Universidad de las Ciencias Informáticas

Facultad 3

Centro de Informatización de Entidades (CEIGE)

Componente Gestor del Movimiento de la Mercancía para el Sistema de Importación y Exportación Xedro-Apside v1.0

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autor:

Leonardo Argumedo Reloba

Tutores:

Ing. Virtudes Milagro Figueredo Lara

Ing. Yoendry Betancourt Peña

La Habana, Junio 2015

"Año 57 de la Revolución"

"Inteligencia más carácter,

el objetivo de la verdadera educación"

Martin Luther King

Agradecimientos

"Lo que diga o haga nunca será suficiente para agradecer todo lo que han hecho por mí".

A mi papá que siempre me ha apoyado y ha sabido encaminarme en la vida.

A mi cuñado y mis sobrinos por su apoyo y cariño.

A mis abuelos por los consejos y enseñanzas.

A mis primos, tíos y tías, en especial a mi tía Zaida que ha sido como una madre.

A mis amigos y hermanos del politécnico y la universidad, a Yany que ha sido como mi hermana.

A mi novia por todo el cariño y paciencia que me ha brindado.

A Beatris y Betsy por aguatarme cada fin de semana.

A mis tutores y mi oponente por la ayuda brindada en estos meses.

A mi hermana del alma Aniuska, por siempre estar ahí y saberme cuidar como lo hizo nuestra madre, mientras pudo.

Por último y más importante a mi mamá, que es el gran amor de mi vida, me educó y enseñó; gracias a ella estoy aquí convirtiéndome en ingeniero.

Dedicatoria

Le dedico este trabajo a todas las personas que han ayudado a que hoy pueda escribir estas líneas.

En especial a mi mamá y mi papá por apoyarme en el trayecto de mi vida y por siempre haber confiado en mí.

A mi hermana Aniuska, mi tía Zaida y Nelita, por todo el cariño y ayuda que siempre me han brindado y por saber ocupar el lugar de mi mamá mientras ella no ha estado.

A mis sobrinos para que le sirva de ejemplo y el día de mañana puedan ser unos dignos profesionales.

A mis abuelos por enseñarme el camino y ofrecerme tantos consejos.

A todos ellos y a mis compañeros y profesores de escuela, desde la primaria les dedico este trabajo.

Declaración de autoría.

Por este medio declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo. Para que así conste firmo la presente a los __ días del mes de ___ del año 2015**.**

Firma del autor

Ing.

Ing.

Resumen

Los sistemas de gestión son herramientas de utilidad para empresas tanto a nivel nacional como internacional. Un sistema de gestión permite mejorar el flujo de información entre las áreas de una entidad y un uso adecuado de estos sistemas puede lograr un incremento en los indicadores de productividad. El Sistema de Importación y Exportación Xedro-Apside v1.0 es un sistema de gestión destinado a las entidades comercializadoras cubanas. Uno de los objetivos del Xedro-Apside v1.0, es facilitar la gestión secuencial y centralizada de los procesos de Importación de la mercancía y lograr un mayor seguimiento y control del cumplimiento de los ciclos comerciales; para este fin se desarrolló el Componente Gestor del Movimiento de la Mercancía. Dicho componente permite el seguimiento de la mercancía, ya que se puede conocer en cada momento, en qué estado se encuentra el movimiento e ir pasando por cada uno de ellos hasta que la mercancía es facturada. Otras funcionalidades que posee el componente, es la de asignar un apoderado de aduana al movimiento y exportar en formato Excel los datos de los movimientos que se deseen. Además el componente genera una serie de gráficas con datos estadísticos en fechas determinadas por el usuario, lo que facilita y favorece la toma de decisiones a la gerencia de la entidad.

Palabras claves: Importación, Movimiento de la Mercancía, Sistema de Gestión, Xedro-Apside

Contenido

Resumen IV

Contenido V

Introducción 1

Capítulo 1. Fundamentación Teórica 6

1.1 Marco conceptual 6

1.2 Estudio de sistemas de comercialización 6

1.2.1 Software de Administración de Cumplimiento de Regulaciones de Importación (Integration Point) 7

1.2.2 StockBase POS 8

1.2.3 Mistral Import 1.0.1.96 8

1.3 Valoración del estudio de tendencia de sistemas de comercialización 9

1.4 Modelo de desarrollo de software 11

1.4.1 Fases 11

1.4.2 Disciplinas 11

1.5 Herramientas y Tecnologías para el desarrollo de la aplicación 12

1.5.1 Marco de trabajo SAUXE v2.2 12

1.5.2 Marco de trabajo ExtJS 4.2 13

1.5.3 Marco de trabajo Zend Framework 1.5 13

1.5.4 Marco de trabajo Doctrine 1.2.2 13

1.5.5 Lenguajes de programación 13

1.5.6 Biblioteca PHPExcel 1.8.0 14

1.5.7 Lenguaje de Modelado UML 2.1 14

1.5.8 Visual Paradigm for UML 8.0 14

1.5.9 Servidor de aplicaciones Apache 2.2 15

1.5.10 Sistema Gestor de Base de Datos PostgreSQL 9.2 15

1.5.11 Navegador Web Mozilla Firefox 30.0 15

1.5.12 Entorno de Desarrollo Integrado (IDE) NetBeans 8.0 15

1.6 Patrones 16
1.7 Conclusiones parciales 17
Capítulo 2. Análisis y Diseño de la Solución Propuesta 18
2.1 Modelo conceptual 18
2.2 Requisitos de software 19
2.2.1 Técnicas de captura de requisitos 19
2.2.2 Requisitos funcionales 19
2.2.3 Requisitos no funcionales 23
2.2.4 Técnicas de validación de requisitos 25
2.3 Diagrama de clases del diseño 25
2.4 Patrones utilizados 27
2.4.1 Patrón arquitectónico 27
2.4.2 Patrones de diseño 27
2.5 Modelo de datos 28
2.6 Conclusiones parciales 29
Capítulo 3. Implementación y Validación de la Solución Propuesta 31
3.1 Diagrama de componentes 31
3.2 Diagrama de despliegue 32
3.3 Estándares de codificación 32
3.3.1 Nomenclatura de las clases 32
3.3.2 Nomenclatura de las funciones y las variables 33
3.4 Interfaces del componente 33
3.5 Validación del diseño 36
3.5.1 Métrica relaciones entre clases (RC) 36
3.5.2 Resultados de la aplicación de la métrica RC 37
3.6 Pruebas de software 40
3.6.1 Pruebas de caja blanca 41
3.6.3 Pruebas de caja negra 44
3.7 Pruebas de aceptación con el cliente 48

3.8 Aporte de la solución y beneficios sociales 48

3.9 Conclusiones parciales 48

Recomendaciones.....50

Bibliografía 51

Anexos 53

Índice de Tablas

Tabla 1. Descripción del Software de Administración de Cumplimiento de Regulaciones de Importación 7

Tabla 2. Descripción de StockBase POS 8

Tabla 3. Descripción de Mistral Import 8

Tabla 4. Resumen del estudio de tendencia.....10

Tabla 5. Descripción del RF6 Graficar estadística de los movimientos de carga diario.....21

Tabla 6. Descripción del diagrama de clases del diseño 26

Tabla 7. Métrica RC 37

Tabla 8. Resultados de la aplicación de la métrica RC.....38

Tabla 9. Resultados de las pruebas de caja blanca.....43

Tabla 10. Diseño de caso de prueba “Exportar movimiento de carga diario” 45

Índice de Figuras

Figura 1. Marco de Trabajo SAUXE 12
Figura 2. Marco conceptual 18
Figura 3. Complejidad de los requisitos funcionales 21
Figura 4. Prototipo de interfaz para el RF6, fechas de recibo de los documentos de embarque 22
Figura 5. Prototipo de interfaz para el RF6, gráficos estadísticos 23
Figura 6. Diagrama de clases del diseño 26
Figura 7. Diagrama entidad-relación 29
Figura 8. Diagrama de componentes 31
Figura 9. Diagrama de despliegue 32
Figura 10. Método asignarApoderadoAction(), notación *camelCasing* 33
Figura 11. Interfaz, funcionalidad listar de movimientos 33
Figura 12. Interfaz, funcionalidad actualizar movimiento 34
Figura 13. Interfaz, funcionalidad asignar apoderado de aduana 34
Figura 14. Ejemplo de un documento Excel generado por el componente 35
Figura 15. Ejemplo de una gráfica de pastel generada por el componente 35
Figura 16. Ejemplo de una gráfica de barra generada por el componente 36
Figura 17. Resultados de la métrica RC, indicador Acoplamiento 38
Figura 18. Resultados de la métrica RC, indicador Complejidad de mantenimiento 39
Figura 19. Resultados de la métrica RC, indicador Reutilización 39
Figura 20. Resultados de la métrica RC, indicador Cantidad de pruebas 40
Figura 21. Método asignarApoderadoAction() con los nodos 42
Figura 22. Grafo de flujo del método asignarApoderadoAction() 42
Figura 23. Resultados de las pruebas de caja negra, pruebas internas 47
Figura 24. Resultados de las pruebas de caja negra, pruebas de liberación 47

Introducción

Los avances de la ciencia y las tecnologías influyen directamente en el progreso de la humanidad. Los últimos años han marcado un adelanto significativo en el campo de las Tecnologías de la Informática y las Comunicaciones (TIC), lo que posibilita una mejor gestión de los procesos, tanto a nivel gubernamental, empresarial como particular. Muchas corporaciones y empresas han llevado a cabo procesos de informatización, para lograr mejoras en los indicadores de productividad y así lograr mayores ingresos.

La informática se ha insertado en cada sector de la sociedad, posibilitando satisfacer necesidades de comunicación y lograr beneficios económicos. Cuba no está ajena a este fenómeno y el gobierno ha tomado medidas que posibilitan el fortalecimiento de la industria cubana del software, que a su vez tiene como uno de sus principales objetivos la informatización de la sociedad. Varios centros e instituciones tienen como una de sus misiones apoyar el proceso de informatización, ejemplo de ello es la Universidad de las Ciencias Informáticas (UCI).

La UCI está envuelta en el desarrollo de disímiles proyectos informáticos; con ello aporta a la economía del país y brinda beneficios a la sociedad cubana. Dicha Universidad posee centros de producción de software, cada uno de ellos dedicados a sectores específicos, distribuidos por facultades. El Centro de Informatización de Entidades (CEIGE) perteneciente a la Facultad 3, tiene como misión: “Desarrollar productos, servicios y soluciones informáticas para la gestión de entidades, contribuyendo a la formación de estudiantes y profesionales, permitiendo un posicionamiento en el mercado nacional e internacional (1)”. Dentro de los proyectos que se desarrollan están los sistemas de gestión para determinadas empresas o instituciones, los cuales posibilitan ahorro de tiempo y de esfuerzo, y pueden mejorar el flujo de información entre las áreas de una entidad.

El Sistema de Importación y Exportación Xedro-Apside v1.0 es un proyecto de gestión desarrollado en el CEIGE. El mismo está dirigido a las entidades comercializadoras cubanas, tiene como objetivo ser un sistema para la Gestión Comercial de la Importación y la Exportación, facilitando la gestión secuencial y centralizada de los procesos de Importación de la mercancía. También logrará una mayor coordinación y conciliación de la información generada durante las operaciones comerciales, así como un mayor seguimiento y control del cumplimiento de los ciclos comerciales y la disminución de los tiempos de gestión de los documentos de la cartera de clientes y proveedores (1). El sistema anteriormente mencionado cuenta con un módulo para gestionar las importaciones, permite generar reportes referentes a los proveedores, productos, solicitudes de importación, entre otros aspectos de interés. Uno de los clientes del Sistema de Importación y Exportación Xedro-Apside v1.0 es la empresa BK Import-Export perteneciente a la Cámara de Comercio de la República de Cuba.

La empresa BK Import-Export se encarga de la importación y exportación de mercancías, según establece la Resolución 50 del 2014. Entre los procesos que desarrolla dicha empresa se encuentra el proceso de Importación, en el cual se realizan las actividades de tramitación aduanal y recepción de la mercancía. Este proceso genera gran cantidad de documentos, que en su mayoría son elaborados de forma manual, por lo que están propensos a contener errores de redacción y propiciar incoherencia en los registros de entrega, recepción y solicitud de las mercancías. Además no se cuenta con una base de datos centralizada para la manipulación y gestión de los datos generados, puesto que la información es almacenada en formato duro o en hojas de cálculo utilizando la herramienta Microsoft Excel. Un problema identificado es la poca integridad de la información ya que los documentos pueden ser manipulados por personas no autorizadas, esto ocurre por la ausencia de un mecanismo que establezca niveles de accesibilidad a dichos documentos. Los problemas antes mencionados generan consecuencias negativas en la ejecución del proceso de Importación: se afecta el seguimiento y control de la mercancía y a su vez se incrementa el tiempo de ejecución y supervisión del proceso. Lo que conlleva a la siguiente interrogante:

¿Cómo lograr el seguimiento y control de los movimientos de carga durante las actividades de tramitación aduanal y recepción de la mercancía que se efectúan en el proceso de Importación gestionado por las entidades comercializadoras cubanas?

Se propone como **objeto de estudio** de la investigación: la actividad comercial de importación de la mercancía y como **campo de acción:** el proceso de tramitación aduanal y recepción de la mercancía.

Para dar solución al problema planteado se define como **objetivo general:** desarrollar el Componente Gestor del Movimiento de la Mercancía para el Sistema de Importación y Exportación Xedro-Apside v1.0, que permita el seguimiento y control de los movimientos de carga durante las actividades de tramitación aduanal y recepción de la mercancía en el proceso de Importación gestionado por las entidades comercializadoras cubanas.

A su vez para darle cumplimiento al objetivo general se abordarán los siguientes **objetivos específicos (OE):**

1. Elaborar el marco teórico de la investigación.
2. Realizar el análisis y diseño del Componente Gestor del Movimiento de la Mercancía.
3. Implementar el Componente Gestor del Movimiento de la Mercancía en el Sistema de Importación y Exportación Xedro-Apside v1.0.
4. Validar la propuesta de solución a través de métricas, pruebas de software y pruebas de aceptación con el cliente.

Para desarrollar los objetivos planteados se llevarán a cabo las siguientes **tareas investigativas**, distribuidas por objetivos específicos:

1. Estudio del estado del arte referente a soluciones informáticas relacionadas con el campo de acción especificado.
2. Análisis de la arquitectura definida para el Sistema de Importación y Exportación Xedro-Apside v1.0 para conocer las características fundamentales, marcos de trabajo, herramientas y tecnologías definidas para el desarrollo, así como la estrategia de integración entre los diferentes componentes.
3. Estudio de la metodología de desarrollo de software definida para la UCI.
4. Estudio de los patrones de diseño.
5. Elaboración del Modelo conceptual asociado a la solución.
6. Levantamiento y especificación de requisitos.
7. Diseño de los prototipos de interfaz de usuario.
8. Estudio y actualización del Modelo de Datos del Sistema de Importación y Exportación Xedro-Apside v1.0.
9. Elaboración del diseño de las clases asociadas a la solución.
10. Implementación de los requisitos identificados.
11. Selección de los métodos y técnicas de pruebas para validar la solución.
12. Validación de la solución.

Se plantea como **idea a defender** que si se desarrolla el Componente Gestor del Movimiento de la Mercancía para el Sistema de Importación y Exportación Xedro-Apside v1.0, se logrará un seguimiento y control de los movimientos de carga durante las actividades de tramitación aduanal y recepción de la mercancía que se efectúan en el proceso de Importación gestionado por las entidades comercializadoras cubanas.

Para dar cumplimiento a los objetivos y tareas planteadas se utilizaron varios métodos de investigación, los cuales se relacionan a continuación.

Métodos teóricos:

- **Histórico-Lógico:** la utilización de este método fue esencial para estudiar el comportamiento y funcionamiento de los sistemas que llevan a cabo actividades comerciales de importación de mercancías.
- **Analítico-Sintético:** posibilita dividir la investigación en sus múltiples relaciones y después unir las partes analizadas para obtener nuevo conocimiento. Este método se utilizó para realizar el análisis de documentos y leyes que rigen el proceso de Importación de mercancías en Cuba.
- **Inductivo-Deductivo:** a partir del estudio de hechos aislados, permite hallar proposiciones generales y después con ello arribar a conclusiones particulares que pueden demostrarse en la práctica. Este método fue utilizado para realizar generalización de la teoría estudiada y llegar a conclusiones específicas, que posibilitaron conocer la necesidad de implementar el componente y su posterior desarrollo.
- **Modelación:** se empleó para la representación gráfica de los productos de trabajo necesarios en las disciplinas por las que transitó la investigación. Su uso permitió entender con más claridad el negocio, lo que facilitó el trabajo en el momento de implementar la solución.

Métodos empíricos:

- **Entrevista:** se aplicó a especialistas del sector logístico de la empresa BK Import-Export para obtener conocimientos del negocio y aportar al proceso de obtención de requisitos.

El documento está estructurado en tres capítulos que a continuación se describen:

- **Capítulo 1. Fundamentación Teórica**

El Capítulo 1 expone conceptos fundamentales para entender el problema en cuestión. Se realiza un estudio a diferentes sistemas de comercialización, que posibilitan ampliar el conocimiento en cuanto a sus funcionalidades y ello permite una mejor toma de decisiones en el diseño y la implementación del componente. Se explica además la metodología de desarrollo a seguir, se describen las herramientas y tecnologías a utilizar.

- **Capítulo 2. Análisis y Diseño de la Solución Propuesta**

En el Capítulo 2 se realiza un estudio del negocio que posibilita mejorar el entendimiento del mismo. En este capítulo se generan una serie de productos de trabajo esenciales para el comienzo de la

implementación. Además se aplicaron y explicaron patrones para el diseño y construcción de algunos productos de trabajo.

- **Capítulo 3. Implementación y Validación de la Solución**

En el Capítulo 3 se explican los estándares de codificación a utilizar en la implementación del componente. Se realizan todas las validaciones de la propuesta de solución, se exponen los resultados obtenidos de las pruebas realizadas y los análisis de dichos resultados. Además se explica el aporte y beneficios sociales logrados con el componente y se presenta el resultado de las pruebas de aceptación con el cliente.

Capítulo 1. Fundamentación Teórica

En este capítulo se abordan descripciones referentes al problema en cuestión; así como conceptos necesarios para la comprensión de dicho problema. Se realiza un estudio a sistemas de comercialización y con ello una valoración de los mismos. Además se explica la metodología de desarrollo de software a utilizar y las tecnologías y herramientas empleadas para la modelación e implementación del componente.

1.1 Marco conceptual

En el marco conceptual se muestran conceptos esenciales, para una mejor comprensión del contenido del capítulo.

- **Comercialización:** es la acción y efecto de comercializar (poner a la venta un producto o darle las condiciones y vías de distribución para su venta). Por ejemplo: “La empresa LABIOFAM comenzará la comercialización de un nuevo producto en los próximos días”, “La comercialización del vino de arroz fue un éxito”, “Tenemos un buen producto, pero todavía fallamos en la comercialización” (2).

- **Importación:** es un término que procede del verbo importar (introducir productos o costumbres extranjeras en un país). Se trata de la acción de importar mercancías o cuestiones simbólicas de otra nación. Por ejemplo: “El gobierno planea introducir trabas a la importación de calzado para no perjudicar a los productores locales” (3).

1.2 Estudio de sistemas de comercialización

En la actualidad existen varios sistemas de comercialización, los cuales favorecen la gestión de los procesos involucrados en esta actividad. Los sistemas a estudiar pueden lograr mejoras en los indicadores de eficiencia y productividad, por lo que son utilizados en múltiples empresas tanto a nivel nacional como internacional. En este acápite se describen y analizan distintos sistemas de comercialización, con el propósito de identificar requisitos, para la informatización del proceso de Importación. De los mismos se analizarán los siguientes aspectos:

- Sistema Operativo (SO) en el cual funciona.
- Empresa desarrolladora.
- Licencia.
- Funcionalidades principales para el proceso de Importación:

  F.1 Gestión de documentos.

  F.2 Gestión de contratos.

F.3 Gestión de contenedores.

F.4 Gestión de carga de la mercancía.

F.5 Representación gráfica de datos estadísticos.

1.2.1 Software de Administración de Cumplimiento de Regulaciones de Importación (Integration Point)

Integration Point propone una solución muy completa, diseñada para facilitar el flujo de información de importaciones y exportaciones a través de la cadena de suministros global, en tiempo real. El software administra el proceso completo, desde la preparación de las entradas, a través del llenado de documentos para Aduanas, hasta la validación después de la entrada, al consolidar toda la información necesaria en un sistema automatizado (4).

Tabla 1. Descripción del Software de Administración de Cumplimiento de Regulaciones de Importación

SO en el cual funciona	Windows
Empresa desarrolladora	Integration Point
Licencia	Privativa
Funcionalidades principales para el proceso de Importación	- Gestión del proceso de Importación completo. - Cuenta con acceso a información de clasificación global y específica de cada país vía internet. - Utiliza información de comercio internacional actualizada y en tiempo real para la clasificación, preferencias arancelarias y tratados de libre comercio. - Da seguimiento a la comunicación con el agente aduanal. - Contacta con Autoridades Aduaneras alrededor del mundo. - Gestiona los documentos relativos al proceso aduanal.

1.2.2 StockBase POS

StockBase POS es un sistema fácil de utilizar, muy intuitivo y de interfaz amigable. Este software permite la gestión de inventarios, clientes, proveedores, cuentas corrientes, vendedores, cajeros, compras y ventas. Fue creado en 1994 y hasta el momento es utilizado por más de 10 000 empresas en 21 países (5).

Tabla 2. Descripción de StockBase POS

SO en el cual funciona	Windows, se está trabajando en próximas versiones que serán compatibles con Linux y MacOS.
Empresa desarrolladora	EGA Futura
Licencia	Privativa, pero el producto tiene una versión gratis de por vida
Funcionalidades principales para el proceso de Importación	No gestiona nada referente al proceso de Importación

1.2.3 Mistral Import 1.0.1.96

Mistral Import es una aplicación orientada a la Gestión Comercial y logística de una central de compras; diseñada para pequeñas y medianas empresas. Este sistema informático agiliza la cadena logística automatizando el proceso de compras (6). Además en Cuba es empleada por varias empresas.

Tabla 3. Descripción de Mistral Import

SO en el cual funciona	Windows
Empresa desarrolladora	Mistral Caribe Holding S.A
Licencia	Privativa
Funcionalidades principales para el proceso de Importación.	- Control de contratos de proveedores y clientes. - Condiciones de pago, cronograma de entregas, e incidencias. - Generación por el sistema del texto del contrato.

	- Control del cumplimiento del proveedor en tiempo y con la calidad requerida de las entregas pactadas. - El sistema permite gestionar las solicitudes de pago al departamento económico para cumplir con las obligaciones de pago contraídas. - Posee un potente módulo orientado al control de los embarques y gestión de contenedores. - Utilización de plantillas para representar los modelos de control que posee la empresa y la impresión de estos desde la aplicación. (Excel generado por el sistema con la solicitud de oferta y pliego de concurrencia). - El sistema posee la capacidad de auto-documentarse con los documentos legales, de referencia y de consulta. - Obtención del cronograma de pagos pendientes de las facturas recibidas, según condiciones de pago establecidas en el contrato. - Referencias cruzadas entre puertos y aeropuertos permitiendo conocer duración estimada de travesías en días, de un origen a un destino.

1.3 Valoración del estudio de tendencia de sistemas de comercialización

En la siguiente tabla se resume el estudio de las herramientas vistas en el acápite anterior:

Tabla 4. Resumen del estudio de tendencia

Sistema	SO en el cual funciona.	Empresa desarrolladora.	Licencia	Funcionalidades				
				F.1	F.2	F.3	F.4	F.5
Integration Point.	Windows	Integration Point	Privativa	Si	Si	No	Si	No
Stock Base POS	Windows	EGA Futura	Privativa	No	No	No	No	No
Mistral Import 1.0.1.96	Windows	Mistral Caribe Holding S.A	Privativa	Si	Si	Si	Si	No

A partir del estudio realizado a los diferentes sistemas se analizó que:

- El software de la empresa Integration Point hace buen manejo y control del proceso de Importación, ayudando en la generación de documentos necesarios en este proceso.
- El sistema Mistral Import 1.0.1.96 gestiona bien los documentos involucrados en los procesos que implementa. Al ser utilizado por empresas cubanas se puede obtener retroalimentación en cuanto a su utilización.
- Las licencias que utilizan son privativas.
- El sistema Mistral Import 1.0.1.96 implementa el 80% de las funcionalidades seleccionadas para el estudio; el de Integration Point implementa el 60% y el de EGA Futura aunque es muy utilizado para la comercialización, no implementa ninguna de las funcionalidades seleccionadas para el estudio.
- Ninguno de los sistemas es multiplataforma.
- Aunque el proceso de Importación es parecido en todo los países, porque hay normas internacionales que lo regulan, en Cuba existen regulaciones internas que ninguno de estos sistemas implementa.

Sobre la posible solución:

- Debe incluir las cinco funcionalidades evaluadas a los sistemas estudiados.

- Gestionar el subproceso de movimiento de la mercancía, para con ello facilitar el seguimiento y control de la misma y brindar reportes de los distintos estados por los que transcurre la mercancía en dicho movimiento.

- El mismo deberá estar incorporado al Sistema de Importación y Exportación Xedro-Apside v1.0, el cual está desarrollándose con tecnologías libres, es multiplataforma y cuenta con un módulo destinado a gestionar la seguridad, que permitirá establecer niveles de accesibilidad a los datos.

- Brindará facilidades en el manejo de la documentación que se necesite generar.

- Contará con plantillas predefinidas, lo cual ahorra tiempo en el proceso y elimina errores de redacción.

1.4 Modelo de desarrollo de software

El modelo de desarrollo de software es un estándar que establece las distintas fases por las que debe transitar un proyecto durante su ciclo de vida, así como el conjunto de artefactos a generar en cada una de ellas (7). La metodología a utilizar en la elaboración del componente será la "Metodología de desarrollo para la Actividad Productiva de la UCI". Esta metodología se basa en una variación de la metodología AUP (Proceso Unificado Ágil, siglas en inglés) en unión con el modelo CMMI-DEV v1.3 (Integración de Modelos de Madurez de Capacidades, siglas en inglés), con el objetivo de lograr una adaptación a la UCI y contribuir así al proceso de mejoras de la Universidad. Cuenta con tres fases, ocho disciplinas y 11 roles. A continuación se describen las fases y las disciplinas (8):

1.4.1 Fases

De las cuatro fases que propone AUP (Inicio, Elaboración, Construcción, Transición), se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes tres fases de AUP en una sola, la que se llamará Ejecución y se agrega una fase de Cierre (8).

1.4.2 Disciplinas

AUP propone siete disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener ocho disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP, están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba, se desagrega en tres disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional. Las restantes

tres disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el Nivel II, serían: CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (8).

1.5 Herramientas y Tecnologías para el desarrollo de la aplicación

Para la elaboración de un proyecto es necesario escoger las herramientas y tecnologías a emplear, estas deben satisfacer las necesidades elementales para la confección del proyecto. Las tecnologías y herramientas a utilizar están concebidas por el proyecto Sistema de Importación y Exportación Xedro-Apside v1.0; su uso apoya la política de soberanía tecnológica que impulsa el gobierno cubano, ya que poseen licencias de software libre.

1.5.1 Marco de trabajo SAUXE v2.2

El marco de trabajo SAUXE da solución a un sin número de escenarios o aspectos arquitectónicos como:

- Integración de componentes de forma distribuida o no distribuida.
- Acceso a bases de datos a través de una capa de abstracción.
- Gestión dinámica de las trazas generadas por los sistemas.
- Implementación de mecanismos de autenticación y autorización.
- Implementación de mecanismos de mensajería y control de excepciones.
- Gestión y configuración de flujos de trabajo.

Además cuenta con una arquitectura en capas como se muestra en la *Figura 1*. Su arquitectura está basada en el patrón MVC (Modelo Vista Controlador). SAUXE contiene 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. Siguiendo el paradigma de independencia tecnológica por el cual apuesta el país, utiliza las siguientes tecnologías libres (9):

Figura 1. Marco de Trabajo SAUXE

1.5.2 Marco de trabajo ExtJS 4.2

En el mundo de aplicaciones Web, ExtJS destaca fácilmente como una biblioteca de JavaScript que ofrece a los desarrolladores un potente conjunto de herramientas. Posibilita la creación de una interfaz de usuario personalizable, similares a los encontrados en sistemas operativos de escritorio, un modelo efectivo de enlace de datos, una interfaz completa para manipulación del DOM (Modelo de Objetos del Documento, siglas en inglés) y la comunicación con el servidor. Los datos son obtenidos con AJAX a través de XML. Una de las grandes ventajas de utilizar ExtJS, es que permite crear aplicaciones complejas utilizando componentes predefinidos. La carga de procesamiento se distribuye, permitiendo que el servidor pueda manejar más clientes al mismo tiempo (10).

1.5.3 Marco de trabajo Zend Framework 1.5

Se utilizará el marco de trabajo Zend Framework para el manejo de la lógica de negocio. Este marco de trabajo de código abierto brinda facilidades de uso y poderosas funcionalidades. Está diseñado para la versión 5 de PHP y posee buenas capacidades de ampliación. Proporciona un sistema de caché de forma que se puedan almacenar diferentes datos, así como los componentes que forman la infraestructura del patrón Modelo-Vista-Controlador. Consta de mecanismos de filtrado y validación de entradas de datos. Permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX y provee capacidades de búsqueda sobre documentos y contenidos (11).

1.5.4 Marco de trabajo Doctrine 1.2.2

Para la capa de acceso a datos se empleará Doctrine. Es un sistema ORM (Mapeo de Objeto Relacional, siglas en inglés) para PHP 5.2 o superior que incorpora una DBL (Capa de Abstracción a Base de Datos, siglas en inglés). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las consultas de la base de datos orientada a objeto. Esto proporciona una alternativa poderosa a diseñadores de SQL, manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Además, exporta una base de datos existente a sus clases correspondientes (12).

1.5.5 Lenguajes de programación

Un “lenguaje de programación” es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo (13).

Lenguaje de programación del lado cliente JavaScript 1.8:

JavaScript es un lenguaje ligero e interpretado, orientado a objetos. Es un lenguaje script multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientado a objetos e imperativo. El JavaScript estándar es ECMAScript. A partir de 2012, todos los

navegadores modernos soportan completamente ECMAScript 5.1. JavaScript es el lenguaje de programación que utiliza ExtJS (14).

Lenguaje de programación del lado servidor PHP 5.4:

PHP es un lenguaje de programación de propósito general, muy popular. Está diseñado para el desarrollo web del lado del servidor. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede utilizarse en la mayoría de los servidores web, al igual que en casi todos los sistemas operativos y plataformas sin ningún costo. Es considerado uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy (15).

1.5.6 Biblioteca PHPExcel 1.8.0

PHPExcel es una biblioteca creada en PHP que permite exportar y leer diferentes formatos excel como por ejemplo: XLS, XLSX, CSV y HTML. Para utilizar la librería es necesario tener instalado PHP 5.2 o superior. Utiliza una licencia de código abierto. Contiene una gran diversidad de funcionalidades dentro de las que se encuentran: la formulación de celdas, la generación de gráficas y el cambio de formato a los datos y las celdas. Es una librería fácil de utilizar y aprender aunque la mayor parte de la documentación está en idioma inglés (16).

1.5.7 Lenguaje de Modelado UML 2.1

El UML (Lenguaje de Modelado Unificado, siglas en inglés) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar aspectos conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reutilizables (17).

1.5.8 Visual Paradigm for UML 8.0

Se empleará Visual Paradigm como herramienta CASE (Computer Aided Software Engineering). Utiliza UML como lenguaje de modelado, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Ayudando a construir aplicaciones de calidad más rápido y con bajo costo. Visual Paradigm también permite modelar el negocio orientado a procesos usando la notación BPMN (Notación del Modelado de Procesos de Negocio, siglas en inglés) (18).

1.5.9 Servidor de aplicaciones Apache 2.2

Se utilizará como servidor web Apache 2.2, pues es una tecnología gratuita de código abierto compatible con muchos Sistemas Operativos. Tiene todo el soporte que se necesita para tener páginas dinámicas; permite personalizar las respuestas ante los posibles errores que se puedan dar en el servidor. Tiene una alta capacidad de configuración en la creación y gestión de registros de actividad. Apache permite la creación de ficheros de registro a medida del administrador, teniendo así un mayor control sobre lo que sucede en el servidor (19).

1.5.10 Sistema Gestor de Base de Datos PostgreSQL 9.2

Como Sistema Gestor de Base de Datos (SGBD) se empleará la versión 9.2 de PostgreSQL. Este es un sistema de gestión de bases de datos relacional orientada a objetos. Es una herramienta de código abierto, de bajo costo y multiplataforma. Se destaca en ejecutar consultas complejas y uniones de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Soporta transacciones, claves ajenas con comprobaciones de integridad referencial y almacenamiento de objetos de gran tamaño (20).

1.5.11 Navegador Web Mozilla Firefox 30.0

Mozilla Firefox es un navegador web de código abierto y software libre, desarrollado por la Fundación Mozilla. Proviene de Mozilla Application Suite, luego de que este proyecto fuera abandonado el 10 de marzo del 2005. El proyecto cuenta con miles de colaboradores en todo el mundo, lo que permite que cualquier error sea detectado y corregido rápidamente. Firefox es compatible con varios estándares web, incluyendo HTML, XML, XHTML, SVG 1.1 (parcial), CSS 1, 2 y 3, ECMAScript (JavaScript), DOM, MathML, DTD, XSLT, XPath, e imágenes PNG con transparencia alfa. Firefox también incorpora las normas propuestas por el WHATWG, y canvas element (21).

1.5.12 Entorno de Desarrollo Integrado (IDE) NetBeans 8.0

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000. NetBeans IDE es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso (22).

1.6 Patrones

Un patrón es una regla que consta de tres partes y expresa una relación entre un contexto, un problema y una solución (23). El empleo de patrones enriquece el diseño y le agrega calidad, garantizando estándares y buenas prácticas.

Patrón arquitectónico

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular, el cual es recurrente dentro de un cierto contexto. Este esquema se especifica describiendo los componentes, con sus responsabilidades y relaciones (23).

El patrón MVC (Modelo Vista Controlador), es utilizado para implementar sistemas donde se requiere el uso de interfaces y permite crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos. Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, llamadas: Modelo, Vista y Controlador (24).

El **Modelo** representa las estructuras de datos, reglas de negocio y las funcionalidades correspondientes para la comunicación con Doctrine.

La **Vista** son las interfaces mediantes las cuales los usuarios interactúan con el sistema. Para implementar la vista se utilizará ExtJS.

El **Controlador** es el encargado de intercambiar la información entre el modelo y la vista. Para ello se utilizará Zend Framework.

Patrón de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Brindan una solución probada y documentada a problemas de desarrollo de software, que están sujetos a contextos similares. Su uso pretende establecer un lenguaje común entre los programadores, contribuir a la reutilización, ahorrar tiempo en la implementación y obtener un producto con calidad. Es muy importante tener presente: su nombre, el problema (cuándo aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios) (25).

Patrones GOF

Los patrones GOF (Grupo de los Cuatro, siglas en inglés) ayudan a eliminar problemas recurrentes en el desarrollo de software. Estos patrones están definidos en tres categorías (26):

- **Creacionales:** Se ocupan del proceso de creación de clases y objetos. Los patrones que hacen parte de esta categoría son: Fábrica de Métodos, Fábrica Abstracta, Constructor, Prototipo y Solitario (26).

- **Estructurales:** Tratan de la composición de clases y objetos, se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes. Los patrones que hacen parte de esta categoría son: Adaptador, Puente, Compuesto, Decorador, Fachada, Poco Peso y Proxy (26).

- **Comportamiento:** Caracterizan las formas en que las clases o los objetos interactúan y distribuyen la responsabilidad. Los patrones que se agrupan en esta categoría son: Interprete, Método de Plantilla, Cadena de Responsabilidad, Comando, Iterador, Mediador, Recuerdo, Observador, Estado, Estrategia, Visitante (26).

Patrones GRASP

Los patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades, siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones (23).

Dentro de los patrones GRASP se encuentran los siguientes: Creador, Experto, Controlador, Alta cohesión, Bajo acoplamiento, Polimorfismo, Fabricación pura, Indirección, No hables con Extraños.

1.7 Conclusiones parciales

Se realizó un análisis y estudio a determinados sistemas de comercialización; analizando ventajas y deficiencias de cada uno respecto a varios aspectos y funcionalidades necesarias para el proceso de Importación. Como resultado de este estudio surge la necesidad de implementar un componente para el Sistema de Importación y Exportación Xedro-Apside v1.0, que debe contar con las funcionalidades: Gestión de documentos, Gestión de contratos, Gestión de contenedores, Gestión de carga de la mercancía y Graficar datos estadísticos. El componente es desarrollado con tecnologías y herramientas de código abierto, evitando con esto posibles problemas de licencias. Además en el capítulo se describió el modelo de desarrollo de software a utilizar, para así establecer el ciclo de vida de la propuesta de solución y las disciplinas por las que transitará.

Capítulo 2. Análisis y Diseño de la Solución Propuesta

En este capítulo se expone el diseño del Componente Gestor del Movimiento de la Mercancía para el Sistema de Importación y Exportación Xedro-Apside v1.0. Se describen los requisitos funcionales y no funcionales del componente. Se explican los patrones utilizados en la investigación, tanto el arquitectónico como los de diseño. Además se verán los productos de trabajo generados en la disciplina de Análisis y diseño siguiendo la "Metodología de desarrollo para la Actividad Productiva de la UCI".

2.1 Modelo conceptual

Un modelo conceptual permite un mejor entendimiento del negocio, ya que describe sus principales conceptos y sirve de base para construir el modelo de datos final de la solución. El modelo conceptual que a continuación se presenta (*Figura 2*), expone los principales conceptos involucrados en el movimiento de mercancía y sus relaciones.

El flujo de datos inicial del movimiento proviene del contrato, en cuanto este es confirmado entonces comienza la gestión del movimiento. La descripción de cada uno de los conceptos reflejados en el diagrama se encuentra en el documento entregable "CEIGE_SGIEBK_Modelo_conceptual_Mmercancia.doc".

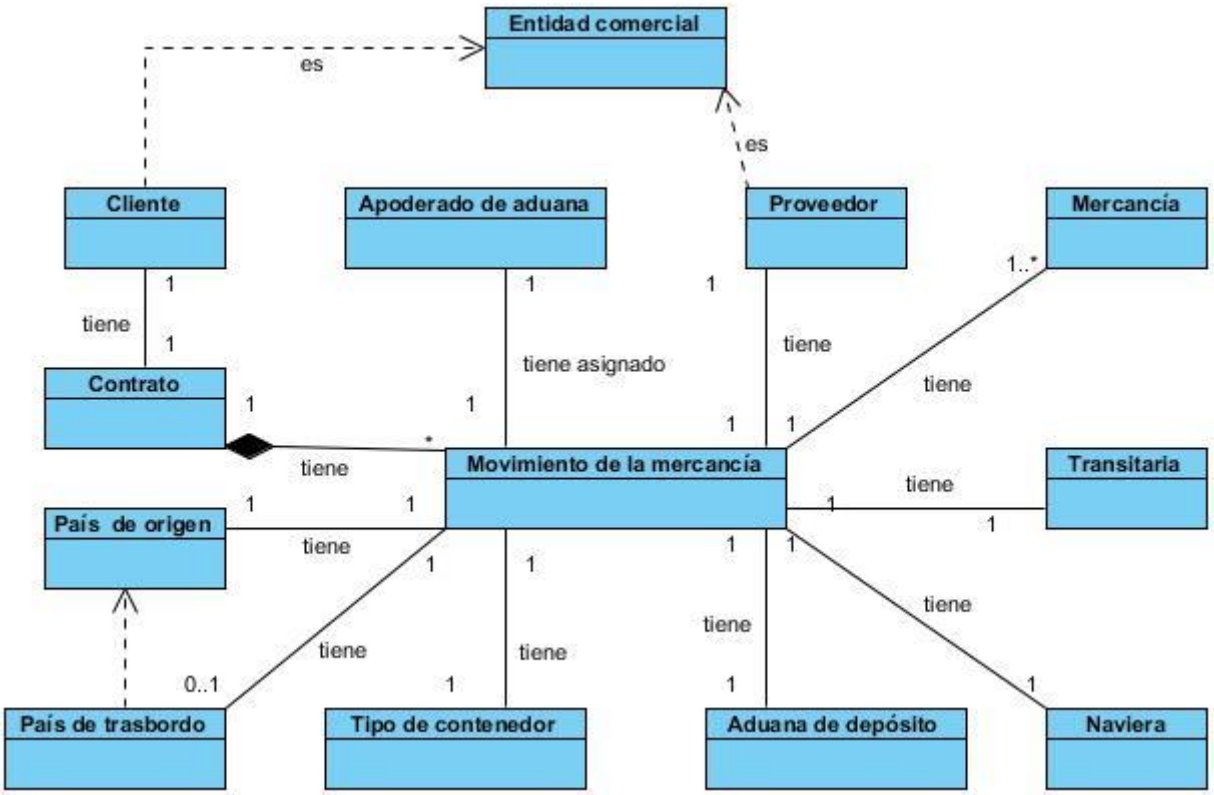


Figura 2. Marco conceptual

2.2 Requisitos de software

Los requisitos funcionales son definidos por el equipo de desarrollo, teniendo en cuenta la información brindada por el cliente. Partiendo de esta información se elaboran los documentos de especificación de requisitos, finalmente se valora y valida la información en busca de errores, inconsistencias o faltas, para evitar omitir algún requerimiento del cliente (27).

2.2.1 Técnicas de captura de requisitos

Las técnicas de captura de requisitos facilitan la obtención de información referente al cliente. A continuación se presentan las técnicas utilizadas para la obtención de los requisitos del componente:

- **Entrevista:** permite al equipo de desarrollo interpretar las necesidades del cliente mediante un intercambio abierto, realizando preguntas que esclarecen con precisión el funcionamiento del negocio. La entrevista se puede tornar provechosa dependiendo de las habilidades del entrevistador.

- **Tormenta de ideas:** consiste en la acumulación de ideas sin prejuicios y valoraciones. Aunque no evidencia los detalles concretos que se pueden necesitar del sistema, es muy común en los comienzos del proceso de ingeniería de requisitos.

La entrevista se realizó entre el equipo de proyecto y especialistas de la empresa BK Import-Export, se logró identificar requisitos y conocer los procesos de dicha empresa. La tormenta de ideas se efectuó mediante reuniones y encuentros con especialistas y directivos de la empresa BK Import-Export, se obtuvieron requisitos y se intercambiaron ideas sobre posibles soluciones.

2.2.2 Requisitos funcionales

Los requisitos funcionales son servicios que debe brindar el sistema, estos cumplen funciones específicas de acuerdo a los datos que se les proporcionan. Aplicando las técnicas de captura de requisitos entrevistas y tormenta de ideas se definieron los siguientes requisitos funcionales:

RF1. Actualizar movimiento de carga diario.

RF2. Listar movimiento de carga diario.

RF3. Exportar movimiento de carga diario.

RF4. Realizar búsqueda avanzada de movimiento de carga diario.

RF5. Asignar apoderado.

RF6. Graficar estadística de los movimientos de carga diario.

RF7. Adicionar tipo contenedor.

RF8. Modificar tipo contenedor.

RF9. Eliminar tipo contenedor.

RF10. Listar tipo contenedor.

RF11. Buscar tipo contenedor.

RF12. Adicionar aduana de depósito.

RF13. Modificar aduana de depósito.

RF14. Eliminar aduana de depósito.

RF15. Listar aduana de depósito.

RF16. Buscar aduana de depósito.

RF17. Adicionar naviera.

RF18. Modificar naviera.

RF19. Eliminar naviera.

RF20. Listar naviera.

RF21. Buscar naviera.

RF22. Adicionar transitaria.

RF23. Modificar transitaria.

RF24. Eliminar transitaria.

RF25. Listar transitaria.

RF26. Buscar transitaria.

Para calcular la complejidad de los requisitos funcionales se utilizó la hoja de cálculo CEIGE_SGIEBK_Evaluacion_de_requisitos.xls, que se encuentra como documento entregable y está basada en la plantilla definida por el programa de mejoras de la UCI para el cálculo de la complejidad de los requisitos. Los resultados se muestran en la siguiente gráfica (*Figura 3*):

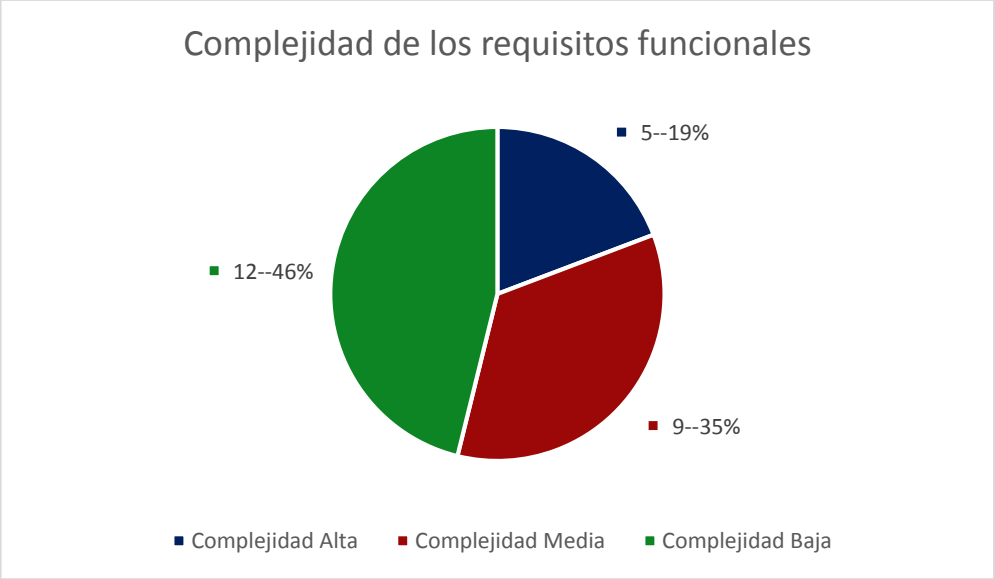


Figura 3. Complejidad de los requisitos funcionales

De un total de 26 requisitos funcionales, 12 poseen complejidad baja, nueve complejidad media y cinco complejidad alta. Este análisis permite tomar decisiones en cuanto a la prioridad de los requisitos para su implementación.

Descripción de requisitos

La "Metodología de desarrollo para la Actividad Productiva de la UCI" plantea que el esfuerzo principal de la disciplina de Requisitos, es desarrollar un modelo que describa el sistema que se va construir. Para contribuir a la construcción del modelo antes mencionado se realizó la descripción de los requisitos que además es una actividad de la disciplina Requisitos.

A continuación se presenta la descripción del RF6 "Graficar estadística de los movimientos de carga diario". La descripción de todos los requisitos funcionales implementados se pueden consultar en los documentos entregables "CEIGE_SGIEBK_Descripcion_de_requisitos_por_proceso (Gestionar movimientos de carga diario).doc", "CEIGE_SGIEBK_Descripcion_de_requisitos_por_proceso (Gestionar tipos de contenedores).doc", "CEIGE_SGIEBK_Descripcion_de_requisitos_por_proceso(Gestionar aduana de depósito).doc", "CEIGE_SGIEBK_Descripcion_de_requisitos_por_proceso(Gestionar naviera).doc" y "CEIGE_SGIEBK_Descripcion_de_requisitos_por_proceso(Gestionar transitaria).doc".

Tabla 5. Descripción del RF6 Graficar estadística de los movimientos de carga diario

Precondiciones	El usuario se ha autenticado en el sistema y tiene permiso para realizar esta acción. Se ha registrado al menos un movimiento de carga diario en el sistema
Flujo de eventos	
Flujo básico Adicionar	

1	Se selecciona la opción Graficar.
2	El sistema muestra una ventana con dos campos para introducir fechas.
3	El usuario escribe las fechas entre las cuales desea hacer el análisis estadístico.
4	Aparece una ventana donde el usuario selecciona el criterio por el que desea graficar.
5	El usuario selecciona el criterio y el sistema grafica.
Pos-condiciones	
1	N/A
Flujos alternativos	
Flujo alternativo 2.a No escribió ninguna fecha	
1	El sistema notifica que las fechas son obligatorias.
Pos-condiciones	
Flujo alternativo 2.b El usuario cancela la acción	
4	Concluye el requisito
Pos-condiciones	
4	N/A
Validaciones	
4	N/A

Conceptos	**Entidad comercial**	Visibles en la interfaz: Proveedor País de origen Naviera Transitaria Estado Aduana depósito Peso bruto Utilizados internamente: Fecha de recibo de los documentos de embarque (Inicio) Fecha de recibo de los documentos de embarque (Fin)
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Las siguientes figuras *(Figura 4 y 5)* muestran los prototipos de interfaz de usuario para el RF6:

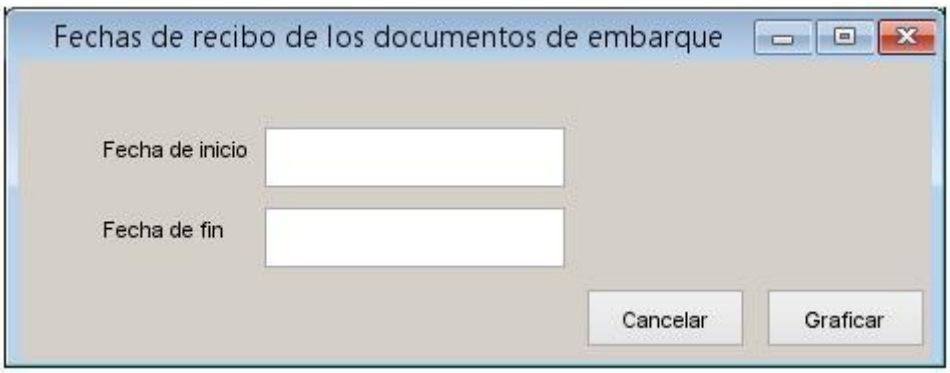


Figura 4. Prototipo de interfaz para el RF6, fechas de recibo de los documentos de embarque

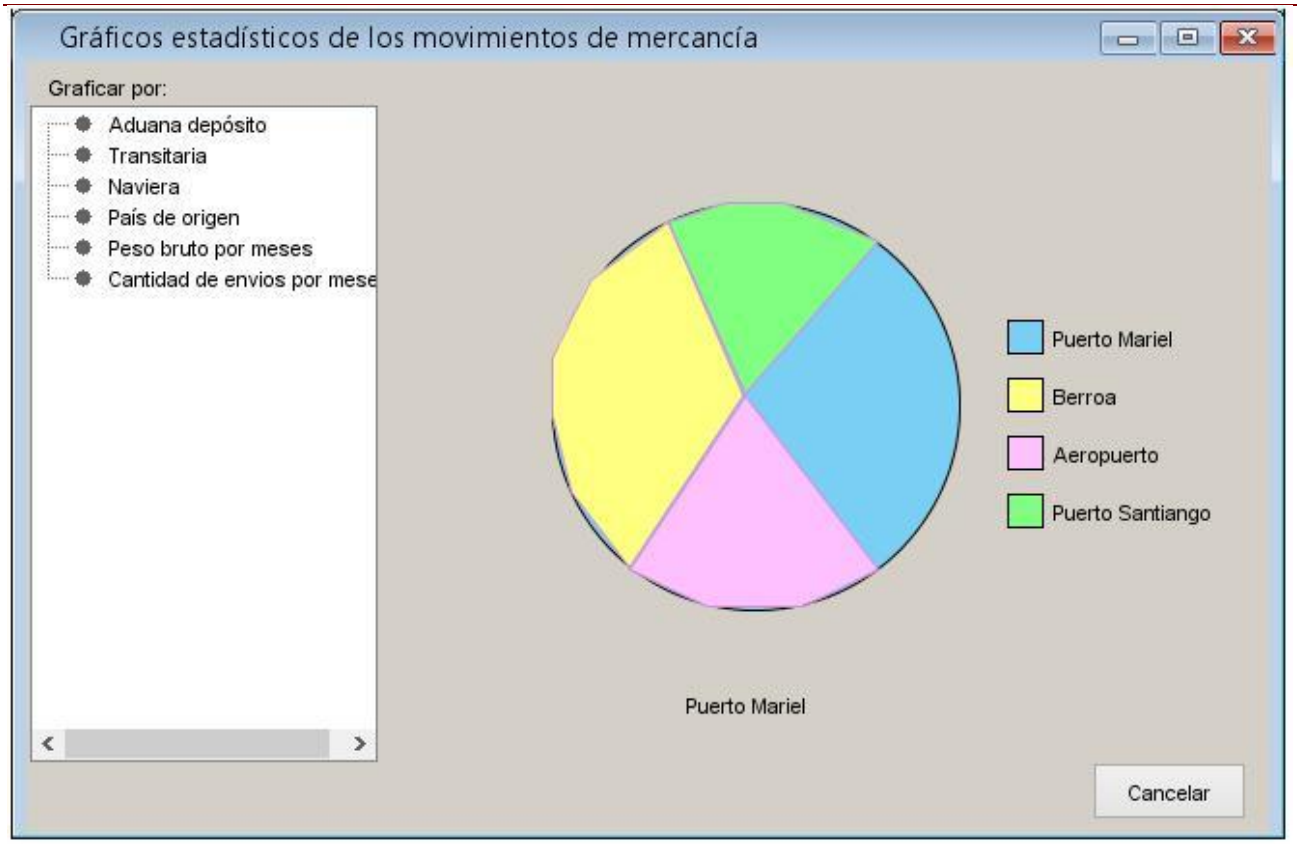


Figura 5. Prototipo de interfaz para el RF6, gráficos estadísticos

2.2.3 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Permiten que clientes y usuarios valoren las características no funcionales del producto, pues aunque cumpla con las funcionalidades requeridas, las características no funcionales pueden hacer una gran diferencia entre un producto que posea la aceptación de todos y uno con poca aceptación (28). A continuación se listan los requisitos no funcionales del Sistema de Importación y Exportación Xedro-Apside v1.0, para obtener más información de los mismos consultar el documento entregable CEIGE_SGIEBK_Especificacion_de_requisitos_de_software.odt.

RNF1. Requisitos de software: la PC cliente debe contar con un navegador instalado, se recomienda Mozilla Firefox en su versión 30.0 o superior. El PC servidor debe contar con la aplicación Apache v2.2 configurada correctamente con el módulo para PHP 5.4 y con las extensiones necesarias para que funcione el Sistema de Importación y Exportación Xedro-Apside v1.0. El servidor además debe tener en funcionamiento el Sistema Gestor de Base Datos Postgresql 9.2.

RNF2. Requisitos de hardware: las PC clientes deben contar con un procesador Pentium IV o superior, 512 Mb de RAM. Mientras que el servidor debe poseer 80 GB como mínimo, de

espacio en el disco duro; el microprocesador debe de ser como mínimo un Celeron Dualcore a 2.0 GHz y la RAM igual o superior a 2 GB.

RNF3. Requisitos de funcionalidad:

- Idoneidad: capacidad del software para mantener un conjunto apropiado de funciones para las tareas y los objetivos especificados por el usuario.
- Seguridad: capacidad del producto de software para que sea seguro, teniendo en cuenta mecanismos de encriptación de contraseñas y gestión de roles y usuarios.

RNF4. Requisitos de usabilidad:

- Comprensibilidad: capacidad del software para permitirle al usuario entender si el software es idóneo, y cómo puede usarse para las tareas y condiciones de uso particulares.
- Cognoscibilidad: capacidad del software para permitirle al usuario aprender a usarlo.
- Operabilidad: capacidad del software para permitirle al usuario operarlo y controlarlo.
- Atractivo: capacidad del software para ser atractivo o amigable para el usuario.

RNF5. Requisitos de confiabilidad:

- Madurez: capacidad del producto para evitar un fallo total como resultado de haberse producido un fallo del software.
- Tolerancia ante fallos: capacidad del software para mantener un nivel de ejecución o desempeño especificado en caso de fallos del software o de infracción de su interfaz.
- Recuperabilidad ante fallos: capacidad del software para restablecer un nivel de ejecución especificado y recuperar los datos directamente afectados en caso de fallo total.

RNF6. Requisitos de eficiencia:

- Rendimiento: capacidad del software para proporcionar tiempos apropiados de respuesta y procesamiento, así como tasas de producción de resultados, al realizar su función bajo condiciones establecidas.

- Utilización de recursos: capacidad del producto para utilizar la cantidad y el tipo apropiado de recursos cuando el software realiza su función bajo las condiciones establecidas.

2.2.4 Técnicas de validación de requisitos

La validación de requisitos es un proceso de gran importancia para poder comenzar la implementación del componente, ya que dicho proceso certifica que el requisito satisface la necesidad del cliente. A continuación se exponen las técnicas de validación de requisitos utilizadas:

- **Revisión técnica formal:** consiste en una revisión formal entre el equipo de proyecto que desarrolló los requisitos y el cliente final. Si el cliente está satisfecho valida el requisito, en caso que no lo esté, indica sus inconformidades y estas son arregladas para una nueva revisión.

- **Generación de casos de prueba:** se crean una serie de pruebas con acciones y valores reales, que se le aplican al sistema para observar cuales son las respuestas o salidas de este y si están en concordancia con la descripción del requisito evaluado.

- **Prototipos:** Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final.

La revisión formal se efectuó por el personal involucrado en el requisito, tanto los clientes como el equipo de desarrollo, los requisitos aprobados estaban descritos de forma correcta, clara y consistente. La creación de prototipos permitió al cliente tener una idea de cómo quedaría la solución y se logró obtener recomendaciones. Mediante la generación de casos de prueba se consiguió verificar si cada requisito cumplía su función y sirvió de ayuda para las disciplinas que implementan las pruebas.

2.3 Diagrama de clases del diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación (23). La figura que a continuación (*Figura 6*) se presenta, muestra el diagrama de clases del diseño con estereotipos web del Componente Gestor del Movimiento de la Mercancía y la *Tabla 6* describe las clases y los archivos contenidos en el diagrama.

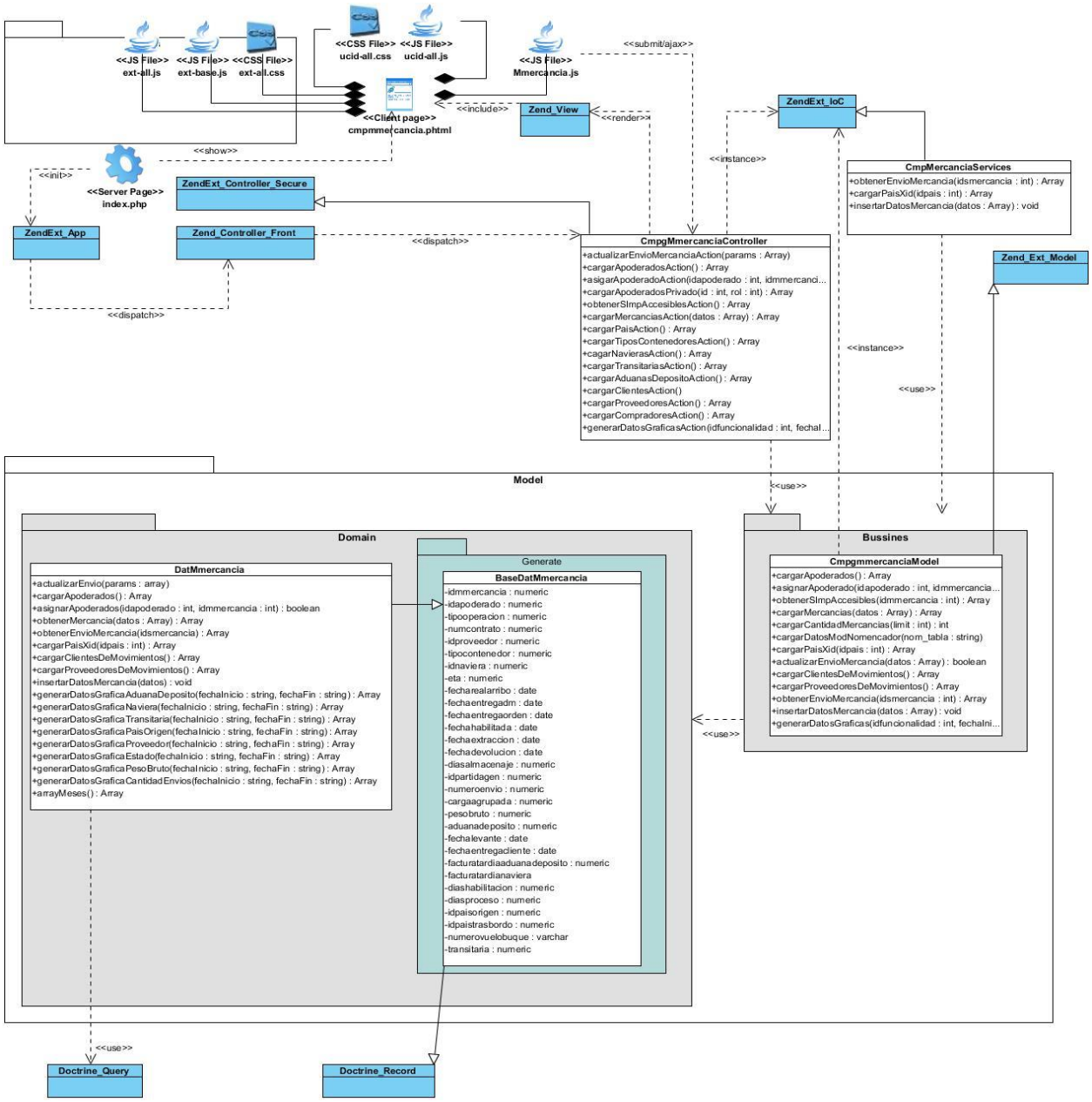


Figura 6. Diagrama de clases del diseño

Tabla 6. Descripción del diagrama de clases del diseño

Clase o archivo	Descripción
cmpmmercancia.phtml	Es el archivo que se carga en el navegador. Incluye los archivos .js y .css necesarios para mostrar la interfaz de la aplicación y sus datos.
Mmercancia.js	Contiene los códigos javascript correspondientes a la vista de la aplicación e

	incluye otros archivos .js que complementan las funciones de la vista.
CmpgMmercanciaController	Es la clase controladora del componente y se encarga de manejar el flujo de información entre la vista y los modelos.
DatMmercancia	Se encarga de realizar las operaciones con la base de datos.
CmpgmmercanciaModel	Se encarga de manejar el negocio, hace las conversiones u operaciones necesarias a los datos para que estén acorde a los requisitos.
BaseDatMmercancia	Contiene los atributos del mapeo de la base de datos.
CmpMercanciaServices	Es la clase que se utiliza para brindar los servicios web del componente.

2.4 Patrones utilizados

2.4.1 Patrón arquitectónico

El patrón arquitectónico utilizado es MVC. Con la utilización de este patrón se logra una arquitectura en tres capas, donde cada una de ellas se encarga de funciones específicas.

En la capa de la **vista** se encuentra la página cmpmmercancia.phtml y el archivo Mmercancia.js, entre ambos brindan una interfaz comprensible al usuario, en correspondencia con los datos ofrecidos por la clase controladora. Como **controlador** está la clase CmpgMmercanciaController, que se encarga de manipular y vincular la información entre la vista y el modelo. En la capa del **modelo** se encuentran las clases DatMmercancia y CmpgmmercanciaModel, entre ambas se encargan de manejar el flujo de información con la base de datos y brindar a la clase controladora la información necesaria para sus funciones.

2.4.2 Patrones de diseño

En el capítulo anterior se expusieron los patrones de diseño GRASP y GOF. A continuación se describen los patrones utilizados en la solución, haciendo referencia al diagrama de clases del diseño *(Figura 6)*:

Patrones GOF

- **Fachada:** Es una clase definida que ofrece una interfaz común para las demás clases que la utilizan. Esto se pone de manifiesto en la clase CmpgmmercanciaModel que sirve de fachada a la clase controladora y a la clase que implementa los servicios web**.**
- **Estado:** El comportamiento de un objeto depende de su estado**.** Este patrón se pone de manifiesto en la clase DatMMercancia; dependiendo del estado en que se encuentre el movimiento se podrán actualizar determinados datos y/o realizar ciertas operaciones.

Patrones GRASP

- **Creador:** Asigna responsabilidades de tal manera que ayuda a identificar que clase debe ser la creadora de determinado objeto. El uso de este patrón se evidencia en las clases del modelo, donde la clase DatMmercancia se encarga de crear los objetos Doctrine_Query, para realizar las consultas a la base de datos.
- **Experto:** Este patrón asigna responsabilidades a las clases que tienen la información necesaria para realizar una acción. El patrón se evidencia en la clase controladora y en las clases del modelo. Su utilización favorece la reutilización.
- **Controlador:** Asigna la responsabilidad de controlar los eventos y acciones del componente a una o varias clases. Dicho patrón se evidencia en la clase CmpgMmercanciaController que funge como controladora en el componente.
- **Alta cohesión:** La alta cohesión es representada por las clases que tienen responsabilidades específicas en un componente, pero colaboran con otras clases para realizar ciertas acciones. El patrón asigna responsabilidades de manera que cada clase deba realizar las labores que le corresponden. Se puede evidenciar en la clase DatMMercancia, la cual se encarga de manejar el acceso a datos pero colabora con la clase CmpgmmercanciaModel.
- **Bajo acoplamiento:** Este patrón plantea que entre las clases debe existir pocos vínculos. Con el uso de este patrón se logra un fácil mantenimiento al código y mayor reutilización. El uso de este patrón se puede observar en las clases del modelo.

2.5 Modelo de datos

El modelo de datos o diagrama entidad-relación es un factor importante en la construcción de un sistema. Este contiene una serie de entidades o tablas relacionadas que posibilitan que el almacenamiento de la información sea lo menos redundante posible. Además la creación del diagrama entidad-relación brinda conocimiento a los desarrolladores, ya que pueden identificar como serán estructurados los datos para su permanencia. A continuación se presenta el diagrama

entidad-relación (*Figura 7*) de la solución propuesta, el cual cuenta con ocho tablas y sus respectivas relaciones.

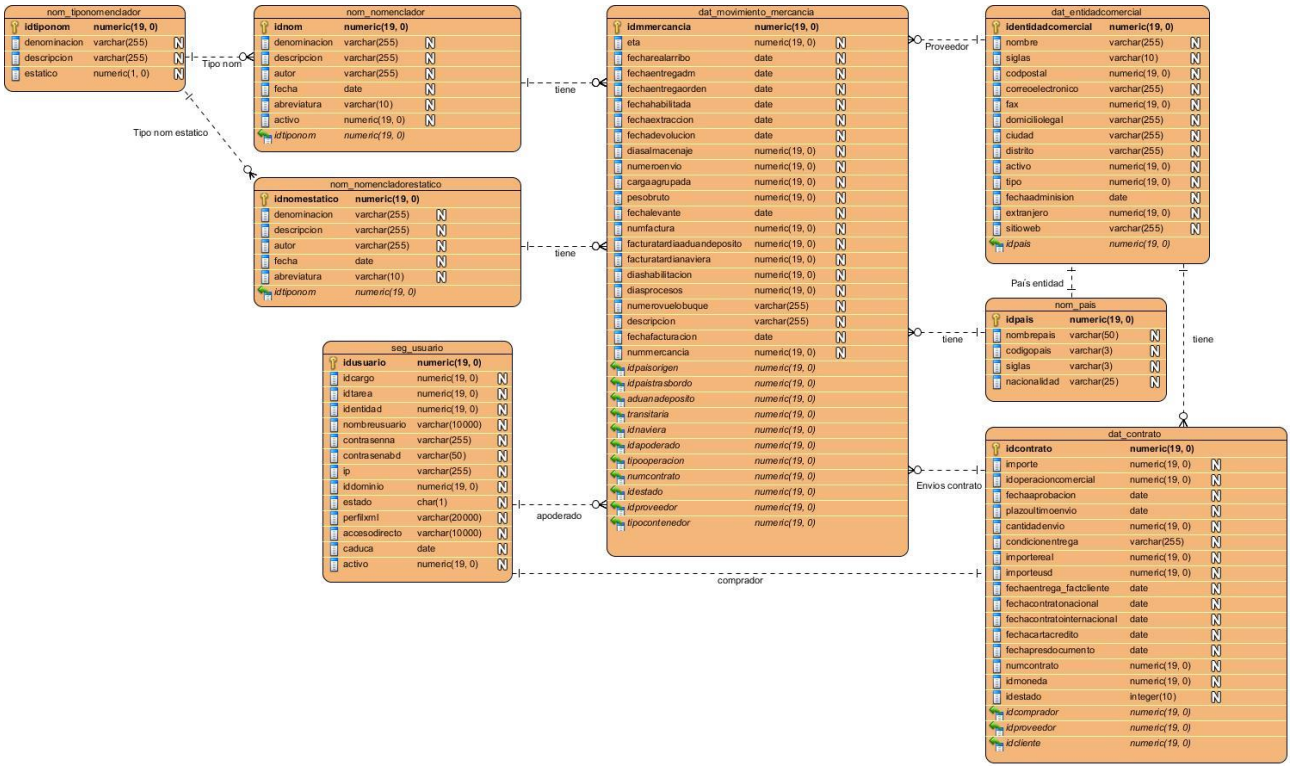


Figura 7. Diagrama entidad-relación

La base de datos de la solución se encuentra en la tercera forma normal, se eliminaron los datos repetidos, los campos de las tablas son dependientes de su llave primaria y no existe transitividad entre los valores de los campos. La tabla dat_movimiento_mercancia contiene datos referentes al movimiento y al unirla con las demás tablas involucradas se obtiene toda la información referente al movimiento. Las tablas nom_nomenclador y nom_nomencladorestatico que a su vez se relacionan con nom_tiponomenclador, brindan datos importantes al movimiento como son: aduanadeposito, transitaria, idnaviera, tipocontenedor, tipooperacion e idestado. De la tabla nom_pais se obtienen tanto el idpaisorigen como idpaistrasbordo. La tabla seg_usuario se relaciona con dat_movimiento_mercancia por el campo idapoderado, ya que los apoderados son usuarios del sistema. La tabla dat_contrato facilita varios datos al movimiento y de ella provienen los datos iniciales del movimiento. La tabla dat_entidadcomercial brinda los datos del cliente y del proveedor ya que ambos son entidades comerciales.

2.6 Conclusiones parciales

Se identificaron 26 requisitos funcionales al aplicar las técnicas de entrevista y tormenta de ideas. Con la validación de los requisitos por parte del cliente, se obtuvieron las funcionalidades básicas a implementar en el componente. La elaboración del diagrama de clases del diseño logró un mejor

entendimiento de las clases y sus relaciones. Además se mostró como quedarán estructurados los datos para su almacenamiento a través de la confección del modelo de datos.

Capítulo 3. Implementación y Validación de la Solución Propuesta

En el capítulo que se desarrolla a continuación se expone la implementación del componente y los resultados obtenidos con el mismo. Se explican los estándares de codificación con ejemplos de código y se evidencia el resultado de las pruebas y validaciones al componente. Se describen también los resultados de las pruebas de aceptación con el cliente y los aportes económicos y sociales de la solución.

3.1 Diagrama de componentes

El desarrollo basado en componentes se está adoptando cada vez más como una aproximación fundamental a la ingeniería del software. Los componentes son independientes para que no interfieran en su funcionamiento unos con otros. Un diagrama de componentes muestra el sistema dividido por componentes, así como la relación entre estos y la forma en la que se comunican (29). El diagrama de componentes que a continuación se presenta *(Figura 8)*, muestra parte de los componentes del Sistema de Importación y Exportación Xedro-Apside v1.0 y la manera en que interactúan con el Componente Gestor del Movimiento de la Mercancía (cmpmmercancia).

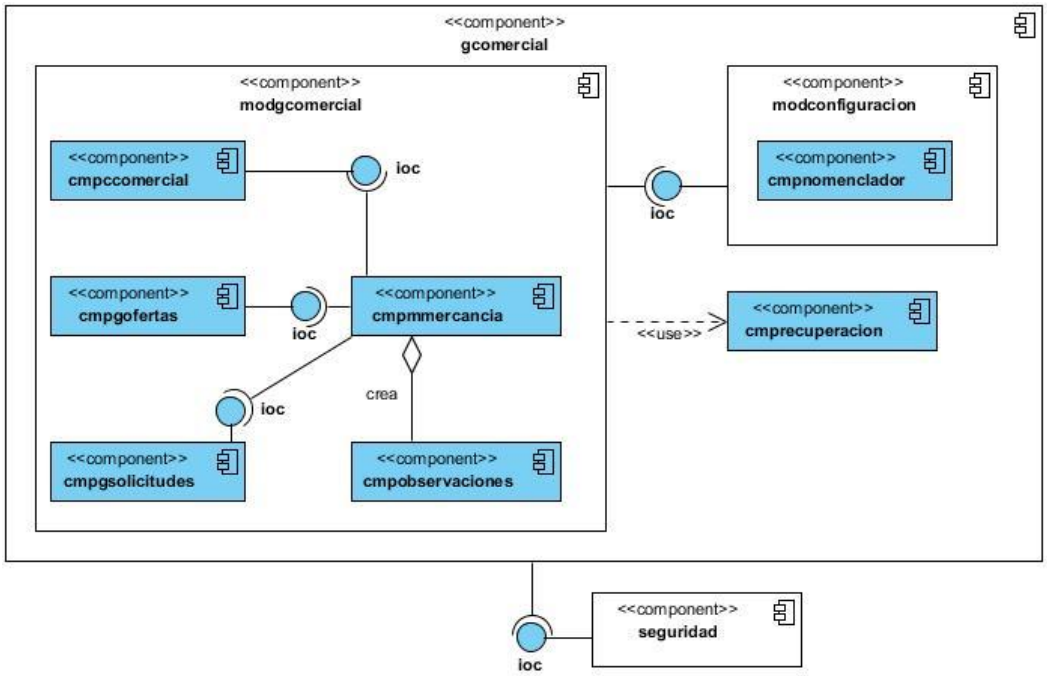


Figura 8. Diagrama de componentes

El cmpmmercancia brinda un servicio que permite obtener los datos de un movimiento de la mercancía, facilitando esta información a otros componentes que la necesiten. También brinda un servicio para insertar la información inicial de un movimiento de la mercancía, que es consumido por el componente cmpgofertas donde se gestionan los contratos.

El componente consume servicios que le posibilitan obtener la información necesaria para mostrar al usuario y poder contribuir al seguimiento y control del movimiento. Uno de los servicios

consumidos es cargar nomenclador por id**,** con el cual se adquiere la información de los estados, la transitaria, la naviera y otras informaciones referentes al movimiento. Otro servicio que se consume es obtener los datos del contrato, del cual se consigue la información referente al cliente, el comprador, el número de contrato y otros datos de interés para el usuario del sistema.

3.2 Diagrama de despliegue

En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados; es decir se sitúa el software en el hardware que lo contiene. Cada hardware se representa como un nodo. Dicho diagrama modela la arquitectura en tiempo de ejecución de un sistema (30). La *Figura 9* muestra el diagrama de despliegue de la solución propuesta, el mismo cuenta con tres nodos, un cliente con el navegador web Mozilla Firefox 30.0 que realiza llamadas https a un servidor de aplicaciones, en este caso Apache 2.2, el cual gestiona el intercambio de información con el servidor de base de datos que utiliza como gestor PostgreSQL 9.2.

Figura 9. Diagrama de despliegue

3.3 Estándares de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Un código fuente debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez (31). Un estándar de codificación o estilo de programación homogéneo en un proyecto permite que todos los integrantes del mismo lo puedan entender en menos tiempo y favorece considerablemente el mantenimiento del código. En este acápite se presentarán y explicarán los estándares utilizados en la implementación de la propuesta de solución.

3.3.1 Nomenclatura de las clases

Se utilizó la notación *PascalCasing* que consiste en que la primera letra de la clase y la primera letra de las siguientes palabras concatenadas están en mayúsculas, el resto de las letras de las palabras están en minúscula, por ejemplo:

```
class DatmovimientoMercancia extends BaseDatMovimientoMercancia
```

Para las clases controladoras y del modelo se utilizó la misma notación, pero a la clase controladora se le agregó al final la palabra *Controller;* a la clase del paquete *Bussines* se le agregó la palabra *Model* al final y a la del paquete *generated* se le agregó la palabra *Base* al comienzo.

3.3.2 Nomenclatura de las funciones y las variables

Para las funciones y las variables se empleará la notación *CamelCasing,* que consiste en poner la primera letra del nombre en minúscula y la primera letra de las siguientes palabras concatenadas en mayúscula. Para las funciones de la clase controladora que son llamadas mediante acciones de la vista se le agregó la palabra *Action* al final, por ejemplo:

```
function asignarApoderadoAction() {
    $datos = $this->_request->getPost();
            try {
        $res = $this->mercanciamodel->asignarApoderado($datos);
        if ($res) {
            echo("{'codMsg':1,mensaje:'Operaci&oacute;n satisfactoria.'}");
        } else {
            echo("{'codMsg':3,mensaje:'Operaci&oacute;n insatisfactoria.'}");
        }
    } catch (Exception $exc) {
        echo json_encode($exc);
    }
}
```

Figura 10. Método asignarApoderadoAction(), notación *camelCasing*

3.4 Interfaces del componente

En este acápite se explicará el flujo de trabajo en el componente, exponiendo las principales funcionalidades y mostrando las interfaces de la aplicación.

La información inicial del movimiento proviene del contrato, en cuanto este es confirmado se insertan los datos del movimiento de la mercancía.

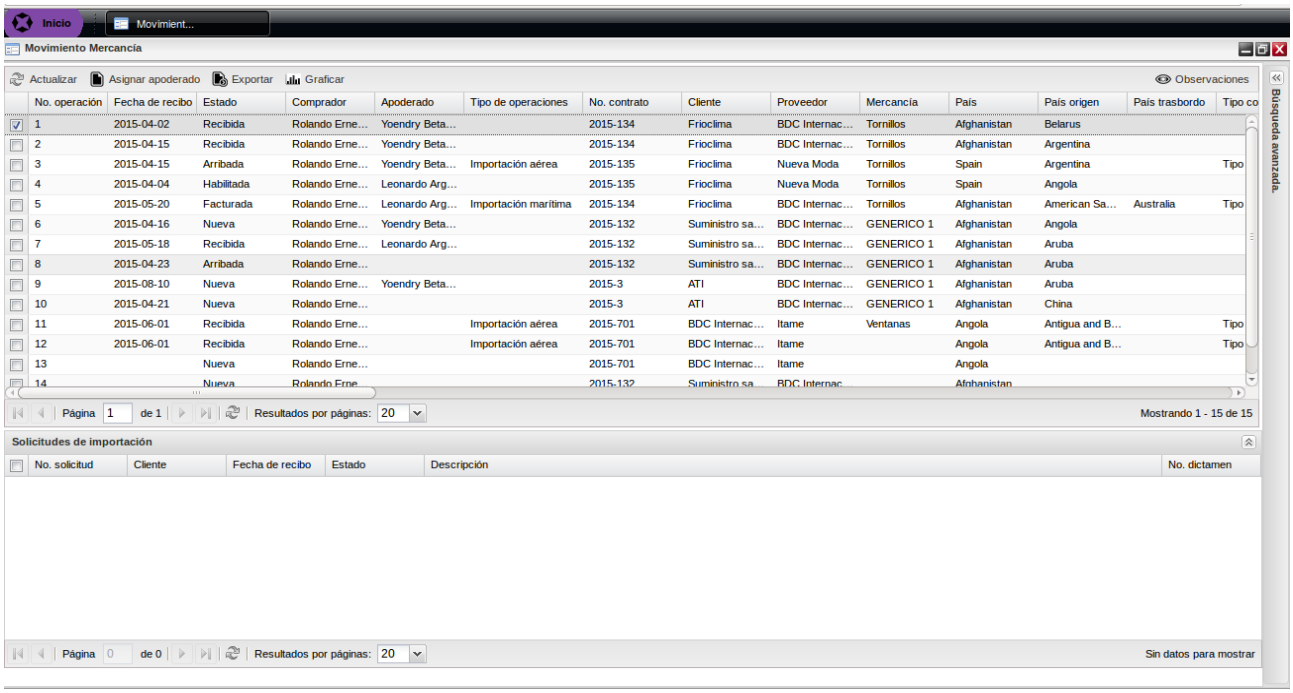


Figura 11. Interfaz, funcionalidad listar de movimientos

Para ir realizando el seguimiento y control del movimiento es necesario introducir datos, dependiendo del estado en que se encuentre el movimiento. Esta actividad se realiza mediante la funcionalidad Actualizar, según los datos proporcionados, la mercancía va transcurriendo de un estado a otro.

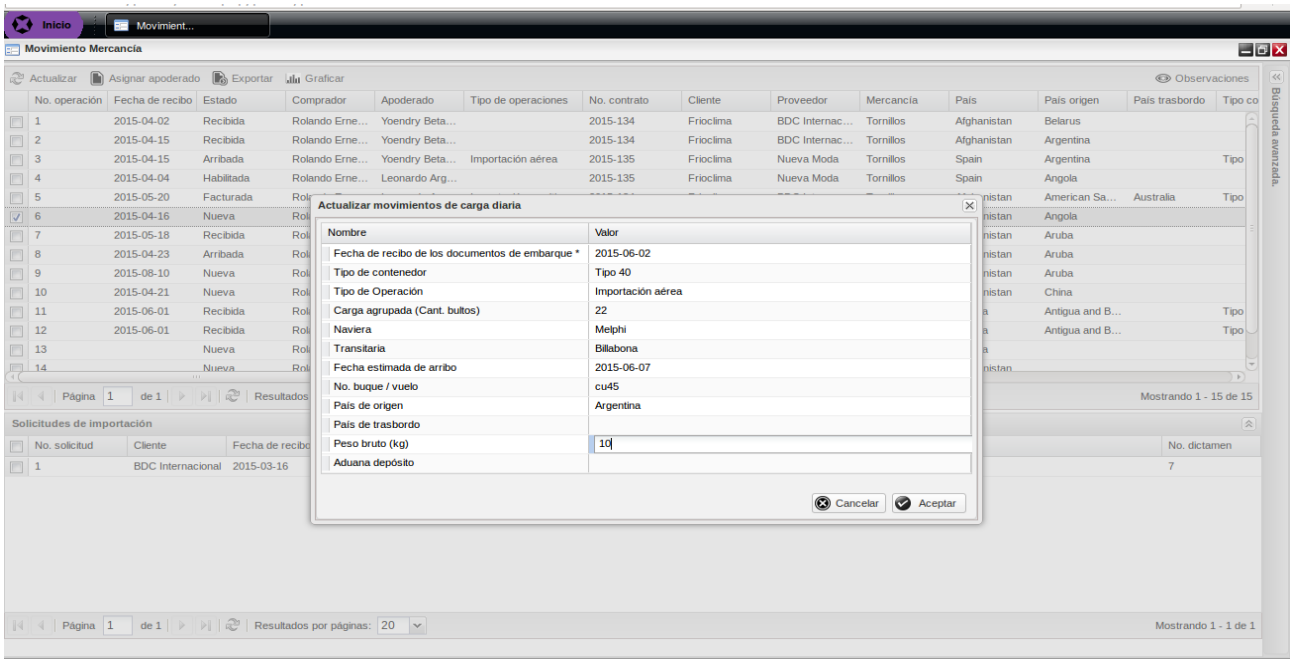


Figura 12. Interfaz, funcionalidad actualizar movimiento

Una funcionalidad necesaria para el movimiento de la mercancía es Asignar apoderado de aduana, esta funcionalidad permite delegar las responsabilidades del seguimiento y control del movimiento a un trabajador del área de logística que posea rol de apoderado.

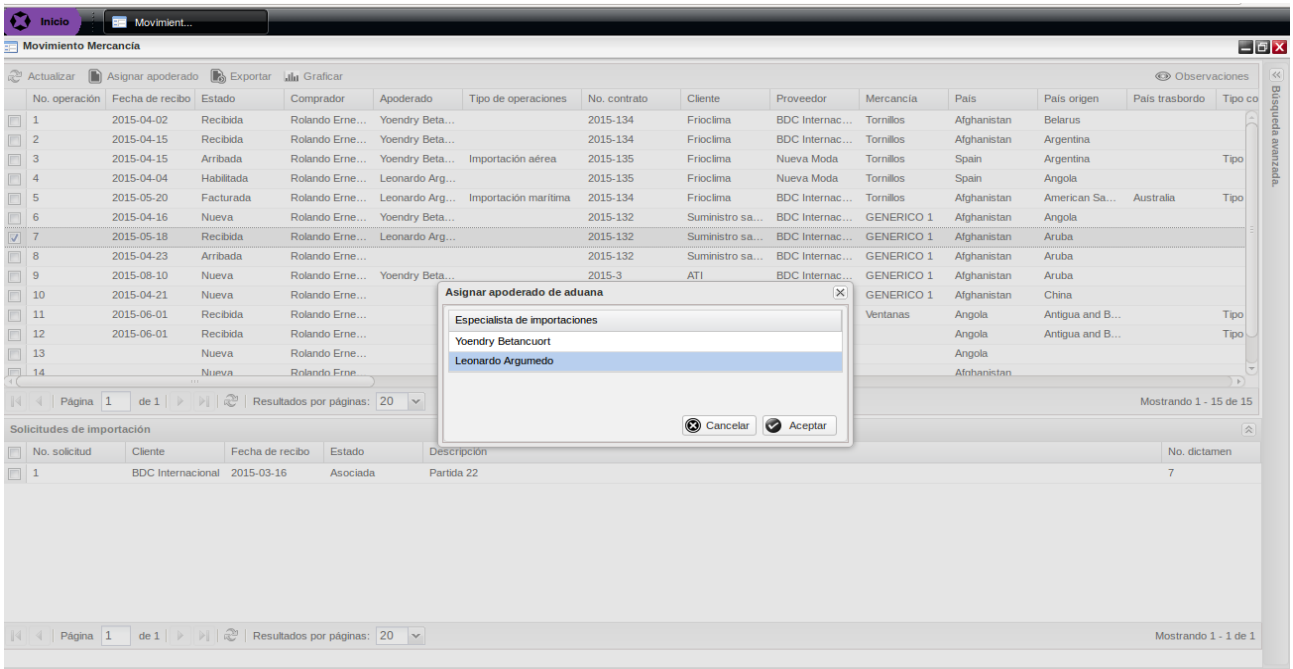
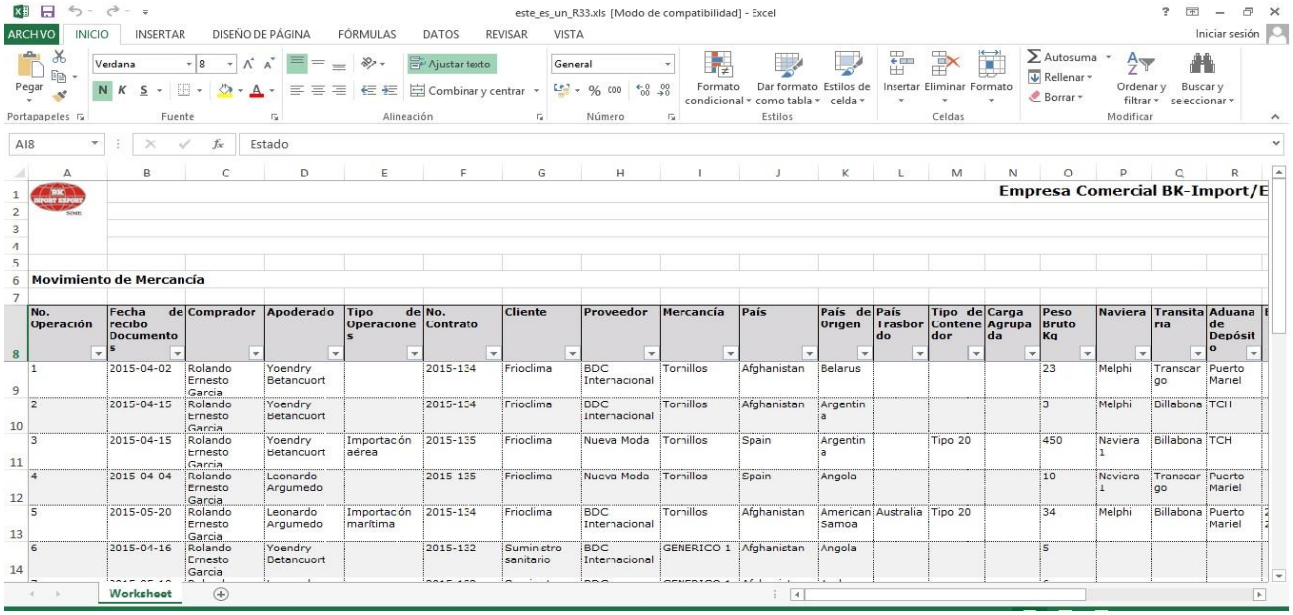


Figura 13. Interfaz, funcionalidad asignar apoderado de aduana

Exportar los movimientos de carga diarios en formato Excel, es esencial para el estudio a posteriori de los movimientos y contribuye a la supervisión de los mismos.



No. Operación	Fecha de recibo Documentos	Comprador	Apoderado	Tipo de Operaciones	No. Contrato	Cliente	Proveedor	Mercancía	País	País de Origen	País trasbordo	Tipo de Contenedor	Carga Agrupada	Peso bruto Kg	Naviera	Transitaria	Aduana de Depósito
1	2015-04-02	Rolando Ernesto Garcia	Yoendry Betancuort		2015-134	Frioclima	BDC Internacional	Tornillos	Afghanistan	Belarus				23	Melphi	Transcargo	Puerto Mariel
2	2015-04-15	Rolando Ernesto Garcia	Yoendry Betancuort		2015-134	Frioclima	BDC Internacional	Tornillos	Afghanistan	Argentina				3	Melphi	Billabona	TCH I
3	2015-04-15	Rolando Ernesto Garcia	Yoendry Betancuort	Importación aérea	2015-135	Frioclima	Nueva Moda	Tornillos	Spain	Argentina		Tipo 20		450	Naviera 1	Billabona	TCH
4	2015 04 04	Rolando Ernesto Garcia	Leonardo Argumedo		2015 135	Frioclima	Nueva Moda	Tornillos	Spain	Angola				10	Naviera 1	Transcargo	Puerto Mariel
5	2015-05-20	Rolando Ernesto Garcia	Leonardo Argumedo	Importación marítima	2015-134	Frioclima	BDC Internacional	Tornillos	Afghanistan	American Samoa	Australia	Tipo 20		34	Melphi	Billabona	Puerto Mariel
6	2015-04-16	Rolando Ernesto Garcia	Yoendry Betancuort		2015-132	Suministro sanitario	BDC Internacional	GENERICO 1	Afghanistan	Angola				5			

Figura 14. Ejemplo de un documento Excel generado por el componente

Almacenando la mayor cantidad de datos posibles de los movimientos, se logra una información más exacta de las estadísticas. El sistema construye gráficas con información estadística, que puede resultar valiosa para el usuario. Esto se realiza mediante la funcionalidad Graficar, la cual necesita un rango de fechas de recibo de los documentos de embarque, para graficar la información contenida en ese rango.

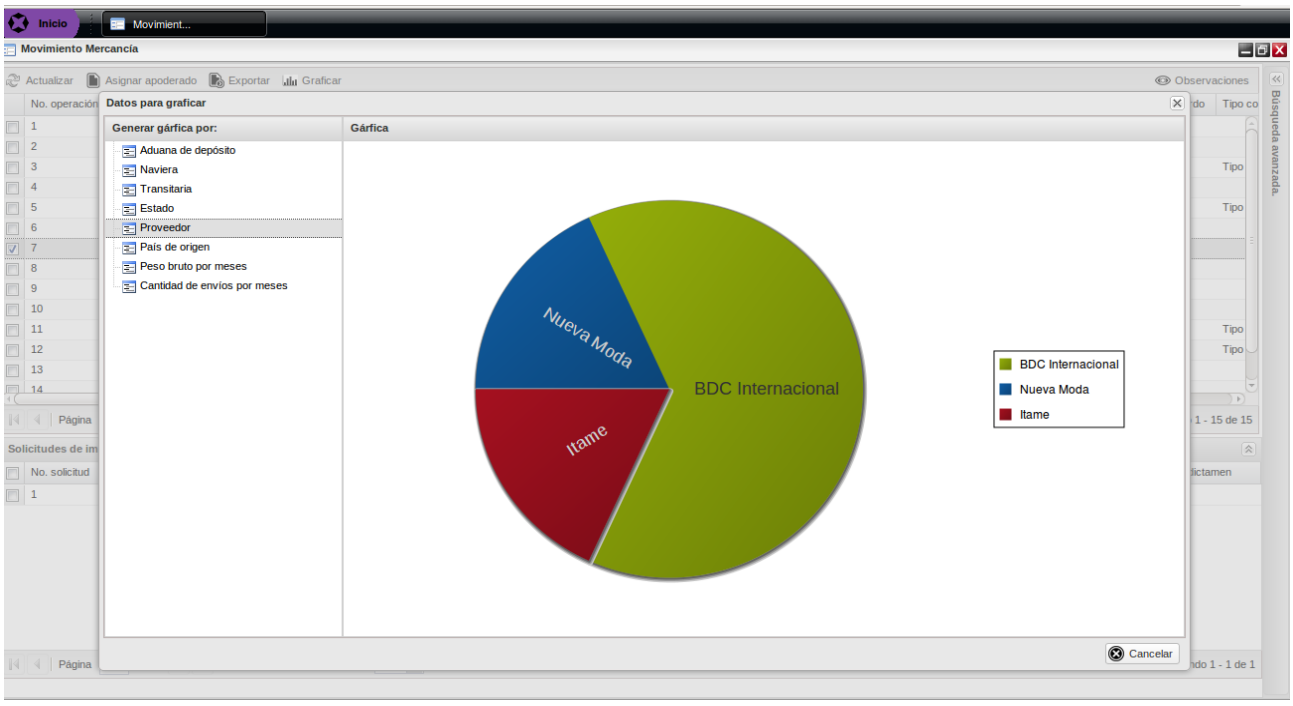


Figura 15. Ejemplo de una gráfica de pastel generada por el componente

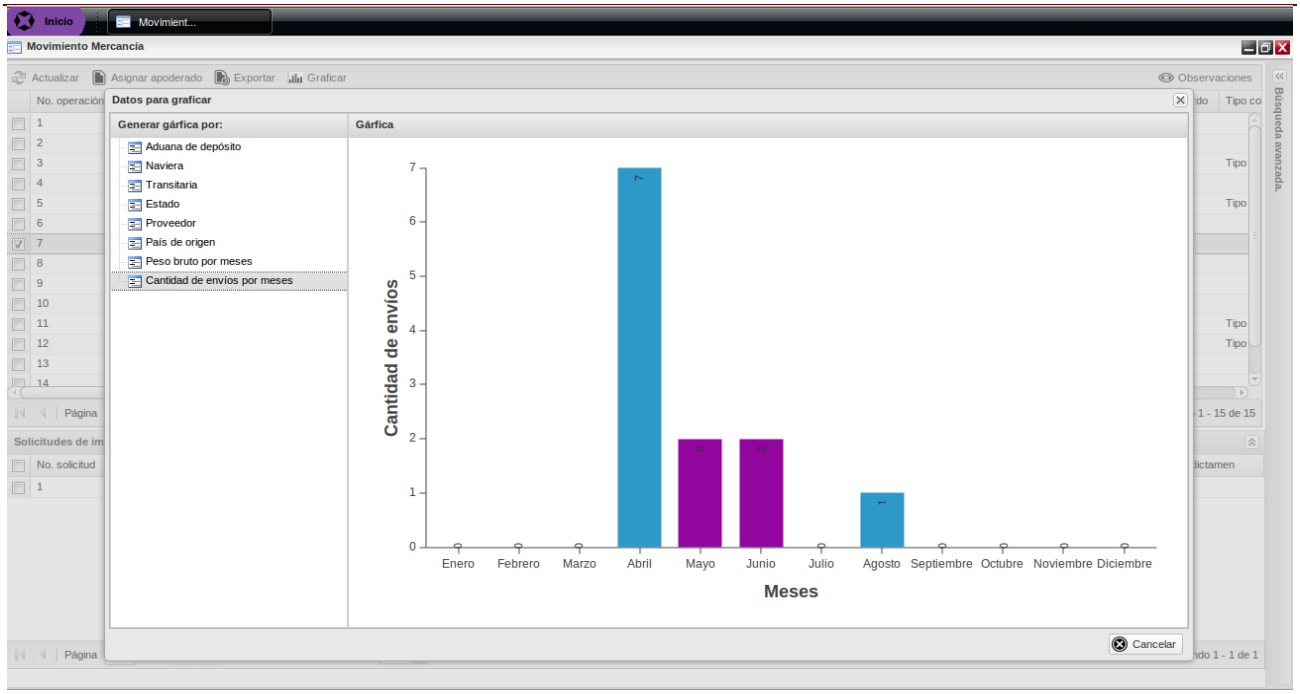


Figura 16. Ejemplo de una gráfica de barra generada por el componente

3.5 Validación del diseño

Las métricas ayudan en la validación de los modelos de análisis y de diseño, donde proporcionan una indicación de la complejidad de diseños procedimentales y de código fuente, favorecen el diseño de pruebas más efectivas. Una métrica se define como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. Se dice que la medición es esencial, si es que se desea realmente conseguir la calidad en un software. Es por eso que existen distintos tipos de métricas para poder evaluar, mejorar y clasificar al software final, en donde serán manejadas dependiendo del entorno de desarrollo del software al cual pretendan orientarse (32).

El componente fue implementado mediante la POO (Programación Orientada a Objetos), las clases son la estructura fundamental de este paradigma, por lo que la métrica a utilizar estará dirigida a ellas.

3.5.1 Métrica relaciones entre clases (RC)

Esta métrica se basa en la medición de distintos indicadores de calidad, dependiendo del número de relaciones de uso de una clase con otra, a continuación se explican los indicadores que mide (33):

- **Acoplamiento:** Un aumento del RC implica un aumento del acoplamiento de la clase.
- **Complejidad de mantenimiento:** Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.

- **Reutilización:** Un aumento del RC implica una disminución en el grado de reutilización de la clase.

- **Cantidad de pruebas:** Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

La siguiente tabla *(Tabla 7)* muestra cómo están definas las categorías según el criterio a evaluar:

Tabla 7. Métrica RC

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

3.5.2 Resultados de la aplicación de la métrica RC

La siguiente tabla (*Tabla 8*) muestra las clases con las cantidades de relaciones de uso que tienen y la categoría asignada en cada indicador.

Tabla 8. Resultados de la aplicación de la métrica RC

Clase	Cantidad de relaciones de uso	Acoplamiento	Complejidad de mantenimiento	Reutilización	Cantidad de pruebas
CmpmmercanciaController	1	Bajo	Baja	Alta	Baja
DatMovimientoMercanciaModel	2	Medio	Baja	Alta	Baja
DatMovimientoMercancia	2	Medio	Baja	Alta	Baja
BaseDatMovimientoMercancia	1	Bajo	Baja	Alta	Baja
CmpMercanciaServices	1	Bajo	Baja	Alta	Baja

Las gráficas que a continuación se presentan *(Figura 17, 18, 19 y 20)* se relacionan con la *Tabla 8* y representan gráficamente los resultados de la aplicación de la métrica RC según la categoría.

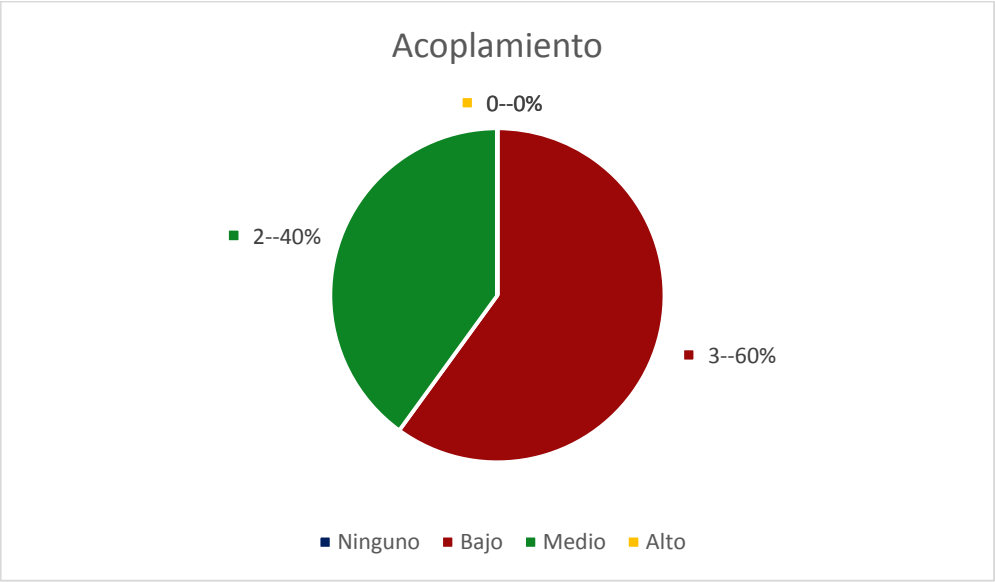


Figura 17. Resultados de la métrica RC, indicador Acoplamiento

Figura 18. Resultados de la métrica RC, indicador Complejidad de mantenimiento

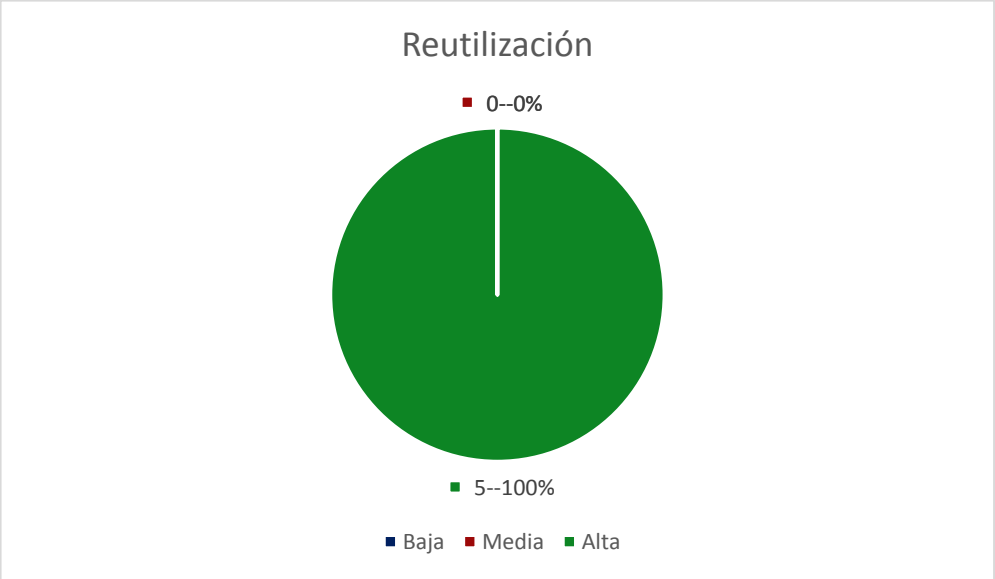


Figura 19. Resultados de la métrica RC, indicador Reutilización

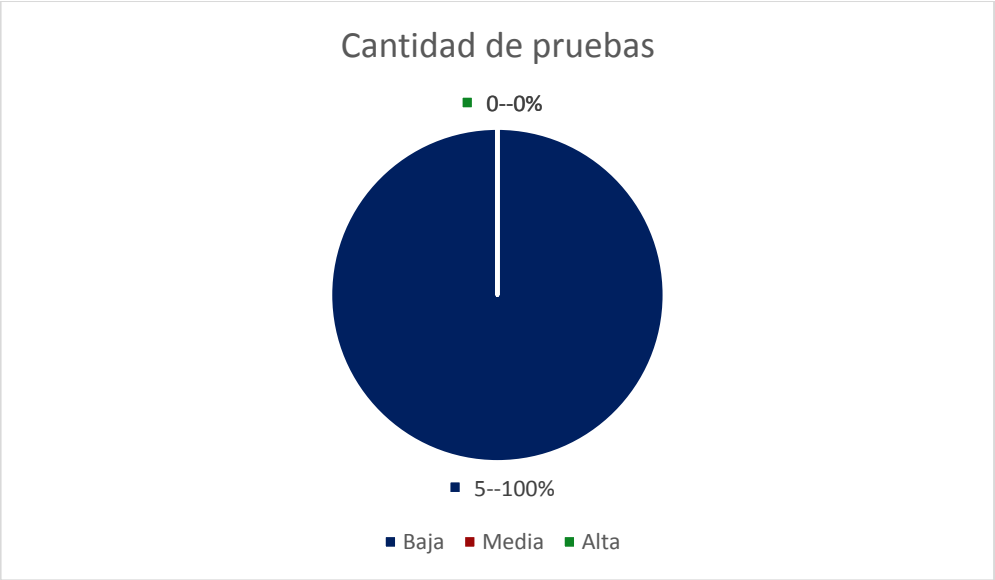


Figura 20. Resultados de la métrica RC, indicador Cantidad de pruebas

Análisis de los resultados de la aplicación de la métrica RC

Dada la evaluación de los indicadores se obtuvieron buenos resultados, los cuales arrojaron un 60% de bajo acoplamiento y un 100% de reutilización, significando que el diseño del componente cumple con el patrón bajo acoplamiento y es reutilizable para futuras integraciones. En cuanto a la complejidad de mantenimiento y la cantidad de pruebas, la métrica devolvió 100% en la categoría baja para ambos casos, logrando con esto un fácil mantenimiento al código y la implementación de pocas pruebas para evaluar el desempeño de las clases.

3.6 Pruebas de software

El objetivo de las pruebas, expresado de forma sencilla, es encontrar el mayor número posible de errores con una cantidad razonable de esfuerzo, aplicado sobre un lapso de tiempo realista (28). El proceso de pruebas tiene como objetivos fundamentales, planificar las pruebas necesarias en cada iteración, incluyendo las pruebas de integración y las pruebas de sistema. Diseñar e implementar las pruebas, creando los casos de pruebas que especifican qué probar. Además de realizar las diferentes pruebas y manejar los resultados de cada una sistemáticamente de forma que las no conformidades puedan ser corregidas (27).

Cualquier producto de ingeniería puede probarse de dos formas:

1. Conociendo la función específica para la que fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa y al mismo tiempo buscar errores en cada función (28).

2. Conociendo el funcionamiento del producto se pueden desarrollar pruebas que aseguren que todas las piezas encajan, o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada (28).

El primer enfoque se denomina “prueba de caja negra” y el segundo “prueba de caja blanca”, y para validar la solución propuesta se realizaron ambos tipos de pruebas.

3.6.1 Pruebas de caja blanca

La prueba de caja blanca es un método de diseño de casos de prueba, que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero de software puede obtener casos de prueba que garanticen que (28):

1. Se ejecute por lo menos una vez todos los caminos independientes de cada módulo.
2. Se ejecuten todas las decisiones lógicas en sus vertientes (verdadera y falsa).
3. Se ejecuten todos los bucles en sus límites y con sus límites operacionales.
4. Se ejecuten las estructuras internas de datos para asegurar su validez.

Las pruebas de caja blanca son consideradas como elementales para disminuir el número de errores en una aplicación y lograr así mejoras en la calidad del software.

Técnica del camino básico

La técnica del camino básico es un método de prueba de Caja Blanca propuesta por Tom McCabe. Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico de caminos de ejecución (28).

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes, por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes, se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico, garantizan que durante la prueba se ejecute por lo menos una vez, cada sentencia del programa (28).

Aplicación y análisis de las pruebas de caja blanca

En el acápite anterior se explicó cómo se realizan las pruebas de caja blanca mediante el método del camino básico. A continuación se ejemplifica la aplicación de dichas pruebas y se analizan los resultados obtenidos.

Primer paso: Realizar el grafo de flujo.

Se realizó como ejemplo el grafo de flujo del método *asignarApoderadoAction()* de la clase *CmpMMercanciaController,* que es la clase controladora del componente. Este método recibe por parámetro un arreglo con dos valores, uno es el identificador del movimiento de la mercancía y el otro es el identificador del apoderado de aduana. Después llama a un método de la clase CmpgmmercanciaModel, que es el encargado de asignar el apoderado al movimiento, acto seguido evalúa si la operación se realizó correctamente o no y emite un mensaje para notificar al usuario. A continuación se muestra una figura *(Figura 21)* que muestra el código del método, con los nodos señalados.

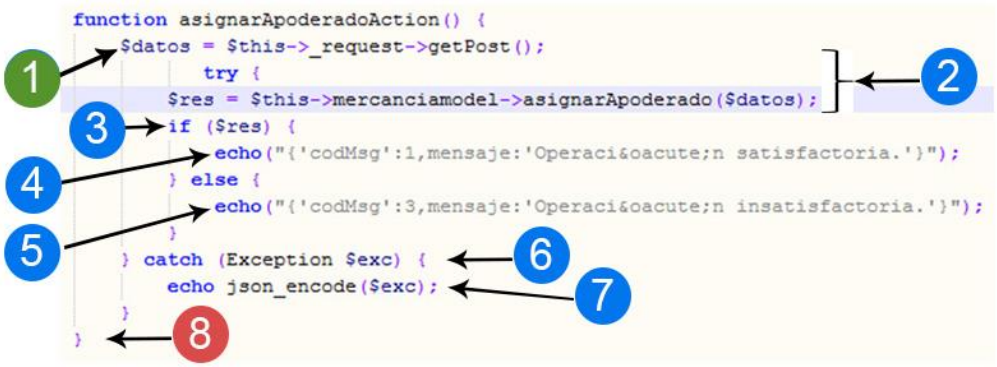


Figura 21. Método asignarApoderadoAction() con los nodos

Después de analizar el código fuente del método, se construyó el siguiente grafo de flujo *(Figura 22)*:

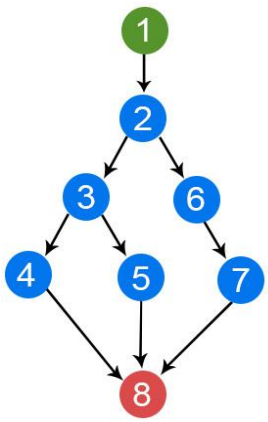


Figura 22. Grafo de flujo del método asignarApoderadoAction()

Segundo paso: Calcular la complejidad ciclomática.

Se utilizó para calcular la complejidad ciclomática la siguiente formula V(G) = A – N + 2, donde A es la cantidad de aristas del grafo y N la cantidad de nodos. Al sustituir los valores queda de la siguiente forma: V(G)=9-8+2.

La complejidad ciclomática da valor tres. Esto significa que la cantidad de casos de prueba que se necesitan realizar para transitar cada uno de los caminos independientes del grafo, es tres.

Tercer paso: Determinar un conjunto básico de caminos independientes.

Cada camino independiente debe recorrer al menos una arista que no haya sido recorrida con anterioridad. Los caminos independientes son:

1. 1, 2, 3, 4, 8
2. 1, 2, 3, 5, 8
3. 1, 2, 6, 7, 8

Cuarto paso: Preparar y realizar los casos de prueba, con los caminos básicos

En el paso anterior fueron obtenidos los caminos básicos, ahora se realizan los casos de prueba utilizando dichos caminos. La *Tabla 9* muestra los resultados obtenidos al aplicar los casos de prueba.

Tabla 9. Resultados de las pruebas de caja blanca

Cami-no	Entrada	Descripción de la entrada	Respuesta esperada	Respuesta del sistema
1	90000000002, 12340000000008	Dos números correctos	Muestra un mensaje con el texto siguiente: Operación satisfactoria.	Operación satisfactoria.
2	90000000001, 12340000999999	Uno de los dos números no existen en la base de datos	Muestra un mensaje con el texto siguiente: Operación insatisfactoria.	Operación insatisfactoria.
3	90000000004, A	Un número y una letra	Devuelve una excepción del sistema, controlada.	Mensaje de alerta: Ha ocurrido una

				excepción inesperada.

Como se pudo observar en la *Tabla 9* después de aplicarle las pruebas al método seleccionado, el resultado fue satisfactorio. De los 53 métodos que posee el componente, 40 fueron sometidos a estas pruebas, de los cuales el 75% resultó correcto y un 25% no devolvían los resultados esperados, por lo que fueron corregidos. Con esto queda demostrado la importancia de este tipo de pruebas, además del valor y calidad que le agregan a los productos de software.

3.6.3 Pruebas de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada, que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores a los que encuentran los métodos de caja blanca (28).

La prueba de caja negra intenta encontrar errores de las siguientes categorías:

1. Funciones incorrectas o ausentes.
2. Errores de interfaz.
3. Errores en estructuras de datos o en accesos a bases de datos externas.
4. Errores de rendimiento.
5. Errores de inicialización y de terminación.

A diferencia de la prueba de caja blanca, la prueba de caja negra tiende a aplicarse durante fases posteriores de la prueba. Ya que la prueba de caja negra ignora intencionadamente la estructura de control y centra su atención en el campo de la información (28).

Método de partición equivalente

La partición equivalente es un método de prueba de caja negra, que divide el campo de entrada de un programa en clases de datos, de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores (por ejemplo, proceso incorrecto de todos los datos de carácter) que, de otro modo, requerirán la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (28).

Para realizar las pruebas de caja negra se utilizó el método de partición equivalente y la tabla *(Tabla 10)* que a continuación se muestra, expone el diseño de caso de prueba para el RF3 Exportar movimiento de carga diario. El resto de los diseños de casos de pruebas se encuentran en el documento entregable "CEIGE_SGIEBK_Disenno_de_casos_de_prueba (Gestionar movimiento de carga diario).ods".

Tabla 10. Diseño de caso de prueba "Exportar movimiento de carga diario"

Escenario	Descripción	Nombre del fichero	Formato del archivo	Respuesta del sistema	Flujo central
1.1 Registrando datos válidos	El sistema permite exportar en un fichero excel, la vista actual donde está posicionado el usuario.	V	V	El sistema genera un fichero con el listado de la vista actual y lo guarda en la ubicación definida por el navegador o el usuario. El sistema muestra una notificación indicando que la acción se ha realizado satisfactoriamente: "Operación satisfactoria".	Clic en el botón Exportar. El sistema muestra una ventana. Se registran los datos correctamente. Clic en el botón Aceptar.
		Este es un listado	.xls		
1.2 Registrando datos inválidos	El sistema no permite exportar la vista actual si se registran datos inválidos.	I	N/A	El sistema muestra una alerta en cada campo que se intente introducir valores inválidos de la siguiente forma: Nombre del fichero: "Este campo solo debe contener letras y números.".	Clic en el botón Exportar. El sistema muestra una ventana. Se intenta registrar datos inválidos.
		$%&			

				Se muestra el mensaje: “Por favor verifique nuevamente existe(n) valor(es) incorrecto(s).”.	Clic en el botón Aceptar.
1.3 Dejando campos obligatorio vacíos	El sistema no permite exportar la vista actual si se dejan campos obligatorios vacíos.	I Vacío	I Vacío	El sistema no permite dejar campos obligatorios vacíos. Se muestra la alerta “Este campo es obligatorio.” Se muestra el mensaje: “Por favor verifique nuevamente existe(n) valor(es) incorrecto(s).	Clic en el botón Exportar. El sistema muestra una ventana. Se dejan campos obligatorios vacíos. Clic en el botón Aceptar.
1.4 Cancelar	El sistema cancela la operación.	El sistema cancela la operación.	El sistema cancela la operación.	El sistema cierra la ventana, cancelando así la operación.	Clic en el botón Exportar. El sistema muestra una ventana. Clic en el botón Cancelar.

Aplicación y análisis de las pruebas de caja negra

Las pruebas de caja negra fueron divididas en dos disciplinas diferentes: Pruebas Internas y Pruebas de Liberación.

Las Pruebas Internas fueron aplicadas por el equipo de aseguramiento de la calidad del CEIGE, donde se realizaron tres iteraciones. A continuación la *Figura 23* muestra los resultados por iteraciones.

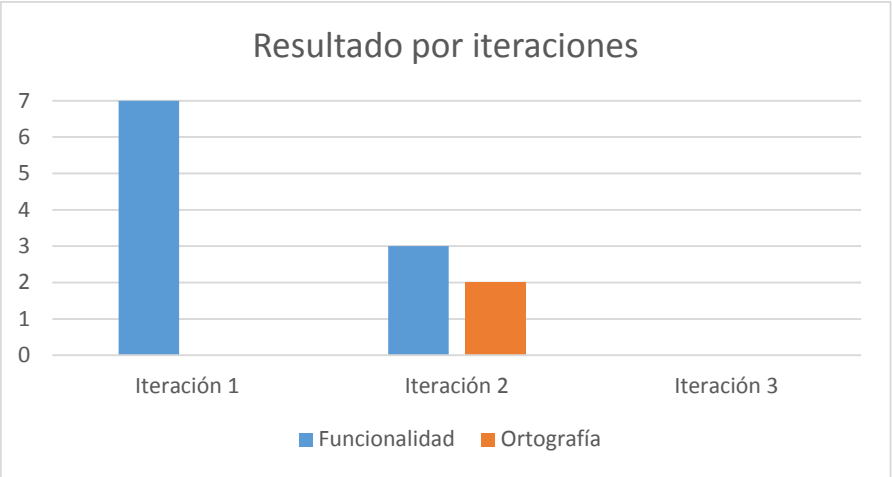


Figura 23. Resultados de las pruebas de caja negra, pruebas internas

En la iteración uno se detectaron siete no conformidades, las cuales fueron resueltas, al igual que las cinco detectadas en la iteración dos. Al realizar la iteración tres quedaron resueltas todas las no conformidades, logrando con ello que la interfaz del componente responda de manera correcta a cada petición del usuario. Se emitió un acta de liberación por parte del equipo que realizó las pruebas *(Anexo 1 al 3).*

Las Pruebas de Liberación fueron aplicadas en la Dirección de Calidad UCI. Se realizaron tres iteraciones, detectando un total de 12 no conformidades. La *Figura 24* muestra una gráfica que detalla las no conformidades de manera más explícita.

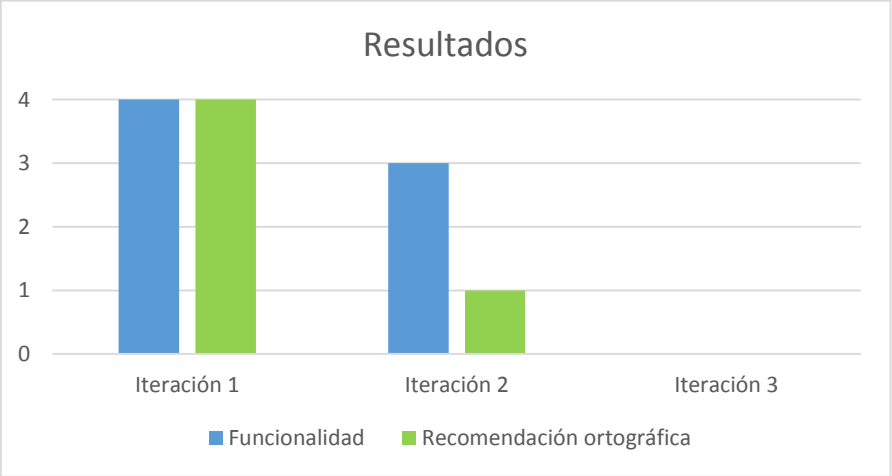


Figura 24. Resultados de las pruebas de caja negra, pruebas de liberación

En la iteración uno y dos se identificaron ocho y cuatro no conformidades respectivamente, todas fueron resueltas, mientras que la iteración tres resultó libre de no conformidades. Esta prueba permitió un análisis detallado del Sistema de Importación y Exportación Xedro-Apside v1.0 y logró un aumento de la calidad del componente, el mismo quedó libre de errores.

3.7 Pruebas de aceptación con el cliente

En la disciplina de Pruebas de Aceptación se realizó la prueba de aceptación con el cliente. El director del área de logística de la empresa BK Import-Export Rolando Nuñes Cruz, fue el encargado de validar las funcionalidades de la solución. Los resultados fueron satisfactorios, el componente cumplió el 100% de las expectativas del cliente. Además el cliente expresó recomendaciones que se deben tener en cuenta para futuras versiones del componente. Se firmó un acta de aceptación como constancia que se puede visualizar en el *Anexo 4*.

3.8 Aporte de la solución y beneficios sociales

Con el desarrollo del Componente Gestor del Movimiento de la Mercancía para el Sistema de Importación y Exportación Xedro-Apside v1.0, se logró un sistema capaz de apoyar el seguimiento y control de los movimientos de carga que realiza la empresa BK Import-Export. Además el componente puede ser utilizado por cualquier entidad comercializadora cubana, con esto se logra un aporte al proceso de informatización de la sociedad.

El componente forma parte del Sistema de Importación y Exportación Xedro-Apside v1.0, lo que posibilita la integración con los componentes existentes y con los de futura creación. Además está desarrollado con tecnologías de código abierto, con lo que tributa a la soberanía tecnológica por la que aboga el país.

La utilización del componente ahorra esfuerzo a los supervisores de movimientos y a los apoderados de aduana, ya que muestra en cada momento el estado en que se encuentra el movimiento de la mercancía y posibilita el estudio de cada uno de ellos mediante los documentos Excel que genera la solución, además de las estadísticas, que brindan comparaciones necesarias que facilitan la toma de decisiones. Una interpretación correcta de las estadísticas puede conllevar a un ahorro significativo para la entidad.

3.9 Conclusiones parciales

La implementación resultó satisfactoria, con el uso de los estándares de codificación se estableció un lenguaje común entre los programadores, que facilita el mantenimiento del código. Se obtuvo un diseño de calidad avalado por los buenos resultados alcanzados al aplicar la métrica RC. El componente quedó libre de errores después de la aplicación de las pruebas de caja blanca y de caja negra. El componente cumplió el 100% de las expectativas del cliente, lo que se demuestra en la prueba de aceptación. El desarrollo de la solución se realizó con herramientas de código abierto, lo que contribuye a la política de soberanía tecnológica de Cuba.

Conclusiones generales

Para la realización del presente trabajo se trazaron varios objetivos y tareas que tributaban a un objetivo general y que permitieron arribar a las siguientes conclusiones:

- El estudio realizado a diferentes sistemas de comercialización permitió identificar la necesidad de implementar un componente, para el Sistema de Importación y Exportación Xedro-Apside v1.0, de forma tal que permitiera el seguimiento y control de los movimientos de la mercancía y que su implementación fuera con tecnologías y herramientas de código abierto.

- Se obtuvo un total de 26 requisitos funcionales al aplicar las técnicas tormenta de ideas y entrevistas. Estos requisitos fueron validados y el cliente quedo satisfecho. Además se logró un diseño de calidad avalado por la métrica RC.

- Se desarrolló un producto web que cumple con las especificaciones del cliente y que contribuye a la política de soberanía tecnológica por la que aboga Cuba. Se estableció además un lenguaje común entre los desarrolladores que permite un fácil mantenimiento al código.

- La solución cuenta con la calidad requerida, ya que fue sometida a varias pruebas de caja blanca, caja negra y una prueba de aceptación con el cliente. Logrando un componente libre de errores y que cumple con las especificaciones del negocio.

Recomendaciones

En la realización de la prueba de aceptación el cliente planteó dos ideas, las cuales deben tenerse en cuenta para próximas versiones del componente:

- Agregar una funcionalidad que permita guardar las gráficas como imágenes.
- Agregar notificaciones al usuario, en caso de que no se cumpla algún indicador de eficiencia fijado por la entidad.

Bibliografía

1. CEIGE. Gespro - CEIGE. [En línea] [Citado el: 11 de 05 de 2015.] http://gespro.ceige.prod.uci.cu/.

2. Definicion.de. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://definicion.de/comercializacion/.

3. Definicion.de. [En línea] [Citado el: 25 de 03 de 2015.] http://definicion.de/importacion/.

4. Integration Point. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://www.integrationpoint.com/es/products/importcompliance.html.

5. Sistema ERP y CRM para PyMEs | EGA Futura. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://www.egafutura.com/.

6. Mistral Caribe Holding S.A. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://www.mistralcaribe.com/index.php?option=com_content&view=article&id=124&Itemid=49.

7. Subdirección de producción, CEIGE. *Modelo de Desarrollo de Software v1.1.* La Habana : s.n., 2012.

8. Sánchez, Tamara Rodríguez. *Metodología de desarrollo para la Actividad Productiva de la UCI v1.1.* La Habana : s.n., 2014.

9. Baryolo, Oiner Gómez. *Solución Informática de Autorización en Entornos Multientidad y Multisistema.* La Habana : s.n., 2010.

10. Programadores ExtJS. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://www.extjs.mx/.

11. Programmer´s Reference Guide - Zend Framework. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://framework.zend.com/manual/1.12/en/manual.html.

12. Home -- Doctrine Project. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://www.doctrine-project.org/.

13. Lenguajes de programación. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://es.kioskea.net/contents/304-lenguajes-de-programacion.

14. JavaScript | MDN. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] https://developer.mozilla.org/es/docs/Web/JavaScript.

15. PHP: Hypertext Preprocessor. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] https://php.net.

16. PHPExcel. [En línea] 02 de 03 de 2014. [Citado el: 06 de 05 de 2015.] https://phpexcel.codeplex.com/.

17. Tutorial de UML. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://users.dcc.uchile.cl/~psalinas/uml/introduccion.html.

18. Software Design Tools for Agile Teams, with UML, BPMN and more. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://www.visual-paradigm.com.

19. Documentación del servidor Apache. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://httpd.apache.org/docs/2.0/es.

20. PostgreSQL. [En línea] [Citado el: 08 de 02 de 2015.] http://www.postgresql.org.

21. Navegador firefox. [En línea] [Citado el: 08 de 02 de 2015.] https://www.mozilla.org/es-ES/firefox/desktop/.

22. Portal del IDE Java de Código Abierto. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] https://netbeans.org/index_es.html.

23. Larman, Craig. *UML y Patrones.* 2003.

24. DesarrolloWeb.com. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://www.desarrolloweb.com/articulos/que-es-mvc.html.

25. Patrón de diseño. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] https://msdn.microsoft.com/es-es/library/bb972240.aspx.

26. Información tecnológica. [En línea] La Serena, 28 de 01 de 2013. http://www.scielo.cl/scielo.php?pid=S0718-07642013000300012&script=sci_arttext. ISSN 0718-0764.

27. Jacobson, Ivar, Rumbaugh, James y Booch, Grady. *Proceso Unificado de Desarrollo de Software.* 2002.

28. Pressman, Roger. *Ingeniería de software, un enfoque práctico.* 2002.

29. Sommerville, Ian. *Ingeniería del software séptima edición.* Madrid : Pearson Educación, 2005. ISBN: 84-7829-074-5.

30. Romero, Oscar Casasola. Programación en castellano. [En línea] [Citado el: 05 de 05 de 2015.] http://programacion.net/articulo/introduccion_a_uml_181/7.

31. Microsoft. Revisiones de código y estándares de codificación. [En línea] Microsoft. [Citado el: 30 de 04 de 2015.] https://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

32. desarrolloweb.com. desarrolloweb.com. [En línea] [Citado el: 04 de 05 de 2015.] http://www.desarrolloweb.com/articulos-copyleft/articulo-metricas-de-software.html.

33. Lorenz, Mark y Kidd, Jeff. *Object-Oriented Software Metrics.* 1994. 978-0131792920.

34. wordreference.com. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://www.wordreference.com/definicion/stock.

35. Definicion.de. [En línea] 08 de 02 de 2015. [Citado el: 08 de 02 de 2015.] http://definicion.de/arancel/.

36. Mincex. *Resolución No. 50 del 2014, Reglamento para la Actividad de Importación y Exportación.* La Habana : s.n., 2014.

Anexos


UCi

Informáticas

CENTRO PARA LA INFORMATIZACIÓN DE ENTIDADES

Validación y Verificación de Requisitos (V&V)

ACTA DE LIBERACIÓN DEL COMPONENTE MOVIMIENTO DE MERCANCÍA DEL PRODUCTO XEDRO-APSIDE 1.0

Acta de Liberación

Anexo 1

2 Datos del producto

Emitida a favor de: Movimiento de Mercancía del Sistema de Gestión de Importación y Exportación BK Import/Export
Fecha de emisión del acta: 14/05/2015
Responsable: Leonardo Argumedo Retolaza
Carga: Estudiante

2.1 Clasificado como:

- Aplicación

2.2 Detalle de los elementos probados y su estado final:

Artefacto	Estado Final
Aplicación	0 No Conformidades

2.3 Cantidad de Iteraciones:

Para la revisión se emplearon un total de 3 iteración de trabajo para lograr el resultado de 0 (cero) No Conformidad.

Artefacto	Versión	Estado final	Cantidad Iteraciones	Tipos de pruebas realizadas	Fecha de liberación
Aplicación	Versión 1.0	No Conformidades	3 Iteraciones total	Pruebas Funcionales Pruebas de regresión	14/05/2015

CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES
Universidad de las Ciencias Informáticas

Anexo 2

UCI | DS-AP-CL0001 - Acta de liberación

Ing. Lisandro Pérez Zaldívar — Administrador de configuración

Leonardo Argumedo Restiva — Estudiante

Ing. Yisel Niño Benítez
Asesor de Calidad

Ing. Yirusdis M. Aguenedo Lara
Jefa de Proyecto

CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DE ENTIDADES
Universidad de las Ciencias Informáticas
Carretera a San Antonio de los Baños, Torrens, Boyeros, Ciudad de La Habana, Cuba.
Teléfono: + 53 (7) 837 3600
E-mail: software.gestion@uci.cu

17

Anexo 3

ACTA DE ACEPTACIÓN

En cumplimiento del **Convenio de colaboración** de la Empresa Comercial BK Import/Export y en función de la ejecución del proyecto: Sistema de Gestión Comercial de Importación, se han efectuado las actividades siguientes:

- Revisión y Aprobación de los Requisitos funcionales y no funcionales por el Cliente.
- Realización de pruebas exploratorias al Componente Gestor del Movimiento de la Mercancía desarrollado para lograr un seguimiento y control de los movimientos de carga diarios.

La Parte Cliente, luego de haber revisado los productos de trabajo determina que aceptan la solución realizada.

Entrega	**Recibe**
Nombre y apellidos: Leonardo Argumedo Reloba	**Nombre y apellidos:** Rolando Núñez Cruz
Cargo: Estudiante de la Universidad de las Ciencias Informáticas	**Cargo:** Director de Logística. Empresa Comercial BK Import/Export

Fecha: 20/05/2015

Anexo 4