Universidad de las Ciencias Informáticas
Facultad 3

Título: Desarrollo del módulo Tarjeta de combustible del Sistema XEDRO-ERP.

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

Autores: Roberto Bandera Gómez

Liannet Baez Fernández

Tutora: Ing. Saily Oliva Martínez

Co-Tutor: Ing. Boris Luis Correa Frías

La Habana, 2015

"Año 57 de la Revolución"

"El trabajo en equipo es la habilidad de trabajar juntos hacia una visión común, es el combustible que le permite a la gente común obtener resultados pocos comunes."

Andrew Carnegie

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los _____días del mes de _____ del año _____.

Autores:

_____ _____

Liannet Baez Fernández Roberto Bandera Gómez

Tutores:

_____ _____

Ing. Saily Oliva Martínez Ing. Boris Luis Correa Frías

DATOS DE CONTACTO

Ing. Saily Oliva Martínez

Ingeniera en Ciencias Informáticas, Universidad de las Ciencias Informáticas.

Correo electrónico: soliva@xetid.cu

Ing. Boris Luis Correa Frías

Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas.

Correo electrónico: cfrias@uci.cu.

A *mi ángel de la guarda, mi padre querido que me acompaña desde el cielo, te extraño mucho.*

A *mi mamá por ser mi ejemplo a seguir, por ser mi gran amiga, por darme tanto amor, cuidarme y quererme siempre como su niña chiquita, Te Amo, mamita.*

A *mi segundo padre Arístides por convertirme en la mujer que soy, por darme tanto amor y cariño.*

A *mi hermano Roberto por ser tan especial conmigo, te quiero Tico.*

A *mi bisabuela Aracelis por ser la mejor abuelita del mundo, Te Amo Chela.*

A *mis tías Yusleidis, Yaines, Yanicet y María Elena, por quererme como su hija y aconsejarme en cada etapa de mi vida.*

A *mis abuelos, tíos, primos y familia en general, gracias por tanto amor.*

A *mi novio Orlando por darme tanto amor, por ser mi amigo y apoyarme siempre en las buenas y malas.*

A *mis amigas Liset y Marieta por quererme más de lo que me merezco, gracias por su amistad.*

A *Marieta y su esposo Carlos por quererme como su hija y abrirme las puertas de su corazón.*

A *mis amigas, por luchar junto a mí para lograr este sueño que parecía inalcanzable, Leticia, Yariannis, Margarita, Lisbetty, Rosaidis, Aniuska, Yadira, Yanisley, Liliana y Dayani…*

A *mis todos mis profesores por convertirme en una mejor persona, con valores y sentimientos sinceros, gracias a todos.*

A *mi compañero de tesis por tener la suficiente paciencia para soportarme y por ayudarme a cumplir este sueño.*

Liannet Baez Fernández

A*gradezco a mi familia por siempre estar atenta a mis preocupaciones y ayudarme a alcanzar mis objetivos en la vida, a todas las personas que he conocido en la Universidad ya que de todos aprendí algo, a los profes que se encargaron de esta misión de la cual soy fruto, a las instructoras Maira y Esperanza, a la jefa de medios básicos Ana Delia y la profe Marieta Peña Abreu por su apoyo incondicional y sus buenos consejos en todo momento, por último y no menos importante a mi compañera de tesis por ser tan atenta e inseparable, muchas gracias, sin ustedes no hubiese sido posible.*

Roberto Bandera Gómez

A *nuestros tutores Saily y Boris por su dedicación y empeño para la realización de este trabajo. A Tamara, José Ernesto, Yoenry, Jesús, Alexis y demás miembros del proyecto. A todos los miembros del tribunal, por las recomendaciones y sugerencias, gracias por todo.*

***A** mi mamá por ser mi luz, mi ejemplo de mujer…*

***A** mis padres por convertirme en la mujer que soy.*

***A** mi bisabuela Aracelis, por ser la viejita más linda del mundo.*

***A** todos mis seres queridos que desde el cielo me guían y me cuidan.*

***A** todos mis amigos que han sido incondicionales conmigo.*

***A** mi familia en general por todo su apoyo.*

Liannet Baez Fernández

***D**edico esta victoria en primer lugar:*

***A** mis padres y hermanos Yanelis, Leo, Adrián y Yoandris por haberme apoyado en los buenos y malos momentos.*

***A** mi abuela Adís por el amor y el cariño que me ha dado.*

***A** mis exnovias Carmen y Nosdaly por haber entrado en mi vida y haberme convertido en una mejor persona.*

***F**inalmente a todos mis compañeros de estudio que por un motivo u otro no pudieron llegar a este momento.*

Roberto Bandera Gómez

RESUMEN

La gestión de combustible se basa específicamente en controlar aspectos relacionados con su distribución, uso y consumo. Actualmente en Cuba esta gestión se realiza a través de tarjetas magnéticas, por los beneficios que brindan con respecto al ahorro de combustible. En la Universidad de las Ciencias Informáticas (UCI) el procesamiento de las tarjetas magnéticas para combustible se realiza de forma manual. Este hecho da lugar a una serie de dificultades como la descentralización, duplicación y pérdida de la información que se procesa, evidenciándose desajustes contables en los balances de comprobación. Esto provoca lentitud en la tramitación de los procesos de las tarjetas de combustible. La presente investigación tiene como objetivo desarrollar un módulo dentro del sistema XEDRO-ERP que informatice los requisitos identificados en los procesos de las tarjetas de combustible realizados en la UCI, logrando mejorar su control. Para la creación de este módulo se estableció una metodología de desarrollo de software, tecnologías y herramientas necesarias para realizar la solución. Durante el desarrollo del trabajo fueron generados los artefactos correspondientes a las disciplinas definidas en la metodología. Estos artefactos fueron validados mediante técnicas, métricas de diseño y pruebas de software que demostraron que el módulo Tarjeta de combustible funciona correctamente. Como resultado se obtuvo un módulo que a través de funcionalidades y emisión de reportes gestiona los procesos de las tarjetas de combustible garantizando la centralización e integridad de la información para contribuir a su control en el menor tiempo posible.

PALABRAS CLAVE: combustible, control, procesos, tarjetas.

Índice de contenido

Índice de tablas

Índice de figuras

INTRODUCCIÓN

La gestión empresarial es la actividad que a través de diferentes individuos especializados, como: directores institucionales, consultores, productores y gerentes busca mejorar la productividad y la competitividad de una empresa o negocio. La gestión de portadores energéticos, es un subsistema de la gestión empresarial, que abarca las actividades de administración y aseguramiento de estos portadores (Pérez Del Pino, 2012). Entre los diferentes tipos de portadores energéticos que existen se encuentran: combustible (Diésel y Gasolina), electricidad, gas licuado y los lubricantes, siendo el combustible uno de los portadores más usado en la mayoría de las empresas.

La gestión de combustible se basa específicamente en controlar aspectos relacionados con su distribución, uso y consumo. Actualmente en Cuba esta gestión se realiza a través de las tarjetas magnéticas, mecanismo usado para la distribución y planificación de combustible. La implementación de este mecanismo en todas las entidades cubanas fue uno de los resultados alcanzados en el país a partir de la labor desempeñada por los trabajadores sociales, para apoyar el desarrollo de la Revolución Energética y de otros programas de la revolución.

Las empresas cubanas que usan las Tarjetas Magnéticas para Combustible (TMC), siguen un único procedimiento para realizar su control, aprobado en la Resolución No.60/2009 del Ministerio de Finanzas y Precio (Rodríguez, 2009). Dicho procedimiento tiene como objetivo establecer las disposiciones que normen el pedido, abastecimiento y consumo de combustible para las unidades vehiculares de las empresas.

Las Tecnologías de la Información y las Comunicaciones (TIC) dentro del ámbito empresarial, han propiciado el surgimiento de nuevos sistemas de gestión. Cuba no está exenta de esto, ejemplo de ello es la Universidad de las Ciencias Informáticas (UCI). Para el desarrollo de software la UCI cuenta con varios centros de producción, entre los que se encuentra el Centro de Informatización de Entidades (CEIGE), el cual desarrolla un sistema integral de gestión denominado XEDRO-ERP. Este sistema tiene como objetivo, apoyar la soberanía tecnológica por la que aboga el país y satisfacer las necesidades de las empresas cubanas informatizando sus procesos y actividades.

Para el cumplimiento de sus objetivos y metas la UCI demanda una gran cantidad de recursos materiales, energéticos y financieros, destacando el combustible como uno de los fundamentales debido a que la entidad lo requiere para el traslado de especialistas hacia las empresas clientes, eventos científicos y transporte obrero. Por la gran cantidad de personal que diariamente se transporta en la universidad, la UCI tiene designados alrededor de 115 vehículos automotores que consumen gasolina y 147 vehículos consumidores de combustible Diésel; existiendo alrededor de 514 tarjetas magnéticas para combustible, según resolución. Estas tarjetas magnéticas son

administradas por las áreas de Finanzas, Caja y Contabilidad de la universidad, con el fin de llevar el control de consumo de combustible asignado. Sin embargo en la ejecución del procesamiento de las tarjetas de combustible se han identificado las siguientes dificultades que afectan su control en la entidad:

- Aunque la universidad cuenta con un sistema informático denominado Assets[1] para apoyar sus procesos contables; los procedimientos de carga, entrega, liquidación y control de las tarjetas de combustible están parcialmente informatizados debido a que se emplean herramientas ofimáticas como Excel y Word para su realización. Esto trae consigo que la información que se obtiene en cada uno de estos procesos antes mencionados se encuentre almacenada en documentos y resulte difícil obtener reportes de la misma.
- La carga de las tarjetas de combustible se realiza en el área de Finanzas, mientras que la entrega y liquidación se efectúan en el área de Caja, almacenándose en ambos lugares los registros correspondientes, por lo que se evidencia descentralización en la información.
- Para intentar llevar la información contable actualizada tanto en formato duro (papel) como digital se registran de forma manual los asientos contables de los procesos de las tarjetas de combustible en el área de Contabilidad, así como en el módulo de contabilidad del sistema Assets, trayendo como resultado que existan desajustes contables en los balances de comprobación al tener información duplicada.
- Actualmente la información generada por los procesos de las tarjetas de combustible es vulnerable al robo, deterioro y alteración, los que constituyen factores que atentan contra la integridad de la misma.

Teniendo en cuenta lo anterior se plantea como **problema científico:** ¿Cómo mejorar el control de los procesos de las tarjetas magnéticas para combustible en la Universidad de las Ciencias Informáticas?

Se propone como **objeto de estudio**: Los procesos para el tratamiento de las tarjetas magnéticas para combustible. Y el **campo de acción** se enmarca en: Los procesos para el tratamiento de las tarjetas magnéticas para combustible en la Universidad de las Ciencias Informáticas.

Para la solución del problema planteado se define como **objetivo general**: Desarrollar un módulo en

[1] **ASSETS:** es un Sistema de Gestión Integral estándar y parametrizado, que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos Fijos, Útiles y Herramientas y Recursos Humanos.

el sistema XEDRO-ERP que contribuya al mejoramiento del control de los procesos de las tarjetas magnéticas para combustible en la Universidad de las Ciencias Informáticas.

Para dar cumplimiento al objetivo general planteado, se proponen los siguientes **objetivos específicos**:

- Elaborar la fundamentación teórica de la investigación para establecer las bases teóricas de la misma.
- Realizar el análisis y diseño del módulo Tarjeta de combustible del sistema XEDRO-ERP para poder modelarlo soportando los requisitos identificados.
- Implementar las funcionalidades del módulo Tarjeta de combustible del sistema XEDRO-ERP en términos de componentes.
- Validar la propuesta de solución mediante las pruebas de caja blanca, caja negra y de aceptación.

Se plantea como **idea a defender**: El desarrollo del módulo Tarjeta de combustible del Sistema XEDRO-ERP permitirá mejorar el control de los procesos de las tarjetas magnéticas para combustible en la Universidad de Ciencias Informáticas.

Métodos de investigación

Métodos teóricos:

- Analítico-sintético: Se empleó en el análisis de la documentación confiable referente al tema de investigación, permitiendo identificar los elementos claves para la compresión de los procesos para el tratamiento de las tarjetas magnéticas para combustible. También se utilizó para el estudio de las herramientas, lenguajes, modelo de desarrollo, tecnologías y sistemas informáticos que gestionen los procesos de tratamiento de tarjetas magnéticas para combustible, con el fin de determinar los elementos más indicados a utilizar en la propuesta de solución.

- Modelación: Se utiliza este método para modelar los componentes esenciales correspondientes al ciclo de vida del software, específicamente en la fase de análisis y diseño de la metodología seleccionada.

- Inductivo – Deductivo: Se utilizó en el estudio de sistemas informáticos que gestionen los procesos de tratamiento de las TMC, para poder hacer generalizaciones a partir de estos casos particulares por inducción que sirvan para confirmar teorías , y de estas teorías poder deducir conclusiones sobre dichos casos particulares que pueden ser verificados en la práctica.

Métodos empíricos:

- Entrevista: Se realizó con el objetivo de consultar con expertos sobre temas relacionados con la gestión de las TMC además de obtener conocimiento acerca de las necesidades y exigencias que debe cumplir el sistema a desarrollar.
- Encuesta: Se utilizó con el fin de identificar las dificultades que trae consigo realizar de forma manual la gestión de los procesos de las TMC.

Estructura del documento:

Capítulo 1. Fundamentación teórica

En este capítulo se describen los principales conceptos relacionados con el procedimiento para el tratamiento de las tarjetas magnéticas para combustible. Se expone una valoración de los sistemas informáticos que gestionan procesos relacionados con las tarjetas magnéticas para combustible. Además se realiza la fundamentación de las herramientas, técnicas, lenguajes y metodología de desarrollo a utilizar para dar solución a la problemática.

Capítulo 2. Modelado del negocio, Análisis y Diseño

En este capítulo se realiza un estudio del negocio y de las necesidades de información a través de la disciplina modelado de negocio. Se definen los requisitos funcionales y no funcionales, describiéndose cada uno de ellos y se exponen las técnicas utilizadas para validarlos. Además, se elabora una propuesta de solución en términos de componentes, diagramas y otros artefactos que guían la implementación. Se detallan los distintos patrones utilizados para el diseño del módulo y se muestran los resultados obtenidos tras la validación del diseño a través de las métricas Tamaño Operacional de Clases y Relaciones entre Clases.

Capítulo 3. Implementación y validación de la solución propuesta

Este capítulo se basa en los resultados obtenidos del diseño del módulo. Se describen aspectos relacionados con la implementación de la solución como: su integración con el sistema XEDRO-ERP, los estándares de codificación y tratamientos de errores empleados. Por último, se reflejan los resultados obtenidos de las pruebas de caja blanca, caja negra y de aceptación, además la validación de la propuesta de solución.

CAPÍTULO1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se abordan los fundamentos teóricos asociados a la investigación y se enuncian las definiciones y conceptos relacionados con el tratamiento de las TMC. Se realiza un estudio de varios sistemas informáticos que informatizan de una forma u otra los procesos que se realizan con las TMC, donde se hace un análisis de sus principales características en busca de elementos que puedan ser reutilizados en la propuesta de solución. Por último se justifican las tecnologías, metodología, los lenguajes, técnicas y herramientas que fueron usadas en la solución propuesta.

1.1. Principales conceptos

Con el objetivo de profundizar en el conocimiento del tema en el cual se enmarca la investigación, se describen a continuación los principales conceptos (López, 2013):

- **Portadores energéticos:** Son recursos naturales o artificiales que pueden ser procesados o transformados en fuentes de energía según la necesidad del ser humano, esta transformación puede tener varios usos, por ejemplo, como combustible para los vehículos (Energéticos, 2008).
- **Combustible:** Cualquier material capaz de liberar energía cuando se oxida de forma violenta con desprendimiento de calor.
- **Tarjeta magnética para combustible:** Pieza rectangular, de plástico, dotada con componentes electrónicos que la convierten en una tarjeta inteligente para ser usada como medio de pago, conteniendo todos los datos pertinentes a esta función. O sea, es una tarjeta magnetizada la cual se carga en establecimientos de FINCIMEX, S.A, es decir, se registra en ella, por medio de una celda magnética, una cantidad determinada de combustible. Las mismas constituyen valores en moneda libremente convertible, equivalentes a su contrapartida material.
- **Comprobantes de pago:** Son los vales que documentan el consumo en cantidad e importe por la compra del combustible, los cuales son proporcionados por los servicentros.

1.2. Marco legal

Para el uso de las TMC en las empresas cubanas, se estableció la Resolución No. 60/2009 (Rodríguez, 2009) que norma el registro contable y control interno, a partir del nivel de utilización en la economía nacional. La resolución tiene como objetivos:

- Aprobar la Norma Específica de Contabilidad No. 4 “Control de las Tarjetas Prepagadas

para Combustible" (NEC. 4), la que se integra la Sección II del Manual de Normas Cubanas de Información Financiera y que como Anexo No. 1 consta de tres (3) páginas, formando parte integrante de esta Resolución.

- Aprobar el Procedimiento de Control Interno No. 3 "Elementos claves para el Control de las Tarjetas Prepagadas para Combustible" (PCI No. 3), el que se integra a la Sección de Procedimientos del Manual de Normas de Control Interno y que como Anexo No. 2 consta de ocho (8) páginas, formando parte integrante de esta Resolución.

1.3. Sistemas informáticos estudiados

Con el avance de las tecnologías y la informática se han ido materializando sistemas informáticos que gestionan los procedimientos para el control de las tarjetas magnéticas para combustible. A continuación se describen algunos de estos sistemas estudiados:

Sistema de Control de Combustibles (SCC)

El Sistema de Control de Combustibles, tiene como misión incrementar la eficiencia y eficacia en la administración de la información asociada al consumo de combustible de los vehículos y maquinarias, que prestan servicios de apoyo a las labores del proceso extractivo de mineral. Desarrollado por Opendat, empresa chilena de Ingeniería de Sistemas, dedicada a asesorar y desarrollar proyectos de informática de gran envergadura.

Estructuralmente el sistema está conformado por cuatro módulos:

- Referencias
- Procesos
- Consultas y Reportes
- Administración del sistema

En el módulo Procesos están las acciones relacionadas con el proceso de cierre de mes. Comenzando con la carga del archivo de consumos (detalle de las cargas de combustible), en esta carga todos los registros son validados respecto de la información de referencia existente, los consumos que no estén correctos, quedan en una etapa de validación y corrección. El ingreso de los valores de combustibles, los valores de los gastos relacionados, y los tipos de cambio, también son ingresados en este módulo. El proceso de cierre es por etapas, entregando información relativa a lo procesado y posibles problemas encontrados, toda la información relacionada con el cierre de mes queda bloqueada para evitar su modificación.

El módulo Consultas y Reporte permite la generación de consultas a la Base de Datos, tanto de los

consumos históricos como la información de cierre de mes. Todas las consultas realizadas en pantalla pueden generar un reporte a papel, y algunas pueden generar un archivo excel con el resultado de la consulta.

SCC es una aplicación web que utiliza el Sistema Gestor de Base de Datos Oracle y sólo es soportada por el sistema operativo Windows.

Sistema para el Control de Combustible por tarjetas magnéticas

El sistema para el Control de Combustible por tarjeta magnética es una aplicación diseñada por especialistas de la Casa Consultora DISAI, de la provincia de Villa Clara. Aplicado en instituciones del Grupo Empresarial de Industria Sidero-Mecánica (Gesime). La aplicación tiene como objetivo establecer un mejor control sobre el consumo de los combustibles (Diésel y Gasolina), a partir de los chips asociado a las diferentes tarjetas magnéticas y equipos de transporte con que cuenta la entidad. Con ella se logra automatizar de forma ágil el procedimiento del control de combustible por tarjetas, y los datos que brinda pueden servir para la toma de decisiones y planificación de esta actividad en función de los intereses de la dirección.

Con la implantación de este sistema se puede conocer:

- El consumo asociado a cada centro de costo de la entidad diferenciado por tipo de combustible y tipo de moneda.
- Los datos de cada tarjeta: saldo inicial, cargue, consumo y saldo final, agrupados por tipo de combustible y tipo de moneda.
- El consumo de los diferentes equipos de transporte que incluye: la fecha del consumo, el importe, la tarjeta utilizada, el destino y los kilómetros que debió recorrer.
- La fecha e importe, solamente de las tarjetas que se cargaron, agrupadas por tipo de combustible y moneda.
- El comprobante contable asociado a la actividad para el área económica.
- Un submayor por tarjeta.
- El combustible gastado en cada actividad realizada.
- Tarjeta de estiba de entrada y salida del combustible.
- De forma gráfica se pueden ver los consumos mensuales y gastos por concepto de compra de los combustibles.

Como información técnica se tiene que es una aplicación de escritorio desarrollada sobre el lenguaje de programación Visual Basic 6.0, Sistema Gestor de Bases de Datos Microsoft Access y solo es soportado por sistema operativo Windows (Windows 95 o superior) (DISAIC, 2008).

Sistema para Control de Combustible, Celador S2C

El sistema informático para el control de combustible Celador S2C ha sido desarrollado por la empresa de Desarrollo de Software (DESOFT) en Villa Clara, aplicado en las empresas Combinado Textil, Fábrica de Antenas, Artes Gráficas y Empresa de Calderas. Este permite controlar el consumo de los portadores energéticos relacionados con el combustible. Su objetivo principal es llevar el control del gasto de combustible por cada una de las unidades de costo, así como por vehículos. Además, emitir el comprobante contable de cada una de las operaciones que se ejecutan y el control del listado del submayor de cada tarjeta. Celador S2C es una aplicación de escritorio que utiliza un Sistema Gestor de Base de Datos SQL Server 2000 y sólo es soportada por el sistema operativo Windows (Celador, 2013).

Funcionalidades del sistema relacionadas con la problemática:

- Control de consumo de los equipos y vehículos
  - ✓ Genéricos de portadores
  - ✓ Portadores energéticos
  - ✓ Suministradores de combustible
  - ✓ Chóferes y responsables de tarjetas de combustible
  - ✓ Centros de costo
  - ✓ Unidades de costo
  - ✓ Cuentas
  - ✓ Chapas
- Control de tarjetas magnéticas
  - ✓ Introducir datos de la tarjeta
  - ✓ Cambios de portador de una tarjeta
  - ✓ Facturación de tarjetas
  - ✓ Activar y desactivar tarjetas
  - ✓ Ajustes a una tarjetas
  - ✓ Traspaso de saldo entre tarjetas
  - ✓ Ticket de consumo de combustible

Sistema de Control de Portadores Energéticos (SISCOPE)

Es un sistema desarrollado por la Universidad de Matanzas “Camilo Cienfuegos”, a solicitud de la Empresa de Proyectos de Arquitectura e Ingeniería (EMPAI) de Matanzas. El sistema tiene como principal objetivo ayudar a la planificación, control y toma de decisiones de los diferentes portadores

energéticos. El combustible como portador más importante para esta empresa cuenta con un módulo para su gestión dentro del sistema. El cual tiene como objetivo establecer un mejor control sobre el consumo de los combustibles (Diésel y Gasolina), a partir de las diferentes operaciones realizadas con las tarjetas magnéticas y equipos de transporte con que cuenta la entidad. Las operaciones que registra son las asociadas al proceso de compra y consumo de combustible, logrando así una mejor manipulación y control con las informaciones para facilitar la contabilización de este portador energético. Los datos y reportes de salidas se pueden utilizar para la toma de decisiones y planificación de esta actividad en función de los intereses de la dirección. Para su descripción y construcción se utilizó la metodología ágil de desarrollo de software Programación Extrema (XP), el ambiente de programación ASP.NET con el lenguaje C# y gestor de base de datos SQL Server Express (Pérez Del Pino, 2012).

Valoración de los sistemas estudiados

El análisis de los sistemas informáticos mencionados anteriormente se realizó con el objetivo de identificar buenas prácticas o elementos positivos que apoyen el desarrollo del módulo Tarjeta de combustible en el sistema XEDRO-ERP. Para dicho análisis se tuvieron en cuenta los siguientes indicadores: si incorporaban la gestión de los procesos de tratamiento de las tarjetas magnéticas para combustible en su negocio, tipo de tecnología sobre la cual están desarrollados y si se pueden integrar o no al sistema XEDRO-ERP.

Por tanto los resultados obtenidos a partir de los indicadores anteriormente definidos fueron:

En cuanto a la gestión de los procesos de tratamiento de las tarjetas magnéticas para combustible:

Los sistemas SCC, Celador S2C, SISCOPE y Sistema para el Control de Combustible por tarjetas magnéticas cuentan con funcionalidades que permiten gestionar parcialmente los procesos para el tratamiento de tarjetas magnéticas para combustible. Sin embargo los procesos de carga, entrega y liquidación de tarjetas magnéticas para combustible no se tienen en cuenta por la mayoría de estos sistemas, por lo que no cubren todas las necesidades definidas en la problemática.

En cuanto al tipo de tecnología sobre la cual están desarrollados:

Los sistemas SCC, Celador S2C, SISCOPE y Sistema para el Control de Combustible por tarjetas magnéticas son soportados por el sistema operativo Windows y exceptuando el sistema SISCOPE los demás utilizan un Sistema Gestor de Bases de Datos privativo. Por lo tanto se puede concluir que dichos sistemas usan tecnologías privativas, no ajustándose a las políticas de independencia

tecnológica por las que aboga la dirección del país.

En cuanto a su integración con el sistema XEDRO-ERP:

Los sistemas estudiados no cuentan con características compatibles con el sistema XEDRO-ERP como: la implementación de una arquitectura basada en componentes, la utilización del patrón IoC y de los componentes de seguridad y tecnología, donde estos componentes logran una administración segura y definen la plataforma tecnológica sobre la que se desarrollan, respectivamente. Concluyendo que ninguno de dichos sistemas cumple con las características requeridas para poder integrarse con el sistema XEDRO-ERP.

1.4. Metodología de desarrollo

La metodología de desarrollo de software utilizada fue la llamada "Metodología de desarrollo para la Actividad productiva de la Universidad de las Ciencias Informáticas". La cual tiene entre sus objetivos aumentar la calidad del software, basándose en el Modelo CMMI-DEV v1.3 y en la metodología "Proceso Unificado Ágil" (AUP por sus siglas en inglés).

AUP-UCI, como también se le conoce propone para el ciclo de vida de los proyectos 3 fases fundamentales (*Inicio, Ejecución y Cierre*) donde la propuesta de solución incidirá en la fase de:

Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto.

Este modelo propone siete (7) disciplinas, las cuales se desarrollan en la Fase de Ejecución, de ahí que se realicen iteraciones y se obtengan resultados incrementales. Las disciplinas por las que transita la solución son:

Disciplina Modelado del negocio

Destinada a comprender los procesos de negocio que se desean informatizar de una organización, para tener garantías de que el software desarrollado va a cumplir su propósito (Rodríguez Sánchez, 2014).

Los artefactos generados en esta disciplina son:

- Mapa de procesos
- Descripción de procesos de negocio
- Reglas de negocio

Disciplina Requisitos

Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto (Rodríguez Sánchez, 2014).

Los artefactos generados en esta disciplina son:

- Modelo conceptual
- Descripción de requisitos
- Especificación de requisitos de software
- Salida del sistema

Disciplina de Análisis y diseño

Se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales (Rodríguez Sánchez, 2014).

Los artefactos generados en esta disciplina son:

- Modelo de diseño

Disciplina Implementación

A partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, script, ejecutables y similares (Rodríguez Sánchez, 2014).

Disciplina Pruebas internas

Se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias (Rodríguez Sánchez, 2014).

Los artefactos generados en esta disciplina son:

- Diseño de casos de pruebas

Disciplina Pruebas de aceptación

Son realizadas principalmente por los usuarios con el apoyo del equipo de desarrollo. El propósito es verificar que el software está listo y que puede ser usado por los usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (Rodríguez Sánchez, 2014).

Los artefactos generados en esta disciplina son:

- Especificación de requisitos de software

1.5. Arquitectura del software

Con el objetivo de crear una infraestructura reutilizable que provea aspectos como seguridad,

auditoría, interoperabilidad, concurrencia, administración de transacciones y soporte para varios idiomas, el Departamento de Tecnología del centro CEIGE, desarrolló el marco de trabajo Sauxe. Este marco de trabajo actúa como base tecnológica para la construcción de aplicaciones web de gestión, siguiendo el paradigma de desarrollo orientado a componentes. Para la creación de los componentes que se desarrollan en dicho marco se implementa una arquitectura en capas, donde se hace uso del patrón arquitectónico y de diseño Modelo-Vista-Controlador, (Gómez Baryolo, 2012).

Para el desarrollo del módulo Tarjeta de combustible del sistema XEDRO-ERP se utiliza el marco de trabajo Sauxe 1.5 sobre el cual fue implementado dicho sistema. El cual posee un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo, (Gómez Baryolo, 2012). Para implementar la capa de presentación Sauxe utiliza ExtJS 2.2, se apoya en Zend Framework 1.9.7 para el desarrollo de la lógica del negocio y emplea para la capa de acceso a datos el ORM Doctrine 1.2. Además emplea como herramienta de seguridad el sistema Acaxia[2], el cual está desarrollado sobre software libre e incorpora procesos importantes como la administración de conexiones y de perfiles.

1.6. Tecnologías, lenguajes y herramientas para el desarrollo

Para la implementación de la propuesta se utilizan algunas de las herramientas y tecnologías propuestas por el Departamento de Tecnología del centro CEIGE (Bauta, 2011).

1.6.1. Lenguajes de programación

Un lenguaje de programación es el medio dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis. Se pone a disposición del programador para que este pueda comunicarse con los dispositivos de hardware y software del ordenador que posee. Consta de un léxico, una sintaxis y una semántica, según el número de instrucciones que requiera para realizar una tarea específica, se clasifican en lenguaje de alto nivel y lenguaje de bajo nivel (Labs, 2013).

Programación del lado del cliente

JavaScript 1.6: Es un lenguaje de programación multiplataforma, interpretado y orientado a objetos, lo que permite a los desarrolladores añadir y crear interactividad en el desarrollo y diseño de sitios

[2] **Acaxia:** Sistema de Gestión Integral de Seguridad.

web, así como la validación de datos (JavaScript, 2011). Su característica principal es que ayuda a mejorar el aspecto y la funcionalidad de una página web. La forma de emplear JavaScript en una página web es directamente dentro de las etiquetas HTML[3] (JavaScript, 2010).

JavaScript permite la programación de pequeños scripts y de programas más grandes orientados a objetos, con funciones y estructuras de datos complejas. Además, pone a disposición del programador todos los elementos que forman la página web para acceder a ellos y modificarlos dinámicamente. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado, es soportado por navegadores como: Internet Explorer, Netscape, Opera y Mozilla Firefox.

Programación del lado del servidor

PHP 5.3 (Hypertext Preprocessor): Es un lenguaje de programación que se ejecuta en el servidor, gratuito e independiente de la plataforma de desarrollo, por lo que puede ser utilizado en cualquier marco de trabajo o IDE de desarrollo que lo soporte. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Una de sus características más potentes es su compatibilidad con varias bases de datos, como MySQL, PostgreSQL y Oracle (PHP, 2011).

1.6.2. Tecnologías

JSON (JavaScript Object Notation): Es un formato ligero de intercambio de datos. Actualmente se ha convertido en un estándar en el desarrollo de aplicaciones web donde en ocasiones sustituye a XML[4] por permitir el intercambio de información entre el servidor y las funciones JavaScript (JSON, 2011).

Tiene muchas ventajas como la simplicidad, velocidad y facilidad de lectura. Permite una integración sencilla con otras herramientas mientras que proporciona un aumento en el rendimiento, además posibilita construir aplicaciones más rápidas e interactivas.

AJAX (JavaScript y XML asíncronos): Es una técnica empleada en el desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios, mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

Permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad y velocidad en las aplicaciones (Torres Salas, 2013).

[3] **HTML:** Acrónimo en inglés de Hyper Text Markup Language.

[4] **XML:** Acrónimo en inglés de Extensible Markup Language.

1.6.3. Lenguajes y herramienta CASE para el modelado

Notación de Modelado de Procesos de Negocio (BPMN[5]): Es un estándar basado en los diagramas de flujo, adaptado para suministrar una notación gráfica que describe la lógica de los pasos de un proceso de negocio a través de flujos de trabajo. Su principal objetivo es proveer una notación estándar que sea fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio. BPMN sirve de lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación (Introduction to BPMN, 2014).

Lenguaje Unificado de Modelado 2.1 (UML[6]): Es el lenguaje estándar para el modelado de software que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software (Jacobson, 2000). Posibilita el intercambio de modelos entre las distintas herramientas de modelado orientadas a objetos, constituyendo un lenguaje estándar al definir una notación y semántica común para todas estas herramientas. Este lenguaje se usa específicamente para realizar el diseño de la solución.

Visual Paradigm 8.0: Es una herramienta que utiliza UML, multiplataforma que soporta el ciclo de vida completo del desarrollo de software. Visual Paradigm permite dibujar todos los tipos de diagramas de clases y generar código inverso desde diagramas. Dentro de sus características fundamentales se encuentran que soporta la opción de usar BPMN. Posee como peculiaridad sobre el resto de las demás herramientas que cuenta con una potente funcionalidad para la creación de interfaces de usuarios de las aplicaciones (Paradigm, 2010). Puede ser integrado con el IDE de desarrollo Netbeans y con el gestor de Bases de Datos PostgreSQL.

1.6.4. Herramientas de programación

Netbeans 8.0: Es un entorno de desarrollo integrado (IDE), modular, de base estándar (normalizado), escrito en el lenguaje de programación Java. El proyecto Netbeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general (framework) para compilar cualquier tipo de aplicación. Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones Web, o para dispositivos móviles. Además puede instalarse en varios sistemas operativos: Windows, Linux, Mac OS (Netbeans, 2015).

[5] **BPMN:** Acrónimo en inglés de Business Process Modeling Notation.

[6] **(UML):** Acrónimo en inglés de Unified Modeling Language.

1.6.5. Herramienta para el control de versiones

Subversión 1.6.6: Es un sistema de control de versiones de código fuente abierto. Es software libre bajo una licencia de tipo Apache/BSD. Es un sistema centralizado para compartir información. Gestiona archivos, directorios y sus cambios a través del tiempo. Es un repositorio en forma de árbol con una jerarquía de directorios y ficheros. Entre sus principales características se encuentran: modificaciones atómicas, la creación de ramas y etiqueta. Tiene gran integración con los servidores apache, WebDav y DeltaV, también maneja eficientemente los archivos binarios (Subversión, 2015).

1.6.6. Sistema Gestor de Base de Datos

PostgreSQL 9.1: Es un sistema de gestión de bases de datos relacional orientado a objetos, el cual incluye características como herencia, restricciones, tipos de datos, reglas e integridad transaccional. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y uniones. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Utiliza el modelo cliente servidor y es un manejador de base de datos de código abierto liberado bajo la licencia BSD[7]. PostgreSQL está diseñado para administrar grandes volúmenes de datos (PostgreSQL, 2015).

1.6.7. Servidor Web

Apache 2.2: Es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos HTTP[8]. Es una tecnología gratuita de código abierto. Se ejecuta en varios sistemas operativos, característica que lo hace prácticamente universal. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es altamente configurable en la creación y gestión de logs[9] (Apache, 2015).

1.6.8. Navegador Web

Mozilla Firefox 3.6 o superior: Es un navegador libre y de código abierto. Es usado para visualizar páginas web. Incluye corrector ortográfico, búsqueda progresiva y marcadores dinámicos. Además se pueden añadir funciones a través de complementos desarrollados por terceros. Contiene el plugin Firebug que se utiliza para ver los errores del código. Es multiplataforma, realiza la navegación por pestañas, presenta compatibilidad para múltiples extensiones. Utiliza el sistema

[7] **BSD:** Acrónimo en inglés de Berkeley Software Distribution.

[8] **HTTP:** Acrónimo en inglés de Hypertext Transfer Protocol.

[9] **Log:** Registro oficial de eventos durante un periodo de tiempo.

SSL[10] para proteger la comunicación con los servidores web, utilizando fuerte criptografía cuando se utiliza el protocolo HTTPS[11] (Firefox, 2015).

1.7. Técnicas para la captura y validación de requisitos

La identificación de los requisitos es una actividad que se lleva a cabo fundamentalmente al inicio del desarrollo del sistema. En este proceso los analistas extraen de diferentes fuentes de información los datos que son necesarios para conocer las funcionalidades a desarrollar por el sistema mediante el empleo de técnicas. Además para la validación de los requisitos se emplean técnicas para poder demostrar que los requisitos identificados son los que el usuario realmente necesita.

Las técnicas para la **captura de requisitos** que se utilizan en la presente investigación son:

- **Entrevista:** Consiste en establecer una conversación entre personas de ambas partes para obtener información sobre el negocio y a partir de éstos se definen los requisitos (Toro, 2000).
- **Estudio de documentación:** Consiste en realizar una lectura basada en documentos sobre el dominio del negocio de la organización o sus prácticas profesionales. Ejemplos de algunos documentos que se pueden consultar son: manuales de usuarios del sistema actual, normativas y legislaciones (Andalucía, 2013).
- **Sistemas existentes:** Consiste en analizar distintos sistemas ya desarrollados que estén relacionados con el proceso que se intenta informatizar. Pueden ser analizadas las interfaces de usuario y las distintas salidas que los sistemas producen (listados, consultas, reportes), ya que pueden surgir nuevas ideas sobre la base de éstas (Moreno, 2009).

La técnica para la **validación de los requisitos** que se utiliza en la presente investigación es:

- **Prototipos:** Permite al usuario hacerse una idea de cómo sería el producto final una vez terminado, permitiendo descubrir con rapidez si el usuario se encuentra satisfecho, o no, con los requisitos (Moreno, 2009). Se hace uso en la investigación de prototipos del tipo evolutivos, donde una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final (Andalucía, 2013).

[10] **SSL:** (Security Socket Layer). Seguridad de la capa de transporte.

[11] **HTTPS:** (Hypertext Transfer Protocol Secure) Protocolo de transferencia de hipertexto seguro.

Patrones para la denominación de los requisitos

Existen patrones que pueden ser aplicados a los requisitos. Para la denominación de los requisitos funcionales de la solución se aplican los siguientes patrones:

- **El nombre revela la intención:** Este patrón nombra los casos de uso utilizando un verbo activo o frase que represente la meta del actor primario (Adolph Steve, 2001).
- **Preciso y legible:** Este patrón tiene como objetivo garantizar que cada caso de uso sea escrito lo suficientemente legible a fin de que los clientes los lean, los evalúen y precisen lo suficiente a fin de que entiendan los que los desarrolladores están construyendo (Adolph Steve, 2001).

1.8. Patrones para el diseño del sistema

Los patrones brindan una solución generalmente ya probada y documentada a problemas que se dan durante el proceso de desarrollo de software. Facilitan el trabajo al emplear un conjunto de buenas prácticas, además de presentar la ventaja de ser genéricos, al no depender de ningún lenguaje. Los patrones se dividen fundamentalmente en patrones arquitectónicos y de diseño.

Patrón arquitectónico

El patrón arquitectónico representa, en términos de componentes posibles soluciones para incorporar en el diseño, aspectos que mejoran la usabilidad del sistema final (Sánchez-Segura, 2010).

Patrón Modelo-Vista-Controlador (MVC): Patrón de arquitectura donde todo proceso está dividido en tres componentes de aplicación que separan la interfaz de usuario, el control de la lógica del negocio y el acceso a los datos en componentes diferentes (Gómez Baryolo, 2012).

- **Modelo:** contiene la principal funcionalidad y la información con la que trabaja la aplicación. Integra las clases que contienen las consultas a la base de datos, así como la definición de las tablas, sus relaciones y atributos.
- **Vista:** transforma el modelo en una página web que permite al usuario interactuar con esta, mostrándole así la información. Además determina la interfaz que se muestra finalmente al cliente para interactuar con la aplicación.
- **Controlador:** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista, dándole respuestas a las peticiones de los usuarios.

Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software, por lo que es de suma importancia aplicarlos en la construcción del diseño de un sistema. Estos patrones se dividen de manera general en dos categorías: patrones GRASP y GoF. A continuación se presentan los patrones de diseño que se utilizan para el diseño de la solución (Larman, 2004):

Patrones GRASP: Patrones que asigna responsabilidades a través de la descripción de los principios fundamentales del diseño y la solución a un problema.

- **Experto:** Asigna responsabilidades a la clase que se encarga de gestionar la información referente a un objeto determinado y es capaz de cumplir con la actividad asignada conservando el encapsulamiento y promoviendo clases sencillas, fáciles de comprender y mantener.
- **Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.
- **Bajo acoplamiento:** Es la medida en que cada una de las clases realiza actividades independientes, además de poseer un conocimiento de las actividades que realizan las otras clases del sistema permitiendo la reutilización.
- **Alta cohesión:** Asigna una responsabilidad de modo que la cohesión siga siendo alta. Se define que existe alta cohesión funcional cuando los elementos de un componente colaboran para producir algún comportamiento bien definido. El nivel de cohesión no debe ser independiente de otras responsabilidades ni de otros principios como los patrones Experto y Bajo Acoplamiento.
- **Controlador:** Se encarga de asignar la responsabilidad a una clase en el momento de manejar mensajes correspondientes a eventos en un sistema a clases específicas, facilitando la centralización de actividades. Esta clase es la responsable de manejar la lógica de negocio, decidiendo así que clase es la encargada de ejecutar una tarea determinada.

Patrones GoF: Los patrones de diseño pueden tener propósito creacional, estructural o de comportamiento (García, 2005):

- **Patrones creacionales:** Se encargan de las formas de crear instancias en los objetos.

Su objetivo es: abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados. Ejemplos de patrones de creación: Abstract Factory, Builder, Prototype y Singleton.

- **Patrones estructurales:** Están relacionados y se aprecia cómo las clases y los objetos se combinan para formar nuevas estructuras más complejas y proporcionar nuevas funcionalidades. Ejemplos de patrones estructurales: Adapter, Bridge, Proxy, Decorator y Facade.
- **Patrones de comportamiento:** Están relacionados con algoritmos y asignación de responsabilidades a los objetos. Describen no solamente patrones de objetos o clases, sino también patrones de comunicación entre ellos. Ejemplos de patrones de comportamiento: Chain of Responsibility, Mediator, Observer y Visitor.

1.9. Métricas para la validación del diseño

Las métricas basadas en clases según Lorenz y Kidd se dividen en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones, para luego promediar los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos de clases examinan la cohesión y asuntos relacionados con el código así como las métricas orientadas a los valores externos examinan el acoplamiento y la reutilización. Para la validación del diseño se seleccionaron las métricas Tamaño Operacional de Clase y Relaciones entre Clases que a continuación se describen:

- **Métrica Tamaño Operacional de las Clases (TOC):** La métrica TOC está dada por la cantidad de funcionalidades contenidas en las clases, a partir de las cuales se determina la afectación que ejerce en el diseño. La misma comprende los siguientes atributos de calidad (García Ramírez, 2013).
- **Responsabilidad:** Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.
- **Complejidad de implementación:** Grado de dificultad que tiene que implementar un diseño de clases determinado.
- **Reutilización:** Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.

A continuación se muestra una serie de tablas encaminadas a un mejor entendimiento de la

utilización de esta métrica.

Tabla 1: Métrica Tamaño Operacional de Clase

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

Tabla 2: Rango de valores para la métrica TOC.

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

Promedio: Sumatoria de la cantidad de operaciones por clase entre la cantidad de clases.

Métrica Relaciones entre Clases (RC): La métrica RC está dada por la cantidad de relaciones existentes entre las clases contenidas en el diseño, a partir de las cuales se determina la afectación que éstas ejercen dentro de la eficiencia del sistema. Los indicadores medidos por esta métrica son los siguientes (García Ramírez, 2013).

- **Acoplamiento:** Dependencia o interconexión de una clase o estructura de clase respecto a otras.
- **Cantidad de pruebas:** Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto (componente) diseñado.
- **Complejidad del mantenimiento:** Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la

planificación del proyecto.

- **Reutilización:** Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.

Esta métrica evalúa los siguientes atributos de calidad:

Tabla 3: Relaciones entre clases

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

Tabla 4: Rango de valores para la métrica RC

Atributo	Categoría	Criterio
Acoplamiento	Baja	1
	Media	2
	Alta	> 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

Promedio: sumatoria de la cantidad de relaciones de uso por clase entre la cantidad de clases.

1.10. Pruebas de software

Son esencialmente un conjunto de actividades dentro del desarrollo del software, que involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados. El único instrumento adecuado para determinar el estado de la calidad de un producto de software, es el proceso de pruebas. En este se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requisitos (Enciclopedia cubana, 2009).

Pruebas de caja blanca

Se conocen también como Prueba de Caja Transparente o de Cristal. Esta prueba consiste específicamente en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento. Dentro de la prueba de caja blanca se incluyen las Técnicas de Pruebas que serán descritas a continuación (Enciclopedia cubana, 2009):

- **Prueba del Camino Básico:** Permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos.
- **Prueba de Condición:** Ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método.
- **Prueba de Flujo de Datos:** Se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.
- **Prueba de Bucles:** Se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución de todos los bucles en sus límites operacionales.

Pruebas de caja negra

Se conocen también como Prueba de Caja Opaca o Inducida por los Datos. Se centran en lo que se espera de un módulo, estas pruebas se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo internamente, es decir, solo trabaja sobre su interfaz externa (Pressman, 2005).

En esencia permite encontrar:

- Funciones incorrectas o ausentes
- Errores de interfaz
- Errores en estructuras de datos o en accesos a las bases de datos externas

- Errores de rendimiento
- Errores de inicialización y de terminación

Existen diferentes técnicas de prueba de caja negra descritas por Pressman para validar la funcionalidad del sistema sin entrar a analizar su ejecución interna:

- Métodos de prueba basados en grafos
- Análisis de valores límites
- Tabla ortogonal
- Partición de equivalencia

Conclusiones parciales

Con el estudio de sistemas informáticos que gestionan los procesos de las tarjetas de combustible y el análisis de tecnologías, lenguajes y herramientas de desarrollo, se llegaron a las siguientes conclusiones:

- Los sistemas analizados no pueden ser utilizados ya sea por la plataforma que utilizan o por su gestión parcial de los procesos de las tarjetas de combustible.
- La selección de las tecnologías, herramientas y metodología a utilizar permitió establecer el entorno de desarrollo de la propuesta de solución.

CAPÍTULO 2: MODELADO DEL NEGOCIO, ANÁLISIS Y DISEÑO

Introducción

En el presente capítulo se describen los procesos de negocio a informatizar por el módulo Tarjeta de combustible del sistema XEDRO-ERP. Se elabora el modelo conceptual con los principales conceptos que se manejan en el tratamiento de las tarjetas magnéticas para combustible. A partir del uso de técnicas se identifican los requisitos funcionales y no funcionales que contendrá la solución propuesta, los cuales son descritos y validados. Se modela el diseño de la solución validándose la misma a través de las métricas definidas.

2.1. Modelado del negocio

El modelado del negocio tiene como objetivo describir los procesos existentes u observados, con el propósito de comprenderlos. Permite al analista capturar el esquema general y los procedimientos que gobiernan el negocio (Layola, 2006). En tal sentido en el presente epígrafe se muestran el mapa de procesos, la descripción del proceso de negocio y las reglas de negocio.

2.1.1. Mapa de procesos

Un mapa de procesos muestra cómo se relacionan los procesos de negocio, especificando las entradas y salidas de cada uno. Las entradas son los documentos que se necesitan para realizar el proceso y las salidas son la constancia de lo que se hizo. En la Figura 1 se representan los procesos relacionados con el negocio a modelar:

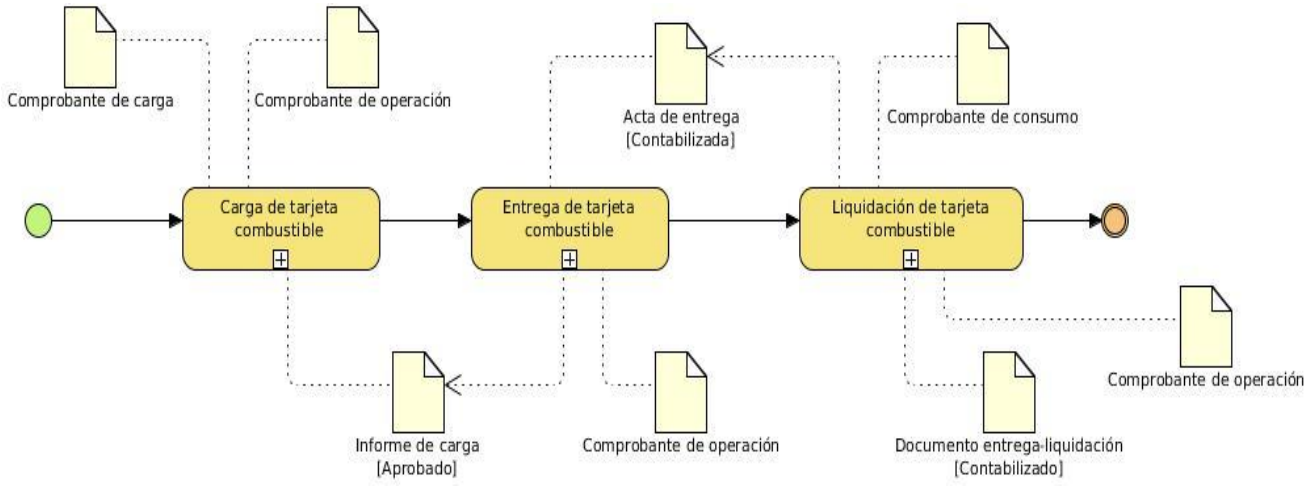

Figura 1: Mapa de procesos de tratamiento de las TMC

2.1.2. Descripciones de los procesos de negocio

Carga de tarjetas magnéticas para combustible

Es el proceso donde se realiza la inserción de una cantidad determinada de combustible en saldo (es decir en dinero) a una tarjeta magnética para que luego el chofer pueda extraerlo en los servicentros y abastecer de combustible el tanque del vehículo. Por cada tarjeta cargada se genera un comprobante de carga y al final de este proceso se genera un informe de carga con la relación de todas las tarjetas. Para realizar los asientos contables de la carga realizada, se genera en el área de Contabilidad de la entidad el comprobante de operaciones.

Para conocer la descripción del proceso de negocio Carga de tarjetas magnéticas para combustible consultar el Expediente de proyecto 3.4 del centro CEIGE (Baez Fernández, 2015).

Entrega de tarjetas magnéticas para combustible

El encargado del control en la entidad es el responsable de realizar la entrega de las tarjetas a partir de un listado de personas autorizadas a recoger las mismas, mediante un documento (Acta de entrega) que se utiliza para exigir la responsabilidad material de la tarjeta entregada. Además para poder realizar la entrega se debe consultar el informe de carga generado en el proceso Carga de tarjetas magnéticas para combustible, para comprobar que la tarjeta solicitada por la persona es la misma que tiene asignada en dicho informe. El proceso de entrega es personal e intransferible excepto en los casos en que se cuente con la correspondiente autorización del usuario de la misma.

Para conocer la descripción del proceso de negocio Entrega de tarjetas magnéticas para combustible consultar el Expediente de proyecto 3.4 del centro CEIGE (Baez Fernández, 2015).

Liquidación de tarjetas magnéticas para combustible

Es la suma total del consumo avalado mediante los comprobantes de consumo que emiten los servicentros en el mes en que fue consumido el combustible, el comprobante debe estar firmado por el chofer del vehículo y contener el número de chapa del auto que fue serviciado, para que el responsable del control realice las anotaciones correspondientes y los entregue en el área contable. Para la ejecución de este proceso es necesario consultar las Actas de entrega de las tarjetas magnéticas para combustible para verificar que los comprobantes de consumo entregados de una tarjeta estén firmados por el chofer al que se le asignó dicha tarjeta. Al final de este proceso se genera el documento que deja constancia de la entrega-liquidación de las tarjetas.

Para conocer la descripción del proceso de negocio Liquidación de tarjetas magnéticas para combustible consultar el Expediente de proyecto 3.4 del centro CEIGE (Baez Fernández, 2015).

2.1.3. Reglas de negocio

Los procesos de negocio se rigen por reglas que garantizan una adecuada ejecución de acuerdo a las estrategias, objetivos y filosofía de la organización; las reglas de negocio establecen los procedimientos que deben ser ejecutados y las condiciones que deben ser evaluadas y controladas en el flujo de proceso (Bizagi, 2015).

Tipos de reglas de negocio identificadas

Reglas Textuales: Contienen "instrucciones", se expresan de forma libre (no estructurada) en lenguaje natural.

Ejemplo:

- Las tarjetas para combustible son entregadas por el encargado del control, mediante un documento que se utiliza para exigir responsabilidad material si procede.

Reglas del modelo de datos: Engloba todas aquellas reglas que se encargan de controlar que la información básica almacenada para cada atributo o propiedad de una entidad u objeto sea válida.

Ejemplo:

- No hay saldos de tarjetas magnéticas para combustible negativos.
- La fecha de vencimiento de la tarjeta magnética debe ser 2 años después de encontrarse operativa.

Reglas de relación: Incluye todas aquellas reglas que controlan las relaciones entre los datos.

Ejemplo:

- Cuando se realiza la carga de las tarjetas magnéticas para combustible deberá generarse un comprobante de operaciones.

Las demás reglas definidas para el negocio en cuestión se encuentran en el Expediente de proyecto 3.4 del centro CEIGE (Baez Fernández, 2015).

2.2. Requisitos de software

Los requisitos de software son características que deben incluirse en un sistema o aplicación, se pueden clasificar según Sommerville en requisitos funcionales y requisitos no funcionales.

2.2.1. Requisitos funcionales

Los requisitos funcionales expresan la esencia y definen el funcionamiento del software, establecen como el sistema debe reaccionar a una entrada en particular y como debe comportarse ante tal situación, es el conjunto de funcionalidades que va a realizar la aplicación para automatizar un

proceso determinado (Sommerville, 2007).

El módulo Tarjetas de combustible del sistema XEDRO-ERP tiene como objetivo llevar el registro y control de las tarjetas magnéticas para combustible; además de gestionar los procesos de carga, entrega y liquidación de combustible asignado a las mismas. A través de las técnicas definidas para la captura de requisitos, se identificaron los siguientes requisitos funcionales:

Tabla 5: Requisitos funcionales del sistema

Código	Descripción	Código	Descripción
	Gestionar portador energético		**Gestionar acta de entrega**
Rf-#1	Adicionar portador energético	**Rf-#31**	Adicionar acta de entrega
Rf-#2	Modificar portador energético	**Rf-#32**	Modificar acta de entrega
Rf-#3	Eliminar portador energético	**Rf-#33**	Listar actas de entrega
Rf-#4	Listar portador energético	**Rf-#34**	Consultar acta de entrega
Rf-#5	Buscar portador energético	**Rf-#35**	Buscar acta de entrega
	Gestionar tipo de portador energético	**Rf-#36**	Realizar búsqueda avanzada del acta de entrega
Rf-#6	Adicionar tipo de portador energético	**Rf-#37**	Imprimir acta de entrega
Rf-#7	Modificar tipo de portador energético	**Rf-#38**	Imprimir listado de actas de entrega
Rf-#8	Eliminar tipo de portador energético	**Rf-#39**	Confirmar acta de entrega
Rf-#9	Listar tipos de portador energético	**Rf-#40**	Contabilizar acta de entrega
Rf-#10	Buscar tipo de portador energético	**Rf-#41**	Cancelar acta de entrega
	Gestionar tarjeta de combustible		**Gestionar documento de entrega-liquidación**
Rf-#11	Adicionar tarjeta de combustible	**Rf-#42**	Adicionar documento entrega-liquidación
Rf-#12	Modificar tarjeta de combustible	**Rf-#43**	Modificar documento entrega-liquidación

Rf-#13	Eliminar tarjeta de combustible	**Rf-#44**	Listar documentos entrega-liquidación
Rf-#14	Listar tarjetas de combustible	**Rf-#45**	Consultar documento entrega-liquidación
Rf-#15	Buscar tarjeta de combustible	**Rf-#46**	Buscar documento entrega-liquidación
	Gestionar informe de carga	**Rf-#47**	Realizar búsqueda avanzada del documento entrega-liquidación
Rf-#16	Adicionar informe de carga	**Rf-#48**	Imprimir documento entrega-liquidación
Rf-#17	Modificar informe de carga	**Rf-#49**	Imprimir listado de documentos entrega-liquidación
Rf-#18	Listar informes de carga	**Rf-#50**	Confirmar documento entrega-liquidación
Rf-#19	Consultar informe de carga	**Rf-#51**	Contabilizar documento entrega-liquidación
Rf-#20	Buscar informe de carga		**Submayor de tarjeta**
Rf-#21	Imprimir listado de informes de carga	**Rf-#52**	Listar tarjetas de combustible en el submayor
Rf-#22	Imprimir informe de carga	**Rf-#53**	Consultar submayor de tarjeta de combustible
Rf-#23	Confirmar informe de carga	**Rf-#54**	Consultar movimientos de la tarjeta de combustible
Rf-#24	Contabilizar informe de carga	**Rf-#55**	Imprimir listado de tarjetas de combustible
Rf-#25	Cancelar del informe de carga	**Rf-#56**	Imprimir movimientos de la tarjeta de combustible
	Gestionar responsable de vehículo	**Rf-#57**	Realizar búsqueda avanzada de tarjeta de combustible
Rf-#26	Adicionar responsable de vehículo		
Rf-#27	Modificar responsable de vehículo		

Rf-#28	Eliminar responsable de vehículo
Rf-#29	Listar responsable de vehículo
Rf-#30	Buscar responsable de vehículo

2.2.2. Descripción de requisitos

A continuación se muestra la descripción del requisito Adicionar informe de carga. Las descripciones restantes se encuentran en el Expediente de proyecto 3.4 del centro CEIGE (Baez Fernández, 2015).

Descripción textual del requisito Adicionar Informe de carga

Precondiciones	Se ha autenticado el usuario del área de Caja responsable de realizar los procesos relacionados con las tarjetas de combustible en la entidad. Debe haberse asociado un instrumento de banco a la caja. Debe existir un listado de tarjetas en estado inactiva y reserva registradas en el sistema.
Flujo de eventos	
Flujo básico Adicionar informe de carga de tarjetas de combustible	
1.	Se selecciona la opción Gestionar informe de carga.
2.	Se presiona el botón Adicionar.
3.	El sistema muestra los siguientes datos: Número Importe Tarjetas cargadas Estado Creador por
4.	Se introducen y/ seleccionan los datos del informe de carga: Cajero Operación Matrícula Asignación (Litros)
5.	El sistema valida los datos introducidos.
6.	Si los datos son correctos el sistema los registra.
7.	Concluye el requisito.
Pos-condiciones	
1.	Se registró en el sistema un nuevo informe de carga.

Flujos alternativos		
Flujo alternativo 6.a Información errónea		
1	El sistema señala los datos erróneos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 5 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo 6.b Información incompleta		
1	El sistema señala los datos vacíos y permite corregirlos.	
2	El usuario corrige los datos.	
3	Volver al paso 5 del flujo básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	No se registran los datos.	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual CIG-ERP-N-CAJ-i2202.doc.	
2	Los siguientes datos son obligatorios: Cajero Operación Matrícula Asignación (Litros)	
3	Se valida que la asignación sea menor o igual que 2000 litros.	
4	Se valida que la matrícula tenga una longitud de 6 caracteres donde los 3 primeros o el primero sean letras mayúsculas y el resto números.	
Conceptos	**Informe de carga**	Visibles en la interfaz: Número Importe Tarjetas cargadas Estado Creador por código operación
Requisitos	N/A	
Asuntos pendientes	Posibles mejoras al requisito.	

2.2.3. Validación de los requisitos

Para la validación de los requisitos funcionales, se tuvieron en cuenta los criterios definidos en los artefactos “Criterios para validar requisitos del cliente” y “Criterios para validar requisitos del producto”, además se utilizaron prototipos de interfaz de usuario. Estos últimos brindan la posibilidad de modelar cómo serán las futuras interfaces del sistema. En la Figura 2 se muestra el prototipo de interfaz de usuario del requisito Adicionar informe de carga, el resto de los prototipos de interfaz de usuario y artefactos referentes a la validación de los requisitos se encuentran en el Expediente de proyecto 3.4 del centro CEIGE (Baez Fernández, 2015).

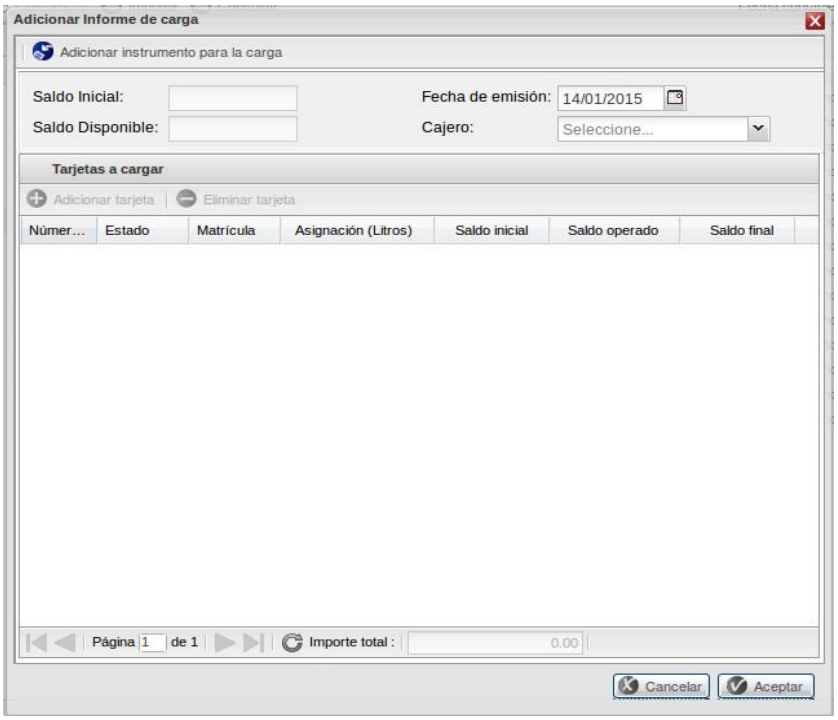

Figura 2: Prototipo del requisito Adicionar informe de carga

2.2.4. Evaluación de requisitos

Luego de haber identificado los requisitos funcionales del módulo, se realizó la evaluación de requisitos siguiendo los criterios de complejidad que establece la plantilla “Evaluación de requisitos” definida por el centro CEIGE (Baez Fernández, 2015).

La clasificación de la complejidad en Alta, Media y Baja permite estimar el esfuerzo de implementación del requisito, estableciendo como requisitos de mayor prioridad aquellos que posean complejidad Alta.

La Tabla 6 muestra los resultados obtenidos de la evaluación de requisitos de acuerdo a los criterios de complejidad a los que se hacían referencia anteriormente.

Tabla 6: Resultados de la complejidad

Resumen	
Cantidad de requisitos con complejidad Alta	23
Cantidad de requisitos con complejidad Media	26
Cantidad de requisitos con complejidad Baja	8

2.2.5. Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que los servicios o funciones del sistema a desarrollar deben poseer al final de su elaboración. Estas propiedades no están directamente relacionadas con las funcionalidades que se derivan del negocio. Le confieren al producto final ciertas características como usabilidad, rapidez, confiabilidad y seguridad (Sommerville, 2007).

Los requisitos no funcionales están definidos por el centro CEIGE para el proyecto XEDRO-ERP. Se seleccionaron sólo los aplicables a la presente solución.

Tabla 7: Requisitos no funcionales

Código	Clasificación
	Apariencia o interfaz externa
Rnf-#1	La interfaz de usuario en cada una de las páginas de la aplicación debe ser sencilla, intuitiva, que le permita al usuario que interactúa con el sistema realizar cualquier labor en el menor tiempo posible y con pocas acciones.
Rnf-#2	Las interfaces cuentan con menús que le permitan al usuario acceder desde cualquier lugar de la aplicación a otro lugar que desee con la mínima cantidad de acciones.
Rnf-#3	En el momento que ocurra un error con los datos de entrada a la aplicación le será informado al usuario de manera detallada.
	Usabilidad

Rnf-#4	El usuario final de la aplicación debe ser técnico medio en contabilidad o algún nivel superior, tener una experiencia profesional sobre los procesos de gestión de tarjetas de combustible de al menos 3 meses y un mínimo conocimiento en el uso de aplicaciones web o de escritorio.
	Confiabilidad
Rnf-#5	El sistema no permitirá la entrada de datos incorrectos.
Rnf-#6	El sistema impondrá campos obligatorios para garantizar la integridad de la información que se introduce por el usuario.
	Eficiencia
Rnf-#7	El sistema, para operaciones de inserción, modificación o búsqueda de datos, deberá tener respuestas a peticiones del cliente en un tiempo máximo de 30 segundos (esta cifra no incluye los retardos por concepto de tráfico de red).
Rnf-#8	El sistema, para operaciones de recuperación de datos, deberá tener respuestas a peticiones del cliente en un tiempo máximo de 15 segundos (esta cifra no incluye los retardos por concepto de tráfico de red).
Rnf-#9	El sistema no excede los 5 minutos para efectuar acciones de salva de la base datos (esta cifra no incluye los retardos por concepto de tráfico de red).
	Soporte
Rnf-#10	El sistema será modular, tomando como criterio para la creación de los módulos los macro-procesos de negocio identificados, favoreciendo así la incorporación, modificación o eliminación de funcionalidades.
Rnf-#11	La codificación del sistema será estándar y las funcionalidades serán comentadas.
	Restricciones de diseño
Rnf-#12	El sistema será una aplicación web centralizada.
Rnf-#13	El sistema se implementó usando el lenguaje PHP.
Rnf-#14	El sistema está regido por un estilo arquitectónico basado en componentes. Cada uno de los componentes sigue el patrón Modelo Vista Controlador.
Rnf-#15	El sistema usa el Framework de Presentación ExtJS para manejar las vistas.
Rf-#16	El sistema usa el Framework Zend para manejar la lógica de negocio.
Rnf-#17	El sistema usa el Framework Doctrine para manejar la persistencia de los datos.
	Hardware

Rnf-#18	El sistema para su instalación en las máquinas clientes requiere:
	• Procesador: 1.40 GHZ
	• RAM: 512 MB (recomendado 1024 GB)
	• Tarjeta de Red: 1
Rnf-#19	El sistema para su instalación en el servidor de aplicación requiere:
	• Procesador: 3.00 GHZ
	• RAM: 1GB (recomendado 4GB)
	• Disco duro: 160 GB (recomendado 500GB)
	• UPS: 1
	• Lector de CD: 1
	• Tarjeta de Red: 1
Rnf-#20	El sistema para su instalación en el servidor de base de datos requiere:
	• Procesador: 3.00 GHZ
	• RAM: 1GB (recomendado 4GB)
	• Disco duro: 160 GB (recomendado 500GB)
	• UPS: 1
	• Lector de CD: 1
	• Tarjeta de Red: 1

2.2.6. Modelo conceptual

El modelo conceptual explica de forma global los aspectos lógicos significativos en el dominio del problema (Buenastareas, 2015). Además estipula una especificación de los requisitos desde la perspectiva de la clasificación por objetos y desde el punto de vista de entender los términos empleados en el dominio. A continuación se muestra la representación conceptual de los procesos de tratamiento de tarjetas magnéticas para combustible:

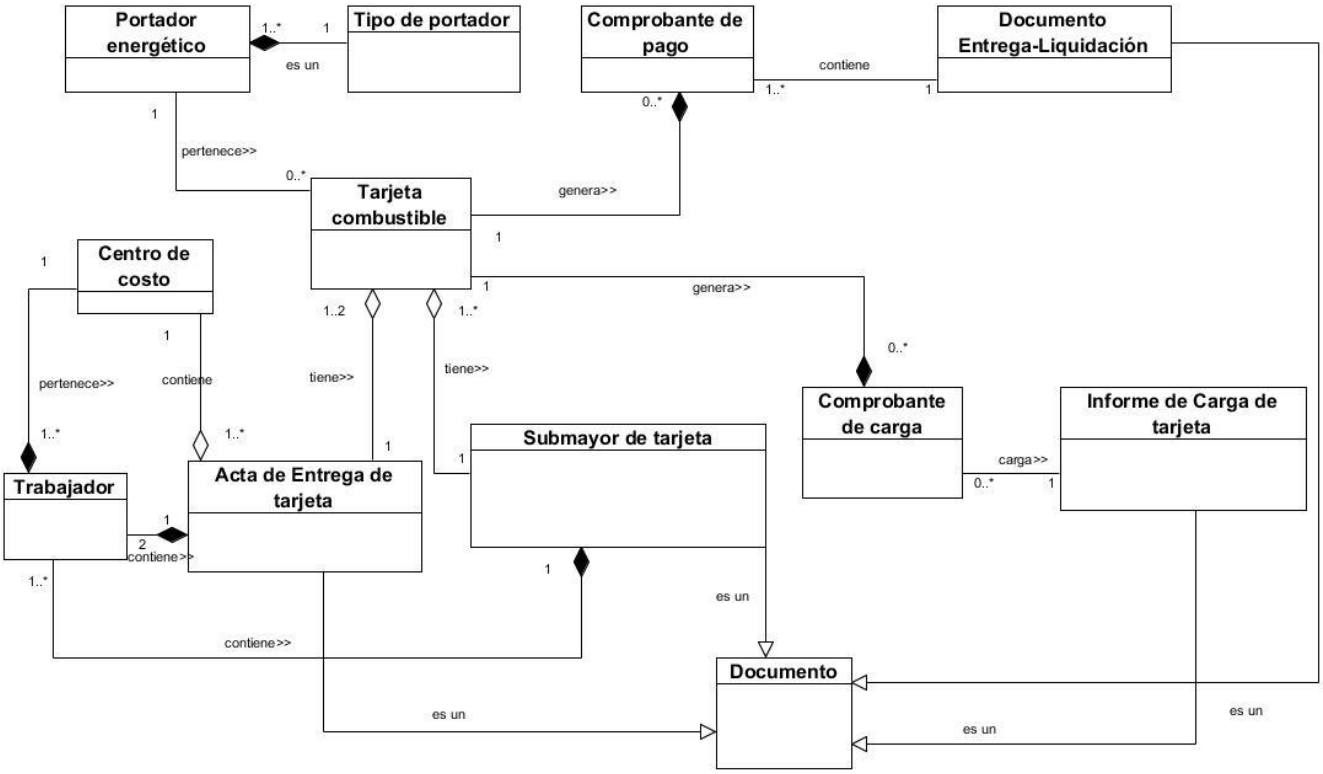

Figura 3: Modelo conceptual

2.3. Análisis y Diseño

El diseño de un sistema se basa en su arquitectura, la cual está compuesta por varios estilos o patrones que posibilitan detallar un software con suficiente claridad, lo que facilita llevar a cabo la realización física del mismo. La elaboración de un correcto modelo de diseño influye de manera significativa en la realización de un software con alta calidad.

2.3.1. Diseño de componentes

El módulo Tarjeta de combustible perteneciente al sistema XEDRO-ERP está compuesto por un único componente denominado Tarjeta de combustible, es el encargado de informatizar los requisitos funcionales relacionados con los procesos establecidos en el negocio. Un modelo de componentes define los parámetros a seguir para la construcción de un componente, su documentación y despliegue, de esta forma se puede asegurar que el conjunto de componentes que conforman el sistema puedan comunicarse e interactuar entre ellos (Hall, 2011).

Para el desarrollo del componente Tarjeta de combustible se requiere de una estrecha relación con algunos componentes existentes en el sistema XEDRO-ERP, por los servicios que estos brindan y que deberán ser consumidos por el componente Tarjeta de combustible para poder realizar las

funcionalidades correspondientes. A continuación se describen brevemente las responsabilidades de cada componente:

Estructura y Composición: Se encarga de organizar las estructuras jerárquicas que conforman la entidad y registrar la información que las componen. (Ejemplo: agrupación, entidad, áreas, cargos y escalas salariales). Información que es consumida por el módulo Tarjeta combustible.

Seguridad: Se encarga de gestionar las acciones a las que tendrán acceso los usuarios según el rol que desempeñen en la entidad. Información que es consumida por el módulo Tarjeta de combustible.

Configuración: Se encarga de registrar y controlar los proveedores, documentos primarios, unidad de medida y reglas contables; datos necesarios para desencadenar los procesos del tratamiento de las tarjetas magnéticas para combustible en el sistema.

Capital Humano: Se encarga del registro y control de los trabajadores de la entidad (Ejemplo: cajero, chofer y director). Información que es consumida por el módulo Tarjeta de combustible.

Contabilidad: Se encarga de registrar el nomenclador (o maestro) de cuentas contables definidas por la entidad. Controla el submayor y mayor de cuentas, además de gestionar los comprobantes de operaciones que son generados por los subsistemas. Las contabilizaciones que se realicen en el módulo Tarjeta combustible serán utilizadas por el módulo Configuración, para que este pueda crear la estructura del comprobante, el cual posteriormente será enviado al módulo Contabilidad para ser asentado.

Banco: Se encuentra dentro del subsistema Finanza. Este componente cuenta con las funcionalidades “Instrumento” y “Configuración”, las cuales son consumidas por el módulo Tarjeta de combustible cuando se realiza un informe de carga.

Multimoneda: Registra y controla las monedas que están vigentes en las operaciones contables de la entidad así como sus tasas; información empleada por el módulo Tarjeta de combustible.

Caja: Se encuentra dentro del subsistema Finanzas. Este módulo contiene al componente “Tarjeta de combustible”, el cual registra las funcionalidades asociadas a los procesos de carga, entrega y liquidación del combustible consumido para un mayor control de los mismos. Además permite generar diferentes reportes los que posibilitan realizar una correcta toma de decisiones en el ahorro del combustible de la empresa.

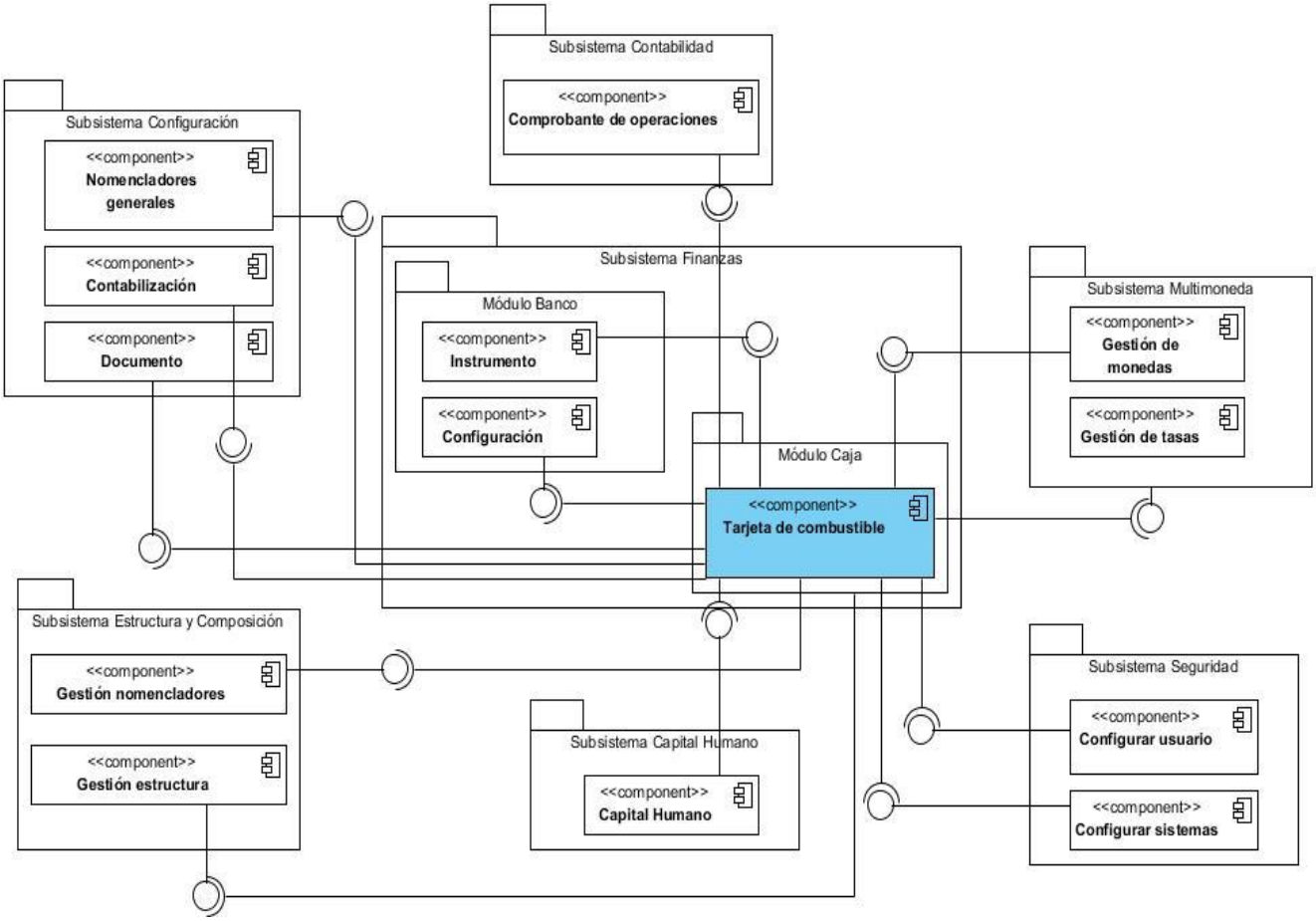

Figura 4: Diagrama de componentes del módulo Tarjeta de combustible

2.3.2. Modelo de datos

El modelo de datos describe de forma abstracta las entidades y sus características, además de las relaciones entre estas dentro de la base de datos. Las entidades son objetos que guardan información necesaria para el sistema y los atributos son las características de una entidad. Las relaciones muestran la asociación entre dos entidades, representadas por una línea discontinua que une a las entidades involucradas (Datos, 2012).

El modelo de datos de la solución cuenta con 19 tablas, de ellas 10 fueron las nuevas entidades creadas en el esquema Caja, donde 4 son nomencladores y el resto destinadas al almacenamiento de la información correspondiente al dominio del problema. Estas entidades además se relacionan con las restantes tablas de los esquemas Capital humano, Finanzas, Datos maestros, Nomencladores y Caja del modelo de datos generado. A continuación se muestra una sección del modelo de datos de la solución, ver Figura 5.

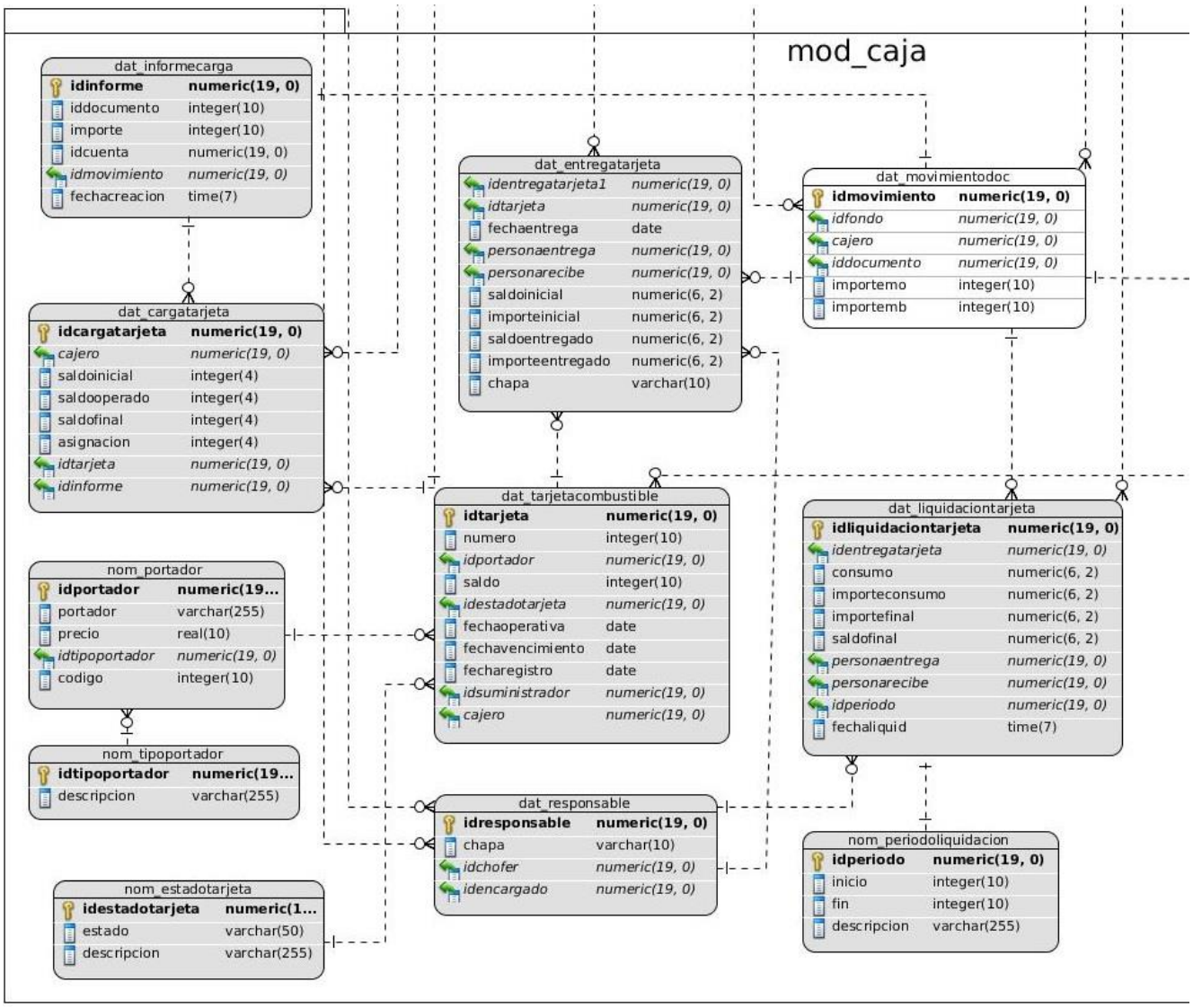

Figura 5: Sección del modelo de datos del módulo Tarjeta de combustible

2.3.3. Diagrama de clases del diseño con estereotipos web

Los diagramas de clases según la clasificación UML, son diagramas de estructura estática donde la representación de los requisitos se lleva a cabo a través de las clases del sistema y sus interrelaciones (Computación, 2014).

A continuación se muestra el diagrama de clases del diseño con estereotipos web correspondiente a la funcionalidad Gestionar carga de tarjeta, ver Figura 6. Los diagramas de las demás funcionalidades identificadas se encuentran en el Expediente de proyecto 3.4 del centro CEIGE (Baez Fernández, 2015).

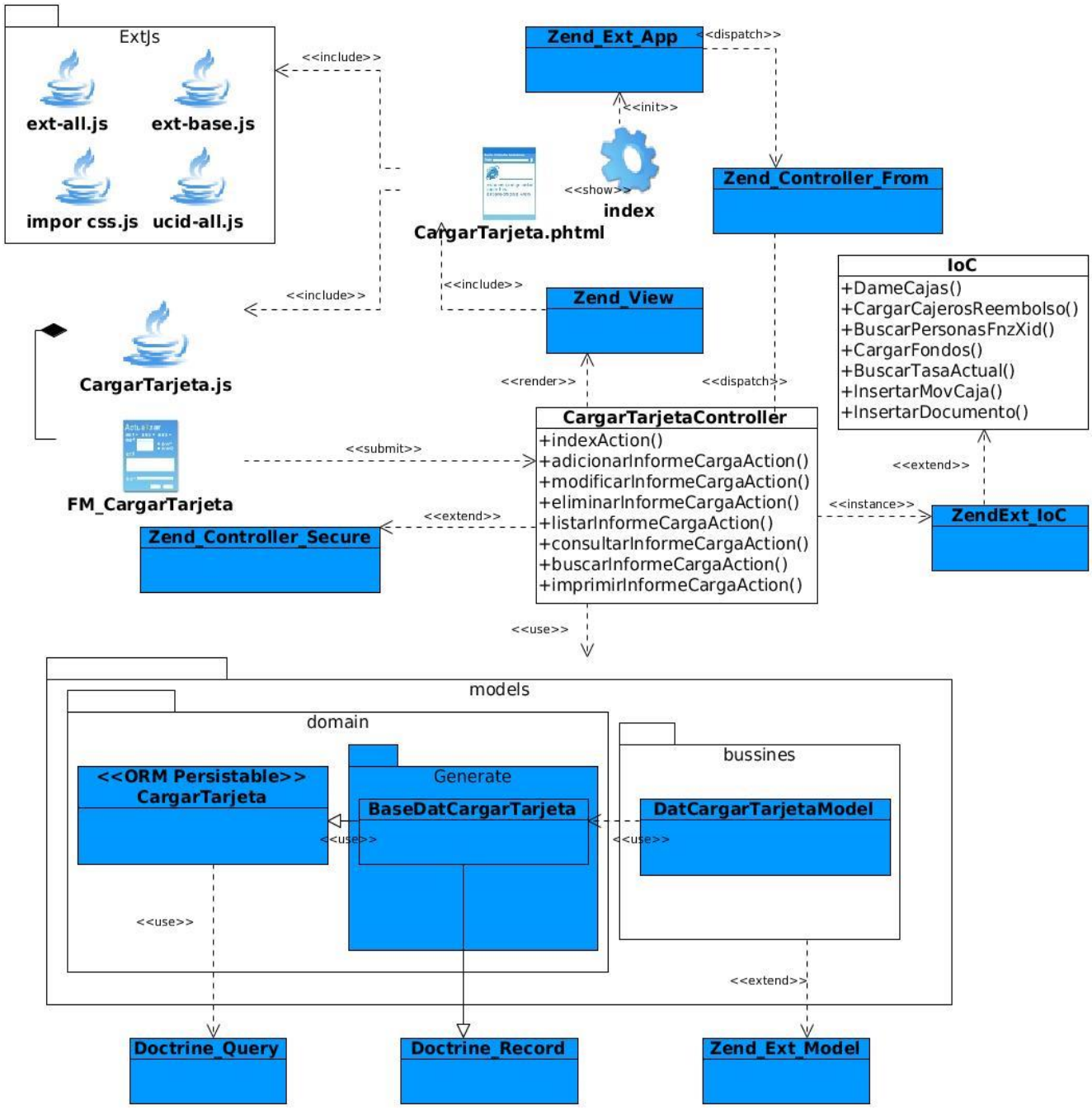

Figura 6: Diagrama de clases del diseño con estereotipos web de la funcionalidad Gestionar carga de tarjeta

La Tabla 8 muestra la descripción de las clases que componen el diagrama de clases del diseño con estereotipos web de la funcionalidad Gestionar carga de tarjeta.

Tabla 8: Descripción de las clases del diagrama de clases del diseño con estereotipos web

Clases	Descripción
Cargartarjeta.phtml	Encargada de visualizar, a través de los js que debe incluir, la información necesaria del proceso carga de tarjetas.
Cargartarjeta.js	Encargada de generar de forma dinámica a través del DOM y utilizando la librería ExtJS los componentes necesarios a través de los cuales se puedan realizar las operaciones que genera cargar tarjetas. Responsable de enviar y recibir los datos de la controladora utilizando las tecnologías AJAX y JSON.
Fm_ Cargartarjeta	Encargado de recoger los datos entrados por el usuario para poder realizar las funcionalidades correspondientes a la carga de tarjeta.
Index.php	Encargado de redireccionar el sistema, comienza con el levantamiento del sistema y va al componente Portal.
CargartarjetaController	Clase controladora responsable de carga las tarjetas, donde a partir de ella se deben ejecutar las diferentes funcionalidades, según las peticiones del usuario.
IoC	Permite fomentar la reutilización de componentes que tienen dependencias de servicios o componentes.
Paquete Models	Encargado de manejar los datos persistentes dentro del componente. Contiene el Bussines y el Domain.

2.3.4. Diagrama de secuencia

Los diagramas se secuencia describen la interacción entre los objetos de una aplicación y los mensajes recibidos y enviados por los objetos. Estos contienen detalles de la implementación del escenario, incluyendo los objetivos y clases que se usan para implementarlo.

Los diagramas de secuencia de los requisitos identificados se encuentran en el Expediente de proyecto 3.4 del centro CEIGE (Baez Fernández, 2015).

2.3.5. Patrones de diseño y arquitectónico

El diseño de la solución fue elaborado siguiendo patrones que constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos.

El patrón arquitectónico MVC gestiona el flujo de información desde la vista (view) a través del negocio (controllers), hacia el acceso a datos (models) y viceversa. Este patrón se manifiesta en el

marco de trabajo Sauxe a través de cuatro nodos de integración: vista-controlador, controlador-modelo, modelo-frameworks Doctrine y Doctrine-base de datos. A continuación se muestra la Figura 7, Figura 8 y Figura 9; las cuales representan la capa modelo, capa vista y capa controladora, respectivamente del componente Tarjeta de combustible.

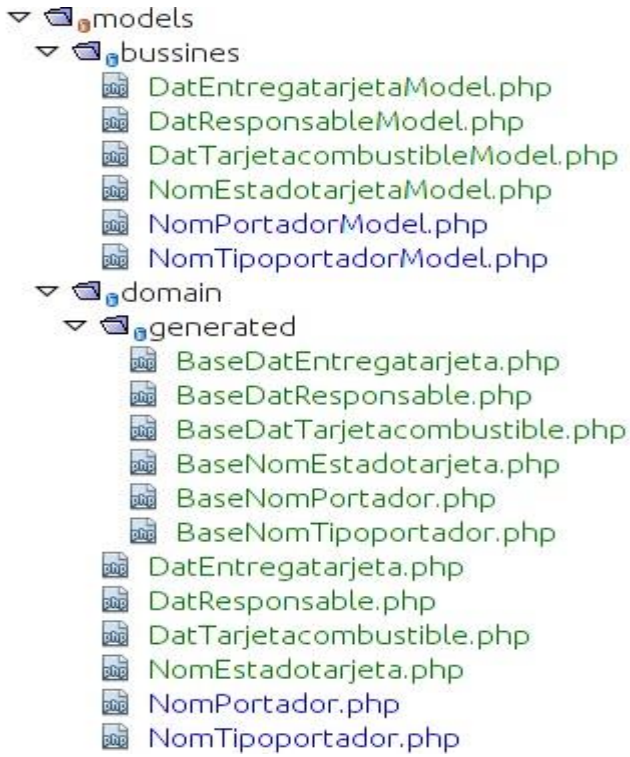

Figura 7: Capa modelo del componente Tarjeta de combustible

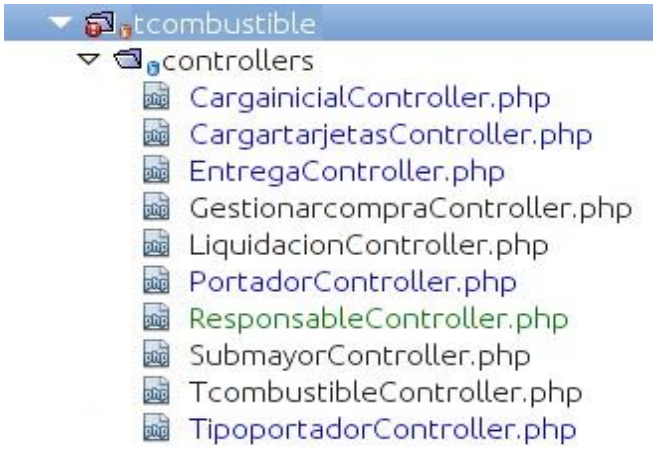

Figura 8: Capa controladora del componente Tarjeta de combustible

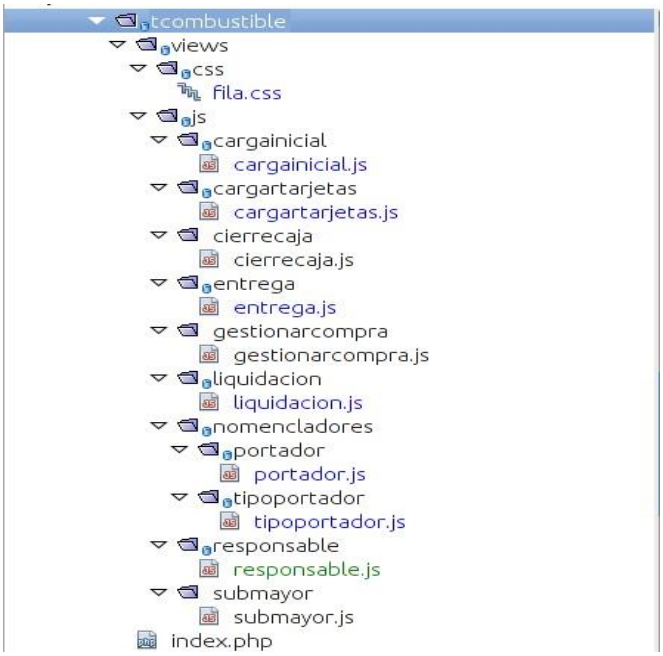

Figura 9: Capa vista del componente Tarjeta de combustible

Para contribuir a que el sistema sea más robusto y flexible se utilizaron los siguientes patrones GRASP:

- **Experto:** La utilización de este patrón se evidencia en las clases de la capa del negocio. Por ejemplo, la responsabilidad de guardar un portador energético se le asigna a la case *NomPortadorModel*, la cual contiene toda la información necesaria para realizar dicha actividad.
- **Creador:** Se evidencia en las clases controladoras del módulo Tarjeta combustible donde cada controladora agrega o contiene a los objetos según la responsabilidad de la misma. Por ejemplo en la clase *ResponsableController* se pueden obtener los responsables de los vehículos.
- **Bajo acoplamiento:** Se observa la utilización de este patrón en las clases de la capa del negocio, donde existen solo las relaciones necesarias entre las clases para mantener bajo el acoplamiento. Por ejemplo entre las clases *ResponsableController* y *DatResponsableModel.*
- **Alta cohesión:** Su empleo se evidencia en las clases controladoras, aunque cada clase es experta en su responsabilidad, puede cooperar con otras clases para poder realizar algunas de sus tareas. En la clase *PortadorController* cuando se elimina un portador energético se hace necesario saber si hay alguna dependencia con las tarjetas magnéticas para combustible.

- **Controlador:** La clase controladora: *TipoportadorController*, es ejemplo de la aplicación de este patrón, la misma tiene a cargo la responsabilidad de gestionar los tipos de portadores energéticos en el componente, haciendo uso de las clases del modelo.

En el diseño del componente además de hacer uso de los patrones antes mencionados se emplearon los siguientes patrones GoF:

- **Cadena de responsabilidad (Comportamiento):** Evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.

  Se evidencia en la capa de presentación del sistema cuando se realiza una petición al modelo, debido a que el sistema está preparado para responder ante los posibles eventos que puedan suceder en este proceso .Un ejemplo sería en el archivo javascript *tiportador.js* cuando se guarda un tipo de portador energético, ya que en su clase controladora existe un mecanismo para dar respuesta si este ya existe, sino hubo un error insertándolo en la base de datos o si fue insertado correctamente.
- **Singleton (Creacional):** Patrón que restringe la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto, asegura que solo se cree una instancia de una clase y proporciona un punto de acceso global a dicha instancia.

  La utilización de este patrón se evidencia en las clases modelo del módulo Tarjeta de combustible, mediante la creación del método getInstance(), el cual asegura que solo se cree un único objeto de cada clase.
- **Fachada (Estructural):** Patrón estructural que provee una interfaz unificada para un conjunto de interfaces en un subsistema. Fachada define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.

  La utilización de este patrón está reflejada en la creación y empleo de la clase IoC, por la necesidad de integración entre los módulos del Sistema XEDRO-ERP, era necesaria la implementación de una clase fachada donde fueran publicados los servicios necesarios por estos, facilitando su interacción, donde la relación que existe entre las clases controladoras y al fachada, permite acceder a métodos que están implementados en otros componentes dentro y fuera del subsistema Finanzas.

2.3.6. Validación del diseño

Con el fin de evaluar la calidad del diseño se utilizaron las métricas de software siguientes que

permiten medir de forma cuantitativa la calidad de los atributos internos del sistema propuesto:

Aplicación de la métrica Tamaño Operacional de Clases

Se le aplicó la métrica Tamaño Operacional de Clases a un total de 25 clases y 160 operaciones, para un promedio de procedimientos por clases de 6.32, aproximándolo al valor entero 6.

En la Figura 10 se muestra una representación gráfica de los resultados obtenidos agrupados en los intervalos de procedimientos definidos. Esta refleja que la mayoría de las clases tienen de 0 a 6 procedimientos, lo que demuestra que el funcionamiento general del sistema está distribuido equitativamente entre las diferentes clases.

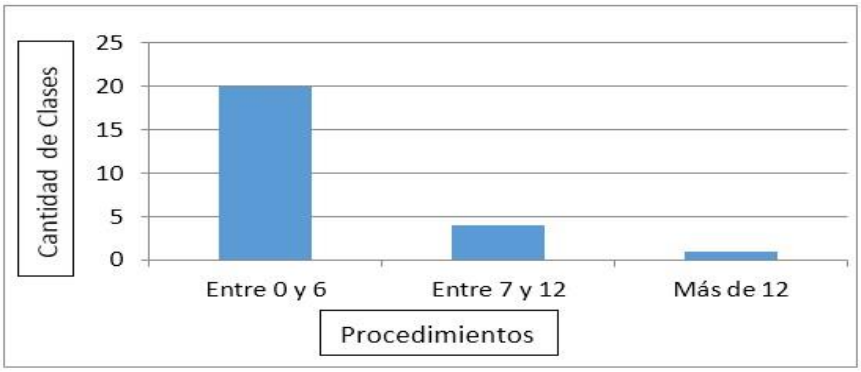

Figura 10: Representación de la métrica TOC

Luego de haber realizado un análisis de los resultados obtenidos para los atributos de la métrica, se observa que el 90% de las clases que conforman el módulo Tarjeta de combustible, para los atributos responsabilidad y complejidad de implementación están dentro de la categoría Media y Baja, mientras que el atributo reutilización cuenta con igual por ciento de clases en las categorías Alta y Media, demostrando así que el módulo cuenta con una elevada reutilización, baja complejidad de implementación y responsabilidad en el diseño propuesto. Se concluye que los resultados obtenidos según esta métrica son satisfactorios.

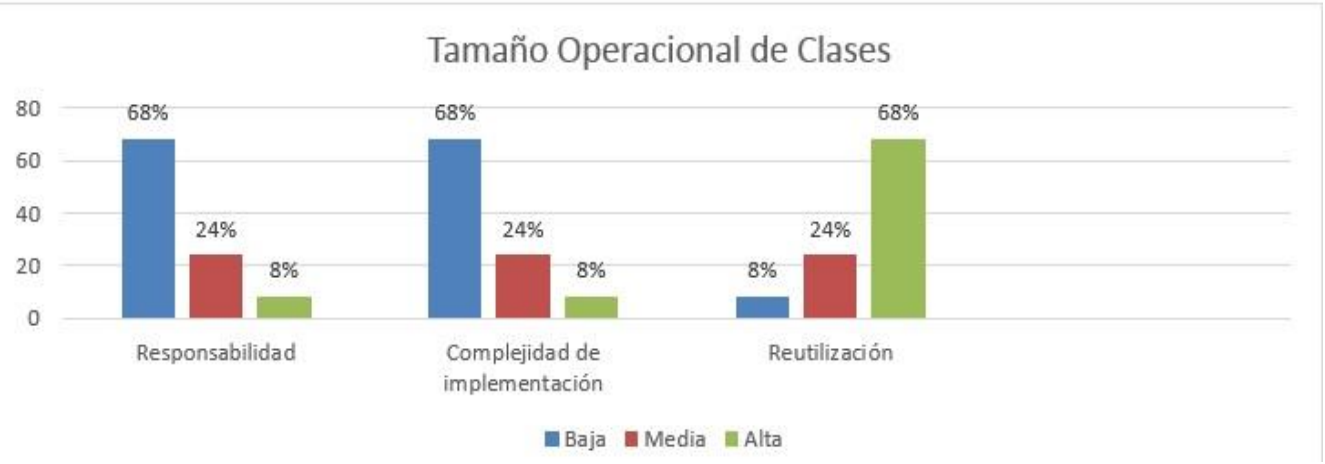

Figura 11: Representación en % de los resultados de la evaluación mediante la métrica TOC en los atributos: Reutilización, Complejidad de implementación y Responsabilidad

Aplicación de la métrica Relaciones entre Clases

Se le aplicó la métrica Relaciones entre Clases a un total de 25 clases y 52 relaciones, para un promedio de relaciones entre clases de 2.08, aproximándolo al valor entero 2.

La Figura 12 muestra una representación gráfica de los resultados obtenidos, los mismos reflejan que la mayoría de las clases tienen una (1) dependencia entre clases, por lo que las clases se encuentran dentro de los niveles aceptables de calidad.

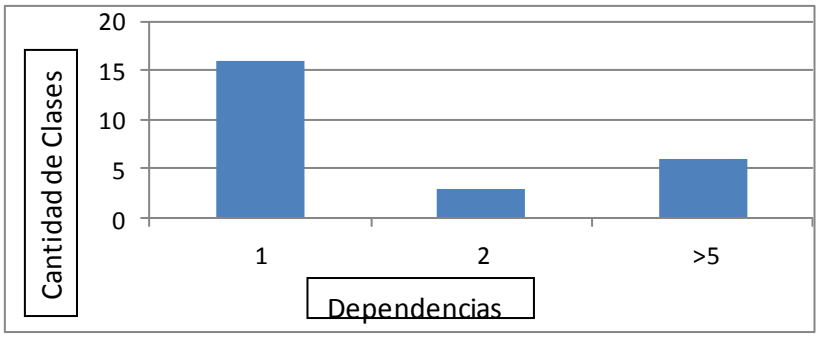

Figura 12: Representación de la métrica RC

Al analizar los resultados obtenidos de la métrica RC, se puede concluir que el diseño realizado tiene una calidad aceptable. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 64% de las clases el nivel de Acoplamiento es mínimo. La Complejidad de mantenimiento y la Cantidad de pruebas son bajas a un 80%, mientras que el atributo Reutilización cuenta con igual por ciento en la categoría Alta. Esto implica que se reduzca el nivel de esfuerzo necesario para sustentar el diseño y se disminuyan la cantidad de pruebas realizadas para mejorar la calidad de software.

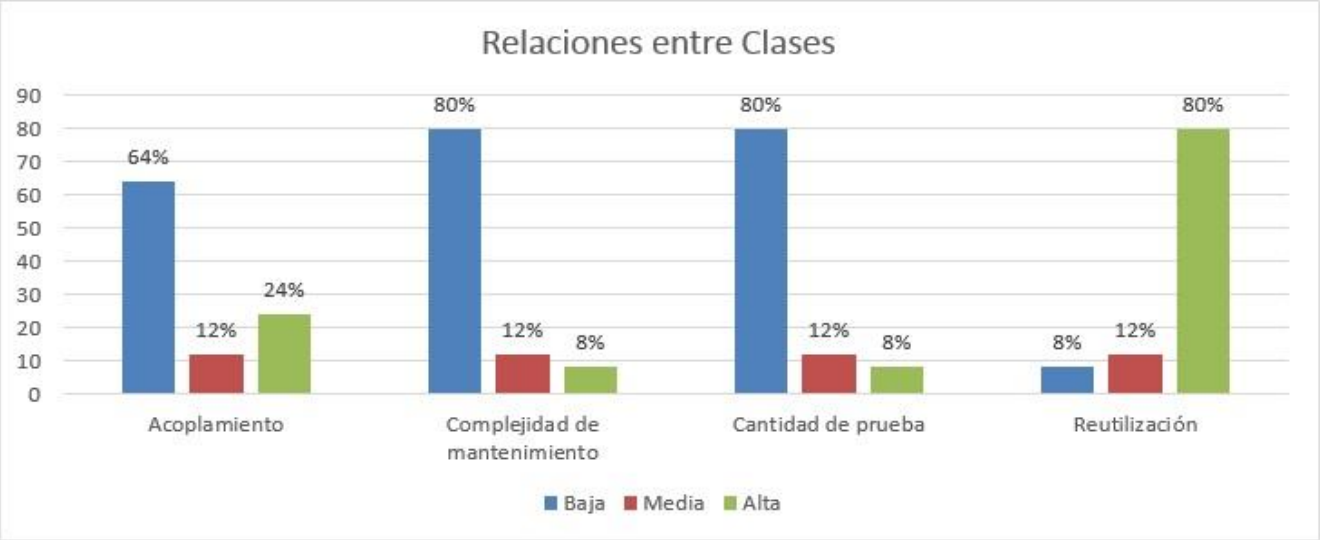

Figura 13: Representación en % de los resultados de la evaluación mediante la métrica RC en los atributos: Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de pruebas

Conclusiones parciales

- La definición, especificación y validación de los requisitos del módulo permitió abarcar todas las funcionalidades requeridas para informatizar los procesos de las tarjetas de combustible.
- La validación del modelo de diseño mediante el uso de métricas permitió determinar que la mayor parte de las clases son reutilizables, poco dependientes entre sí y colaboran para compartir el esfuerzo si la tarea es grande.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA PROPUESTA DE SOLUCIÓN

Introducción

En el presente capítulo se describe cómo realizar una correcta implementación de la solución a partir del uso de estándares de codificación, tratamiento de errores y del principio arquitectónico de inversión y control (IoC) utilizado para lograr la integración entre los componentes del sistema XEDRO-ERP. Se aplican las pruebas de caja blanca, caja negra y de aceptación, lo que permite comprobar si el software cumple con la especificación de los requisitos establecidos en el análisis y si se ajusta con las necesidades del cliente. Además se realiza la validación de las variables de la investigación.

3.1. Integración entre componentes

La integración para la comunicación entre módulos y componentes se realiza empleando el principio arquitectónico de inversión y control (IoC), que está implementado en el framework Zend_Ext y permite realizar la inyección de dependencias y operar sobre distintos esquemas en la base de datos, realizando las transacciones pertinentes. En este componente se define el fichero ioc.xml, que contiene la ubicación de los componentes y los servicios que brindan las clases services correspondientes. La integración se realiza tanto interna entre módulos como externa entre subsistemas. IoC Interno: para la integración entre componentes de un subsistema. IoC Externo: para la integración entre subsistemas (Sierra, 2012).

Servicios entre componentes

Se describen a continuación los servicios que son consumidos por el componente Tarjeta de combustible del sistema XEDRO-ERP:

Tabla 9: Servicios Consumidos

Servicios	Componente que brinda	Descripción
DameCajas	Configuración de Caja	Servicio utilizado para obtener las Cajas pertenecientes a una
CargarFondos	Configuración de Caja	Servicio utilizado para obtener los fondos asociados a una caja.
CargarCajerosReembolso	Metadatos	Servicio utilizado para obtener los cajeros asociados a una caja determinada.

ObtenerOperaciones	Comprobantes_operaciones	Servicio utilizado para obtener las operaciones asociadas a un concepto.
DameOperacPorIdFondo	Comprobantes_operaciones	Servicio utilizado para devolver las operaciones que están configuradas el fondo.
BuscarPersonasFnzXid	Parametrizaciones	Servicio utilizado para obtener un trabajador de la entidad dado un
OperacionesPorId	Comprobantes_operaciones	Servicio utilizado para obtener la denominación de una operación.
ObtenerMovimientosporIdfondo	Recuperaciones	Servicio utilizado para obtener los movimientos asociados a un
DameInstrumentosPorIdMovCaja	Instrumento	Servicio utilizado para obtener los instrumentos asociados a un movimiento en caja.
DameAnexosPorDesglose	Moviemientos_genericos	Servicio utilizado para obtener los anexos asociados a un movimiento en caja.
BuscarTasaActual	Consolidacion	Servicio utilizado para obtener la tasa correspondiente a la moneda del fondo de efectivo.
InsertarMovCaja	Moviemientos_genericos	Servicio utilizado para insertar el movimiento caja asociado a una operación.
InsertarDocumento	Comun_finanzas	Servicio utilizado para insertar un documento finanza.
InsertarMovDoc	Comun_finanzas	Servicio utilizado para insertar un movimiento que relaciona una operación con un documento.
Buscartrabajadores	Configuracion de Finanza	Servicio utilizado para obtener los trabajadores de la entidad.

ObtenerProveedores	Configuracion	Servicio utilizado para obtener los proveedores asociados a una entidad.
dameMovCajaArrIdMovDoc	Comun_finanzas	Servicio utilizado para obtener los movimientos en caja asociados a una lista de documentos movimiento.
DameInstrumentosPorIdMovCaj	Instrumento	Servicio utilizado para obtener los movimientos de instrumentos realizados por la caja.

3.2. Tratamiento de errores

En los formularios, como una de las vías para introducir información al sistema por los usuarios, pueden ocurrir errores. En aras de un correcto funcionamiento del sistema, el tratamiento de errores se convierte en un mecanismo de vital importancia. Es fundamental identificar y controlar los posibles problemas que puedan presentarse a la hora de interactuar con el software.

Las validaciones realizadas del lado del servidor se encargan de controlar el flujo de los datos recibidos en el controlador para evitar la inconsistencia de la información almacenada en la base de datos. Las validaciones realizadas en las vistas se encargan de que el usuario haga una entrada correcta o parcialmente correcta de los datos. Tienen como función principal evitar que se introduzcan datos incorrectos en los diferentes campos de los formularios para impedir que datos inconsistentes o incorrectos lleguen al servidor. Un ejemplo de estas validaciones se pueden observar en la Figura 14 y Figura 15 respectivamente.

PortadorController.php

```
function guardarPortadorAction() {
    $objesc = new NomPortador();
    $objesc->portador = $this->_request->getPost('portador');
    $aux1 = $objesc->portador;
    $objesc->precio = $this->_request->getPost('precio');
    $objesc->codigo = $this->_request->getPost('codigo');
    $objesc->idtipoportador = $this->_request->getPost('idtipoportador');
    $accion = $this->_request->getPost('accion');
    $aux2 = $objesc->verExiste($aux1);
    $n = count($aux2);
    $mtipoPortador = new NomPortadorModel();
    if ($n == 0) {
        if ($mtipoPortador->insertar($objesc)) {
            echo("{'codMsg':1,'mensaje': 'Portador energético insertado correctamente.'}");
        } else {
            echo("{'codMsg':3,'mensaje': 'Error insertando Portador energético.'}");
        }
    } else {
        echo("{'codMsg':3,'mensaje': 'Ya existe un Portador energético con este nombre.'}");
    }
}
```

Figura 14: Validación del formulario Portador del lado de las vistas

Figura 15: Validación del formulario Portador del lado del servidor

3.3. Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. A continuación se especifican los estándares de codificación que fueron utilizados en el desarrollo del módulo Tarjeta de combustible, normado por el documento "Estándares de codificación para proyectos con el marco de trabajo Sauxe del centro CEIGE":

Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se emplea notación PascalCasing[12]. Con solo leerlo se reconoce el propósito de la misma. Ejemplo: Portador.

- Según el tipo de clases:

Las clases controladoras después del nombre lleva la palabra "Controller", ver Figura 16.

```
<?php

class PortadorController extends ZendExt_Controller_Secure {
```

Figura 16: Notación PascalCasing aplicada en el nombre de la clase PortadorController

A las clases que se encuentran dentro de la carpeta Business (Negocio) se les incorpora después del nombre la palabra "Model", ver Figura 17.

[12] PascalCasing es un procedimiento de programación común en el lenguaje Java y .Net. La nomenclatura está compuesta por tantas palabras como sean necesarias. La letra inicial del identificador y la primera letra de cada una de las palabras que lo forman irán siempre en mayúsculas y se obvia el uso de artículos.

```
<?php

class NomPortadorModel extends ZendExt_Model {
```

Figura 17: Notación PascalCasing aplicada en el nombre de la clase NomPortadorModel

Las clases que se encuentran dentro de la carpeta “Domain” reciben su nombre de acuerdo a la tabla de la base datos que representan, ver Figura 18.

```
<?php
/**
 *this class has been auto-generated by the Doctrine ORM Framework
 */
class NomPortador extends BaseNomPortador {
```

Figura 18: Notación PascalCasing aplicada en el nombre de la clase NomPortador

Las clases que se encuentran dentro del directorio “Domain/generated” reciben su nombre de acuerdo a la tabla en la base de datos que representan y a la vez se les agrega la palabra Base al inicio, ver Figura 19.

```
<?php
/**
 *this class has been auto-generated by the Doctrine ORM Framework
 */
abstract class BaseNomPortador extends Doctrine_Record {
```

Figura 19: Notación PascalCasing aplicada en el nombre de la clase BaseNomPortador

Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se emplea notación CamelCasing[13]. Con solo leerlo se reconoce el propósito de la misma. Ejemplo: guardarPortador.

En caso de ser funcionalidades de la clase controladora se le pone el nombre y seguida la palabra “Action”, ver Figura 20.

```
public function guardarPortadorAction() {
```

Figura 20: Notación CamelCasing aplicada en el nombre de la función guardar portador

[13] CamelCasing es un estilo de escritura que se aplica a frases o palabras compuestas. La nomenclatura está compuesta por tantas palabras como sean necesarias. La letra inicial del identificador irá en minúscula y la primera letra del resto de las palabras que lo forman irán siempre en mayúsculas y se obvia el uso de artículos.

Nomenclatura de variables

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se emplea notación CamelCasing, y comenzando con un prefijo según el tipo de datos, ver Figura 21.

```
$intCodigo = $objesc->codigo;
$realprecio = $objesc->precio;
$intIdPortador = $this->_request->getPost('idportador');
```

Figura 21: Notación CamelCasing aplicada en los nombres de las variables

Nomenclatura de los comentarios

Es una necesidad comentar todo lo que se haga dentro del desarrollo, es decir, establecer las pautas que conlleven a lograr un código más legible y reutilizable, de manera que se facilite su mantenibilidad a lo largo del tiempo. Los comentarios deben ser claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando.

Antes de la declaración de una clase se escribe una breve descripción donde se explique el propósito de la misma, ver Figura 22.

```
<?php
/**
 *NomPortadorModel
 *Clase modelo del negocio para gestionar los portadores energéticos
 *@author Roberto Bandera Gómez
 *@package Finanzas
 *@version 1.0
 */
class NomPortadorModel extends ZendExt_Model {
```

Figura 22: Comentarios de la clase NomPortadorModel

Antes de la declaración de la función se escribe una breve descripción donde se explique el propósito de la misma, ver Figura 23.

```
/**
*verExiste
*Busca los portadores energéticos cuyo nombre sea el pasado por parámetros.
*@param $portador - nombre del portador buscado
*@return array datos - retorna todos los portadores energéticos cuyo nombre sea el pasado por parámetros.
*/
public function verExiste($portador) {
```

Figura 23: Comentarios de la función verificar si existen portadores

3.4. Pruebas aplicadas

Prueba de Caja blanca

Dentro de la prueba de caja blanca, la técnica que se utilizó fue camino básico. Para aplicar esta técnica se debe introducir la notación para la representación del flujo de control, este puede representarse por un grafo de flujo en el cual:

- Cada nodo del grafo corresponde a una o más sentencias de código fuente.
- Todo segmento de código de cualquier programa se puede traducir a un grafo de flujo.
- Se calcula la complejidad ciclomática del grafo.

Un grafo de flujo está formado por tres componentes fundamentales que ayudan a su elaboración y comprensión, estos brindan información para confirmar que el trabajo se está haciendo adecuadamente. Componentes del grafo de flujo:

- **Nodo:** Son los círculos representados en el grafo, el cual contiene una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.
- **Aristas:** Son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.
- **Regiones:** Son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

Para realizar la prueba del camino básico es preciso calcular la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se muestra el código del método “actualizarEstadoTarjetaEntregada” encargado de cambiar al estado “En explotación” los estados de las tarjetas entregadas, ver Figura 24.

```
    public function actualizarEstadoTarjetaEntregada($idtarjeta) {

        $objTarjeta = new DatTarjetacombustible();//1
        //Obtengo la tarjeta que fue cargada.
        $objTarjeta = Doctrine::getTable('DatTarjetacombustible')->find($idtarjeta);//1
        $objEstadoTarjeta = new NomEstadotarjeta(); //cargarEstado//1
        $arrayObjEstado = $objEstadoTarjeta->cargarEstado();//1
        foreach ($arrayObjEstado as $objEstado) {//2
            if ($objEstado[estado] == 'En explotación') {//3
                $objTarjeta->idestadotarjeta = $objEstado[idestadotarjeta];//4
            }
        }
        $tarjeta = DatTarjetacombustibleModel::GetInstancia();//5
        $tarjeta->insertar($objTarjeta);//6
        return true;//7
    }

}
```

Figura 24: Método actualizarEstadoTarjetaEntregada

Para el cálculo de la complejidad ciclomática es necesario representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones, quedando como se muestra en la Figura 25.

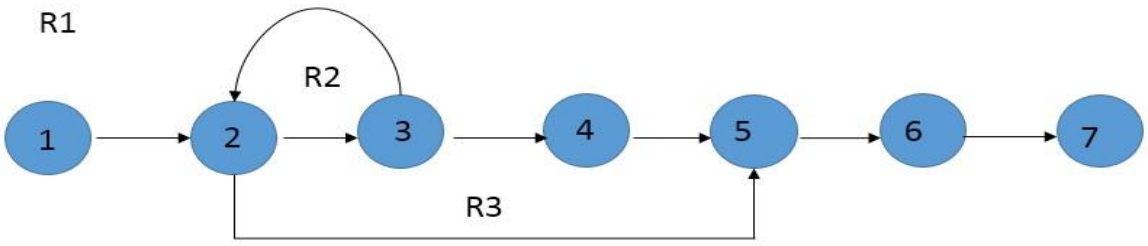

Figura 25: Grafo de flujo asociado al método actualizarEstadoTarjetaEntregada

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, el cálculo es necesario efectuarlo mediante tres fórmulas, se debe utilizar el mismo grafo en cada caso:

Fórmula 1: V (G) = (A – N) + 2 Donde “A” es la cantidad total de aristas y “N” la cantidad total de nodos.

Resultado:

V (G) = (8-7) + 2

V (G) = 3.

Fórmula 2: V (G) = P + 1 Donde “P” es la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

Resultado:

V (G) = 2 + 1

V (G) = 3.

Fórmula 3: V (G) = R Donde “R” es la cantidad total de regiones, se incluye el área exterior del grafo, contando como una región más.

Resultado: V (G) = 3.

Seguidamente es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución:

Camino básico # 1: 1, 2, 3, 2, 3, 4, 5, 6, 7.

Camino básico # 2: 1, 2, 3, 4, 5, 6, 7.

Camino básico # 3: 1, 2, 5, 6, 7.

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico. Para realizarlos es necesario cumplir con las siguientes exigencias:

- Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- Entrada: Se muestran los parámetros que entran al procedimiento.
- Resultados esperados: Se expone el resultado que se espera que devuelva el procedimiento.
- Resultados: Se muestra el resultado obtenido.
- Salida: Se presenta el valor final.

Tabla 10: Caso de prueba para el camino básico 1

Camino básico # 1: 1, 2, 3, 2, 3, 4, 5, 6,	
Descripción	Se actualiza el estado de la tarjeta que ha sido entrega para tener constancia de que no se encuentra en la Caja.
Condición de ejecución	Debe estar creada y cargada al menos una tarjeta de combustible.

Entrada	$objTarjeta = new DatTarjetacombustible(); $objTarjeta = Doctrine::getTable(' DatTarjetacombustible ')->find($idtarjeta); $objEstadoTarjeta = new NomEstadotarjeta(); $arrayobjEstado = $objEstadoTarjeta->cargarEstado();
Resultados esperados	Se debe actualizar el estado de la tarjeta de combustible entregada.
Resultados	Se actualizó el estado de la tarjeta de combustible entregada luego de varias iteraciones en busca del estado “En explotación”.
Salida	True.

Tabla 11: Caso de prueba para el camino básico 2

Camino básico # 2: 1, 2, 3, 4, 5, 6, 7	
Descripción	Se actualiza el estado de la tarjeta que ha sido entrega para tener constancia de que no se encuentra en la Caja.
Condición de ejecución	Debe estar creada y cargada al menos una tarjeta de combustible.
Entrada	$objTarjeta = new DatTarjetacombustible(); $objTarjeta = Doctrine::getTable(' DatTarjetacombustible ')->find($idtarjeta); $objEstadoTarjeta = new NomEstadotarjeta(); $arrayobjEstado = $objEstadoTarjeta->cargarEstado();
Resultados esperados	Se debe actualizar el estado de la tarjeta de combustible entregada.
Resultados	Se actualizó el estado de la tarjeta de combustible entregada en una única iteración en busca del estado “En explotación”.
Salida	True.

Tabla 12: Caso de prueba para el camino básico 3

Camino básico # 3: 1, 2, 5, 6, 7	
Descripción	Se actualiza el estado de la tarjeta que ha sido entrega para tener constancia de que no se encuentra en la Caja.
Condición de ejecución	Debe estar creada y cargada al menos una tarjeta de combustible.
Entrada	$objTarjeta = new DatTarjetacombustible(); $objTarjeta = Doctrine::getTable(' DatTarjetacombustible ')->find($idtarjeta); $objEstadoTarjeta = new NomEstadotarjeta(); $arrayobjEstado = $objEstadoTarjeta->cargarEstado();
Resultados esperados	Se debe actualizar el estado de la tarjeta de combustible entregada.
Resultados	No se actualizó el estado de la tarjeta de combustible entregada porque no encontró el estado “En explotación”, por lo que la tarjeta se mantiene con su estado actual.
Salida	False.

Resultado de la ejecución de las pruebas de caja blanca

Luego de haber ejecutado el método de camino básico a las funcionalidades más complejas en el desarrollo del módulo Tarjeta de combustible, se pudo comprobar que el flujo de trabajo de las funcionalidades es correcto, se probó que cada sentencia es ejecutada al menos una vez con los parámetros de entrada y las salidas esperadas. Con las pruebas de caja blanca se comprobó que la implementación se realizó de manera correcta, se examinó el estado de la aplicación en varios puntos de la misma determinando que el estado real coincidía con el esperado.

Prueba de caja negra

Dentro de la prueba de caja negra, se seleccionó la técnica partición de equivalencia la cual permite examinar los valores válidos e inválidos de las entradas existentes en el software. Para la aplicación de esta técnica se realizan los diseños de casos de prueba los cuales se basan en una evaluación de las clases de equivalencia para una condición de entrada.

- Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.
- Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.
- Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:
    1. Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, por encima y en el rango.
    2. Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, por encima y en el rango.
    3. Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
    4. Si una entrada es booleana, hay 2 clases: sí o no.

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases. Se realizaron 57 diseños de casos de prueba para validar el sistema atendiendo a la técnica partición de equivalencia.

Resultado de la ejecución de las pruebas de caja negra

Las pruebas de caja negra a la solución fueron desarrolladas por el Grupo de Aseguramiento de la Calidad del Centro de Informatización de Entidades (CEIGE). Estas se realizaron en dos iteraciones, en la primera iteración se detectaron 10 no conformidades 7 fueron de aplicación y 3 de documentación. De las no conformidades referentes a las de aplicación se encuentran 6 de validación y 1 de funcionalidad. En cuanto a las de documentación, las 3 no conformidades fueron de redacción. Finalmente se comprobó en la segunda iteración que el módulo estaba libre de no conformidades culminando así la fase de pruebas internas. Prueba de ello lo constituye el acta de liberación expedida por el grupo de calidad de CEIGE, consultar el Expediente de proyecto 3.4 del centro CEIGE (Baez Fernández, 2015).

3.5. Pruebas de Aceptación

Luego de la culminación de las pruebas de caja blanca y caja negra se procedió a realizar las pruebas de aceptación; estas últimas bajo un ambiente controlado son capaces de determinar si el software realiza lo que desea el cliente. Terminadas estas pruebas el cliente avaló la solución con la entrega del Acta de Aceptación del cliente, ver Anexo 1. Para esta investigación, el cliente lo constituyó las Áreas Caja, Contabilidad y Finanzas representadas por los Técnicos Generales María

M. Utra Morell y Orlando José Valdés Pérez, y los Especialistas Generales Iraida Rondón Sosa y Alexis Alcón Rodríguez todos de la UCI.

3.6. Validación de la solución

El control es el proceso de verificar el desempeño de distintas áreas o funciones de una organización. Usualmente implica una comparación entre un rendimiento esperado y un rendimiento observado, para verificar si se están cumpliendo los objetivos de forma eficiente y eficaz y tomar acciones correctivas cuando sea necesario (Control, 2015).

Para validar la solución se realizaron encuestas con el objetivo de caracterizar el control de los procesos de las tarjetas de combustible en la actualidad y después de desarrollada la propuesta de solución. Para la presente investigación se desglosa el control en los siguientes indicadores: tiempo de ejecución de los procesos relacionados con las tarjetas magnéticas para combustible, descentralización de la información, desajustes contables en los balances de comprobación y pérdida de la información.

Análisis de los resultados de la investigación

A continuación en la siguiente tabla se muestran los resultados obtenidos luego de haber tabulado la información recogida en la encuesta aplicada a 6 especialistas de las áreas de Finanzas, Caja y Contabilidad de la UCI.

Tabla 13: Comparación respecto a la variable dependiente

Indicadores	Evaluación del control de las tarjetas de combustible actualmente	Evaluación del control de las tarjetas de combustible usando la solución propuesta
Tiempo	• 5,24 horas	• 2,07 horas
Descentralización	• Finanzas • Caja • Contabilidad	• Módulo (Finanzas) • Módulo (Caja) • Módulo (Contabilidad)

Desajustes contables en los balances de comprobación	• Registro manual de los asientos contables • Registro de los asientos contables en el sistema Assets • Duplicación de información	• Integración entre los módulos del sistema XEDRO-ERP • Se generan los asientos contables de forma automática • Evitando la posibilidad de errores de cálculo al procesar la información
Pérdida de información	• Alto volumen de datos • Información vulnerable al robo, deterioro y alteración • Afectación de la integración de la información	• Bases de Datos donde se guardan todos los movimientos realizados a las tarjetas • Se emplea el sistema de seguridad Acaxia

Al realizar un análisis de los indicadores con respecto a los procesos (carga, entrega y liquidación de las tarjetas de combustibles, así como sus asientos contables) para una muestra de 90 tarjetas con una demora de 4 horas para realizar la carga, entre 4 a 5 minutos para la entrega y liquidación y 1,08 horas para realizar los asientos contables de estos procesos se obtuvo:

- La realización de los procesos de las tarjetas de combustible demora actualmente 5,24 horas, mientras que estos mismos procesos se demoran 2,07 horas usando el módulo propuesto, debido que solo se necesita 2,043 horas para realizar la carga de las tarjetas, 40, 35 y 54 segundos para efectuar la entrega, liquidación y sus asientos contables respectivamente, lo que implica una disminución de 3,10 horas aproximadamente.
- Los procesos de las tarjetas de combustible se realizan en áreas diferentes, la carga en el área de Finanzas, la entrega y liquidación en la Caja y los asientos contables de estos procesos en Contabilidad, evidenciando la descentralización de la información. La propuesta de solución centraliza la información debido a que pertenece al sistema XEDRO-ERP, el cual

cuenta con varios módulos que gestionan las áreas de una empresa, por lo que todos los procesos se podrán realizar de forma íntegra en dicho sistema.

- Las causas que provocan los desajustes contables en los balances de comprobación son: registro manual de los asientos contables, registro de los asientos contables en el sistema Assets y duplicación de información. La solución resuelve este problema al integrarse con los módulos y componentes del sistema XEDRO-ERP, generando los asientos contables de forma automática para evitar la posibilidad de cometer errores de cálculo al procesar la información.
- Actualmente la información es vulnerable al robo, deterioro y alteración, los que constituyen factores que atentan contra su integridad. La propuesta de solución contribuye a la integridad de la información al contar con una base de datos que guarda la información relacionada con las tarjetas de combustible. También emplea el sistema de seguridad Acaxia para restringir el acceso a la base de datos y a la aplicación por roles y usuarios.

Conclusiones parciales

- La integración entre componentes permitió consumir servicios de otros módulos y componentes del sistema XEDRO-ERP mediante el principio arquitectónico de inversión y control (IoC).
- Los estándares de codificación utilizados contribuyeron a la trazabilidad de los requisitos y organización del código para que otros desarrolladores puedan entenderlo.
- Se corrigieron los errores detectados durante las pruebas lo que permitió obtener un módulo con mayor calidad y confiabilidad.
- Validar la investigación demostró que el módulo Tarjeta de combustible mejora la función del control de los procesos de las tarjetas magnéticas para combustible en la UCI.

CONCLUSIONES

- La elaboración del marco teórico de la investigación, permitió conceptualizar los elementos fundamentales que fueron utilizados y evidenció la necesidad de desarrollar un módulo que se adaptara a las características del sistema XEDRO-ERP.
- Se diseñaron los modelos de la propuesta de solución para realizar una representación técnica de los requisitos identificados.
- La implementación del módulo Tarjeta de combustible permitió una integración entre componentes del sistema XEDRO-ERP.
- La verificación de la solución permitió comprobar la correcta funcionalidad del módulo Tarjeta de combustible.
- La validación de la investigación demostró que el módulo Tarjeta de combustible da respuesta a las necesidades planteadas en la problemática.

RECOMENDACIONES

Se recomienda:

- Realizar los cierres contables del módulo Tarjeta de combustible.
- Gestionar notificaciones para informar al usuario las tarjetas que están por vencer y los choferes que faltan por liquidar en un periodo determinado.

REFERENCIAS

Adolph Steve, Bramble Paul y otros. 2001. *Patters for Effective Use Cases.* s.l. : Cockburn.Highsmith, 2001. 0-201-72184-8.

Andalucía, Marco de Desarrollo de la Junta de. 2013. Junta de Andalucía. [En línea] 2013. [Citado el: 06 de 03 de 2015.] http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/415.

Baez Fernández, Liannet. 2015. Expediente de proyecto 3.4 del centro CEIGE. Universidad de las Ciencias Infórmaticas, La Habana, Cuba: s.n., 2015.

Bauta, René Rodrigo. 2011. Arquitectura de software. Vista entorno de desarrollo tecnológico. Proyecto SAUXE. La Habana: CEIGE: s.n., 2011.

Bizagi. 2015. bizagi. [En línea] 2015. [Citado el: 18 de 03 de 2015.] http://help.bizagi.com/bpmsuite/es/index.html?definir_reglas_de_negocio.htm.

Bizagi,2014. [En línea] http://www.buenastareas.com/ensayos/Bizagi/4235294.html.

Buenastareas. 2015. buenastareas. [En línea] 2015. [Citado el: 18 de 03 de 2015.] http://www.buenastareas.com/ensayos/Modelos-Conceptuales/23455.html.

Celador. 2013. Sistema Control de Combustible, Celador S2C. [En línea] 2013. [Citado el: 05 de 03 de 2015.] http:// Sistema Control de Combustible, Celador S2C _ ExpoMatanzas.html.

Computación, Universidad Central de Chile. Departamentos de ciencias de la. 2014. 2011. Universidad de Chile. Departamentos de ciencias de la computacion.Sitio Web DCC. [En. *dcc.uchile.cl.* [En línea] 2014. [Citado el: 27 de 03 de 2015.] www.dcc.uchile.cl/~psalinas/uml/modelo.html.

Control. 2015. [En línea] 10 de 05 de 2015. http://www.zonaeconomica.com/control.

Datos, Modelo de. 2012. CHRONOTECH, el futuro esta en nuestras manos. [En línea] 2012. [Citado el: 20 de 03 de 2015.] https://sites.google.com/site/jalexiscv/modelosdedatos.

DISAIC. 2008. Casa Consultora DISAIC. [En línea] 2008. [Citado el: 05 de 03 de 2015.] http://Casa Consultora DISAIC.html.

Enciclopedia cubana, EcuRed. 2009. Proceso de Desarrollo y Gestión de Proyectos de Software. [En línea] 2009. [Citado el: 04 de 03 de 2015.] http://www.ecured.cu/index.php/Pruebas_de_software.

García Ramírez, Christian y Pavón Reyes, Raydel. 2013. Componente dinámico para la interoperabilidad de procesos. La Habana: s.n., 2013.

GoF. 2013. Patrones del "Gang of Four". Facultad de Informática-Universidad Politéctica de Madrid : s.n., 2013.

Gómez Baryolo, Oiner, Morejón Borbón, Yoandry y García Tejo, Darien. 2012. Arquitectura tecnológica para el desarrollo de software. Universidad de las Ciencias informáticas, Habana, Cuba: s.n., 2012.

—. 2012. Arquitectura tecnológica para el desarrollo de software. Universidad de las Ciencias informáticas, Habana, Cuba: s.n., 2012.

Hall, Richard S., y otros. 2011. OSGi in action. Stamford : Manning Publications Co. 2011.

Jurisco Natalia, Moreno Ana M., Vegas Sira. 2005. Técnicas de evaluación de software. 2005.

Labs, Sensio. 2013. The Book. 2013.

Larman, Craig. 2004. UML y Patrones. Introducción al análisis y diseño orientado a objetos. La Habana : s.n., 2004.

Layola, William. 2006. Maestria en Sistemas de Información Gerencial. 2006.

López, Lic. Rafael Antonio Almaguer. 2013. Consultor electrónico del contador y el auditor. DISAIC. [En línea] 04 de 2013. [Citado el: 05 de 03 de 2015.] http://Consultor del Contador y el Auditor - Abril 2013/index.htm.

Moreno, Adarlis Fernández y Fontanills, Edwing Robert Odelín. 2009. Modelado de negocio y Levantamiento de requisitos del subsistema Activos Fijos Tangibles del sistema Cedrux. La Habana : s.n., 2009.

Pérez Del Pino, Ing. Sissi, Tarifa Lozano, DrC. Lourdes, Pérez Sosa, MSc. Teresa. 2012. DISEÑO Y UTILIZACIÓN DE UN SISTEMA INFORMÁTICO PARA EL CONTROL DE PORTADORES ENERGÉTICOS. Universidad de Matanzas “Camilo Cienfuegos” (UMCC). Matanzas. Cuba: s.n., 2012.

Pressman, R. 1999. Software Engineering.A Practitioner's Approach. USA: s.n., 1999.

Pressman, Roger S. 2005. *Ingenieríade Software, Un enfoque práctico.* s.l.: McGraw-Hill Companies, 2005. ISBN: 8448132149.

Rodríguez Sánchez, Ing.Tamara. 2014. Metodología de desarrollo para la Actividad productiva de la UCI. Universidad de las Ciencias Informáticas. La Habana, Cuba: s.n., 2014.

S. Pressman, Roger. 2001. Ingeniería de Software un enfoque práctico. Madrid: s.n., 2001.

Sánchez-Segura, Ana M. Moreno y Maribel. 2010. Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el momento de Arquitectónico. Madrid : Universidad Politécnica de Madrid, España: s.n., 2010.

Sierra, Paola. García, Madeline. 2012. Diagrama de componentes. 2012.

Sommerville, I. 2007. *Software Engineering. Octava Edición.* 2007.

Toro, Amador Durán. 2000. Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información. Sevilla: s.n., 2000.

ANEXOS

Anexo 1: Acta de Aceptación del cliente

UCI
Universidad de las Ciencias Informáticas

La Habana, 25 de mayo del 2015.
"Año 57 de la Revolución".

Acta de Aceptación

De una parte, las áreas de Caja, Contabilidad y Finanzas de la Universidad de las Ciencias Informáticas (UCI), representadas en este acto por la:

Técnica General María M. Utra Morell, Orlando José Valdés Pérez y los Especialistas Generales Iraida Rondón Sosa y Alexis Alcón Rodríguez y de otra parte los estudiantes Roberto Bandera Gómez y Liannet Baez Fernández.

Primero: Que en cumplimiento de los requisitos funcionales han sido efectuadas las implementaciones correspondientes.

CONSIDERANDO: Que los hitos realizados han sido desarrollados con la calidad requerida y bajo las condiciones pactadas y aprobadas por **Las Partes.**

CONSIDERANDO: Que los hitos que se han ejecutado cumplen con los requerimientos establecidos.

CONSIDERANDO: Que el producto constituye un aporte relevante para el mejoramiento del control los procesos de las tarjetas de combustible en la UCI.

POR TANTO: Las partes acuerdan formalizar mediante la presente Acta, la aceptación del producto: Módulo Tarjeta de combustible del sistema Xedro-ERP.

Y para que así conste, se extiende la presenta Acta en dos (2) ejemplares, rubricados por **Las Partes.**

Roberto Bandera Gómez

Liannet Baez Fernández

María M. Utra Morell

Orlando José Valdés Pérez

Alexis Alcón Rodríguez

Iraida Rondón Sosa

UCI Universidad de las Ciencias Informáticas
DEPARTAMENTO DE FINANZAS

Anexo 2: Encuesta aplicada a las áreas de Finanzas, Caja y Contabilidad de la UCI.

Encuesta:

La siguiente encuesta tiene como objetivo caracterizar la gestión de los procesos de tratamiento de las tarjetas magnéticas para combustible en la Universidad de las Ciencias Informáticas (UCI). Entiéndase como procesos la Carga, Entrega, Liquidación y Control de tarjetas magnéticas para combustible. Por ello le pedimos su contribución y que sea lo más sincero posible en sus planteamientos. Le garantizamos confidencialidad y anonimato. Gracias de antemano por su colaboración.

Datos de interés:

Área a la que pertenece: _____ Rol: _____

Preguntas:

1. ¿Los procesos de tratamiento de las tarjetas magnéticas para combustible se realizan en áreas diferentes?

Sí___ No___

a. En caso de ser positiva la respuesta especificar en qué área se realizan cada uno por separado.

1.1. ¿Las áreas se relacionan unas con otras?

Sí___ No___

a. En caso de ser positiva la respuesta explicar brevemente como ocurre este proceso.

2. ¿Qué tiempo se demora realizar un informe de carga de un mes determinado?

Tiempo (Horas): _____

3. ¿Qué tiempo se demora realizar los proceso de entrega y liquidación de una tarjeta magnética para combustible?

Tiempo (Horas): Entrega: _____ Liquidación: _____

3.1 ¿Existen varios periodos para realizar la liquidación de las tarjetas?

Sí___ No___

a. En caso de ser positiva la respuesta especificar cuáles son.

4. ¿Qué tiempo se demora conocer la información referente a una tarjeta magnética para combustible con respecto al chofer, responsable, consumo, chapa del vehículo asociado a la misma entre otras?

Tiempo (Horas): _____

4.1. ¿Obtener la información anterior es un proceso engorroso?

Sí___ No___

a. En caso de ser positiva la respuesta explicar brevemente las razones.

5. ¿En qué formato se registra la información relacionada con los procesos de las tarjetas magnéticas para combustible?

Excel___ Word___ txt___ PDF____

5.1 ¿Se genera mucha documentación en la realización de dichos procesos?

Sí___ No___

a. En caso de ser positiva la respuesta mencionar algunos inconvenientes que esto puede causar.

6. ¿Actualizar los asientos contables de los procesos de las tarjetas magnéticas para combustible en el sistema Assets, además de tenerlos registrados en formato duro (papel), propicia en ocasiones obtener desajustes contables en los balances de comprobación?

Sí___ No___

a. En caso de ser positiva la respuesta mencionar que consecuencia trae consigo obtener desajustes contables en los balances de comprobación.

7. ¿Qué ventajas a su consideración cree que le puede propiciar el desarrollo de un software que informatice los procesos relacionados con la gestión de las tarjetas magnéticas para combustible?