidas respuestas de los desarrolladores de software o de usuarios expertos.
- Dispone de una gran cantidad de componentes entre los cuales se pueden mencionar a: Combos, Editor HTML, Árbol de Datos, Barras de Herramientas, que al ser extensibles permiten crear nuevos componentes adaptándose a las necesidades del desarrollador (Sencha Inc, 2011).

1.6.4 Doctrine.

Doctrine es un sistema de mapeado de objetos relacionales (ORM) para PHP, que permite obtener una capa de abstracción independiente, a la base de datos que se utilice. Entre sus principales características se encuentra la posibilidad de escribir de manera opcional las consultas a la base de datos. Lo que proporciona a los desarrolladores una alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código de forma innecesaria. 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 (Domínguez Medina, 2010). Se utiliza este sistema para el acceso a las tablas de la base de datos ya que se encuentra definido en el marco de trabajo que se va a utilizar.

1.6.5 Sauxe.

[13] HTML Dinámico o DHTML (del inglés Dynamic HTML) designa el conjunto de técnicas que permiten crear sitios web interactivos utilizando una combinación de lenguaje HTML estático.

El marco de trabajo Sauxe es creado para el desarrollo de aplicaciones web orientadas a componentes, permite y facilita el desarrollo de sistemas de gestión, ya que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor agilidad en el proceso de desarrollo. Además incluye el marco de trabajo Zend, utiliza en la capa de acceso a datos el Lenguaje de Consulta de Datos (DQL) que implementa Doctrine y utiliza ExtJS en la capa de presentación, por la gran cantidad de componentes que se pueden reutilizar y para mostrarle al usuario una interfaz más amigable (Alfonso Alfonso, 2011). Este marco de trabajo es usado para el desarrollo del Sistema de Gestión de Mantenimiento Vehicular, ya que además de las características mencionadas, fue creado por el CEIGE y definido por el mismo para el desarrollo de los sistemas de gestión.

1.7 Lenguajes.

1.7.1 Lenguaje de desarrollo JavaScript.

JavaScript es un lenguaje de programación "ligero" y orientado a objetos. El intérprete del lenguaje se encuentra en los navegadores, para interpretar los códigos (scripts) escritos en las páginas. Estos códigos pueden hacer que exista interactividad dinámica entre el usuario y la página, controlar el navegador o crear páginas completas HTML sin que se tenga que ir nuevamente al servidor (Grupo Anaya S.A, 2002).

JavaScript brinda facilidades a los usuarios como son:

- Permite la verificación de los datos introducidos por el usuario antes de enviar el formulario al servidor.
- Permite el manejo de pequeñas cantidades de información al igual que en una base de datos.
- Permite el manejo de applets y plug-ins dentro de múltiples marcos de HTML.
- Permite el preprocesado de información antes de enviarla al servidor (Grupo Anaya S.A, 2002).

Este lenguaje por sus características y posibilidades permite un desarrollo completo de cualquier aplicación web por compleja que sea, facilita el manejo de información en el navegador sin necesidad de realizar llamadas redundantes a una base de datos, por lo cual se va a utilizar para el desarrollo del Sistema de Gestión de Mantenimiento Vehicular posibilitando una mayor rapidez del mismo.

1.7.2 Lenguaje de desarrollo PHP.

La necesidad que tienen los sitios web de ser robustos, dinámicos y flexibles ha hecho que PHP se convirtiera en uno de los lenguajes de desarrollo más difundidos (Monso, 2005). Este es un lenguaje de programación interpretado de alto nivel, embebido en páginas HTML y ejecutado en el servidor, es muy similar en su sintaxis al Lenguaje C, Java o Perl, con algunas diferencias, no compila como el lenguaje C, es un intérprete, por lo tanto cada vez que debe ejecutar un programa, lo interpreta, verificando toda su sintaxis.Es un lenguaje de programación del lado del servidor sumamente funcional y gratuito con una gran librería de funciones y mucha documentación, que se ejecuta en el servidor web, justo antes de que se envíe la página al cliente, que verá solamente páginas con el código HTML. PHP brinda soporte a una gran cantidad de base de datos como: mSQL, MySQL, Oracle, Informix y PostgreSQL (Maestrosdelweb, 1997). También ofrece la integración con varias librerías externas, que permiten que el desarrollador haga casi cualquier cosa, desde generar documentos en PDF hasta analizar código XML. Todas estas características posibilitan el desarrollo del Sistema de Gestión de Mantenimiento Vehicular con este lenguaje, permitiendo al sistema ser implantado en cualquier lugar sin tener en cuenta ningún sistema operativo en específico.

1.8 Herramientas.

1.8.1 Eclipse.

La plataforma Eclipse consiste en un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés), compuesto por un conjunto de herramientas útiles para un desarrollador de software. Como elementos básicos, un IDE cuenta con en un editor de código, un compilador/intérprete y un depurador. Eclipse sirve como IDE Java y cuenta con numerosas herramientas de desarrollo de software. También da soporte a otros lenguajes de programación, como son C/C++, Cobol, Fortran, PHP o Python. A la plataforma base de Eclipse se le pueden añadir extensiones (plug-ins) para extender la funcionalidad. Fue creado inicialmente por IBM 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 una arquitectura basada en plug-ins y posibilita integrar diversos lenguajes de programación. Por estas facilidades que brinda a la hora de desarrollar, además de permitir un buen control de versiones, posibilita el desarrollo del Sistema de Gestión de Mantenimiento Vehicular (Dirección Provincial Cultura, 2007-2010).

1.8.2 Visual Paradigm.

Visual Paradigm para UML[14] es una herramienta CASE[15] 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 de desarrollo. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación (Visual Paradigm, 1999-2011).

Principales características de Visual Paradigm:

- Soporte de UML versión 2.1
- Es una herramienta de uso profesional.
- Permite redactar especificaciones de casos de uso del sistema.
- Posibilita la Sincronización entre diagramas de entidad-relación y diagramas de clases.
- Generación de código e Ingeniería inversa.
- Interoperabilidad con otras aplicaciones (Visual Paradigm, 1999-2011).

Por estas características es adoptada como herramienta CASE en el desarrollo del Sistema de Gestión de Mantenimiento Vehicular, ya que su interfaz de usuario intuitiva y la capacidad de modelado le permiten desarrollar la calidad de los diseños de software en forma eficiente y profesional.

1.8.3 Apache 2.0.

Apache es la plataforma de servidores web de código fuente abierto más poderosa del mundo.Su alta configuración, robustez y estabilidad hacen que más del 64% de toda web reiteren su confianza en este programa (netcraft, 2003-2012). 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 permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute una determinada secuencia de comandos cuando ocurra un error en concreto. Tiene una alta configuración en la creación y gestión de registros. Permite monitorizar servidores web y

[14] Lenguaje Unificado de Modelado.

[15] Ingeniería de Software Asistida por Computadora, de sus siglas en inglés Computer Aided Software Engineering.

adaptar archivos de registro para análisis. Apache está preparado para todas las novedades que puedan surgir en el protocolo HTTP (Anaya Multimedia, 2003). Al ser este servidor web uno de los más potentes en el mundo para aplicaciones web y brindar las características que lo hacen una herramienta confiable, será usado en el desarrollo del Sistema de Gestión de Mantenimiento Vehicular.

1.8.4 PostgreSQL.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley. PostgreSQL es un sistema objeto-relacional, ya que incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar de esto, PostgreSQL no es un sistema de gestión de bases de datos puramente orientado a objetos. PostgreSQL está considerado como una de las bases de datos de código abierto más avanzadas del mundo. Proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle (Domínguez Medina, 2010).

PostgreSQL es el gestor de base de datos que se utiliza para el desarrollo del Sistema de Gestión de Mantenimiento Vehicular ya que además de las características mencionadas anteriormente es un servidor de base de datos relacional libre, que se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y uniones de gran tamaño. Fue diseñado para ambientes de alto volumen y tiene mejor soporte para vistas, procedimientos almacenados en el servidor, transacciones y almacenamiento de objetos de gran tamaño (PostgreSQL, 1996-2011).

1.8.5 Mozilla Firefox.

Mozilla Firefox es un navegador multiplataforma disponible para varios sistemas operativos que será usado para el acceso al Sistema de Gestión de Mantenimiento Vehicular mediante navegación web. Algunas de las ventajas que ofrece y que permiten su uso son (Mozillia firefox, 1998-2011):

- Es de software libre, ágil, práctico y en renovación constante.
- Permite una fácil navegación.
- Se puede modificar según las necesidades del usuario.
- Presenta navegación por pestañas.
- Presenta gran seguridad.

- Está diseñado para realizar un bajo consumo de recursos.

1.9 Conclusiones parciales.

Se realiza un análisis de los sistemas internacionales y nacionales que gestionan el mantenimiento vehicular, mostrando sus características, principales funcionalidades, ventajas y desventajas que impiden a estos sistemas ser una solución a implantar en el CPNB. Se aborda además la arquitectura de software, modelo de desarrollo, tecnologías y herramientas que serán utilizadas para dar solución a la problemática planteada, describiendo los aspectos de cada una de ellas que permiten ser utilizadas para el desarrollo de la solución propuesta.

CAPÍTULO 2: DISEÑO E IMPLEMENTACIÓN.

2.1 Introducción.

En el presente capítulo se realiza el diseño e implementación del componente Ejecución del Sistema de Gestión de Mantenimiento Vehicular, presentando los diagramas de clases del diseño realizados utilizando los estereotipos web, diagramas de componentes y modelo de datos que facilitan la implementación de este componente. Además se realizará un análisis de los artefactos entregados por los analistas. También se tratarán los mecanismos de diseño utilizados para la realización de los diagramas de clases del diseño así como los beneficios que trae su aplicación.

2.2 Análisis de los artefactos entregados por los analistas.

Para realizar el diseño de la propuesta de solución planteada por los analistas del Sistema de Gestión de Mantenimiento Vehicular se analizaron los artefactos entregados por los mismos:

- Modelo Conceptual, ya que en este artefacto se identificaron los conceptos que se manejaban en el negocio y era necesario dominar, debido a que los analistas en el artefacto Especificación de requisitos funcionales hacían referencia a los mismos.
- Descripción del proceso Ejecución del mantenimiento, en aras de comprender cómo funciona este proceso dentro del área de Transporte del CPNB para entender mejor qué es lo que se quería informatizar.
- Especificación de requisitos funcionales, se analizó porque a partir de la información que contiene este artefacto, dígase descripción de los requisitos funcionales y prototipos de interfaz de usuario se va a realizar el diseño de la propuesta de solución planteada por los analistas.

El análisis realizado permitió comprobar que no existía inconsistencia o ambigüedad en las especificaciones de los requisitos funcionales identificados, todo esto debido a que fueron revisados anteriormente, aprobados y liberados por el cliente y por Calisoft[16], por cuanto no se propusieron cambios y se da paso al desarrollo del diseño y la implementación de la propuesta de solución.

2.3 Diseño.

[16] Empresa que se dedica a evaluar la calidad de los software que se realizan en la UCI.

El diseño es una representación significativa de ingeniería de algo que se va a construir, es un proceso iterativo mediante el cual los requisitos se traducen en un plano para construir el software, donde se puede evaluar su calidad antes de que comience la implementación (Pressman, 2005).

El diseño posee los siguientes propósitos:

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y de interfaz de usuario.
- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo (Pressman, 2005).

2.3.1 Mecanismos de diseño.

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

Para el diseño del componente Ejecución se definieron los siguientes mecanismos de diseño:

Mecanismo de diseño para las páginas clientes.

Imagen 3. Mecanismos de diseño para clases clientes.

ext-all.js: Es la encargada de la creación de los componentes visuales de la vista. Está incluida dentro de las clases que trae ExtJS.

ext-base.js: Contiene lo que necesita ExtJS para su correcto funcionamiento.

ucid-all.js: Encargada de mostrar la interfaz estándar.

Mecanismo de diseño para los nombres de clases.

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

En el caso de la agrupación de requisitos conformadas por adicionar, modificar, eliminar, listar y buscar, se divide en dos clases, una que agrupa los requisitos adicionar, modificar, y eliminara 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 nombra LB_Nombre_de_la_clase.

Mecanismo de diseño para las clases controladoras.

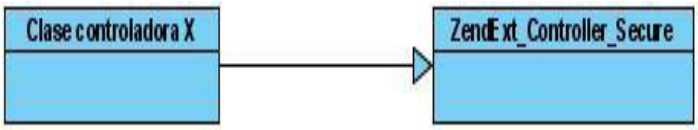


Imagen 4. Mecanismo de diseño para las clases controladoras**.**

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

Mecanismo de diseño para las clases modelos.

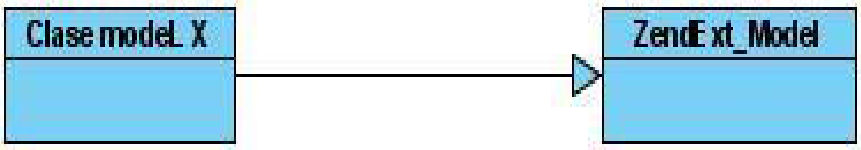


Imagen 5. Mecanismo de diseño para las clases del modelo.

Las clases del modelo definidas en el diseño heredan de la clase ZendExt_Model, esta incluye las principales funciones para el manejo de los datos.

2.3.2 Diagrama de clases del diseño.

Los diagramas de clases del diseño describen gráficamente las especificaciones de las clases de software, las interfaces y sus relaciones. Se utilizan para modelar la vista de diseño estática de un

sistema. Son importantes para visualizar, especificar, documentar modelos estructurales y construir sistemas ejecutables aplicando ingeniería directa e inversa.

Diagrama de clases de diseño: Generar Orden de trabajo.

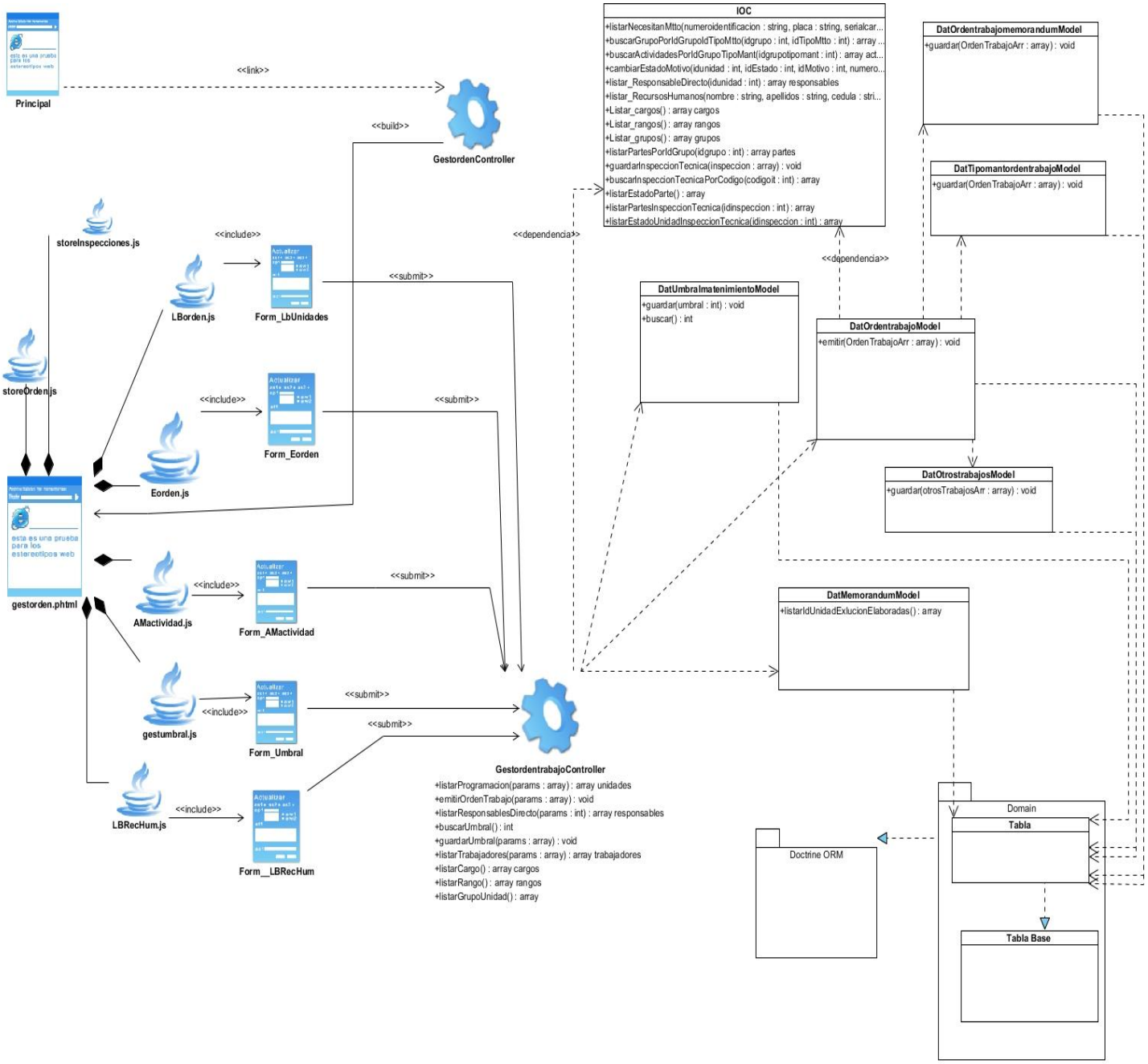


Imagen 6. Diagrama de clases del diseño: Generar Orden de Trabajo.

Diagrama de clases de diseño: Gestionar Orden de trabajo.

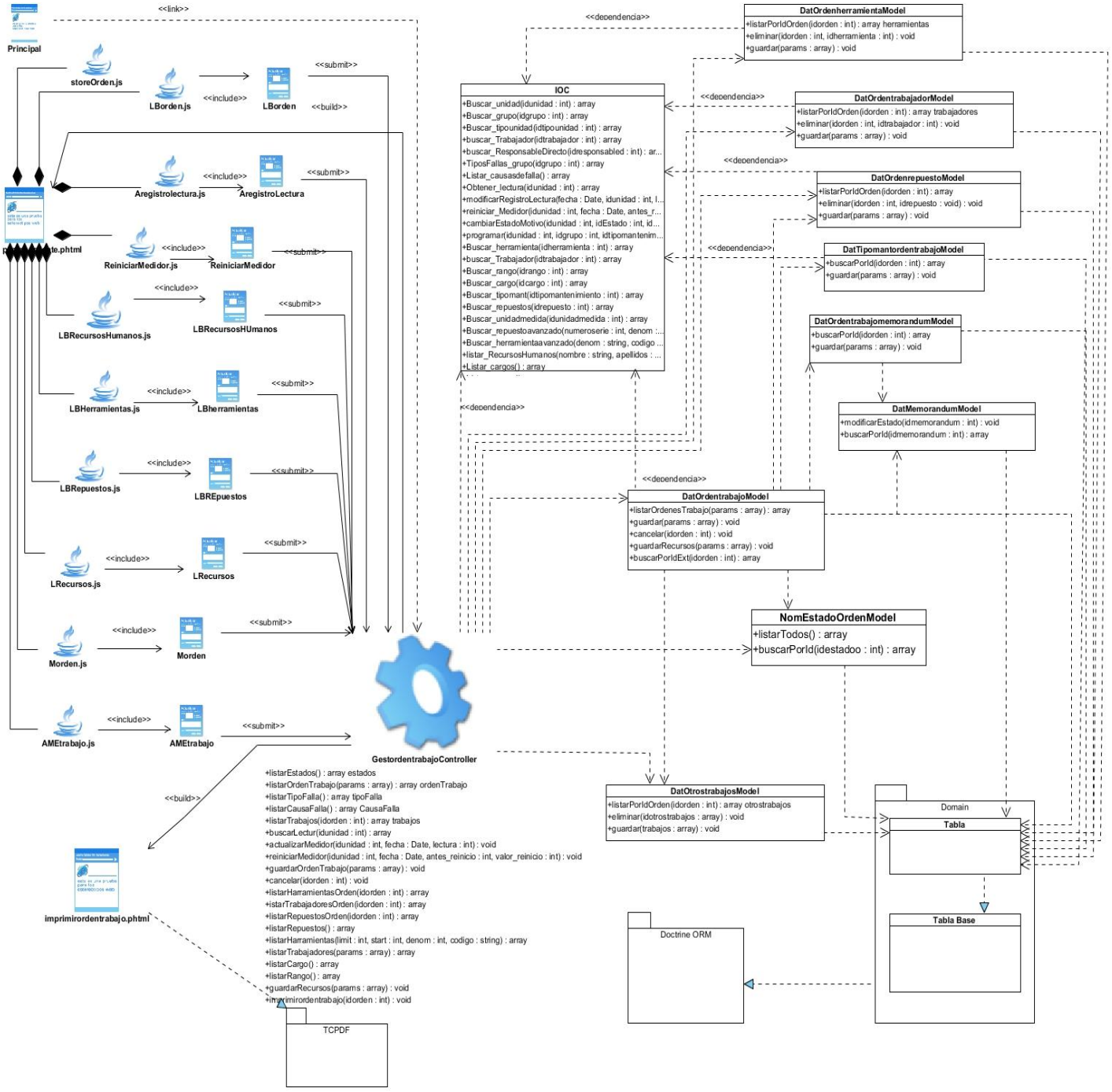


Imagen 7. Diagrama de clases de diseño: Gestionar Orden de Trabajo.

Diagrama de clases de diseño: Gestionar Memorándum.

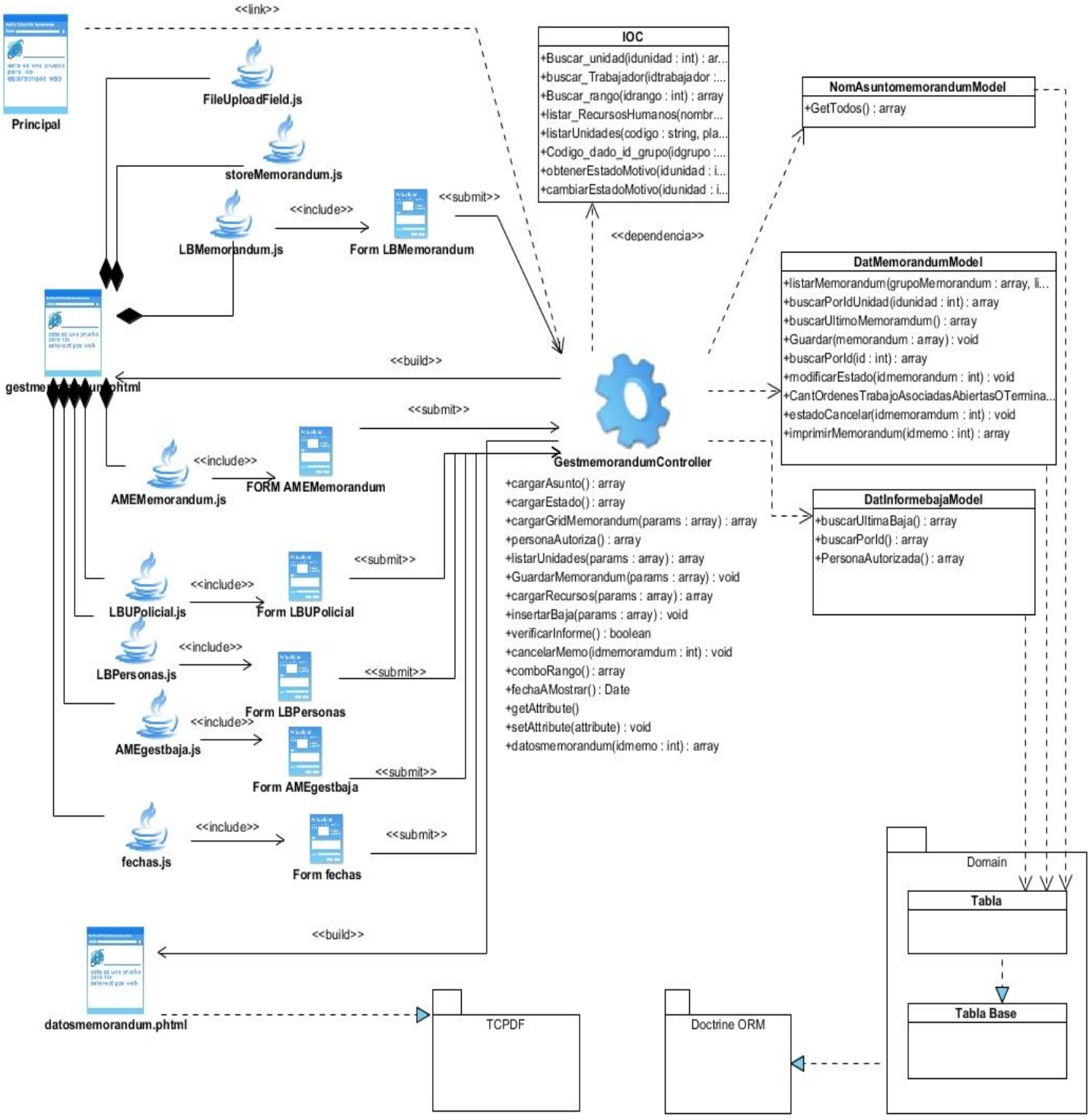


Imagen 8. Diagrama de clases de diseño: Gestionar Memorándum.

Diagrama de clases de diseño: Recuperaciones.

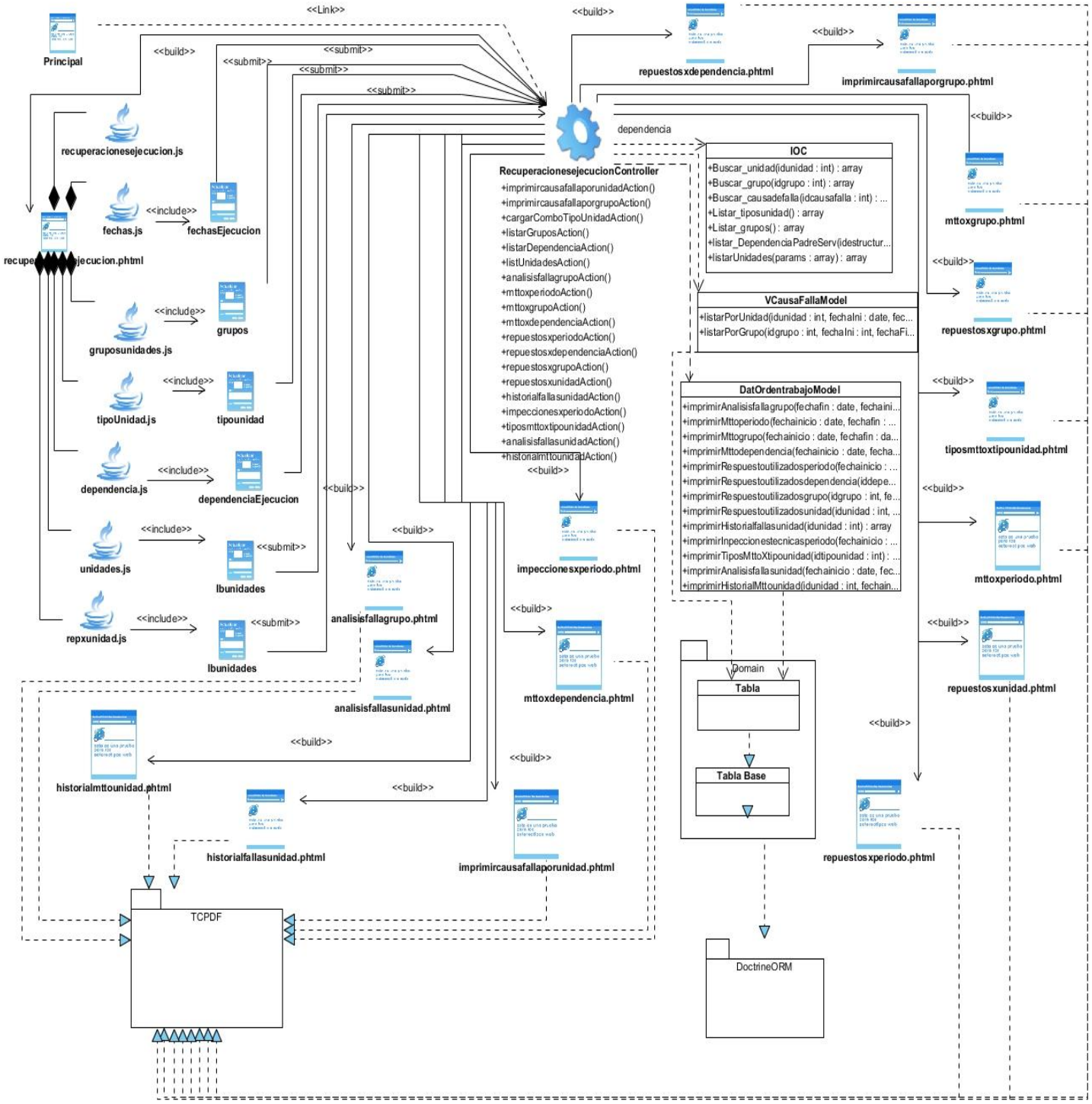


Imagen 9. Diagrama de clases de diseño: Recuperaciones.

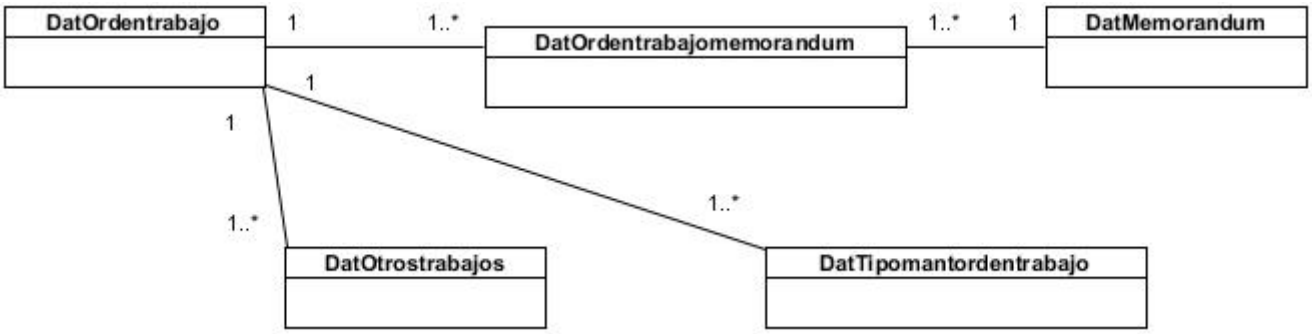


Imagen 10. Diagramas de clases del paquete de dominio para Generar Orden de Trabajo.

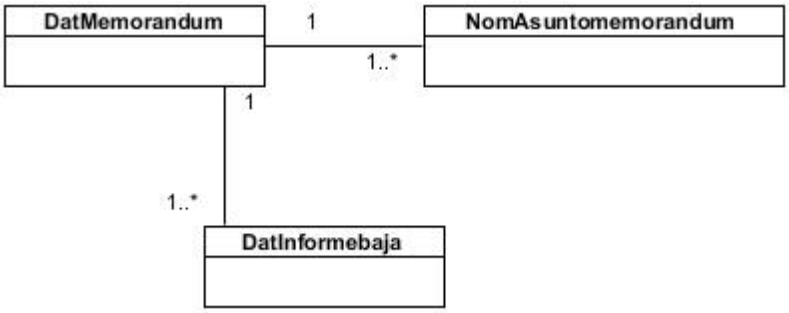


Imagen 11. Diagramas de clases del paquete de dominio para Gestionar Memorándum.

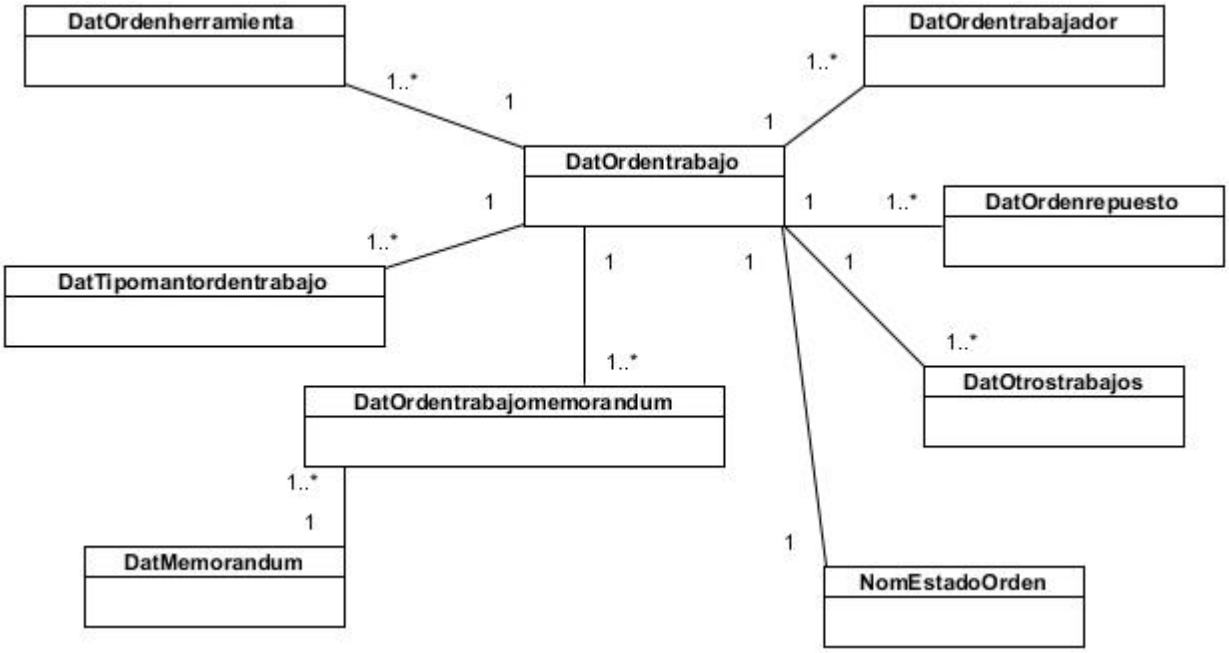


Imagen 12. Diagramas de clases del paquete de dominio para Gestionar Orden de Trabajo.

2.3.3 Descripción de las clases del diseño.

Tabla 1. Descripción de la clase GestordentrabajoController.

Nombre	GestordentrabajoController
Tipo de clase	Controladora
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
listarProgramacionAction()	Permite obtener la lista de las unidades policiales a las cuales se les puede generar una Orden de trabajo, mostrándolas en la vista GenerarOT.
emitirOrdenTrabajoAction ()	Permite obtener los valores necesarios para emitir una Orden de trabajo a una unidad policial seleccionada, enviando estos valores a la clase DatOrdenTrabajoModel para guardarlos.
guardarUmbralAction ()	Permite modificar un umbral de mantenimiento.
buscarUmbralAction ()	Permite obtener mediante POST el umbral de mantenimiento y mostrarlo al cargar la interfaz de GenerarOT.
guardarOrdenTrabajoAction ()	Permite obtener mediante POST los valores para modificar una Orden de trabajo, enviándolos a la clase DatOrdenTrabajoModel para modificar los datos guardados inicialmente en la Orden.
cancelarOrdenTrabajoAction ()	Permite cancelar una Orden de trabajo seleccionada desde la interfaz GestionarOT.
guardarRecursosAction ()	Permite actualizarle los recursos utilizados a una Orden de trabajo seleccionada en la interfaz GenerarOT, obteniendo mediante POST una lista de recursos utilizados en la Orden, los cuales se

	envían a la clase DatOrdenTrabajoModel para guardar los valores de los recursos.
listarOrdenTrabajoAction ()	Permite listar las Órdenes de trabajo devuelta por la clase DatOrdenTrabajoModel y mostrarlas en un listado en la interfaz GestionarOT.
listarTrabajosAction ()	Permite listar los trabajos asociados a una Orden de trabajo, devueltos por la clase DatOtrostrabajosModel, para mostrarlos y poder modificar esa lista a la hora de modificar la Orden seleccionada.
listarTrabajadoresOrdenAction ()	Permite listar los trabajadores asociados a una Orden de trabajo, al seleccionar una orden desde la interfaz GestionarOT, y actualizar los recursos de esa orden, esta lista se puede modificar.
listarRepuestosOrdenAction ()	Permite listar los repuestos asociados a una Orden de trabajo, al seleccionar una orden desde la interfaz GestionarOT, y actualizar los recursos de esa orden, esta lista se puede modificar.
listarHarramientasOrdenAction ()	Permite listar las herramientas asociadas a una Orden de trabajo, al seleccionar una orden desde la interfaz GestionarOT, y actualizar los recursos de esa orden, esta lista se puede modificar.
listarEstadosAction ()	Permite listar los estados para mostrar en la interfaz de GestionarOT.
listarCausaFallaAction ()	Permite listar las causas de fallas asociadas a una Orden de trabajo que se muestran a la hora de modificar la orden.
listarTipoFallaAction ()	Permite listar los tipos de fallas asociadas a una Orden de trabajo

	que se muestran a la hora de modificar la orden.
listarResponsablesDirectoAction ()	Permite listar los responsables directos que se mostraran al emitir una Orden de trabajo.
listarTrabajadoresAction ()	Permite listar los trabajadores que pueden asociarse a una Orden de trabajo cuando se actualizan los recursos utilizados en la Orden.
listarRepuestosAction ()	Permite listar los repuestos que pueden asociarse a una Orden de trabajo cuando se actualizan los recursos utilizados.
listarHarramientasAction ()	Permite listar las herramientas que pueden asociarse a una Orden de trabajo cuando se actualizan los recursos utilizados.
listarCargoAction ()	Permite listar los cargos que pueden ser utilizados como criterios de búsqueda al buscar los trabajadores que pueden asociarse a una Orden de trabajo.
listarRangoAction ()	Permite listar los rangos que pueden ser utilizados como criterios de búsqueda al buscar los trabajadores que pueden asociarse a una Orden de trabajo.
listarGrupoUnidadAction()	Permite listar los grupos de unidad para mostrarlos como un criterio de búsqueda de las unidades a las que se le puede emitir una Orden de trabajo.
buscarLecturaAction()	Permite buscar los datos de una lectura cuando al modificar una Orden de trabajo se registra una nueva lectura o se reinicia medidor.
actualizarMedidorAction	Permite al modificar una Orden de trabajo registrar una nueva lectura.

reiniciarMedidorAction()	Permite al modificar una Orden de trabajo reiniciar el medidor.

2.3.4 Patrones de diseño.

Para realizar un diseño más eficiente es conveniente utilizar uno o varios patrones de diseño. Los patrones de diseño describen un problema que ocurre repetidas veces en algún contexto determinado de desarrollo de software, y entregan una buena solución ya probada. Esto ayuda a diseñar correctamente en menos tiempo, a construir problemas reutilizables y extensibles, facilita la documentación, y facilita la comunicación entre los miembros del equipo de desarrollo (Larman, 1999).

- **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. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio. En fin se puede decir que separa la lógica de negocio de la presentación o interfaz.

  Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requisitos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

  Vista: Maneja la visualización de la información. Muestra la información del modelo al usuario. Se encarga de presentar la interfaz al usuario.

  Controlador: Son las clases que gestionan el manejo de la lógica del negocio. Gestionan las entradas del usuario.

Se utilizaron además patrones GRASP[17], los cuales se utilizan para la asignación de responsabilidades.

- **Experto:** Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos. La aplicación del patrón Experto

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

consiste en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Este tiene como beneficio que se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece tener sistemas más robustos y de fácil mantenimiento. Además el comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más fáciles de comprender y mantener (Larman, 1999). En estos diagramas se emplean en las clases controladoras y del modelo ya que son las que cuentan con la información necesaria para cumplir sus responsabilidades. Ejemplo: DatOrdentrabajoModel.

- **Creador:** Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Permite crear instancias de otras clases según la responsabilidad de la misma. Esto posibilita un bajo acoplamiento, mejores oportunidades de reutilización (Larman, 1999). En estos diagramas se emplean principalmente en las clases controladoras ya que estas manejan instancias de otros objetos. Ejemplo: GestordentrabajoController.
- **Controlador:** La aplicación del patrón Controlador consiste en asignar la responsabilidad de administrar un mensaje de eventos del sistema a una clase que represente: el negocio, la organización global, o el sistema global (Larman, 1999). Se emplean en las clases controladoras ya que son las encargadas de manejar los eventos del sistema. Ejemplo: GestordentrabajoController.
- **Bajo acoplamiento:** Asignar una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. Este patrón soluciona el problema de lograr una dependencia escasa y un aumento de la reutilización, asignando responsabilidades a las clases de tal forma que la dependencia entre las mismas sea la menor posible (Larman, 1999). Este patrón se emplea en la parte del negocio, ya que solo existen las relaciones necesarias entre las clases para mantener un bajo acoplamiento. Ejemplo: GestordentrabajoController y DatOrdentrabajoModel.
- **Alta cohesión:** Asigna una responsabilidad de modo que la cohesión siga siendo alta. En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. La utilización de este patrón permite que las

clases contengan las responsabilidades estrechamente relacionadas, sin tener que realizar un trabajo excesivo, posibilitando la claridad y facilidad con que se entiende el diseño, simplificando el mantenimiento y las mejoras en funcionalidad, logrando en ocasiones un bajo acoplamiento (Larman, 1999). Se emplea en las clases controladoras y del modelo ya 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. Ejemplo: DatOrdentrabajoModel.

También fueron utilizados **Patrones GOF,** pandilla de los cuatro o Gang of Four (GOF) son una herramienta fundamental para cualquier programador. Estos patrones son una descripción de clases y objetos que se comunican entre sí, adaptada para resolver un problema general de diseño en un contexto particular. En el diseño realizado se emplean los siguientes patrones GOF:

Patrones Creacionales:

- **Solitario:** Una clase de las que sólo una sola instancia puede existir, especificar el tipo de objetos a crear utilizando una instancia prototípica, y crear nuevos objetos copiando este prototipo. En este fue utilizado en las clases del dominio, donde los métodos contenidos en estas se hicieron de forma estática, con lo cual se garantiza que pudieran ser accedidos sin crear instancia de las mismas. Ejemplo DatOrdentrabajo.

Patrones Estructurales:

- **Fachada:** Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema. La utilización de este patrón está reflejada en la creación y empleo de la clase IOC, ya que por la necesidad de integración entre los módulos del Sistema de Gestión de Mantenimiento Vehicular, era necesaria la implementación de una clase donde fueran publicados los servicios necesarios por estos, facilitando su interacción.

Patrones de Comportamiento:

- **Mediador:** Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto. Este patrón se ve reflejado en la clase DatOrdentrabajomemorandum, permitiendo una relación de muchos a muchos entre las clases DatOrdentrabajo y DatMemorandum, lo que posibilita la comunicación entre ambas clases de manera menos compleja.

- **Cadena de responsabilidad:** Permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada. La cadena de responsabilidad se encarga de evitar el acoplamiento del remitente de una petición a su receptor, dando a más de un objeto la posibilidad de manejar la petición. Este patrón se refleja en las clases controladoras y las del modelo donde se establece la una cadena a seguir para las peticiones de la clase controladora al modelo y del modelo a la entidad. Ejemplos: GestordentrabajoController, DatOrdentrabajoModel.

2.3.5 Modelo de datos.

Un modelo de datos es la representación abstracta de los datos en un sistema gestor de base de datos. Básicamente el modelo de datos está formado por tres elementos fundamentales que son: los objetos, que son todas las entidades que manipulan los datos a persistir; los atributos, que son las características básicas de los objetos antes mencionados; y las relaciones, que son las que enlazan a dichos objetos entre sí (Pressman, 2005).

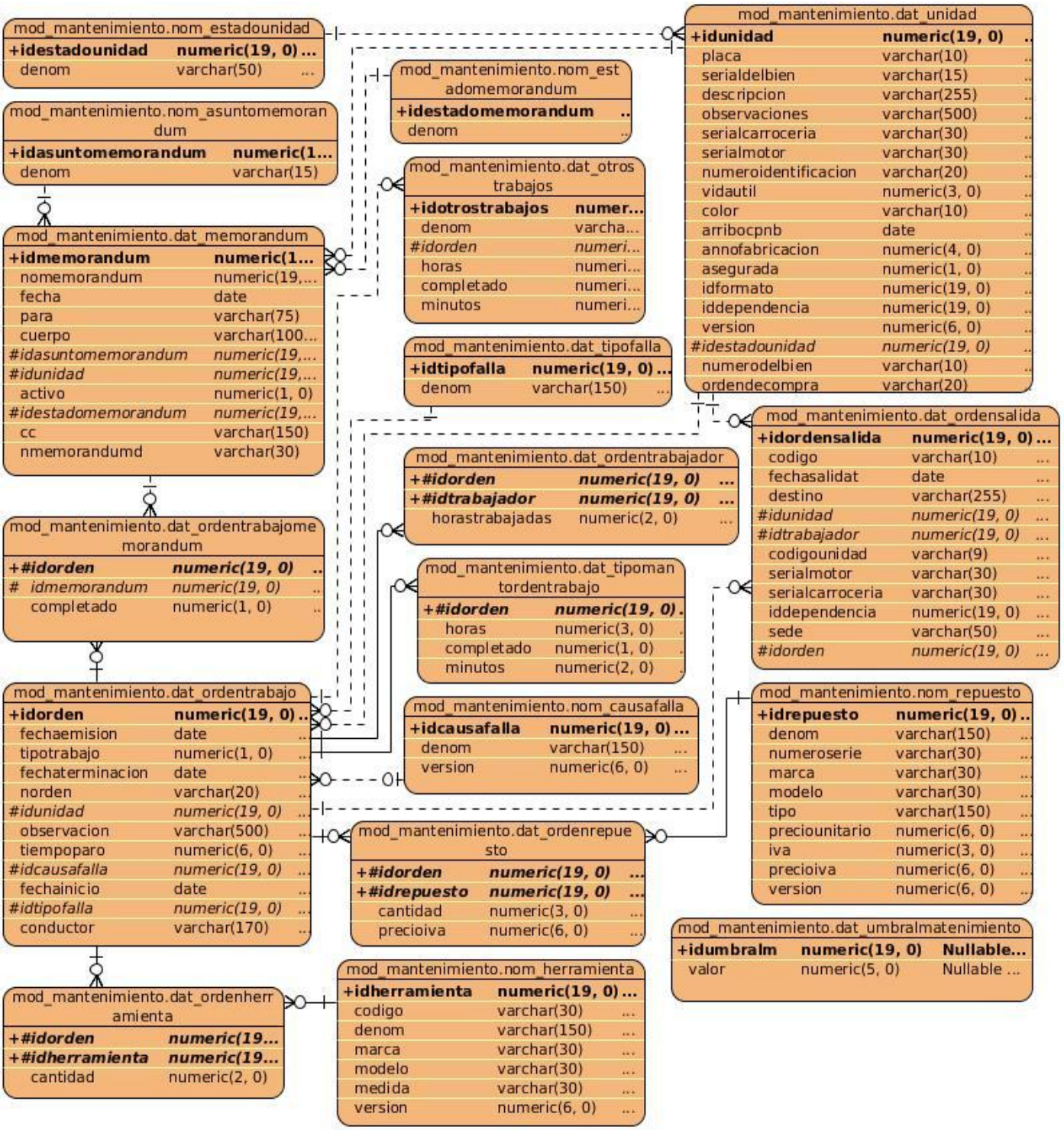


Imagen 13. Diseño de base de datos.

2.4 Implementación.

2.4.1 Diagrama de componentes.

Un diagrama de componentes se utiliza para modelar los componentes de un sistema y mostrar las dependencias entre ellos. Un componente representa un módulo de software con una interfaz bien definida (Mancha, 2011). La Imagen 14 muestra el diagrama de componentes desde la perspectiva del componente Ejecución.

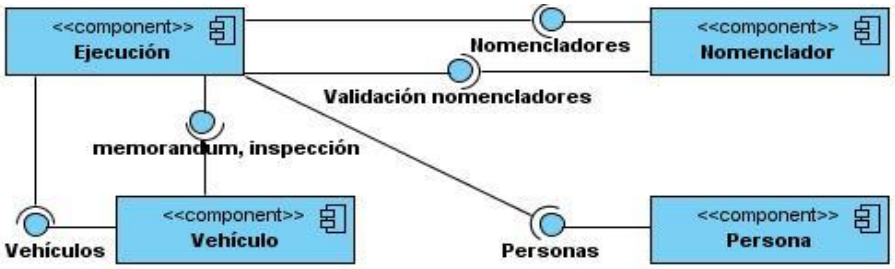


Imagen 14. Diagrama de componentes desde la perspectiva del componente Ejecución

2.4.2 Diagrama de despliegue.

Un diagrama de despliegue es un tipo de diagrama en el cual queda representado un modelado del hardware utilizado en la elaboración del software. Es una vista panorámica que describe la configuración del sistema para su ejecución en un ambiente del mundo real. El usuario desde su estación de trabajo podrá acceder al sistema que estará desplegado en el servidor web. Este servidor se conectará al servidor de base de datos que es necesario para el funcionamiento interno del sistema. A continuación se muestra el diagrama de despliegue confeccionado.

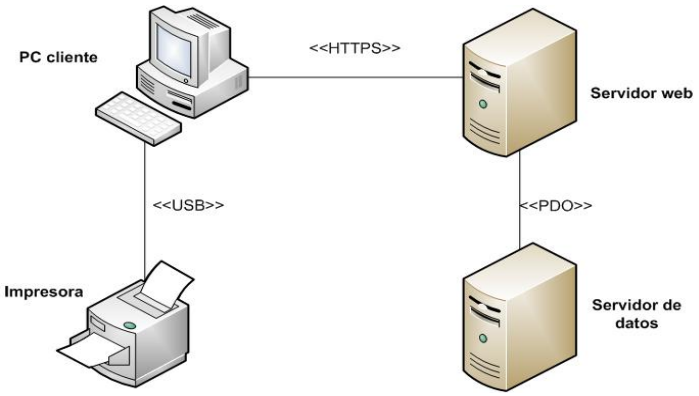


Imagen 15. Diagrama de despliegue.

2.4.3 Estándares de codificación.

Un estándar de código se basa en la estructura y apariencia física de un programa con el fin de facilitar la lectura, comprensión, mantenimiento del código, reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa. Un estándar de programación no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y legibilidad del código escrito (Mairelys Fernández González, 2010).

Los estándares de codificación utilizados en el módulo Ejecución se describen a continuación.

2.4.3.1 Nomenclatura de las variables

Las variables serán nombradas utilizando la nomenclatura CamelCasing en las variables que no son específicamente para la entrada de datos como los botones, gridpanels, ventanas, stores y formularios.

- Los botones se nombrarán según su funcionalidad: btn+Nombre. Ejemplo: btnEmitirOrden.
- Los estores se nombrarán: st+Nombre. Ejemplo: stUnidades.
- Los combobox se nombrarán según lo que almacenen, ejemplo: estado.
- Los campos de texto se nombrarán según lo que se registra en ellos, ejemplo numeroidentificacion.
- Los gridpanel se nombrarán: grid+Nombre. Ejemplo: gridUnidades.
- Las ventanas se nombrarán: ventana+Nombre. Ejemplo: ventanaBaja.

2.4.3.2 Nomenclatura de las funciones

- Las funciones o métodos de las clases controladoras se nombrarán según la acción que realizan, comenzando en minúscula y seguidos por la palabra Action. En caso de que la acción sea compuesta por varias palabras se utilizará la nomenclatura CamelCasing, o sea, seguido del primer nombre, comenzará el segundo con mayúscula y finalizando con la palabra Action. Ejemplo: emitirOrdenTrabajoAction.
- El nombre de los métodos o funciones de una clase no controladora comenzará en minúscula, en caso de que sea compuesto se utilizará la notación CamelCasing, o sea, seguido del primer nombre, comenzará el segundo con mayúscula. Ejemplo: emitir, imprimirInformeBaja.

2.4.3.3 Nomenclatura de las clases

- El nombre de las clases de las vistas debe comenzar con una letra mayúscula según el requisito que realiza, seguido el nombre iniciado con mayúscula y lo demás en minúscula. En caso de estar compuesta por varios requisitos se utilizará la notación PascalCasing, se le agregarán al inicio las letras que representen los requisitos que cumple la clase en mayúscula y seguido el nombre comenzando con mayúscula. Ejemplo: EOrden (emitir), LBOrden (listar y buscar).
- El nombre para las clases controladoras comenzará con mayúscula, seguido el resto en minúscula y en dependencia de su función, agregando al final la palabra Controller. Ejemplo: GestordentrabajoController.
- El nombre de las clases del modelo que son generadas mediante un mapeo a la base de datos realizado por la herramienta Doctrine Generator v 3.0, la cual fue desarrollada en el CEIGE, se define con la palabra Nom si se nombra a un nomenclador, de lo contrario comenzará con la palabra Dat, seguido el nombre iniciado en mayúscula y al final se le agregará la palabra Model. Ejemplo: DatOrdentrabajoModel.
- El nombre de las entidades que son generadas mediante un mapeo a la base de datos realizado por la herramienta Doctrine Generator v 3.0, se define con la palabra Nom si se nombra a un nomenclador, de lo contrario comenzará con la palabra Dat y seguido del nombre comenzando con mayúscula. Ejemplo: DatOrdentrabajo.

2.4.4 Descripción de la implementación por funcionalidades.

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

2.4.4.1 Generar Órdenes de trabajo.

Este componente es el encargado de obtener toda la información necesaria en el área de Transporte del CPNB para generar una Orden de trabajo para las unidades policiales a partir de un tipo de mantenimiento determinado. La imagen 16 muestra la vista de cuando se accede al módulo Generación de Órdenes de trabajo, esta vista principal mostrará las unidades a las cuales se le puede generar una Orden de trabajo y seleccionando alguna de estas unidades brinda la opción de emitir una Orden y realizar una inspección técnica a esa unidad. Además permite realizar una búsqueda a las unidades policiales según los filtros número de identificación, placa, serial de carrocería, grupo y umbral de mantenimiento.

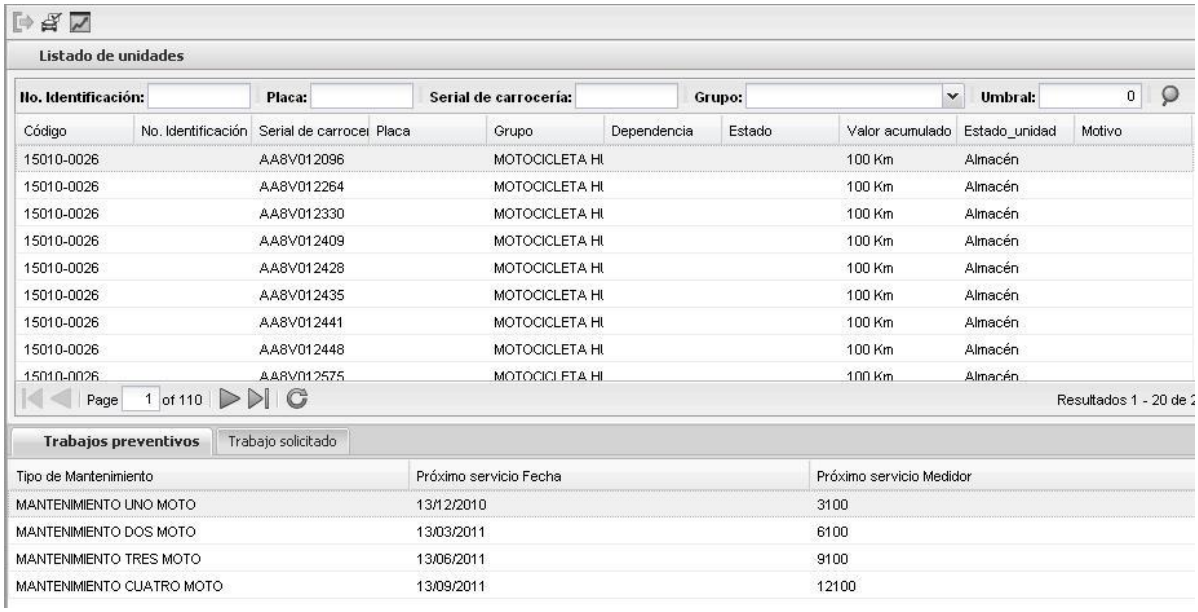


Imagen 16. Listado de unidades policiales.

2.4.4.1.1 Emitir Orden de trabajo.

Al seleccionar una unidad policial y posteriormente seleccionar la opción Emitir, se muestra una interfaz, mediante la cual se introducen los datos necesarios para emitir una Orden de trabajo. Para llenar el campo “Realizado por” se muestra una interfaz con un listado de los Recursos Humanos que pueden realizar la Orden. Al dar clic en el botón Aceptar se emite la Orden de trabajo mostrando un mensaje de información al usuario. En el Anexo 2 se muestran las diferentes vistas para emitir una Orden de trabajo.

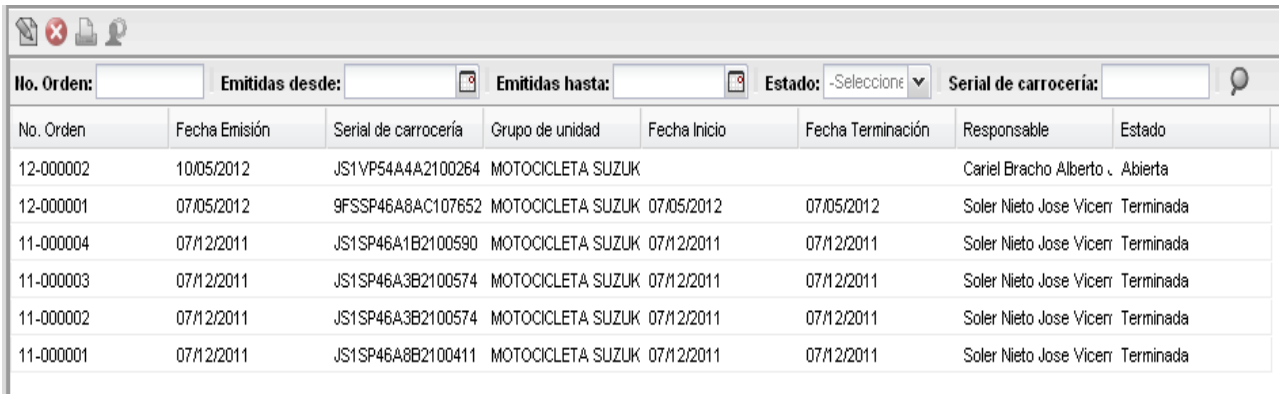
2.4.4.1.2 Adicionar Umbral de mantenimiento.

Al seleccionar la opción Umbral se muestra una interfaz que mediante ella se permite registrar un nuevo umbral de mantenimiento. Al dar clic en el botón Aceptar se adiciona un nuevo umbral mostrando un mensaje de información al usuario. En el Anexo 3 se muestra la vista para adicionar un nuevo Umbral de mantenimiento.

2.4.4.2 Órdenes de trabajo.

Este componente es el encargado de obtener toda la información necesaria en el área de Transporte del CPNB para gestionar las Órdenes de trabajo generadas para las unidades policiales. La imagen 17 muestra la vista de cuando se accede al módulo Órdenes de trabajo, esta vista principal mostrará las Órdenes de trabajo generadas, a las que una vez seleccionada una de ellas, permite mostrar un reporte

completo de la misma, modificar sus datos, cancelarla y actualizar los recursos utilizados en esta. Además permite realizar una búsqueda a las Órdenes de trabajo según los filtros número de la Orden, rango de fechas de emisión, estado de la Orden y serial de carrocería.



No. Orden	Fecha Emisión	Serial de carrocería	Grupo de unidad	Fecha Inicio	Fecha Terminación	Responsable	Estado
12-000002	10/05/2012	JS1VP54A4A2100264	MOTOCICLETA SUZUK			Cariel Bracho Alberto .	Abierta
12-000001	07/05/2012	9FSSP46A8AC107652	MOTOCICLETA SUZUK	07/05/2012	07/05/2012	Soler Nieto Jose Vicen	Terminada
11-000004	07/12/2011	JS1SP46A1B2100590	MOTOCICLETA SUZUK	07/12/2011	07/12/2011	Soler Nieto Jose Vicen	Terminada
11-000003	07/12/2011	JS1SP46A3B2100574	MOTOCICLETA SUZUK	07/12/2011	07/12/2011	Soler Nieto Jose Vicen	Terminada
11-000002	07/12/2011	JS1SP46A3B2100574	MOTOCICLETA SUZUK	07/12/2011	07/12/2011	Soler Nieto Jose Vicen	Terminada
11-000001	07/12/2011	JS1SP46A8B2100411	MOTOCICLETA SUZUK	07/12/2011	07/12/2011	Soler Nieto Jose Vicen	Terminada

Imagen 17. Registro de Órdenes de trabajo.

2.4.4.2.1 Modificar Orden de trabajo.

Al seleccionar una Orden de trabajo en estado abierta y seleccionar la opción modificar se muestra una ventana mediante la cual se le pueden modificar los datos a la Orden de trabajo, así como agregarle o quitarle trabajos a la misma. Al dar clic en el botón Aceptar se modifican los datos en la Orden de trabajo mostrando un mensaje de información al usuario. En el Anexo 4 se muestran las vistas relacionadas con modificar Orden de trabajo.

2.4.4.2.2 Cancelar Orden de trabajo.

Al seleccionar una Orden de trabajo en estado abierta y seleccionar posteriormente la opción cancelar, se muestra un mensaje de confirmación para cancelar la Orden de trabajo. Al dar clic aceptando esa confirmación la Orden es cancelada mostrando un mensaje de información al usuario. En el Anexo 5 se muestra la interfaz para cancelar una Orden de trabajo.

2.4.4.2.3 Actualizar recursos utilizados.

Al seleccionar una Orden de trabajo en estado abierta y seleccionar la opción Actualizar recursos utilizados, se muestra una ventana con los repuestos, herramientas y recursos humanos utilizados en la Orden de trabajo. Al dar clic en el botón Aceptar se actualizan los recursos utilizados en la Orden. En el Anexo 6 se muestran las vistas relacionadas con actualizar los recursos utilizados en la Orden.

2.4.4.3 Gestionar Memorándum.

Este componente es el encargado de obtener toda la información necesaria en el área de Transporte del CPNB para gestionar los memorándum que se manejen en esta área. La imagen 18 muestra la vista mostrada al acceder al módulo Memorándum, esta vista principal muestra los memorándum generados, a los cuales al ser seleccionados se permite modificar sus datos, cancelarlos, mostrarlos en un reporte y generarles un informe de baja en caso de que su asunto sea de exclusión y su estado no sea cancelado. Además permite buscar los memorándum que cumplan con los filtros número de memorándum, fecha, asunto y estado del memorándum.

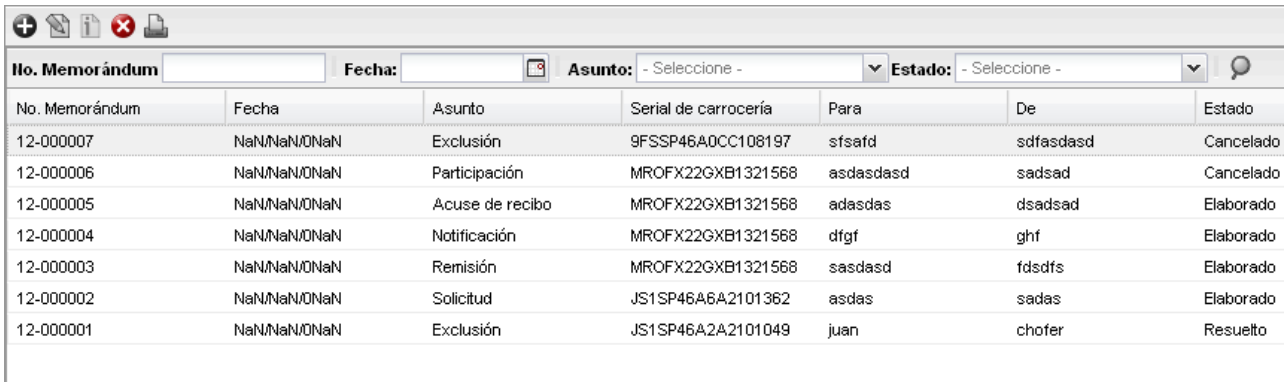


Imagen 18. Gestionar Memorándum.

2.4.4.3.1 Adicionar Memorándum.

Al seleccionar la opción Adicionar Memorándum se muestra una ventana mediante la cual se introducen todos los datos necesarios para crear un nuevo memorándum para una unidad policial, la cual es seleccionada al dar clic en el campo “Unidad policial”, donde se muestra una ventana con las unidades. En el Anexo 7 se muestran las vistas relacionadas con adicionar un nuevo memorándum.

2.4.4.3.2 Modificar Memorándum.

Al seleccionar un memorándum y dar clic en la opción Modificar se muestra una ventana cargando todos los datos que pueden ser modificados, al dar clic en el botón Aceptar se modifica el memorándum mostrando un mensaje de información al usuario. En el Anexo 8 se muestra la vista para la funcionalidad modificar memorándum.

2.4.4.3.3 Informe de Baja.

Al seleccionar la opción Informe Baja después de seleccionar un memorándum en estado elaborado y asunto de exclusión, se muestra una ventana mediante la cual se introducen los datos necesarios para generar un informe de baja. Al dar clic en el campo “Realizado por” se muestra una ventana con los recursos humanos existentes. En el Anexo 9 se muestran las vistas relacionadas con la funcionalidad Informe de Baja.

2.4.4.3.4 Cancelar Memorándum.

Al seleccionar un memorándum y seleccionar la opción Cancelar se muestra un mensaje de confirmación para cancelar el memorándum. Al dar clic aceptando cancelar se cancela el memorándum mostrando un mensaje de información al usuario. Ver Anexo 10.

2.4.4.4 Recuperaciones.

Este componente es el encargado de mostrar los reportes relacionados con la ejecución del mantenimiento. Esta interfaz principal muestra un listado con los nombres de los reportes que pueden ser mostrados. Al seleccionar un reporte de los que se encuentran en este listado y seleccionar la opción Imprimir se muestra una ventana para introducir los parámetros necesarios en dependencia del reporte a mostrar.

2.5 Publicación de servicios entre componentes.

La comunicación dentro de un mismo componente se ejecuta de forma directa, sin embargo, la que se establece entre los diferentes componentes de la aplicación va más allá de un simple llamado a un servicio, esta se basa en el empleo de un registro de datos de los subsistemas contenidos en un fichero xml mapeado por el marco de trabajo para el funcionamiento de los componentes, llamado: inversión de control (IOC). El IOC registra las funcionalidades que ofrecen los métodos de las clases de servicios (services) de cada componente del sistema, especifica respuestas deseadas a sucesos o solicitudes de datos concretas, orden necesario y el conjunto de sucesos que tienen que ocurrir según los parámetros requeridos para poder hacer uso de un determinado servicio.

2.5.1 Servicios que consume el componente Ejecución como parte de la integración.

Tabla 2. Servicios que consume el componente Ejecución.

Servicios del componente	Descripción

Nomencladores.	
Buscar_tipomant	Recibe un id de tipo de mantenimiento para buscar un tipo de mantenimiento y devolver un arreglo con los valores id y denominación del tipo de mantenimiento encontrado.
Listar_causasdefalla	Lista las causas de falla nomencladas devolviéndolas en un arreglo con los valores id y nombre.
Buscar_causadefalla	Busca una causa de falla según un id de causa de falla para devolver un arreglo con los valores id y nombre de la causa de falla encontrada.
Buscar_unidadmedida	Busca una unidad de medida según un id de una unidad de mediad recibido para devolver un arreglo con los valores id, nombre, abreviatura y categoría de la unidad de medida encontrada.
Listar_tiposunidad	Lista los tipos de unidades devolviendo un arreglo con los valores id y denominación de cada tipo de unidad.
Buscar_tipounidad	Busca un tipo de unidad según un id de tipo de unidad para devolver un arreglo con los valores id y denominación del tipo de unidad.
Buscar_herramienta	Busca una herramienta según un id de herramienta para devolver un arreglo con los valores id, denominación, código, marca, modelo y medida de la herramienta.
Buscar_repuestos	Busca un repuesto según un id de repuesto para devolver un arreglo con los valores id, denominación, número de serie, marca, modelo, id de unidad de medida y tipo del repuesto.
Buscar_repuestoavanzado	Busca los repuestos que cumplan con los filtros recibidos (número de serie, denominación, tipo) para devolver un arreglo con los valores id, denominación, número de serie, marca, modelo, id de unidad de medida

	y tipo de los repuestos encontrados.
Listar_rangos	Lista los rangos devolviendo un arreglo con los valores id y denominación de cada rango.
Buscar_rango	Busca un rango dado el id de rango para devolver un arreglo con los valores id y denominación del rango.
Listar_cargos	Lista los cargos devolviendo un arreglo con los valores id y denominación de cada cargo.
Buscar_cargo	Busca un cargo dado el id de cargo para devolver en un arreglo los valores id y denominación del cargo.
Buscar_herramientaavanzado	Busca las herramientas que cumplan con los filtros denominación y código para devolver un arreglo con los valores denominación, código, marca, modelo y medida de las herramientas.
Servicios del componente Configuración.	**Descripción**
Listar_grupos	Lista los grupos de unidades devolviendo un arreglo con los valores nombre, marca, modelo y tipo de unidad de cada grupo.
Buscar_grupo	Busca un grupo según un id devolviendo un arreglo con los valores denominación, marca, modelo, id de tipo de unidad, grupo, umbral, tipo de mantenimiento, frecuencia por fecha, frecuencia de medidor y régimen del grupo.
Codigo_dado_id_grupo	Busca un grupo dado el id para devolver un arreglo con el código.
TiposFallas_grupo	Busca los tipos de fallas por grupo dado el id de grupo para devolver un arreglo con los valores id y tipo de falla.

Buscar_tipofalla	Busca un tipo de falla dado el id para devolver un arreglo con los valores id y tipo de falla.
listarPartesPorIdGrupo	Lista las partes dado un id de grupo para devolver un arreglo con los valores id y denominación de las partes.
buscarGrupoPorIdGrupoIdTipoMtto	Busca un grupo dado el id de grupo y el id de tipo de mantenimiento para devolver un arreglo con los valores nombre, modelo, marca, tipo de unidad y régimen.
buscarActividadesPorIdGrupoTipoMant	Busca las actividades dado un id de grupo por tipo de mantenimiento para devolver un arreglo con los valores id, nombre y duración de las actividades.
Servicios del componente Vehículo.	**Descripción**
Obtener_lectura	Permite Obtener una lectura dado el id de unidad para devolver un arreglo con los valores código, número de identificación, placa, unidad de medida, serial de carrocería, serial de motor, valor base, valor actual, valor acumulado y fecha de lectura.
modificarRegistroLectura	Modifica un Registro de Lectura dado la fecha de lectura, id de unidad y lectura actual.
reiniciar_Medidor	Reinicia un Medidor dado id de unidad, fecha, valor antes de reinicio y valor de reinicio.
Listar_vehiculos	Lista vehículos devolviendo un arreglo con los valores id de unidad, código, número de identificación, placa, grupo, serial de carrocería, serial de motor, dependencia, sede, estado, estado de unidad y motivo de todos los vehículos.

Buscar_unidad	Busca una unidad dado el id para devolver un arreglo con los valores código, número de identificación, placa, grupo, serial de carrocería, serial de motor, dependencia, sede, estado, estado de unidad y motivo.
Codigo_dado_id_it	Obtiene el código dado el id de inspección para devolver un arreglo con el valor del código.
guardarInspeccionTecnica	Guarda una Inspección Técnica dado el id de inspección.
listarUnidades	Lista las unidades que cumplan con los filtros código, placa, número de identificación, id de unidad, id de grupo, número de inventario, serial de carrocería, serial de motor, año de fabricación, id de dependencia, id de estado, id de motivo y color.
listarNecesitanMtto	Lista las unidades que necesitan mantenimiento, dado el número de identificación, placa, id de grupo y umbral.
buscar_ResponsableDirecto	Busca un responsable directo dado el id de responsable para devolver un arreglo con los valores de ese responsable.
obtenerEstadoMotivo	Obtiene el estado y motivo dado el id de una unidad para devolver un arreglo con los valores id de estado de unidad, id de motivo estado de unidad y código asociado.
cambiarEstadoMotivo	Cambia el estado y motivo dado el id de unidad, id de estado, id de motivo, número de documento y si es cancelado.
programar	Programa dado el id de unidad, id de grupo, id de tipo de mantenimiento y fecha de servicio.
ultimoPorIdComprobante	Obtiene el último número de comprobante dado el id de comprobante.
devolverIdDependenciaFec	Devuelve el id de dependencia de fecha de asignación última dado el id

haAsigUltima	de unidad.
Servicios del componente Persona.	**Descripción**
listar_RecursosHumanos	Lista los recursos humanos dados los filtros nombre, apellidos, cédula, rango para devolver un arreglo con los valores id de persona, cédula, nombre, apellidos y rango los recursos humanos.
buscar_Trabajador	Busca un trabajador dado el id de persona, para devolver un arreglo con los valores cédula, nombre, apellidos y rango del trabajador.

2.5.2 Servicios que brinda el componente Ejecución como parte de la integración.

Tabla 3. Servicios que brinda el componente Ejecución.

Servicios que consume el componente Nomencladores.	**Descripción**
repuestoAsociado	Busca dado el id de repuesto si el repuesto está asociado para devolver verdadero o falso.
tipofallaAsociado	Busca dado el id de tipo de falla si el tipo de falla está asociado para devolver verdadero o falso.
causafallaAsociado	Busca dado el id de causa de falla si la causa de falla está asociada para devolver verdadero o falso.
tipoMttoAsociado	Busca dado el id de tipo de mantenimiento si el tipo de mantenimiento está asociado para devolver verdadero o falso.
cargoAsociadoOrdenTrabajador	Busca dado el id de cargo si el cargo está asociado para devolver verdadero o falso.
herramientaAsociadaOrden	Busca dado el id de herramienta si la herramienta está asociada

	para devolver verdadero o falso.
Servicios que consume el componente Vehículo.	**Descripción**
buscarUltimoInforme	Busca el último Informe.
guardarInforme	Guarda un informe dado un arreglo con los valores de un informe de Inspección.
comprobarInformaInspeccion dado idinspeccion	Comprobar informe de Inspección dado el id de Inspección para devolver verdadero o falso.
unidadAsociadaMemorandum	Busca dado el id de unidad si la unidad está asociada a un Memorándum para devolver verdadero o falso.
Servicios que consume el componente Persona.	**Descripción**
personaAsociadaInformeInspeccion	Busca dado el id del trabajador si la persona está asociada a un Informe de Inspección para devolver verdadero o falso.
personaAsociadaMemorandum	Busca dado el id del trabajador si la persona está asociada a un Memorándum para devolver verdadero o falso.
personaAsociadaOrdenSalida	Busca dado el id del trabajador si la persona está asociada a una Orden de salida para devolver verdadero o falso.
personaAsociadaOrdenTrabajador	Busca dado el id del trabajador si la persona está asociada a una Orden de trabajador para devolver verdadero o falso.
personaAsociadaOrdenTrabajo	Busca dado el id del trabajador si la persona está asociada a una Orden de trabajo para devolver verdadero o falso.

2.6 Conclusiones parciales

Se realiza un análisis de los artefactos entregados por los analistas que permitió comprobar que no existía ambigüedad en las especificaciones de los requisitos funcionales. Se describen los mecanismos de diseño que serán utilizados. Se exponen los artefactos generados del diseño del módulo Ejecución, así como la descripción de los mismos. Se dan a conocer los patrones de diseño empleados, así como el modelo de datos elaborado. Se especifican además el diagrama de componentes y despliegue para el sistema. Se describen los estándares de codificación que serán usados en la implementación de la propuesta solución, así como los servicios que se brindan y se consumen por el componente Ejecución.

CAPÍTULO 3: VALIDACIÓN Y PRUEBAS.

3.1 Introducción

En el presente capítulo se hace referencia inicialmente a las métricas Tamaño de clase y Relaciones entre clases, las cuales son usadas para la validación del diseño de la propuesta solución, con el fin de comprobar su correcta realización. Además se especifican las pruebas de software que son usadas para la validación y comprobación de la calidad del sistema desarrollado.

3.2 Métricas para la validación del diseño propuesto

Las métricas de software son una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. Permiten averiguar cuán bien están definidas las clases y el sistema, lo cual tiene un impacto directo en el mantenimiento del mismo, tanto por la comprensión de lo desarrollado como por la dificultad de modificarlo con éxito. Posibilitan al ingeniero evaluar el software al inicio del proceso, haciendo cambios que reducirán la complejidad y mejorarán la viabilidad, a largo plazo, del producto final. También permiten descubrir y corregir problemas potenciales antes de que se conviertan en defectos catastróficos. Estas métricas tienen como propósito entender y mejorar la calidad del producto, evaluar la efectividad del proceso y mejorar la calidad del trabajo llevado a cabo al nivel del proyecto (Pressman, 2005).

Una vez realizado el diseño del componente Ejecución, se dio paso a su validación, para esto se utilizaron las métricas orientadas a objetos, específicamente las orientadas a clases, ya que las medidas y métricas para una clase individual, la jerarquía de clases y las colaboraciones de estas permiten medir la calidad del diseño propuesto (Pressman, 2005).

3.2.1 Métricas propuestas por Lorenz y Kidd

En su libro sobre las métricas orientadas a objetos, separan las métricas basadas en clases en cuatro amplias categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas al tamaño se centran en el recuento de atributos y operaciones para cada clase individual y los valores promedio para el sistema orientado a objetos como un todo (Pressman, 2005).

Las métricas empleadas para evaluar el diseño realizado fueron la métrica **Tamaño de clase (TC)** y la métrica **Relaciones entre clases (RC),** ya que se ajustan para evaluar al diseño realizado, siendo las mismas de fácil utilización.

Estas métricas están diseñadas para evaluar los siguientes atributos de calidad:

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 la planificación del proyecto.

Cantidad de pruebas. Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto (componente, módulo, clase, conjunto de clases) diseñado.

Debido a la cantidad de clases que posee el módulo, solo se le aplicó a las clases controladoras y del modelo de los diagramas realizados.

Para la evaluación de las clases fueron utilizados umbrales para el tamaño general, la responsabilidad, la complejidad y la reutilización de las clases.

Métrica Tamaño de clase (TC).

Tabla 4. Umbrales para la TC.

Clasificación	Valores de los umbrales
Pequeño	<=Promedio de operaciones(PO)
Medio	>PO y <=2*PO
Grande	>2*PO

Tabla 5. Afectación de los atributos.

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

Tabla 6. Rango de valores para los criterios de evaluación de la métrica Tamaño de Clase (TC).

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad de implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Tabla 7. Tamaño de las clases según sus atributos y operaciones.

No	Nombre	Cantidad de atributos	Cantidad de operaciones	Tamaño
1	GestordentrabajoController	0	25	Grande
2	GestmemorandumController	0	13	Medio
3	RecuperacionesejecucionController	0	19	Grande
4	DatUmbralmantenimientoModel	0	2	Pequeño
5	DatOrdentrabajoModel	0	19	Grande
6	DatOrdentrabajomemorandumModel	0	2	Pequeño
7	DatTipomantordentrabajoModel	0	2	Pequeño
8	DatOtrostrabajosModel	0	3	Pequeño
9	DatMemorandumModel	0	10	Medio
10	DatOrdenherramientaModel	0	3	Pequeño
11	DatOrdentrabajadorModel	0	3	Pequeño
12	DatOrdenrepuestoModel	0	3	Pequeño
13	NomEstadoOrdenModel	0	2	Pequeño
14	VCausaFallaModel	0	2	Pequeño
15	NomAsuntomemorandumModel	0	1	Pequeño

16	DatInformebajaModel	0	3	Pequeño

Tabla 8. Cantidad de clases por clasificación.

Clasificación	Cantidad de clases	Responsabilidad de las clases	Complejidad de implementación de las clases	Reutilización de las clases
Pequeño	11	Baja	Baja	Alta
Medio	2	Medio	Medio	Medio
Grande	3	Alta	Alta	Baja

Tabla 9. Resultado general de la métrica.

Cantidad de clases	Cantidad de clases pequeñas	Cantidad de clases grandes	Cantidad de clases medias	Promedio de atributos	Promedio de operaciones
16	11	3	2	0	7

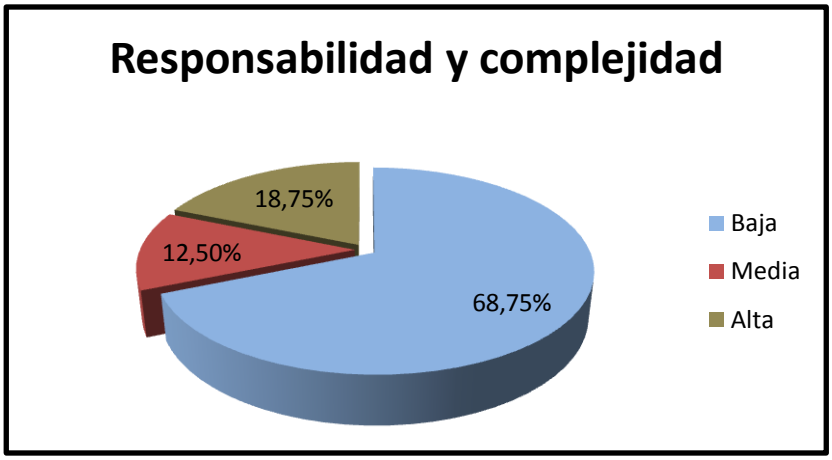


Imagen 19. Resultados obtenidos de la evaluación de la métrica TC para Responsabilidad y Complejidad.

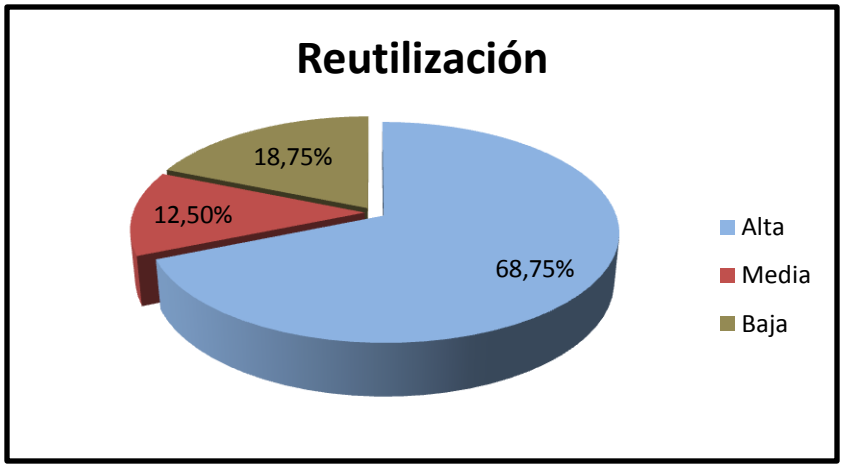


Imagen 20. Resultados obtenidos de la evaluación de la métrica TC para Reutilización.

Al realizar un análisis de los resultados obtenidos al aplicarle esta métrica a las clases controladoras y a las del modelo se demuestra que la mayoría de estas clases se encuentran en la categoría de pequeñas por lo que presentan un bajo nivel de responsabilidad, complejidad y una alta reutilización, siendo este diseño lo más simple posible, permitiendo una sencilla implementación y una amplia realización de pruebas con mayor facilidad.

Métrica Relaciones entre clases (RC).

Esta métrica está dada por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad.

Tabla 10. Criterios y categorías de evaluación de la métrica Relaciones entre clases (RC).

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Tabla 11. Instrumento de evaluación de la métrica RC.

Nombre	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
GestordentrabajoController	9	Alto	Alto	Bajo	Alto
GestmemorandumController	4	Alto	Medio	Medio	Medio
RecuperacionesejecucionController	3	Alto	Medio	Medio	Medio
DatUmbralmantenimientoModel	1	Bajo	Bajo	Alto	Bajo
DatOrdentrabajoModel	8	Alto	Alto	Bajo	Alto
DatOrdentrabajomemorandumModel	2	Medio	Bajo	Alto	Bajo
DatTipomantordentrabajoModel	2	Medio	Bajo	Alto	Bajo
DatOtrostrabajosModel	1	Bajo	Bajo	Alto	Bajo
DatMemorandumModel	1	Bajo	Bajo	Alto	Bajo
DatOrdenherramientaModel	2	Medio	Bajo	Alto	Bajo
DatOrdentrabajadorModel	2	Medio	Bajo	Alto	Bajo
DatOrdenrepuestoModel	2	Medio	Bajo	Alto	Bajo
NomEstadoOrdenModel	1	Bajo	Bajo	Alto	Bajo
VCausaFallaModel	2	Medio	Bajo	Alto	Bajo
NomAsuntomemorandumModel	1	Bajo	Bajo	Alto	Bajo
DatInformebajaModel	1	Bajo	Bajo	Alto	Bajo

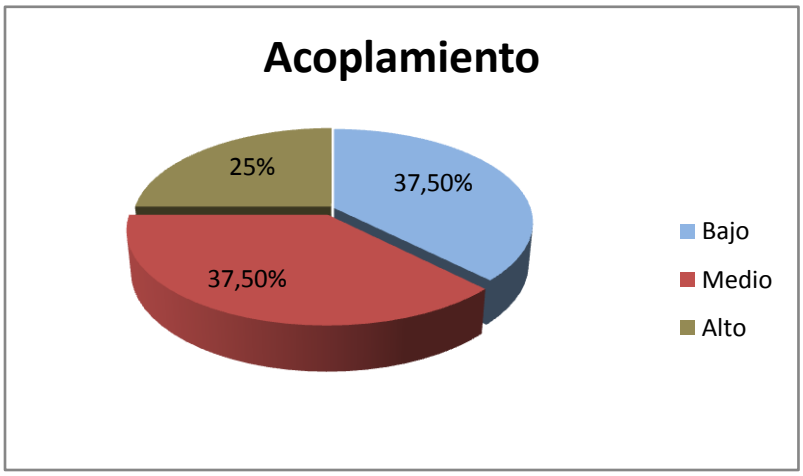


Imagen 21. Resultados de la evaluación de la métrica RC para el atributo Acoplamiento.

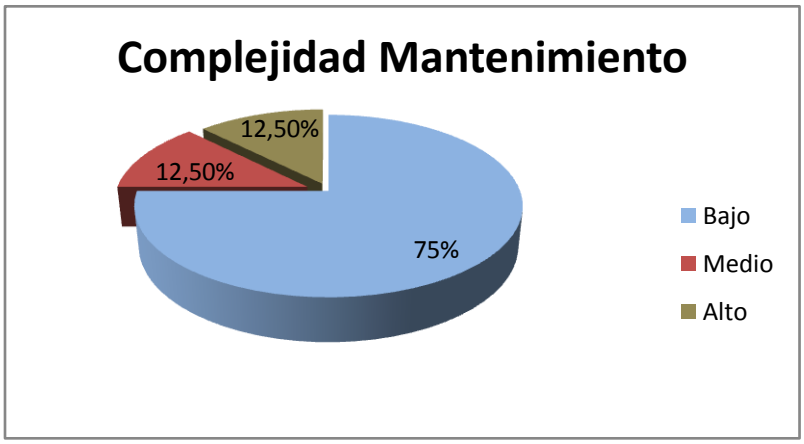


Imagen 22. Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento.

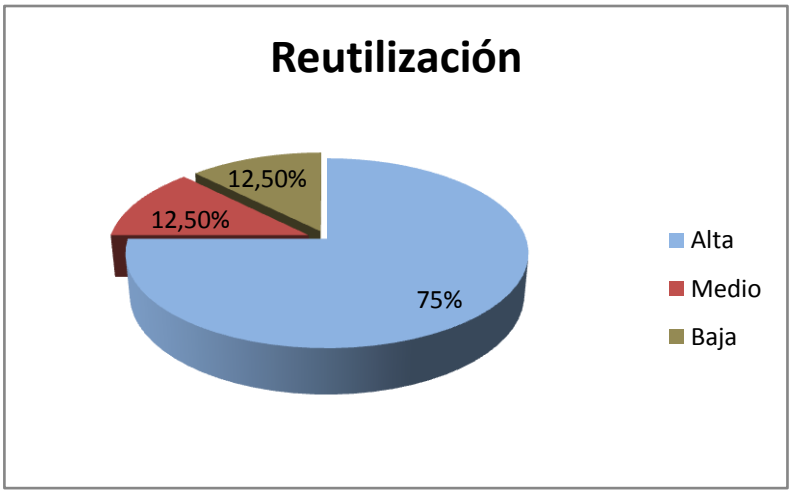


Imagen 23. Resultados de la evaluación de la métrica RC para el atributo Reutilización.

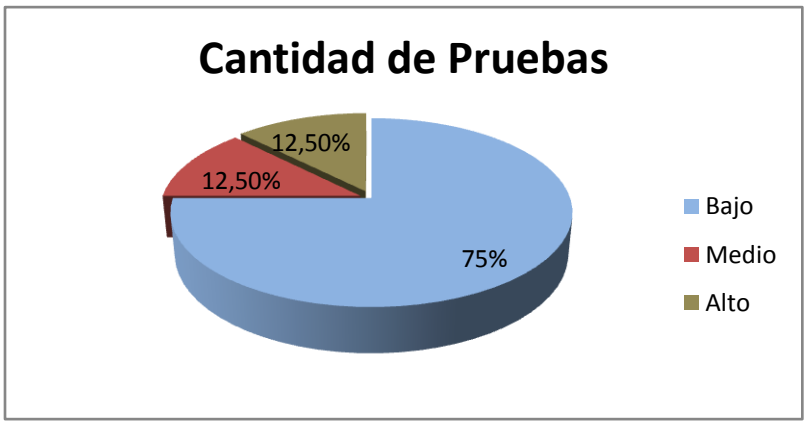


Imagen 24. Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.

Luego de aplicarse la métrica de diseño RC y obtenidos los resultados de la evaluación del instrumento de medición de la métrica, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 75% de las clases empleadas poseen menos de 3 dependencias de otras clases lo que lleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización). Favoreciendo de esta manera la reutilización de las clases así como la modificación e implantación del diseño.

Al aplicar las métricas a las clases del diseño definidas se cumple que las métricas de Tamaño de la clase y Relaciones entre clases cumplen las restricciones para la aceptación de las clases.

3.3 Pruebas de software

3.3.1 Pruebas de caja blanca

La prueba de caja blanca, o pruebas de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Para la solución desarrollada la prueba de Caja Blanca aplicada fue la del camino básico. Esta técnica de prueba permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del

conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (Pressman, 2005).

Para utilizar esta técnica primeramente es necesario el cálculo de la complejidad ciclomática del código fuente que vaya a ser analizado, para lo cual se deben enumerar las sentencias de código del mismo y elaborar el grafo de flujo de la funcionalidad. Las sentencias enumeradas del método guardar($OrdenTrabajoArr) se encuentran en las imágenes 26 y 27, que a continuación son presentadas:

```
public function guardar($OrdenTrabajoArr) {1
    $ordenTrabajo=DatOrdentrabajo::buscarPorId($OrdenTrabajoArr['idorden']);2
    if ($ordenTrabajo->idestadoo==1){3
        if ($OrdenTrabajoArr['idestadoo']>1){4
            $unidad = $this->pIntegrator->vehiculo->Buscar_unidad($ordenTrabajo->idunidad);5
            $estado = empty($unidad['iddependencia'])?1:2;6
            $this->pIntegrator->vehiculo->cambiarEstadoMotivo($ordenTrabajo->idunidad, $estado, 0, '', false);7
        }8
        $ordenTrabajo->idestadoo=$OrdenTrabajoArr['idestadoo'];9
        $ordenTrabajo->fechainicio=Empty($OrdenTrabajoArr['fechainicio'])?Null:$OrdenTrabajoArr['fechainicio'];10
        $ordenTrabajo->fechaterminacion=Empty($OrdenTrabajoArr['fechaterminacion'])?Null:$OrdenTrabajoArr['fecha11
        $ordenTrabajo->tiempoparo=Empty($OrdenTrabajoArr['tiempoparo'])?Null:$OrdenTrabajoArr['tiempoparo']; 12
        $ordenTrabajo->idtipofalla=Empty($OrdenTrabajoArr['idtipofalla'])?Null:$OrdenTrabajoArr['idtipofalla'];13
        $ordenTrabajo->idcausafalla=Empty($OrdenTrabajoArr['idcausafalla'])?Null:$OrdenTrabajoArr['idcausafalla14
        $ordenTrabajo->observacion=$OrdenTrabajoArr['observacion'];15
        $ordenTrabajo->save();16
        if ($ordenTrabajo->tipotrabajo == 0){17
            if (!Empty($OrdenTrabajoArr['idmemorandum'])){18
                $modelo = new DatOrdentrabajomemorandumModel();19
                $modelo->guardar($OrdenTrabajoArr);20
                if ($OrdenTrabajoArr['idestadoo']==2){21
                    $modelo = new DatMemorandumModel();22
                    $modelo->modificarEstado($OrdenTrabajoArr['idmemorandum']);22
                }23
            }24
        }25
    }26
```

Imagen 25. Código fuente de la funcionalidad guardar ($OrdenTrabajoArr).

```
        else{15
            $modelo = new DatTipomantordentrabajoModel();16
            $modelo->guardar($OrdenTrabajoArr);16
            if ($OrdenTrabajoArr['idestadoo']==2){17
                $this->pIntegrator->vehiculo->programar($OrdenTrabajoArr['idunidad']
            }19
        }20
        $modelo = new DatOtrostrabajosModel();21
        if (isset($OrdenTrabajoArr['trabajosEliminados'])){22
            foreach($OrdenTrabajoArr['trabajosEliminados'] as $idotrostrabajos){23
                $modelo->eliminar($idotrostrabajos);24
            }25
        }26

        if (isset($OrdenTrabajoArr['trabajos'])){27
            foreach($OrdenTrabajoArr['trabajos'] as $trabajos){28
                $trabajos['idorden']=$ordenTrabajo->idorden;29
                $modelo->guardar($trabajos);30
            }31
        }32
        $this->conn->commit();33
    }34
    else 35
        throw new ZendExt_Exception('MTTOOTO01');36
}37
```

Imagen 26. Código fuente de la funcionalidad guardar ($OrdenTrabajoArr).

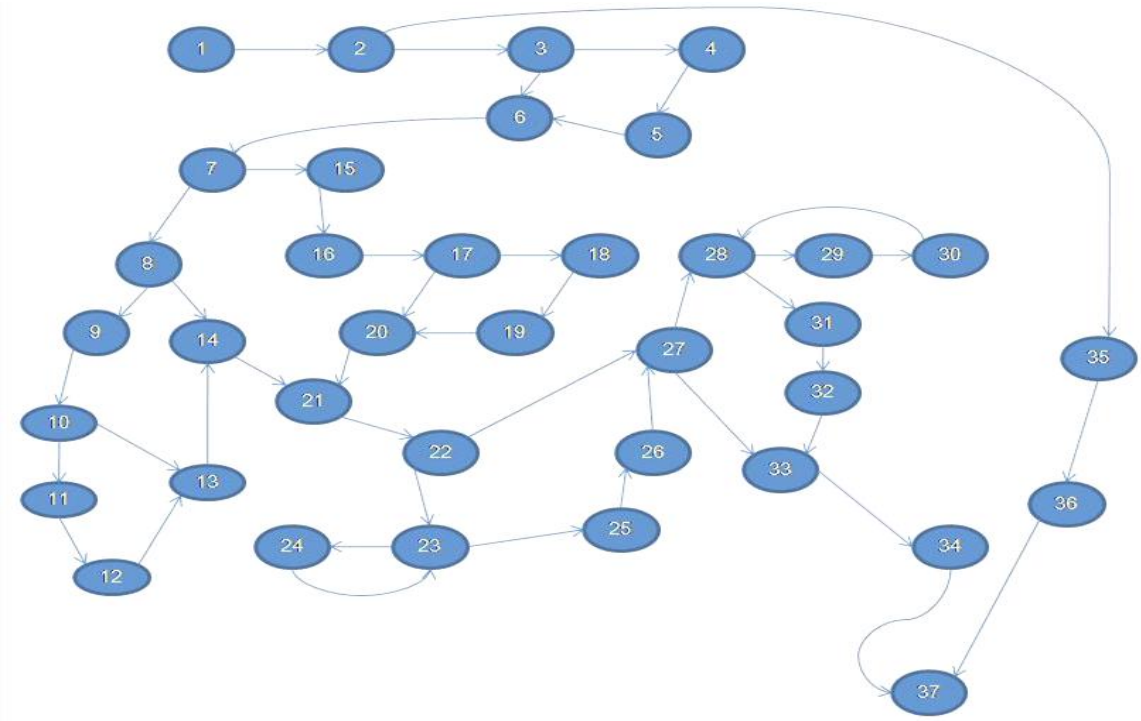


Imagen 27. Grafo de flujo asociado al algoritmo Guardar ($OrdenTrabajoArr).

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

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Pressman, 2005).

La complejidad ciclomática de un código se puede calcular de tres maneras diferentes.

Para calcular la complejidad ciclomática del método Guardar ($OrdenTrabajo) fueron utilizadas estas tres formas para lograr una amplia verificación de los resultados.

Las fórmulas para realizar dicho cáculo son:

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

V (G)= (46 -37)+2

V (G)= 11

Siendo A la cantidad total de aristas del grafo y N la cantidad de nodos.

2. V (G) = P +1

V (G) = 10+1

V (G) = 11

Siendo P la cantidad de nodos predicado (son aquellos de los cuales parten dos o más aristas)

3. V (G) = R

V (G) = 11

Siendo R la cantidad de regiones que posee el grafo.

En cada una de las fórmulas V (G) representa el valor del cálculo. Según los resultados obtenidos en cada uno de estos cálculos se puede concluir que la complejidad ciclomática del código analizado es 11, determinándose a su vez que existen once caminos posibles por donde puede circular el flujo y que esta misma cantidad representa el límite superior de casos de prueba que se le pueden aplicar a dicho código.

A continuación se muestran los caminos básicos por donde puede circular el flujo.

Camino básico # 1: 1-2-35-36-37.

Camino básico # 2: **1-2-3-4-5-6-7-15-16-17-18-19-20-21-22-27-33-34-37.**

Camino básico # 3: 1-2-**3-6**-7-15-16-17-18-19-20-21-22-27-33-34-37.

Camino básico # 4: 1-2-3-4-5-6-7-15-16-**17-20**-21-22-27-33-34-37.

Camino básico **#** 5: …**27-28-29-30-28**-31-32-33-34-37.

Camino básico # 6: ...**27- 28-31**-32-33-34-37.

Camino básico # 7: ...**22-23-25**-26-27-...37.

Camino básico **#** 8: …**22-23-24-23**-…37.

Camino básico **#** 9: …**7-8-14-21**-22-…37.

Camino básico **#** 10: …**8-9-10-13**-14-21-…37.

Camino básico # 11: 1-2-3-4-5-6…**10-11-12-13**-14-21-...37.

En la representación de los caminos básicos los tres puntos suspensivos (…) empleados se utilizan para abreviar, ya que cualquier combinación de caminos utilizada es aceptable.

Para cada uno de los caminos obtenidos se realiza un caso de prueba. Los casos de prueba realizados son los siguientes:

3.3.1.1 Caso de prueba para el camino básico #1.

Camino básico # 1: 1-2-35-36-37.

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

El parámetro $OrdenTrabajoArr no está vacío, contiene todos los datos necesarios para modificar una Orden de trabajo en estado cancelada o terminada, por lo que no se puede modificar la Orden de trabajo.

Entrada: $OrdenTrabajoArr = [[idorden] => 90000000003, [idunidad] => 90000000005, [idgrupo] => 9000000, [idtipomantenimiento] => , [idmemorandum] =>, [tiempoparo] =>, [fechainicio] =>28/04/2012, [fechaterminacion] => 28/04/2012, [idtipofalla] =>, [idcausafalla] =>, [idestadoo] => 2, [observacion] =>, [horas] =>, [minutos] =>, [completado] => 0, [trabajos] => Array([0] => Array([idotrostrabajos] =>

90000000007, [denom] => aasasasasas, [horas] => 0, [completado] => 0, [minutos] => 0)), [trabajosEliminados] => Array()].

Resultados esperados: Se lanza el mensaje “Imposible modificar una Orden de trabajo terminada o cancelada”.

3.3.1.2 Caso de prueba para el camino básico #2.

Camino básico # 2: 1-2-3-4-5-6-7-15-16-17-18-19-20-21-22-27-33-34-37.

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

El parámetro $OrdenTrabajoArr no está vacío, contiene todos los datos de una Orden de trabajo determinada, exceptuando los trabajos y los trabajos eliminados en la Orden. Además este parámetro incluye un nuevo estado de la Orden, para guardar los datos modificados en la Orden de trabajo.

Entrada: $OrdenTrabajoArr = [[idorden] => 90000000010, [idunidad] => 90000000010, [idgrupo] => 9000000, [idtipomantenimiento] =>, [idmemorandum] =>, [tiempoparo] =>00:15, [fechainicio] => 29/04/2012, [fechaterminacion] => 29/04/2012, [idtipofalla] =>, [idcausafalla] =>, [idestadoo] => 2, [observacion] => resuelto, [horas] =>, [minutos] =>, [completado] => 0].

Resultados esperados: Se espera que sea modificada una Orden de trabajo en el sistema, la cual no tendrá trabajos asociados.

3.3.1.3 Caso de prueba para el camino básico #3.

Camino básico # 3: 1-2-**3-6**-7-15-16-17-18-19-20-21-22-27-33-34-37.

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

El parámetro $OrdenTrabajoArr no está vacío, contiene todos los datos de una Orden de trabajo determinada, exceptuando los trabajos y los trabajos eliminados en la Orden. Además este parámetro incluye el estado de la Orden, la cual sigue siendo abierta, para guardar los datos modificados en la Orden de trabajo.

Entrada: $OrdenTrabajoArr = [[idorden] => 90000000010, [idunidad] => 90000000010, [idgrupo] => 9000000, [idtipomantenimiento] =>, [idmemorandum] =>, [tiempoparo] =>00:15, [fechainicio] => 29/04/2012, [fechaterminacion] => 29/04/2012, [idtipofalla] =>, [idcausafalla] =>, [idestadoo] => 1, [observacion] => resuelto, [horas] =>, [minutos] =>, [completado] => 0].

Resultados esperados: Se espera que sea modificada una Orden de trabajo en el sistema, la cual se mantiene en estado abierta y no tiene trabajos asociados.

Caso de prueba para el camino básico #4.

3.3.1.4 Camino básico # 4: 1-2-3-4-5-6-7-15-16-**17-20**-21-22-27-33-34-37.

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

El parámetro $OrdenTrabajoArr no está vacío, contiene todos los datos de una Orden de trabajo determinada, exceptuando los trabajos y los trabajos eliminados en la Orden. Además este parámetro incluye el estado de la Orden, el cual será distinto al estado terminada, para guardar los datos modificados en la Orden de trabajo.

Entrada: $OrdenTrabajoArr = [[idorden] => 90000000010, [idunidad] => 90000000010, [idgrupo] => 9000000, [idtipomantenimiento] =>, [idmemorandum] =>, [tiempoparo] =>00:15, [fechainicio] => 29/04/2012, [fechaterminacion] => 29/04/2012, [idtipofalla] =>, [idcausafalla] =>, [idestadoo] => 1, [observacion] => resuelto, [horas] =>, [minutos] =>, [completado] => 0].

Resultados esperados: Se espera que sea modificada una Orden de trabajo en el sistema, la cual cambiará su estado a cualquier estado distinto al estado terminada y no tendrá trabajos asociados.

3.3.1.5 Caso de prueba para el camino básico #5.

Camino básico # 5:..**27-28-29-30-28**-31-32-33-34-37.

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

El parámetro $OrdenTrabajoArr no está vacío, contiene todos los datos de una Orden de trabajo determinada, la cual tiene trabajos asociados. Además este parámetro no incluye los trabajos eliminados en la Orden.

Entrada: $OrdenTrabajoArr = ([idorden] => 90000000008, [idunidad] => 90000000009, [idgrupo] => 9000000, [idtipomantenimiento] =>, [idmemorandum] =>, [tiempoparo] =>, [fechainicio] =>, [fechaterminacion] =>, [idtipofalla] =>, [idcausafalla] =>, [idestadoo] => 1, [observacion] =>, [horas] =>, [minutos] =>, [completado] => 0, [trabajos] => Array([0] =>([idotrostrabajos] => 90000000011, [denom] => lavado, [horas] => 1, [completado] => 0, [minutos] => 15), [1] =>([denom] => reparación, [horas] => 6, [completado] => 0, [minutos] => 8)))

Resultados esperados: Se espera que sea modificada una Orden de trabajo en el sistema, la cual tendrá trabajos asociados.

3.3.1.6 Caso de prueba para el camino básico #6.

Camino básico # 6: ...**27-28-31**-32-33-34-37.

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

El parámetro $OrdenTrabajoArr no está vacío, contiene todos los datos de una Orden de trabajo determinada, la cual no tiene trabajos asociados. Además este parámetro no incluye los trabajos eliminados en la Orden.

Entrada: $OrdenTrabajoArr = ([idorden] => 90000000008, [idunidad] => 90000000009, [idgrupo] => 9000000, [idtipomantenimiento] =>, [idmemorandum] =>, [tiempoparo] =>, [fechainicio] =>, [fechaterminacion] =>, [idtipofalla] =>, [idcausafalla] =>, [idestadoo] => 1, [observacion] =>, [horas] =>, [minutos] =>, [completado] => 0, [trabajos] => Array ( )).

Resultados esperados: Se espera que sea modificada una Orden de trabajo en el sistema, a la que no se le han asociado trabajos.

3.3.1.7 Caso de prueba para el camino básico #7.

Camino básico # 7: ...**22-23-25**-26-27-...37.

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

El parámetro $OrdenTrabajoArr no está vacío, contiene todos los datos de una Orden de trabajo determinada, la cual no tiene trabajos eliminados. Además este parámetro no incluye los trabajos en la Orden.

Entrada: $OrdenTrabajoArr = ([idorden] => 90000000008, [idunidad] => 90000000009, [idgrupo] => 9000000, [idtipomantenimiento] =>, [idmemorandum] =>, [tiempoparo] =>, [fechainicio] =>, [fechaterminacion] =>, [idtipofalla] =>, [idcausafalla] =>, [idestadoo] => 1, [observacion] =>, [horas] =>, [minutos] =>, [completado] => 0, [trabajosEliminados] => Array ( )).

Resultados esperados: Se espera que sea modificada una Orden de trabajo en el sistema, a la que no se le han asociado ni eliminado ningún trabajo.

3.3.2 Pruebas de caja negra

Las pruebas de caja negra, también denominadas prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca, se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca (Pressman, 2005).

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

Caso de prueba para el requisito Emitir Orden de trabajo.

Condiciones de ejecución.

- Se debe identificar y autenticar ante el sistema, además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar una unidad.
- Se debe haber registrado al menos una herramienta en el sistema.

Tabla 12. Caso de prueba para Emitir Orden de trabajo.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Emitir Orden de trabajo.	El sistema debe permitir emitir Orden de trabajo.	EP 1.1: Emitir Orden de trabajo introduciendo datos válidos.	– Se introducen los datos correctamente. – Se presiona el botón **Imprimir**. – Se presiona el botón **Aceptar**.
		EP 1.2: Emitir Orden de trabajo introduciendo datos inválidos.	– Se introducen los datos inválidos. – Se presiona el botón **Imprimir**.

		– Se muestra un mensaje de error. – Se presiona el botón **Aceptar**.
	EP 1.3: Emitir Orden de trabajo dejando campos vacíos.	– Se introducen los datos dejando campos en blanco. – Se presiona el botón **Imprimir**. – Se muestra un mensaje informando del error. – Se presiona el botón **Aceptar**.
	EP 1.3: Cancelar.	– Se introducen o no los datos. – Se presiona el botón **Cancelar**.

Para más información sobre el caso de prueba para el requisito Emitir Orden de trabajo, consultar el documento CG-SM-DR-041. Los casos de prueba elaborados para los restantes requisitos funcionales se encuentran en los documentos CG-SM-DR-400, CG-SM-DR-397, CG-SM-DR-398, CG-SM-DR-307, CG-SM-DR-311, CG-SM-DR-306, CG-SM-DR-309, CG-SM-DR-176, CG-SM-DR-353, CG-SM-DR-355, CG-SM-DR-354, CG-SM-DR-356, CG-SM-DR-358, CG-SM-DR-357, CG-SM-DR-045, CG-SM-DR-044, CG-SM-DR-325, CG-SM-DR-324, CG-SM-DR-323, CG-SM-DR-326, CG-SM-DR-327, CG-SM-DR-067, CG-SM-DR-401, CG-SM-DR-402, CG-SM-DR-403, CG-SM-DR-399, CG-SM-DR-405, CG-SM-DR-406, CG-SM-DR-404, CG-SM-DR-409, CG-SM-DR-410, CG-SM-DR-408, CG-SM-DR-407.

La aplicación desarrollada fue revisada y probada por Calisoft a partir de los diseños de los casos de prueba, donde se emitió un acta de liberación de software como resultado de las pruebas realizadas, mostrando buena calidad en las mismas. Ver Anexo11.

Ya liberada la aplicación, fue presentada el cliente, el cual mostró estar satisfecho con el producto realizado, quedando esto registrado en los avales otorgados por el mismo y por el gerente general del proyecto por parte de Cuba, en los cuales consta que el sistema cumple con la calidad requerida, dando solución a las necesidades de los clientes, ofreciendo ciertas ventajas para mejorar la gestión de los procesos que se realizan en el área de Transporte del CPNB. Para más información acerca de los avales emitidos consultar. Ver Anexo11.

3.4 Conclusiones parciales

Durante este capítulo se exponen las métricas utilizadas para la validación del diseño de la propuesta de solución, las que generaron como resultado que el diseño realizado era simple y con una calidad aceptable. Además se describieron y aplicaron las pruebas de caja blanca y caja negra, para evaluar el sistema desarrollado, las cuales mostraron que el sistema contaba con un adecuado funcionamiento, demostrando así el cumplimiento de las necesidades del cliente y la calidad requerida en el mismo.

CONCLUSIONES.

Una vez finalizado este trabajo de diploma se puede arribar a las siguientes conclusiones:

- Se realizó un estudio crítico de sistemas nacionales e internacionales que gestionan la ejecución del mantenimiento, demostrando la inexistencia de un sistema adecuado para implantarse en el área de Transporte del CPNB.
- Se elaboró el diseño del componente Ejecución, validando el mismo de acuerdo a las métricas planteadas, las cuales mostraron resultados satisfactorios en su aplicación.
- Se realizó la implementación del componente Ejecución, así como la aplicación de pruebas de software para su validación, dándole solución a las necesidades del área de Transporte del CPNB.
- Se logró la integración entre los procesos Ejecución del Mantenimiento y el proceso Administración de Inspecciones Técnicas, los cuales anteriormente se ejecutaban de forma independiente.
- Se contribuyó a mejorar la gestión de la ejecución del mantenimiento en el área de Transporte del CPNB, ya que anteriormente este proceso era desarrollado de forma manual, por lo que era muy difícil su adecuado control, evidenciando esto en el aval otorgado por los clientes tras haber probado y aceptado la aplicación.

RECOMENDACIONES

Se recomienda que en el desarrollo de futuros sistemas para la gestión del mantenimiento sea utilizado el presente trabajo como referencia en cuanto a los procesos relacionados con la ejecución del mantenimiento.

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 Policía Nacional Bolivariana.

BIBLIOGRAFÍA

Alfonso Alfonso, Inna. 2011. *Desarrollo del componente para la configuración visual de las excepciones en el marco de trabajo Sauxe.* 2011.

Anaya Multimedia. 2003. *La biblia Sevidor Apche.* 2003.

Andrés Vignaga, Daniel Perovich. 2003. *Enfoque Metodológico para el Desarrollo Basado en Componentes.* 2003.

Business Development Software S.L. 2011. Software y programación a medida. [En línea] 2011. [Citado el: 09 de Diciembre de 2011.] http://www.proyectosbds.com/software-y-programacion-a-medida/programacion-php-lamp-zf/mas-sobre-zend-framework/121/.

Citmatel. 2006. *SGestMant.* 2006.

Desarrollo de Software Basado en Componentes. **Montilva C., Jonás A. 2003.** s.l. : Universidad de Zulia, 2003.

Dirección Provincial Cultura, Matanzas, Cuba. 2007-2010. Atenas. [En línea] 2007-2010. http://www.atenas.cult.cu/rl/informatica/manuales/sl/introduccion_al_SL/book1.html.

Domínguez Medina, Lisdanay. 2010. *SISTEMA DE GESTIÓN DEL CAPITAL HUMANO.* 2010.

Empresa Nacional de Software, División Matanzas. 2011. Portal de Matanzas. [En línea] 2011. [Citado el: 02 de Diciembre de 2011.] http://www.expomatanzas.cu/empresa.php?emp=151&prd=249.

Garrett, Jesse James. 2005. Ajax: Un Nuevo acercamiento a las Aplicaciones Web. [En línea] 11 de 06 de 2005. http://www.willydev.net/InsiteCreation/v1.0/descargas/WillyDev_AJAX.pdf.

Grupo Anaya S.A. 2002. *La biblia de JavaScript.* s.l. : Anaya Multimedia, 2002.

Grupo Máximo. 2011. Transportex. [En línea] 2011. [Citado el: 02 de 12 de 2011.] http://www.transportex.net/informacion.html.

InteraSystem. 2011. InteraSystem. [En línea] 2011. [Citado el: 01 de Diciembre de 2011.] http://www.usa.interasystem.com/index.php/mnucaracteristicasproducto.

InterSystem. 2011. InterSystem. [En línea] 2011. [Citado el: 01 de Diciembre de 2011.] http://www.usa.interasystem.com/.

Larman, Craig. 1999. *UML y Patrones.* Mexico : s.n., 1999. 970-17-0261-1.

Maestrosdelweb. 1997. Maestrosdelweb. [En línea] 1997. [Citado el: 06 de Diciembre de 2011.] http://www.maestrosdelweb.com/editorial/phpintro/.

Mairelys Fernández González, Osley Zorrilla Rivera. 2010. Biblioteca de la Universidad de las Ciencias Informática. [En línea] 2010. [Citado el: 16 de Marzo de 2012.] http://biblioteca2.uci.cu/.

Mancha, Universidad de Catilla La. 2011. UCLM. [En línea] 2011. [Citado el: 15 de Febrero de 2011.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

Microsoft. 2011. MSDN. [En línea] 2011. [Citado el: 04 de Diciembre de 2011.] http://msdn.microsoft.com/es-es/library/bb972268.aspx#ref01.

Monso, Martín Ramos. 2005. *Programación PHP.* Buenos Aires, Argentina : MP Ediciones S.A, 2005.

Morera, René Hernández. 2011. Componente para la configuración visual de los servicios de integración en el marco de trabajo Sauxe. Habana : s.n., 2011.

Mozillia firefox. 1998-2011. Mozillia firefox. [En línea] Mozilla Foundation, 1998-2011. [Citado el: 15 de 01 de 2012.] http://www.mozilla.org/es-ES/firefox/central/#highperformance.

netcraft. 2003-2012. netcraft. [En línea] 2003-2012. [Citado el: 14 de Diciembre de 2011.] http://news.netcraft.com/archives/category/web-server-survey/.

Pérez, Javier Eguíluz. 2009. LibrosWeb. *LibrosWeb.* [En línea] 2009. [Citado el: 06 de Diciembre de 2011.] http://www.librosweb.es/ajax/capitulo1.html.

PostgreSQL. 1996-2011. PostgreSQL. [En línea] 1996-2011. [Citado el: 10 de Diciembre de 2011.] http://www.postgresql.org/about/.

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

Sencha Inc. 2011. Sencha. [En línea] 2011. [Citado el: 09 de Diciembre de 2011.]

Systeam - Logística Empresaria. 2010. Systeam - Logística Empresaria. [En línea] Systeam, 2010. [Citado el: 25 de Enero de 2012.] http://www.systeam.com.ar/anago-index.htm.

Visual Paradigm. 1999-2011. Visual Paradigm. [En línea] 1999-2011. [Citado el: 12 de Diciembre de 2011.] http://www.visual-paradigm.com/product/vpuml/editions/community.jsp.

Zend Technologies Ltd. 2006 - 2011. Zend Framework. [En línea] 2006 - 2011. [Citado el: 08 de Diciembre de 2011.] http://framework.zend.com/about/overview.