Universidad de las Ciencias Informáticas

Facultad 3

Título: Diseño e implementación del módulo Despacho Comercial (DC) para el Sistema de Gestión Integral de Aduana.

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

Autor(es): Daniel Alberto Ojeda Estrada.

Reinier Silverio Figueroa.

Tutores: Ing. Eilys Pacheco Rodríguez.

Ing. Boris Enrique Ruiz Guerra.

La Habana, Cuba

Junio, 2012

Síntesis de los tutores

Ing. Eilys Pacheco Rodríguez. (epacheco@uci.cu) Graduada de Ingeniería en Ciencias Informáticas desde el año 2009. Instructor. 2 Años de experiencia en el tema.

Ing. Boris Enrique Ruiz Guerra. (beruiz@uci.cu) Graduado de Ingeniería en Ciencias Informáticas desde el año 2011. Recién Graduado en Adiestramiento.

DECLARACIÓN DE AUTORÍA

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

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

Daniel Alberto Ojeda Estrada.

_____

Firma del Autor

Reinier Silverio Figueroa.

_____

Firma del Autor

Ing. Eilys Pacheco Rodríguez.

_____

Firma del Tutor

Ing. Boris Enrique Ruiz Guerra.

_____

Firma del Co-Tutor

"En la ciencia todo el crédito va al hombre que convence al mundo de una idea, no al que la concibió primero."

Willian Osler

Dedicatoria

A mi madre, a mi novia, a toda mi familia y amigos. A todos los que de una forma u otra han estado conmigo en los momentos buenos y malos.

Daniel

Dedico este trabajo especialmente a mis hermanas, a mi sobrina, a mis padres y a toda la familia. Lo dedico además a mis amigos por ser mi familia en la escuela.

Reinier

Agradecimientos

Primeramente agradecer a las tres mujeres más importantes en mi vida, mi madre por saber educarme y darme todo el amor de madre y padre que necesitaba para poder ser lo que soy actualmente. A mi futura esposa por todo el apoyo y amor brindado en estos 5 años, por ser mi bastón, mi ayuda en todo momento, gracias por permitirme ser parte de tu vida. A mi hermana por hacerme saber en todo momento lo importante que soy para ella en su formación, espero seguir siendo ese ejemplo para ti. A mi otra madre Irma (mi suegra), por toda la confianza, amor, cariño y dejarme ser parte de su familia. A mi padrastro Jose por ser un padre conmigo por cuidarme a mis tres tesoros en el tiempo que no han estado a mi lado, por su gran corazón al mantenerse firme junto a mi familia. A mis tías y tíos por criarme como un hijo más y ayudar a mi mamá en mi educación. A mis abuelos por ser mis segundos padres. A María, Jesús y Nayi por confiar siempre en mí y por cuidarme mucho a mi cosita...

A mi familia de amigos Ari, Katy, Yami, Giorgy, Martin y nuestra madre común LISI…por los momentos inolvidables que pasamos juntos saben que siempre estarán en mi corazón…

A mi otro piquete los ingenieros Vandálicos Berna, Carlos, Andy, Robert y Boris gracias por ser mis amigos, cómplices y hermanos…

A las chicas VIP Yelenis, Analay, Yanet, Nailin gracias por todos los momentos compartidos….y demostrarme que con simples detalles se conquistan grandes cosas….

A Soraya por ser una madre en la universidad…gracias por estar siempre ahí.

A mi compañero de tesis por todo el trabajo y esfuerzo realizado para lograr terminar con éxito nuestros estudios, por compartir todo este tiempo y confiar en que si se puede…

A la profesora Lianet por su ayuda y tutela en la confección de esta tesis….

A todos los que de una forma u otra han contribuido en mi carrera, gracia por todo…

Daniel

AGRADECIMIENTOS

El acto del agradecimiento se define como la acción de mostrar a los implicados de determinado resultado cuán importante ha sido su apoyo, colaboración, actitud, u opinión y sobre todo la idea clara de que siempre se les tiene en cuenta. Es por eso que comienzo este acto aclarando que mis agradecimientos abarcan no solo este periodo sino, desde el primer momento en que comencé a comprender cada fenómeno que ocurría a mi alrededor y por tanto a esas primeras personas que sentaron las bases de mi personalidad, mi forma de ser y mi educación: mis padres. Comenzando por mi madre que siempre me ha demostrado que la vida te pone obstáculos pero; estos surgieron para ser vencidos y que todas las metas propuestas conllevan algún sacrificio, me ha enseñado sobre todo a tener fe y a nunca perder la confianza en mí mismo, ella es quien a través de su actuar ha sido uno de mis principales ejemplos a seguir como persona. A mi padre por ser esa persona conocedora que siempre tiene la respuesta a mis curiosidades e inquietudes y sobre todo por nunca dejar de apoyarme, esas razones lo convierten en otro de mis ejemplos a seguir. A mi familia en general, por creer en mí, por los consejos, por los buenos y malos momentos vividos y por ser la mejor del mundo. A todos mis profesores, por saber incentivar mi apetito de conocimiento, transmitirme experiencias e invitarme a encontrar mis propios pasos como profesional. A mis amistades, las que continúan en la escuela, las que no pero aún se preocupan y se comunican, a las más recientes, a las de primer año, a todas por formar parte de mi vida y por hacerme un espacio en su corazón. A mis tutores, por la comprensión, la disponibilidad y el tiempo dedicado, así mismo a la oponente por todo el apoyo.

Reinier

Resumen

La Aduana General de la República de Cuba (AGR) se ha trazado como meta la informatización de todas las operaciones que forman parte de los diferentes procesos para la gestión del negocio aduanero. El área correspondiente al Despacho Comercial presenta varias dificultades debido a que el sistema desplegado actualmente, Sistema Único de Aduana (SUA), no informatiza la totalidad de sus procesos y los ya informatizados en ocasiones resultan engorrosos en el momento de su realización. Además llevar a cabo la inclusión de mejoras y funcionalidades que suplan dichas dificultades se hace poco factible, razón por la cual se hace necesario su reimplementación.

El presente trabajo tiene como objetivo desarrollar el módulo Despacho Comercial para el Sistema de Gestión Integral de Aduana (GINA), permitiendo la reducción del tiempo de despacho de las mercancías, factor imprescindible que ayudaría no solamente a hacer más eficiente el despacho aduanero, sino que mitigaría los costos derivados del tiempo de espera. Para el desarrollo de la solución se elaboró el diseño que abarca los requisitos identificados en la fase de análisis, sirviendo como base a la fase de implementación, regido por el Modelo de Desarrollo de Software empleado por el Departamento de Soluciones para la Aduana.

La solución desarrollada permitirá erradicar los problemas existentes en el SUA, disponiéndose de un software que gestiona las formalidades del Despacho Comercial para todas las aduanas del país.

Palabras claves: Aduana**,** Despacho Comercial, Sistema de Gestión Integral de Aduana (GINA).

Índice de contenidos

Índice de Figuras

Índice de Tablas

Introducción

En la actualidad los mercados internacionales se han expandido de manera acelerada, incrementando la selectividad y exigencia en sus productos y servicios, haciendo más difícil la participación de las empresas en estos mercados. Los países deben contar con mecanismos que faciliten el comercio en lugar de entorpecerlo. Por lo tanto, se requiere de un aparato industrial y comercial verdaderamente eficiente que cree las condiciones necesarias para salir a los mercados internacionales con un grado de eficiencia tal, que les garantice su permanencia.

Dentro de este contexto, se inserta la necesidad de capacitar a las aduanas para que en un plazo muy corto puedan prestar servicios complejos y eficientes. La dotación de instrumentos y recursos contribuyen favorablemente a mejorar la competencia y la adecuación a los constantes cambios que exige y demanda el comercio mundial. Dentro de tales necesidades se destaca el aporte de las tecnologías de la información, para simplificar y aligerar procedimientos (y con esto el flujo de las operaciones), generando respuestas inmediatas según las necesidades dadas.

La Aduana General de la República de Cuba (AGR) creada el 5 de febrero de 1963, tiene dentro de sus misiones garantizar la seguridad y protección de la sociedad socialista y de la economía nacional, así como la recaudación fiscal y la emisión de las estadísticas del comercio exterior, a través del cumplimiento de las políticas estatales de competencia aduanera para el tráfico internacional de viajeros, mercancías y medios de transporte. Este órgano de control en frontera tiene el reto de informatizar sus principales procesos aduaneros dentro de ellos el Despacho Comercial[1], No Comercial, de Viajeros, de Medios de Transporte, Postal y Envío; incrementando la calidad de sus diferentes funciones debido al desarrollo que ha ido alcanzando la informática y las comunicaciones en el país.

Se hace necesario encontrar una vía factible que relacione la tecnología con los diferentes procesos aduaneros. El Sistema Único de Aduana (SUA) desarrollado por el Centro de Automatización para la Información y la Dirección (CADI), es la respuesta de la AGR a la situación antes descrita.

[1] Formalidades aduaneras que se realizan cuando se comercializan mercancías, tanto en importaciones como exportaciones.

A pesar de que el sistema cuenta con el módulo Despacho Comercial, posee un grupo de características que lo hacen inflexible al cambio:

- Ejecuta validaciones a nivel de base de datos por medio de procedimientos almacenados trayendo como consecuencia que el mantenimiento sea engorroso.
- Realiza las operaciones por pantallas poco orientadas al uso.

Además no informatiza determinados procesos, entre ellos:

- La devolución y reintegro de derechos de aduana.
- La modificación de un régimen cancelado.
- La desanulación de la declaración de mercancías.
- El registro en el sistema del levante manual[2].
- La recepción y validación de manera electrónica de los documentos complementarios[3], adjuntos a la declaración de mercancías[4], eje rector de este proceso e introducidos por el declarante[5].

La AGR requiere nuevas funcionalidades y un trabajo más profundo y perfeccionado. Por estas necesidades, se ha centrado en un proceso de reingeniería del SUA, inicialmente con fuerzas propias y en la actualidad con la cooperación de la Universidad de Ciencias Informáticas (UCI), lo cual permitirá tener un sistema con herramientas más modernas y potentes. Surge entonces la Gestión Integral de Aduana (GINA) como nueva solución, sistema que aún se encuentra en proceso de inclusión de módulos para la gestión del negocio aduanero. El Despacho Comercial ocupa un lugar importante en el sistema a desarrollar, creándose con el fin de incorporar dentro de sus funcionalidades los procesos previamente expuestos que no son gestionados por el módulo Despacho Comercial del SUA actualmente en

[2] Acción de las aduanas de permitir que las mercancías sometidas a despacho se pongan a disposición de la persona concerniente.

[3] Aquellos documentos establecidos por la norma de despacho aduanero de las mercancías, para la formalización de una declaración de mercancía.

[4] Manifestación en la forma prescrita por la Aduana, por la que los interesados indican el régimen aduanero que se ha de aplicar a las mercancías y proporcionan los datos que la Aduana exige para la aplicación de este régimen.

[5] Toda persona natural o jurídica que hace una declaración en aduana o en nombre de la cual esta declaración es hecha, siempre que acredite el derecho a disponer de las mercancías.

explotación en la AGR, permitiendo la reducción del tiempo de despacho de las mercancías en la aduana, factor imprescindible que ayudaría no solamente a hacer más eficiente el despacho aduanero, sino que mitigaría los costos derivados del tiempo de espera.

Por tanto se define como **problema a resolver:** existen dificultades en la rapidez con que se realiza las formalidades del proceso de Despacho Comercial. Centrándose en el **objeto de estudio**: sistemas de gestión aduanera que informatizan el proceso de Despacho Comercial. Enmarcándose en el **campo de acción**: el proceso de Despacho Comercial en el Sistema de Gestión Integral de Aduana.

Se persigue como **objetivo general**: desarrollar el módulo Despacho Comercial del Sistema de Gestión Integral de Aduana para aumentar la rapidez con que se realizan sus formalidades.

Del mismo se derivan los siguientes **objetivos específicos**:

1. Realizar el marco teórico de la investigación para establecer el estado del arte de los sistemas de gestión aduanera existentes.
2. Desarrollar la solución propuesta que permita dar respuesta a la situación problemática existente.
3. Validar la solución obtenida para garantizar el buen funcionamiento de la misma.

Las **tareas de la investigación** definidas para dar cumplimiento a los objetivos propuestos son:

1. Recopilación de la bibliografía necesaria para comprender los principales conceptos relacionados con el proceso de Despacho Comercial.
2. Investigación sobre los sistemas que implementan el Despacho Comercial en el ámbito internacional.
3. Investigación sobre los sistemas que implementan el Despacho Comercial en el ámbito nacional.
4. Estudio del Modelo de Desarrollo de Software a aplicar.
5. Estudio de la arquitectura base definida para el sistema GINA.
6. Diseño del módulo Despacho Comercial.
7. Implementación del módulo Despacho Comercial.
8. Realización de pruebas de calidad de software al módulo Despacho Comercial.
9. Documentación del trabajo realizado.

Métodos de investigación científica empleados:

- Análisis de la Literatura: según el libro ThesisProjects del año 2008, creado por M. Berndtsson y colaboradores este método se refiere a un examen sistemático de un problema determinado, por medio de un análisis de fuentes publicadas, llevado a cabo con un propósito específico en mente. (Berndtsson, 2008) Se utilizará para realizar el estudio a profundidad de las herramientas de trabajo que se usarán en la realización del estado del arte; también para conocer a plenitud los artefactos que servirán para el diseño; la profundización en el lenguaje de programación, los estándares de codificación definidos y en la búsqueda de otros sistemas que realicen funciones de la misma índole.

- Modelado: como su nombre lo indica este método se encarga de realizar una reproducción simplificada de la realidad, es uno de los más importantes en el campo de la construcción de software y se utiliza para la creación de todos los artefactos necesarios en el diseño; el modelo de datos, el diagrama de clases, de componentes y otros.

- Implementación: según el libro ThesisProjects del año 2008, creado por M. Berndtsson y colaboradores el objetivo de este método es demostrar que la solución tiene ciertas propiedades o que (bajo ciertas condiciones) se comporta de una manera específica. (Berndtsson, 2008) Se emplea en la realización de la implementación de todo el diseño que se modeló anteriormente y así dejar plasmado las ventajas obtenidas.

- Histórico – Lógico: combinación del método histórico y el lógico, donde el método lógico descubre las leyes fundamentales de los fenómenos basándose en los datos que le proporciona el método histórico. Se pone en práctica en la realización del estado del arte, vinculándose con la evaluación de las herramientas a utilizar, patrones de diseño y de otros software que realizan los procesos de Despacho Comercial.

El presente documento está compuesto por tres capítulos que dan respuesta a los objetivos específicos mencionados anteriormente. Cada uno de ellos contiene los elementos necesarios que conforman en su conjunto la solución final, partiendo del estudio del problema planteado.

Capítulo 1. Fundamentación Teórica: abarca toda la investigación realizada para establecer el estado del arte de los sistemas de gestión aduanera que implementan el Despacho Comercial a nivel mundial y nacional, se presentarán los conceptos esenciales vinculados con el Despacho Comercial, las herramientas, tendencias y tecnologías actuales que se emplearán en la construcción de la solución y aspectos esenciales que sirvan de soporte a la misma.

Capítulo 2. Diseño e Implementación del sistema: se enfatiza en la parte técnica, se presentan los diferentes artefactos que se obtienen del diseño realizado, también se detallará la implementación desarrollada a partir del diagrama de clases, el modelo de datos, diagrama de componentes y en general toda una descripción del diseño e implementación del módulo Despacho Comercial.

Capítulo 3: Constatación de la solución: se realizarán las pruebas unitarias y funcionales al producto y se dará a conocer los resultados de la solución.

Capítulo 1. Fundamentación teórica

1.1 Introducción

En el capítulo se realizará el estado del arte sobre los sistemas que se utilizan para realizar el Despacho Comercial. Además se definen los principales conceptos asociados al dominio del problema. Se plasmará los resultados de la investigación ejecutada sobre las tecnologías y herramientas a utilizar para el desarrollo de la solución planteada, incluyendo diferentes patrones y metodologías de diseño.

1.2 Conceptos asociados al dominio del problema

Las aduanas

La Resolución No. 33 “Glosario de Términos Aduaneros” emitida por el Jefe de la Aduana General de la República de Cuba, del 18 de octubre de 1996, define como Aduana a: los servicios administrativos responsables de la aplicación de la Normativa Aduanera y de la determinación y percepción de los derechos de aduanas, tasas y demás derechos recaudables. Servicio público encargado de ejecutar el control aduanero aplicable a la entrada, el tránsito, el cabotaje, el trasbordo, el depósito y la salida del territorio nacional de mercancías, viajeros y sus equipajes, bienes y valores sujetos a regulaciones especiales y los medios en que se transporten. (Magna, 2009)

Las aduanas en el contexto mundial son entidades muy antiguas que fueron creadas para el control del intercambio comercial con el exterior. Las naciones modernas desarrollan sus actividades a través de las mismas, como forma de regular la entrada y salida de mercancías en su territorio para alcanzar principalmente sus objetivos económicos, es decir, son las llaves que abren o cierran el comercio exterior, en consecuencia, es innegable la gran importancia que ellas tienen en la actividad económica de cada país. (Rodríguez, 2008)

No es función de la aduana ni de la administración que gira alrededor de ella el establecer trabas al Comercio Internacional, más bien todo lo contrario. En las diferentes fases que abarca las operaciones como embarque de productos, documentación, liquidación de derechos e impuestos, valoración, clasificación, entre otros. Su objetivo es facilitar el comercio, hacerlo más práctico, expeditivo y funcional. (Rodríguez, 2008)

Despacho aduanero

El despacho en las aduanas significa el cumplimiento de las formalidades aduaneras necesarias para importar y exportar las mercancías o someterlas a otros regímenes, operaciones o destinos. (Magna, 2009)

Comercio

El término comercio proviene del concepto del latín *commercĭum* y se refiere a la negociación que se entabla al comprar o vender géneros y mercancías. También se denomina comercio a la tienda, almacén o establecimiento comercial y al conjunto o clase de comerciante. En otras palabras, el comercio es la actividad socioeconómica que consiste en la compra y venta de bienes, ya sea para su uso, venta o transformación. (Real Academia Española)

Despacho Comercial

El Despacho Comercial se puede definir como las formalidades aduaneras que se realizan cuando se comercializan mercancías, tanto en importaciones como exportaciones.

1.3 Sistemas de gestión aduanera que informatizan el proceso de Despacho Comercial

Las grandes posibilidades que ofrecen las tecnologías y las comunicaciones han generado que el comercio se haya convertido en un fenómeno que caracteriza esta época. En este marco, el servicio aduanero cumple una labor de facilitación del comercio internacional. Las aduanas del mundo actualmente han pasado a convertirse de aduanas fiscales en aduanas económicas, siguiendo esta línea como la mejor política para sus procesos. La reducción del tiempo del despacho de las mercancías en la aduana, sería un factor imprescindible que ayudaría no solamente a hacer más eficiente el despacho aduanero, sino que mitigaría los costos derivados del tiempo de espera. En este proceso, las aduanas han tomado gran auge y aprovechando el desarrollo vertiginoso de las tecnologías, han desarrollado varias aplicaciones que facilitan el Despacho Comercial. A continuación se realiza un análisis de algunas de las soluciones extranjeras y cubanas utilizadas con este fin.

1.3.1 Sistemas informáticos extranjeros

1.3.1.1 Sistema Aduanero Automatizado (SIDUNEA)

SIDUNEA constituye una de las soluciones extranjeras más extendidas a nivel mundial, se define como:

"…un sistema informático de gestión aduanera, adecuado a la era de la información global, diseñado para las administraciones de aduanas, cubre la mayor parte de los procedimientos del comercio exterior, maneja manifiestos y declaraciones de aduana, procedimientos de contabilidad, de tránsito de mercancía y regímenes aduaneros suspensivos y genera datos estadísticos de comercio exterior para uso en el análisis económico." (Guerra, y otros, 2006)

Actualmente se utiliza o está en proceso de implementación en 93 países. La región más beneficiada es Europa donde se encuentran 31 de las instalaciones, seguida por Latinoamérica y el Caribe en la cual 23 países lo implementan, en este caso también existen 17 países en la región Asia-Pacífico y 11 de la región de Europa-Central y Oriental con la mayor área de crecimiento. (Guerra, y otros, 2006)

Los resultados alcanzados por este sistema se sustentan sobre la base de una serie de módulos que componen este sistema garantizando la calidad de los servicios brindados. Entre estos módulos tratados se encuentran: (Bolivia, 2011)

MODCBR

Es el Módulo de Aduanas y trabaja principalmente con la Declaración de Mercancías de Exportación (DME), su ingreso al sistema, su verificación local y remota, registro, aplicación del resultado de selectividad y validación. Adicionalmente contiene opciones de reporte para verificar el estado de bienes declarados bajo regímenes suspensivos tales como el depósito de aduanas. Se describe como el módulo central del sistema SIDUNEA, es la base de los procedimientos de control, del cobro y liquidación de los impuestos.

MODBRK

El Módulo del Declarante ha sido diseñado para ser utilizado por un declarante o agente despachante de aduanas, permitiendo una conexión electrónica directa con el sistema SIDUNEA. Esta es una versión modificada del módulo MODCBR.

MODSEL

Este módulo proporciona al usuario la habilidad de controlar la selección y el flujo de declaraciones que pasan por el sistema SIDUNEA en una exportación definitiva. Contiene controles para bloquear la liquidación de aquellas declaraciones seleccionadas para aforo[6] físico y posee un rango de funciones de consulta y reporte.

MODTRS

El Módulo de Tránsitos ha sido diseñado para permitir un monitoreo y control del movimiento de las mercancías dentro el territorio nacional. Estos movimientos controlados incluyen todas las formas de tránsito interno, tales como el tránsito de frontera a frontera (tránsito internacional), de frontera a una aduana interna (importación) o de una aduana interna a una frontera (exportación). Adicionalmente pueden ser incluidos los tránsitos de mercancías entre aduanas internas. Las funciones de tránsito en SIDUNEA están contenidas en el módulo **MODTRS** para las oficinas de aduana y **MODTRB** para funcionarios de aduana y otros operadores.

Además SIDUNEA posee como características adicionales la utilización de códigos internacionales y estándares desarrollados por la Organización Internacional de Estándares (ISO), por la Organización Mundial de Aduanas (OMA) y la Organización de Naciones Unidas (ONU). Cumple con los lineamientos planteados en la Convención de Kioto[7], soporta la arquitectura cliente servidor, es multiplataforma y permite su integración con varios RDBMS[8]. Proporciona intercambio electrónico de datos (EDI) entre los comerciantes y la aduana, usando las normas de EDIFACT (Intercambio Electrónico de Datos para Administración, Comercio y Transporte) y se configura de acuerdo a las características de cada régimen aduanero nacional.

Es apreciable a través de las características de este sistema que tiene la capacidad de mejorar la eficiencia de las instituciones donde se implanta, estas razones propiciaron que fuese una alternativa

[6] Operación que consiste en una, varias o todas las actuaciones siguientes: reconocimiento de las mercancías; verificación de su naturaleza y valor; establecimiento de su peso, cantidad o medida, clasificación en su nomenclatura arancelaria y determinación de los gravámenes que le sean aplicables.

[7] Convenio firmado en el año 1973, que ayuda a la armonización y cooperación entre oficinas aduaneras fronterizas.

[8] (RDBMS) Un sistema de bases de datos relacional es un programa que te permite crear, actualizar y administrar una base de datos relacional.

valorada por la Aduana General de la República de Cuba, sin embargo tras un período de prueba se constató que las funcionalidades que brinda no cumplen del todo con los requerimientos que se necesitan.

Además los términos de uso de esta aplicación establecen que el mantenimiento de la misma debe llevarse a cabo remotamente y por personal ajeno a la aduana, lo cual constituye una brecha de seguridad para este órgano.

1.3.1.2 Korea Customs Service (KCS)

El KCS es un sistema que se destaca por la gestión unificada de la infraestructura aduanera, garantizando la eficiencia de sus procesos a través de un nuevo sistema de Intercambio Electrónico de Datos. Representa una solución que aumenta la eficiencia de los procesos de despacho de aduana, favorece la reducción del tiempo y costo requerido anteriormente para los mismos y reduce costos logísticos. También como parte de un esfuerzo por facilitar aún más los procesos de despacho y reducir la carga de costos para los declarantes, KCS ha estado operando a través de un sistema web de despacho desde octubre de 2005. Este sistema cuenta con varios subsistemas tipo EDI que permiten lograr los resultados ya mencionados, entre estos se encuentran: (Guerra, 2011)

Sistema de Liquidación de Exportación

Permite a las empresas de exportación y las empresas de corretaje de aduanas presentar la declaración de exportación en la aduana y recibir el resultado de la misma electrónicamente.

Sistema de Exportación de Carga

Permite a las compañías navieras, los trazadores de líneas aéreas y agentes presentar informes de manifiesto para su salida del puerto y recibir los resultados de la presentación, la autorización y el permiso de salida por vía electrónica.

Sistema de Importación de Carga EDI

Permite a las compañías navieras, los trazadores de líneas aéreas, transportistas y almacenes generales de depósito presentar los documentos necesarios para la manipulación de la carga como la llegada, descarga y transporte en condiciones de servidumbre y recibir el resultado de dicha presentación como la aprobación y la información sobre la carga en un momento adecuado por vía electrónica.

Desarrollo del Sistema de Declaración de Importación sin papeles

Permite archivar en formato no duro copias de documentos necesarios para las declaraciones de importación a través de una red electrónica EDI, gracias a la simplificación y normalización, así como a la verificación electrónica de los requisitos de importación exigidos por otras agencias del gobierno en cuestión.

El sistema KCS se caracteriza por la rapidez y calidad con que realiza todos los trámites de aduanas, fundamentalmente en la importación y exportación de mercancías. Sin embargo análisis realizados evidencian que cerca del 95% de las declaraciones de exportación e importación se realizan mediante un tratamiento informatizado caracterizado por la no inspección física por parte de los funcionarios de aduanas. (Guerra, 2011) Esto puede traer consigo brechas de seguridad en las mercancías comercializadas; por tanto las formalidades llevadas a cabo por el KCS para la importación y exportación de mercancías no son completamente compatibles con las definidas por la Aduana General de la República de Cuba.

1.3.1.3 Sistema HM Revenue & Customs

El sistema HM Revenue & Customs (CHIEF) es usado en el Reino Unido para la gestión de procesos aduaneros, éste se conecta a 6 sistemas comerciales independientes que propician servicios a cientos de compañías aéreas, hangares de tránsito y transportistas para registrar y seguir el movimiento de las mercancías a través de puertos y aeropuertos. CHIEF controla que la información contenida en las declaraciones aduaneras coincida con lo existente en los inventarios de los diferentes sistemas comerciales, todo esto usando un sistema EDI de mensajería entre sistemas. Funciona sobre la plataforma de Servicios Fujitsu Súper Nova usando el sistema propietario de código abierto VME[9] y el gestor de base de datos IDMSX[10] el cual es jerárquico posibilitando un proceso de mejora viable en el futuro, presenta una interfaz de alto nivel entre el código de la aplicación y el entorno de la máquina. Este sistema soporta Interfaz Hombre-Computadora (HCI) y como interfaz de tráfico EDI, con enlaces que

[9] (Virtual Machine Environment) Entorno de maquina virtual.

[10] Es un gestor transaccional que agrupa tanto el acceso a los datos como el gestor de terminales y transacciones así como la seguridad. Es una base de datos en red aunque en las versiones a partir de la versión 14 integra la posibilidad de accesos en SQL.

benefician las comunicaciones de alta velocidad con Proveedores de Sistema Comunitario (DEP) y las oficinas de aduanas en todo el Reino Unido.

La arquitectura del CHIEF contiene un sofisticado sistema de seguridad y características de resistencia para evitar el sabotaje o el acceso no autorizado y garantizar la continuidad de negocio efectivo. (Unido)

El sistema CHIEF presenta disímiles características propiciando ventajas en cuanto al comercio exterior. Permite a los importadores, exportadores y transportistas completar las formalidades aduaneras necesarias con el mínimo de intervención manual, así como a los principales puertos y aeropuertos realizar las operaciones de manera eficiente y eficaz; sin embargo presenta una gran limitante, estas ventajas no son aplicables a países fuera del Reino Unido, es decir, presenta una estructura marcadamente especializada por lo cual consecuentemente no es adaptable para la AGR. El sistema está montado sobre plataforma privativa, característica que va en contra de los principios seguidos por Cuba de migrar a software libre.

1.3.2 Sistemas informáticos en Cuba

1.3.2.1 Sistema Único de Aduana (SUA)

La Aduana de Cuba presenta su propia solución informática denominada SUA. Este sistema automatiza los cinco procesos aduaneros. Tiene como objetivo automatizar el procesamiento informático referente a todas las operaciones que conforman los diferentes procesos, ya sea de medio de transporte internacional, importación y exportación con y sin carácter comercial, bultos postales y viajeros. El sistema posee una base de datos única y centralizada a la que acceden en línea todas las aduanas del país. (AGR, 2008)

Como se mencionaba anteriormente el SUA ha logrado que el proceso de despacho comercial se encuentre informatizado y cumple con la mayoría de las formalidades necesarias, sin embargo ha presentado un grupo de problemas de mantenimiento y usabilidad. Aún no ha permitido reducir el tiempo de tramitación que utiliza el declarante durante la presentación de la declaración de mercancías, ya que no se realiza de manera electrónica la recepción de los documentos complementarios asociados a ella y en ocasiones no registra el levante manual de las mercancías declaradas. Además no tiene informatizado el proceso de solicitudes de devoluciones y reintegro de derechos de aduana. Solo permite desanular una

declaración de mercancía a nivel de base de datos así como la modificación de un régimen cancelado. De ahí la necesidad que presenta el país de una solución informática que resuelva los problemas planteados.

1.3.3 Valoración crítica

Teniendo en cuenta los sistemas de gestión aduanera previamente expuestos, se puede concluir que a pesar de que brindan una amplia gama de funcionalidades ninguno cumple con las especificaciones del negocio que se necesita en la realización de los procesos de Despacho Comercial en la AGR. Tampoco son compatibles con las políticas que sigue el país de migrar a plataforma libre. Es por ello que se necesita realizar un estudio y análisis para desarrollar una solución donde se pongan de manifiesto las ventajas brindadas por estos sistemas.

1.4 Tendencias actuales y tecnologías empleadas

En la UCI el trabajo de los proyectos productivos se realiza mediante centros de desarrollo que están distribuidos por las distintas facultades que la constituyen. Cada centro tiene sus políticas, normas y estándares definidos para el desempeño de los distintos proyectos que lo componen. En el Centro CEIGE, específicamente en el Departamento de Soluciones para la Aduana se desarrolla el producto GINA que constituye la nueva solución integrada para las operaciones aduanales.

A continuación se muestran algunos de los rasgos característicos del modelo de desarrollo, lenguajes, herramientas y tecnologías utilizadas para el desarrollo de la solución.

Modelo de desarrollo de software

El Departamento de Soluciones para Aduana tiene como principal responsabilidad desarrollar sistemas informáticos para gestionar los procesos de negocio de las entidades aduaneras en su relación con el comercio exterior de nuestro país. La AGR en su condición de órgano de control del comercio exterior establece regulaciones acerca de los procedimientos necesarios para la entrada y salida de mercancías a través de las fronteras nacionales, la documentación que se debe entregar y los plazos para entregarla.

En este departamento se utiliza el Modelo de desarrollo de software como guía rectora para llevar a cabo sus proyectos, dicha definición incluye los flujos de trabajo, encuentros y reuniones básicas, los roles y sus responsabilidades, así como la interacción entre ellos. El modelado se realiza por procesos escritos

en notación BPMN (Business Process Model Notation, por sus siglas en inglés) para lograr una comprensión fácil tanto para todos los usuarios del negocio, como para los analistas implicados y los que realizan el diseño de la solución. Este modelo dirige las actividades que se realizan por rol, les da una responsabilidad, y define qué tareas deberían ser desarrolladas. Presenta un ciclo de vida para los proyectos del departamento el cual se muestra en la **Figura 1**.

Los elementos característicos de este modelo son:

- Roles y responsabilidades
- Actividades
- Descripción de proceso
- Artefactos opcionales

Para más información sobre el Modelo de desarrollo de software utilizado por el Departamento de Soluciones para la Aduana, dirigirse al documento que lleva por título: “Modelo de Desarrollo de Software”.

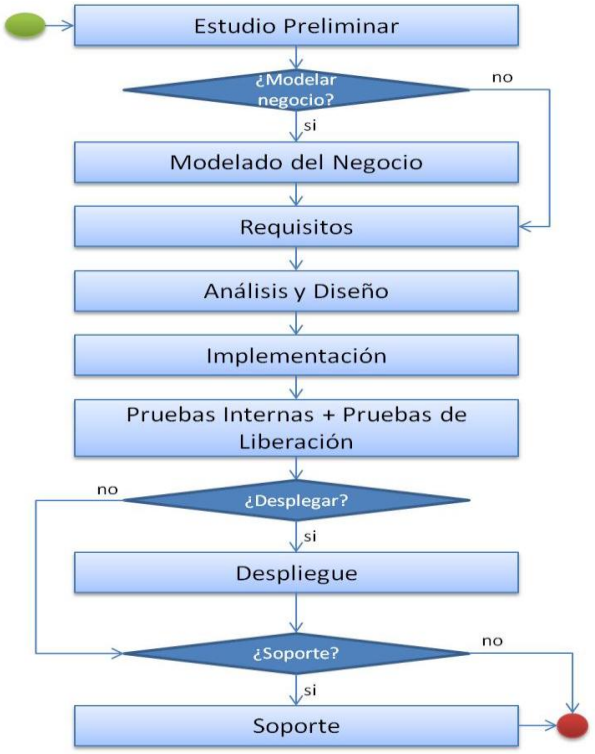

Figura 1: Ciclo de vida para los proyectos del Departamento de Soluciones para Aduana. (CEIGE, 2012)

Metodologías de diseño

Ante un nuevo proyecto se pueden adoptar distintas formas de aproximación denominadas metodologías de diseño. Estas se pueden emplear de forma unitaria o combinándolas entre sí para obtener un nuevo enfoque. Se debe tener en cuenta que también influye el ámbito de aplicación y las disciplinas implicadas en el proceso del proyecto.

A continuación se muestra un listado de las principales metodologías de diseño y sus rasgos característicos:

Diseño Orientado al Uso: se centra en las tareas asociadas al uso y sus objetivos. La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario en condiciones específicas de uso. (Reyero, Eusebio, 2009)

Diseño Centrado en el Usuario: (User Center Design, por sus siglas en inglés) es la metodología de diseño más habitual en el mundo. Este enfoque coloca todas las necesidades, deseos y limitaciones del usuario como núcleo del proceso de diseño. (Reyero, Eusebio, 2009)

Enfoque Ascendente: plantea que la computarización se implanta desde el nivel más bajo. El diseño ascendente se refiere a la identificación de aquellos procesos que necesitan computarizarse conforme vayan apareciendo, su análisis como sistema y su codificación, o bien, la adquisición de paquetes de software para satisfacer el problema inmediato. (Reyero, Eusebio, 2009)

Desarrollo Modular: se trata de un enfoque de concepción modular, se basa en una descomposición de la programación en fracciones lógicas y manejables. Enfatiza en las interfaces entre los módulos, no las ignoradas hasta el final del desarrollo del sistema. Cada módulo debe ser funcionalmente cohesivo de manera que satisfaga solo una función. (Reyero, Eusebio, 2009)

Los ejemplos de metodologías de diseño presentadas satisfacen el trabajo que se desarrolla en el GINA, ya que está basado en las necesidades del cliente y para ello se trabaja en busca de la satisfacción completa, tratando no solo que el software cumpla con los requisitos necesarios sino también que sea agradable y fácil de utilizar. Se realiza una planificación desde una visión global para la definición de la estructura del sistema que incluye subsistemas, módulos y la comunicación entre ellos, comenzando luego el desarrollo desde la raíz hasta la culminación del software completo.

1.4.1 Patrones arquitectónicos

Los patrones de arquitectura están relacionados fundamentalmente con la estructura de un sistema de software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes. Describen un problema particular y recurrente del diseño que surge en un contexto específico y presenta un esquema genérico y probado de su solución.

A continuación se mencionan las principales características del patrón arquitectónico utilizado para el desarrollo de la solución:

Patrón Modelo Vista Controlador (MVC)

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres capas o componentes distintos. El patrón MVC se utiliza frecuentemente en aplicaciones web donde la vista es la página HTML (del inglés, Hypertext Markup Language) que se visualiza en el navegador y el código que proporciona de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio, el controlador es el responsable de recibir los eventos de entrada desde la vista, enviarlos al modelo y manejar también las respuestas del modelo y transmitirlas hacia la vista.

- **El modelo** es un conjunto de clases que representan la información del mundo real que el sistema debe procesar. Desconoce la existencia de las vistas y del controlador. Ese enfoque puede ser interesante, pero en la práctica no es aplicable pues deben existir interfaces que permitan a los módulos comunicarse entre sí. Esta es la recepción específica del dominio de la información sobre la cual funciona la aplicación. El modelo es una forma de llamar a la capa de dominio. La lógica de dominio añade significados a los datos.
- **Las vistas** son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo. Obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada vez que el modelo del dominio cambia por medio de notificaciones generadas por el modelo de la aplicación.

- **El controlador** es el componente que se encarga de dirigir el flujo del control de la aplicación mediante mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador. (Bascón Pantoja, 2004)

En la siguiente figura se muestra la implementación del patrón MVC.

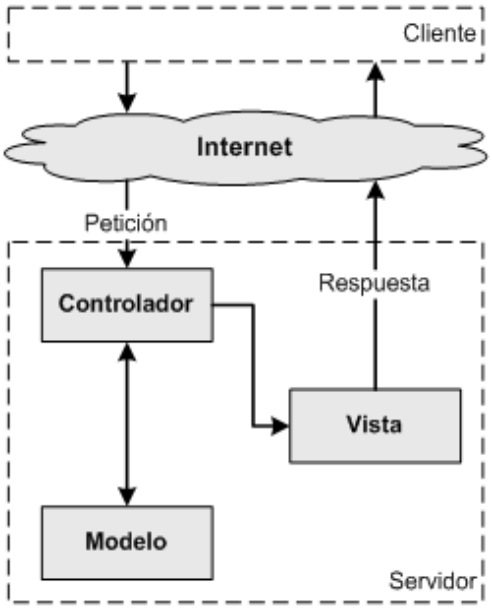

Figura 2: Patrón MVC. (Fabien Potencier, 2008)

1.4.2 Tecnologías utilizadas por capas

Siguiendo la arquitectura base definida por el Departamento de Soluciones para la Aduana y teniendo en cuanta las características del patrón arquitectónico MVC. En cada una de las capas definidas por este se utilizaron un grupo de tecnologías que serán abordadas a continuación.

1.4.2.1 Tecnologías empleadas en la Capa Vista

ExtJS

ExtJS es un framework de JavaScript que permite realizar aplicaciones web enriquecidas basándose en tecnología AJAX (del inglés, Asynchronous JavaScript And XML), JSON (del inglés, JavaScript Object

Notation), DHTML[11]y DOM[12]. ExtJS 3.0 está patentado bajo licencia LGPL[13] lo que posibilita su uso para aplicaciones empresariales privativas de código cerrado.

ExtJS brinda la posibilidad de utilizar un gran número de componentes visuales que mejoran considerablemente la calidad de las aplicaciones. Posibilita validaciones de formularios de todo tipo, basándose en expresiones regulares y tipos de datos. Trae implícitos componentes como vista en árboles, arrastrado y soltado, cambio de tamaño de imágenes, rejillas, paginado, agrupado de objetos, tabs[14], asistentes, entre otros muchos.

La utilización de un framework de JavaScript como ExtJS facilita la separación de la capa de la vista de la del controlador desde el punto de vista productivo, ya que el código utilizado en la primera es solamente JavaScript y no es necesario utilizar ningún tipo de código PHP, los desarrolladores pueden centrarse más en el aprendizaje de un solo lenguaje. Además al soportar serialización[15] de objetos mediante tecnología JSON permite que los datos enviados desde el controlador como respuesta a la vista contengan solo las propiedades de dichos objetos, pero no el comportamiento, minimizando los posibles errores de programación y los accidentes relacionados con la modificación de los objetos desde la vista. (Cobo Rodríguez, 2008)

El framework ExtJS es utilizado en el GINA para la realización de las pantallas de interfaz de usuario. Permite la comunicación con otras capas del proyecto de manera sencilla. Brinda facilidades y dinamismos en las vistas.

[11] (Dynamic HTML) Designa el conjunto de técnicas que permiten crear sitios web interactivos utilizando una combinación de lenguaje HTML estático, un lenguaje interpretado en el lado del cliente (como JavaScript), el lenguaje de hojas de estilo en cascada (CSS) y la jerarquía de objetos de un DOM.

[12] (Document Object Model) Modelo de objetos para la representación de documentos. Es un modelo computacional a través del cual los programas y scripts pueden acceder y modificar dinámicamente el contenido, estructura y estilo de los documentos HTML y XML.

[13] (Lesser General Public License) Licencia Pública General Reducida.

[14] En las interfaces de las aplicaciones se refiere a las pestañas que se utilizan para aumentar la navegabilidad.

[15] Consiste en un proceso de codificación de un objeto en un medio de almacenamiento (como puede ser un archivo o un buffer de memoria) con el fin de transmitirlo a través de una conexión en red como una serie de bytes o en un formato humanamente más legible como XML o JSON.

XHTML

El Lenguaje de Marcado de Hipertexto Extensible (XHTML) es una versión más estricta y limpia del Lenguaje de Marcado de Hipertexto (HTML), que nace precisamente con el objetivo de remplazar al HTML ante su limitación de uso con las diferentes herramientas basadas en XML (del inglés, eXtensible Markup Language). XHTML extiende del HTML 4.0 combinando la sintaxis de HTML diseñado para mostrar datos con la de XML diseñado para describir los datos.

El lenguaje XHTML es muy similar al lenguaje HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje XML. Técnicamente, HTML es descendiente directo del Estándar de Lenguaje de Marcado Generalizado (SGML), mientras que XHTML lo es del XML (que a su vez, también es descendiente de SGML). (Eguíluz Pérez, 2008)

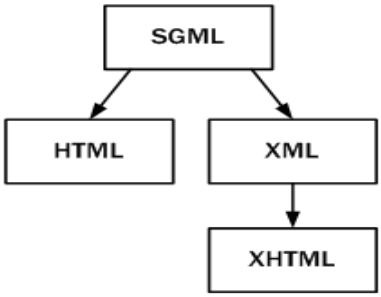

Figura 3: Esquema de la evolución de HTML y XHTML. (Eguíluz Pérez, 2008)

En el sistema GINA es utilizada la versión 1.0 del estándar XHTML. Toda la interfaz de usuario desarrollada en el sistema tiene como base este lenguaje.

CSS

Hojas de estilos en cascada (del inglés, Cascading Style Sheets), es un lenguaje creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados “documentos semánticos”). Además mejora la accesibilidad del documento, reduce la complejidad de su

mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (Eguíluz Pérez, 2009)

A través del CSS se logra una apariencia agradable a las vistas diseñadas e implementadas para el GINA. Describe cómo se va a mostrar un documento en la pantalla o cómo se va a imprimir, incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente JavaScript es un lenguaje de programación interpretado por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (Eguíluz Pérez, 2009)

Entre sus principales características se pueden citar:

- Es interpretado por el cliente.
- Está basado en objetos.
- Su código se integra en las páginas XHTML, incluido en las propias páginas.
- Las referencias a objetos se comprueban en tiempo de ejecución, por lo tanto no se compila.
- Es independiente de la plataforma, por lo que se ejecuta en cualquier sistema operativo. (Universidad Carlos III de Madrid., 2006)

En el sistema GINA se utiliza principalmente para manejar objetos dentro de las páginas web. Dichos objetos facilitan la programación de páginas interactivas, a la vez que se evita la posibilidad de ejecutar comandos que puedan ser peligrosos para la máquina del usuario, tales como formateo de unidades y modificación de archivos.

1.4.2.2 Tecnología empleada en la Capa Modelo

Propel

Propel es un framework programado en PHP5 de persistencia de datos y consulta, lo que significa que brinda un mecanismo para almacenar objetos PHP en una base de datos y un sistema para búsqueda y restauración de los mismos desde una base de datos a partir de un mecanismo de abstracción de los datos y utilizando ORM[16]. Propel permite realizar consultas complejas y manipulación de datos sin escribir una sola cláusula SQL (del inglés, Structured Query Language). Hace más fácil la escritura de aplicaciones, el despliegue y migración de la aplicación a otro Gestor de Base de Datos si alguna vez la situación lo amerita. La versión utilizada en el proyecto es la 1.3 y corre sobre PHP 5.3. Propel puede ser descrito como un mapeado objeto-relacional, una capa DAO [17] o una capa objeto persistente. Su implementación está basada principalmente en los patrones *Row Data Gateway* y *Table Data Gareway*. Está dividido en dos componentes principales: (Cobo Rodríguez, 2008)

1. Un motor generador para construir sus clases y archivos SQL (generador-propel).

2. Un ambiente de ejecución que proporciona herramientas para construir consultas SQL, ejecutando consultas compiladas y herramientas para el manejo de conexiones para múltiples bases de datos simultáneamente.

1.4.2.3 Tecnología empleada en la Capa Controladora

PHP

PHP (del inglés, Hypertext Preprocessor) es un lenguaje del lado del servidor (esto significa que PHP funciona en un servidor remoto que procesa la página web antes de que sea abierta por el navegador del usuario) especialmente creado para el desarrollo de páginas web dinámicas. Puede ser incluido con facilidad dentro del código HTML. (Vazquez, 2003)

[16] (Object Relational Mapping) Mapeo de Objetos Relacional es una técnica de programación para convertir tipos de datos incompatibles entre sistemas de bases de datos y lenguajes orientados a objetos.

[17] (Data Access Objects) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una base de datos o un archivo.

Una de sus características más potentes es su soporte para gran cantidad de bases de datos, pueden mencionarse InterBase, MySQL, Oracle, Informix, PostgreSQL, entre otras. PHP no necesita que el navegador lo soporte, es independiente de este, pero sin embargo para que sus páginas funcionen el servidor donde están alojadas debe soportar este lenguaje.

1.4.3 Framework

Un framework es un conjunto de clases o estructuras que implementan los componentes de una aplicación genérica, componentes concretos que cumplen a cabalidad tareas concretas. Para desarrollar programas completos se buscan e instancian los componentes apropiados.

Realmente no existe una definición oficial de framework pero todos los autores coinciden en la utilización de un tema común: la reutilización. Una definición dada por R. E. Johnson, and B. Foote en 1988 en su publicación *"Designing Reusable Classes"* plantea que: (Foote, 1988)

"Un framework es un conjunto de clases que personifican un diseño abstracto para soluciones de una familia de problemas relacionados (...)"

1.4.3.1 Framework Symfony

Symfony es un framework para PHP5 patentado bajo licencia MTI[18], es compatible con la mayoría de gestores de bases de datos MySQL, PostgreSQL, Oracle y SQL Server de Microsoft, entre otros en dependencia del tipo de abstracción de la base de datos que se utilice.

Symfony está diseñado para optimizar gracias a sus características el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además automatiza las tareas más comunes permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web. (Zaninotto, 2008)

[18] Licencia de Software Libre originaria del Instituto de Tecnología de Massachusetts.

Características de Symfony:

- Symfony incluye un verdadero ORM, Propel, que es otro proyecto de código abierto y probablemente una de las mejores soluciones ORM para PHP.
- Código fácil de leer que incluye comentarios de phpDocumentor[19] y que permite un mantenimiento muy sencillo.
- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de los casos pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar" en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicación empresarial y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

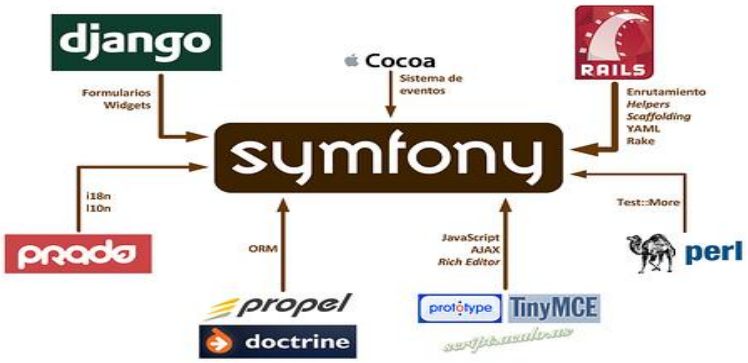

Figura 4: Composición de la estructura de Symfony por otros frameworks. (Eguiluz, 2008)

[19] Es un estándar formal para comentar código PHP.

Symfony en su versión 1.2.8 es el framework utilizado para la creación del GINA. Brinda facilidades para el desarrollo con el lenguaje PHP. Utiliza metodologías y patrones de diseño que brindan la posibilidad de reutilizar el código. Presenta buena seguridad, es multiplataforma y configurable. Todo el soporte del sistema GINA está sobre la base de Symfony. Incluye un trabajo limpio sobre los proyectos a realizar. Permite por medio de Propel el mapeo de las base de datos, transformando las tablas en clases del negocio. Crea formularios seguros para la entrada de información, soporta internacionalización[20] e incluye algunos de los frameworks más usados en el mundo.

1.4.4 Entornos de Desarrollo Integrado (IDE)

Un IDE en un programa informático compuesto por un conjunto de herramientas, utilizadas por los programadores para desarrollar código. Consisten en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Pueden dedicarse en exclusiva a un sólo lenguaje de programación o bien para varios.

NetBeans IDE 6.9

El IDE NetBeans es un entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones web, empresariales, de escritorio y aplicaciones móviles utilizando la plataforma Java, JavaFX, PHP, JavaScript y Ajax, Ruby y RubyonRails, Groovy y Grails y C / C + +. ()

Algunas de las características que presenta integrado a PHP son:

- Creación de proyectos PHP.
- Integración con Symfony y ZenFramework.
- Editor de código fuente.
- Integración con PHPUnitTesting.
- Depuración de PHP.

[20] Es el proceso de diseñar software de manera tal que pueda adaptarse a diferentes idiomas y regiones sin la necesidad de realizar cambios de ingeniería ni en el código.

- Integración con MySQL, PostgreSQL y Oracle.
- Integración con Sistemas de Control de Versiones.

1.4.5 Herramienta CASE para el modelado

Las herramientas CASE (del inglés, Computer Aided Software Engineering) son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software. (Alfaro)

Hoy en día muchas empresas se han extendido a la adquisición de herramientas CASE, con el fin de modelar los aspectos claves de todo el proceso de desarrollo de un sistema, desde el principio hasta el final. Una de estas herramientas es Visual Paradigm.

Visual Paradigm: es una herramienta UML (del inglés, Unified Modeling Language) profesional que soporta el ciclo de vida completo del desarrollo de software, análisis y diseño orientados a objetos, construcción, pruebas y despliegue. La misma facilita una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Algunas características de Visual Paradigm:

- Entorno de creación de diagramas para UML 2.1.
- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs. (A. Vizcaino, y otros, 2008)

1.5 Gestores de Bases de Datos

Un sistema gestor de bases de datos o SGBD (aunque se suele utilizar más a menudo las siglas DBMS procedentes del inglés, Data Base Management System) es el software que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos.

En estos sistemas se proporciona un conjunto coordinado de programas, procedimientos y lenguajes que permiten a los distintos usuarios realizar sus tareas habituales con los datos garantizando además la seguridad de los mismos. (Asenjo, 2009)

Por tanto debe permitir:

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: realizar consultas, actualizarla y generar informes.

En las soluciones implementadas para la AGR el gestor de base de datos que se utiliza es Oracle 11g: sistema de gestión de base de datos relacional desarrollado por Oracle Corporation.

El SGBD **Oracle**, utiliza la arquitectura cliente/servidor. Ha incorporado en su sistema el modelo objeto-relacional, pero al mismo tiempo garantiza la compatibilidad con el tradicional modelo relacional de datos. Ofrece un servidor de bases de datos híbrido. Es uno de los más conocidos y ha alcanzado un buen nivel de madurez y de profesionalidad. Se destaca por su soporte de transacciones, estabilidad y escalabilidad. Los tipos objetos de Oracle son tipos de datos definidos por el usuario que permiten modelar entidades complejas del mundo real en una estructura que trata cada entidad como una unidad atómica simple en la base de datos. A partir de la versión 10 del año 2004, se añade a los servidores la capacidad de funcionar según el paradigma de “Grid” (o rejilla) y se ofrecen mejoras en la administración e integración de algunos elementos que previamente no funcionaban correctamente juntos. (Costilla, 2002)

Ventajas de Oracle:

- Las entidades complejas del mundo real y la lógica se pueden modelar fácilmente, lo que permite reutilizar objetos para el desarrollo de base de datos de una forma más rápida y con mayor eficiencia.
- Los programadores de aplicaciones pueden acceder directamente a tipos de objetos Oracle sin necesidad de ninguna capa adicional entre la base de datos y la capa cliente.
- Las aplicaciones que utilizan objetos de Oracle son fáciles de entender y mantener porque soportan las características del paradigma orientado a objetos.
- Tiene buen rendimiento y hace buen uso de los recursos.

- Posee un rico diccionario de datos.
- Brinda soporte a la mayoría de los lenguajes de programación.
- Es un sistema multiplataforma, disponible en Windows, Linux y Unix.
- Permite tener copias de la base de datos productiva en lugares lejanos a la ubicación principal. Las copias de la base de datos productiva pueden estar en modo de lectura solamente.

Por las características antes expuestas se decidió utilizar para la aplicación el SGBD Oracle además que está definido en la línea base de la arquitectura del departamento de Soluciones para la Aduana por el que este trabajo está integrado y regido.

1.6 Conclusiones parciales

En el capítulo se abordaron todas las herramientas que están definidas para el desarrollo del módulo, estas se caracterizan por las facilidades, ventajas y optimización que le brindan a los desarrolladores, las mejores prácticas que existen mundialmente para la creación de sitios y aplicaciones web. También se evidenció la necesidad de realizar un módulo para la AGR que informatice todos los procesos que tienen lugar en el Despacho Comercial, cumpliendo con las funciones del actual sistema en explotación y supliendo las carencias del mismo.

Capítulo 2. Diseño e implementación

2.1 Introducción

En el presente capítulo se incluyen temas relacionados con el desarrollo de los flujos de trabajo diseño e implementación. Se obtendrán una serie de artefactos[21] de diseño que contienen la solución detallada sobre las clases a utilizar, la comunicación del módulo con otros subsistemas, la composición que presenta la base de datos y los pasos a seguir para poder realizar la elaboración de los diferentes requisitos. El objetivo de todos estos resultados es definir la vía correcta y el orden en que se van a realizar las actividades facilitando la comprensión y rapidez a los programadores logrando un mejor desarrollo de la solución.

2.2 Descripción de la solución

La Universidad de las Ciencias Informáticas (UCI), en específico el Centro de Informatización de Gestión de Entidades (CEIGE), en conjunto con los especialistas de la AGR, están interesados en el desarrollo de un módulo que gestione las formalidades del Despacho Comercial dentro del sistema GINA en todas las aduanas del país. Como parte de estas formalidades se encuentran el almacenamiento y validación de los datos de las Declaraciones de Mercancías (DM), la información será introducida por medio de un sistema de Ventanilla Única facilitando la transferencia ininterrumpida de información comercial. Los agentes comerciales podrán presentar por única vez, de forma electrónica y centralizada la información solicitada (en este caso los documentos complementarios) a una única autoridad designada preferentemente la Aduana. La solución que se propone desarrollar registrará el levante de las mercancías declaradas, posibilitará desanular una declaración de mercancías e incluirá el proceso de devoluciones y reintegros, permitiendo modificar un régimen cancelado además de un grupo de requisitos que se presentan en el **epígrafe 2.3.1** de esta investigación, debe integrarse con algunos de los módulos pertenecientes al GINA, ofrecerá servicios a los mismos y separará la lógica del negocio del gestor de base de datos, garantizando

[21] Un artefacto puede ser un modelo o un elemento del modelo, un documento o todo lo que puede ser generado en un proceso.

su mantenimiento. Además estará provista de una interfaz gráfica orientada al uso, brindando al usuario final una experiencia agradable.

2.3 Modelo del diseño

"Diseño es el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema, con los suficientes detalles como para permitir su realización física". (E.S.Taylor, 1959)

Es el proceso que determina las características principales del sistema final, establece los límites en rendimiento y calidad que la mejor implementación puede alcanzar y determina qué costos se alcanzarán.

El diseño es una actividad que comienza cuando el analista de sistemas ha producido un conjunto de requisitos funcionales lógicos para el mismo y finaliza cuando el diseñador ha especificado los componentes del sistema y las relaciones entre ellos.

Es aquí donde se toman decisiones que afectarán finalmente al éxito de la implementación del programa y con igual importancia la facilidad de mantenimiento que tendrá el software. Estas decisiones se llevan a cabo durante el diseño del software haciendo que sea un paso fundamental de la fase de desarrollo.

La importancia del diseño del software se puede definir con una única palabra: calidad. El diseño es el proceso en el que se asienta la calidad del desarrollo del software. Produce las representaciones del software en las que puede evaluarse su calidad.

El diseño sirve como base para todas las posteriores etapas del desarrollo y de la fase de mantenimiento. Sin diseño se corre el riesgo de construir un sistema inestable, que falle cuando se realicen pequeños cambios, que pueda ser difícil de probar, cuya calidad no pueda ser evaluada hasta más adelante en el proceso de ingeniería de software cuando quede poco tiempo y se haya gastado ya mucho dinero. (Torossi)

2.3.1 Requisitos

En el glosario de Terminología de Ingeniería de Software de la IEEE se define requisito como:

"Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo."

"Una condición o capacidad que debe estar presente en un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formal." (ieee.org, 1990)

Los requisitos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer. (Sommerville, 2005)

Son capacidades o condiciones que el sistema debe cumplir, describen con detalle la función del mismo, sus entradas y salidas, excepciones, entre otros. Deben estar bien determinados y ser convenientemente comprendidos, tanto por los implicados para con el sistema como por los desarrolladores del mismo para que exista un entendimiento común.

A continuación se presentan los requisitos funcionales pertenecientes al módulo Despacho Comercial del sistema GINA:

- Aprobar/Rechazar devolución de derechos de aduana.
- Cancelar temporalidad de oficio.
- Aprobar/Rechazar reintegro de derechos de aduana.
- Presentar documentación correspondiente a la solicitud de reintegro de derechos de aduana.
- Presentar documentación correspondiente a la solicitud de devoluciones de derechos de aduana.
- Ajustar temporalidad.
- Modificar temporalidad pagada.
- Registrar levante manual.
- Pagar facilidad anticipada.
- Desanular Declaración de Mercancías.
- Confirmar Exportación.

- Registrar Factura Comercial Provisional.
- Registrar Factura Empaque.
- Registrar Lista Empaque.
- Registrar Factura de Venta.
- Registrar Declaración Jurada.
- Registrar Permiso.
- Modificar Factura Comercial Provisional.
- Modificar Factura Empaque.
- Modificar Lista Empaque.
- Modificar Factura de Venta.
- Modificar Declaración Jurada.
- Modificar Permiso.
- Anular Documento Complementario.

2.3.2 Diagrama de paquetes

El diagrama de paquetes es utilizado para mostrar las agrupaciones lógicas en las que se encuentra dividido un sistema y las dependencias entre ellas. Suministran una descomposición de la jerarquía lógica de un sistema. Los paquetes contienen elementos del modelo al más alto nivel, tales como clases y sus relaciones, máquinas de estado, diagramas de casos de uso, interacciones, y colaboraciones: cualquier elemento que no esté contenido en otro. Los elementos como atributos, operaciones, estados, líneas de vida y mensajes están contenidos en otros elementos y no aparecen como contenido directo de los paquetes. (BOOCH, y otros, 2000)

El diagrama de paquetes que se representa en la **Figura 5**, contiene la interacción del módulo Gestión de Documentos Complementarios (GDC) perteneciente al Despacho Comercial con los diferentes subsistemas que presenta el GINA, así como la interacción con los paquetes y librerías de sí mismo.

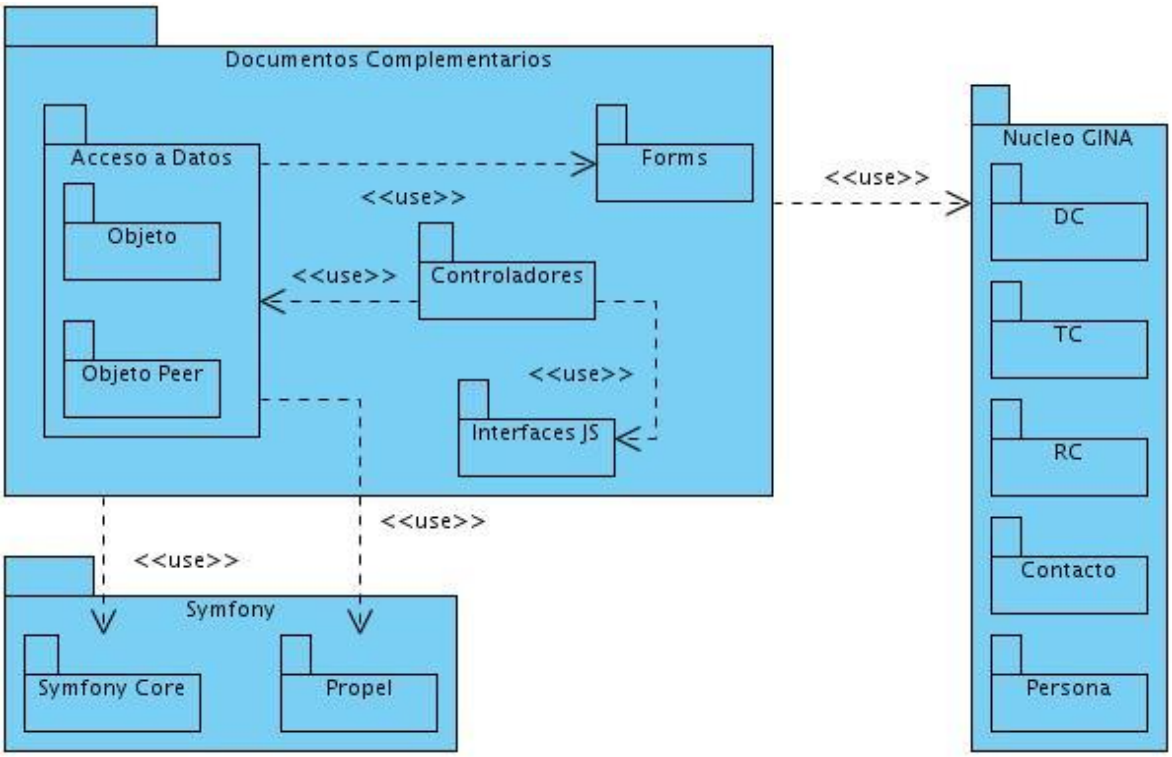

Figura 5: Diagrama de paquetes.

Dentro de los subsistemas que se relacionan con el módulo GDC se encuentra Tablas de Control (TC) que es el encargado de administrar las clases que son nomencladoras y comunes para todos los subsistemas del GINA. También se halla Registro Central (RC) brindando servicios e información referente a las entidades que interactúan con la AGR y sus trabajadores. El subsistema Persona brinda todo tipo de información referente al personal de la AGR y el historial de incidencias que presenten. Además de Contacto subsistema que gestiona los contactos (domicilio, teléfono, email) asociados a las entidades y las personas, por último DC encargado de gestionar las declaraciones de mercancías en las operaciones de importación y exportación, así como el paquete Symfony que agrupa las principales clases que garantizan el funcionamiento del framework. Dentro de los paquetes de mayor importancia se encuentra, el Interfaces JS, encargado de la capa de presentación o vista, en el se encuentran todas las

interfaces de usuarios desarrolladas, el Controladores que pertenece a la capa controladora, presenta las clases y objetos para el control de las peticiones y el flujo de acciones a realizar y el Acceso a Datos que es la capa del modelo, donde se encuentran las clases del negocio, los formularios y algunas de las encargadas del mapeo y abstracción de la base de datos.

2.3.3 Diagrama de clases del sistema

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema. Las clases están compuestas por los atributos pasivos y los activos mientras que las relaciones pueden ser de diferentes tipos como herencia, composición, agregación, asociación y uso.

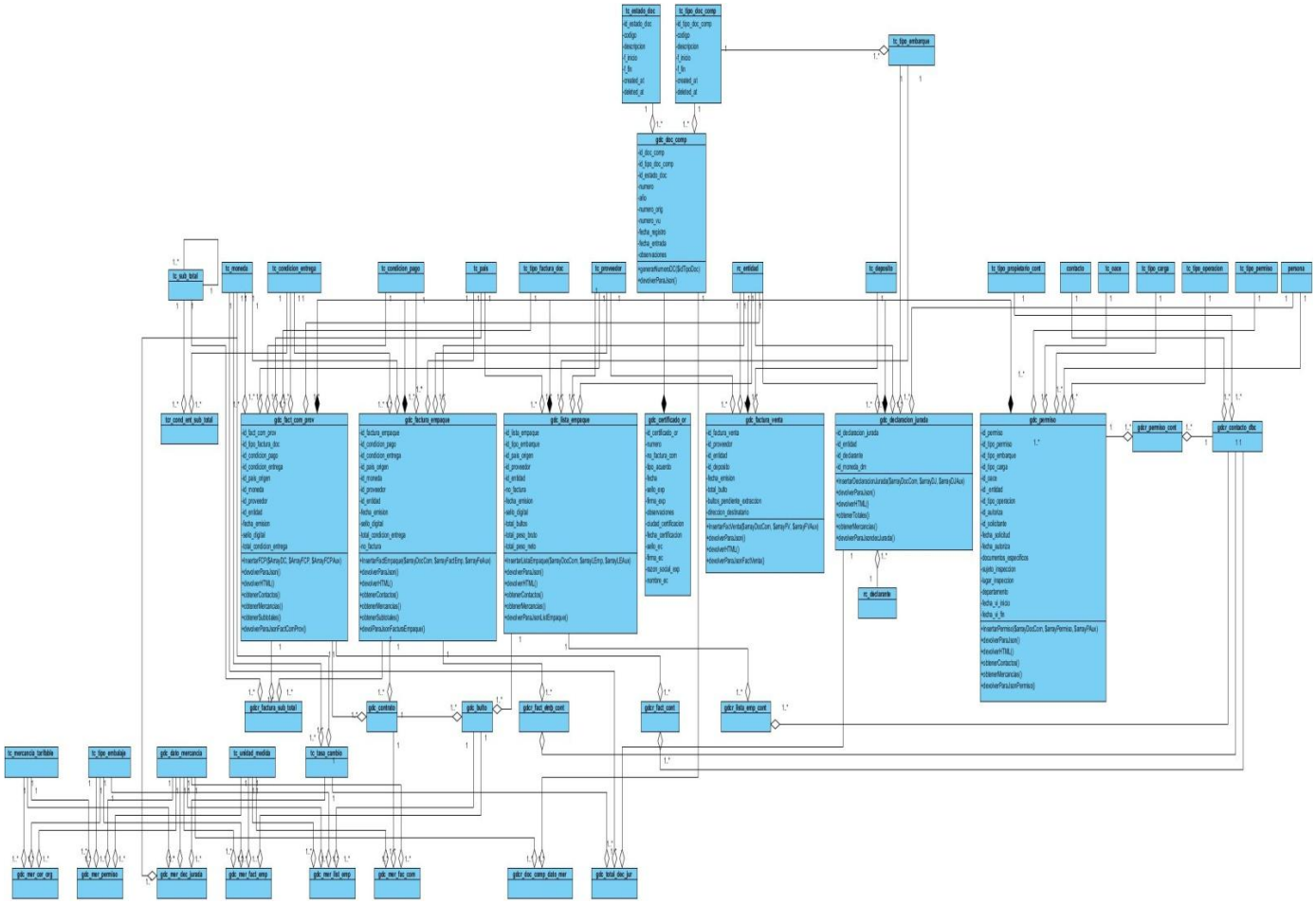

Figura 6: Diagrama de clases del sistema de la capa de modelo.

El diagrama de clases mostrado anteriormente **Figura 6**, es el usado en el módulo Documentos Complementarios del Despacho Comercial y forma parte de la capa de Modelo. Presenta un total de 50 clases, de ellas 24 son nomencladores de las cuales 20 pertenecen al subsistema TC y 2 al subsistema RC y las 2 restantes a los subsistemas Contacto y Persona, donde se obtiene información a través de servicios que se brindan. De las clases más significativas se encuentran la GdcDocComp que es contenedora de todas las funcionalidades y atributos comunes de GdcFactComProv, GdcFacturaEmpaque, GdcListaEmpaque, GdcCertificadoOr, GdcFacturaVenta, GdcDeclaracionJurada y GdcPermiso y estas a su vez contienen sus especificaciones y en su conjunto las 8 contienen toda la implementación principal que se necesita para la realización de los Documentos Complementarios, la relación que existe entre estas clases debido a la utilización del ORM Propel es la simulación de herencia a través de composición.

2.3.4 Diagrama de Secuencia Orientado a Actividades

Estos diagramas son empleados dentro de las nuevas regulaciones realizadas al Modelo de Desarrollo de Software empleado por el Departamento de Soluciones para la Aduana. Están compuestos por calles que representan las clases y la comunicación entre ellas se ve reflejada en las llamadas a funcionalidades necesarias para la realización de la actividad en cuestión. Cada actividad propia del diseño que se realiza es graficada con todos los pasos que necesita para culminar la tarea. Este diagrama trata de evitar el engorroso trabajo que realizan los diseñadores a la hora de diagramar una solución y que resulte rápido y eficaz para el programador.

Presenta dos soluciones gráficas: diagrama visual y diagrama del negocio, ambos enfocados en lograr un mejor entendimiento de las bifurcaciones que en ocasiones son imprescindibles en un diseño de aplicaciones web.

2.3.4.1 Diagrama de Secuencia Orientado a Actividades de Componentes Visuales

Este tipo de diagrama es más corto en su ejecución y debe ser más rápido por la necesidad de responder a los diagramas del negocio. Trabaja fundamentalmente en la capa visual de Symfony.

Este diagrama presenta las mismas políticas del orientado a negocio. Está dirigido principalmente a los programadores de las interfaces de usuario para que tengan una base por donde guiarse a la hora de

brindar las funcionalidades y la forma de comportamiento de las pantallas. Se evidencian en ellos una serie de comentarios y acciones que incluyen nombre de las funcionalidades a llamar, así como de los valores a enviar y mensajes que mostrar. En la **Figura 7**, se presenta un ejemplo de la interfaz Mostrar solicitud de devoluciones de derechos de aduana aprobadas.

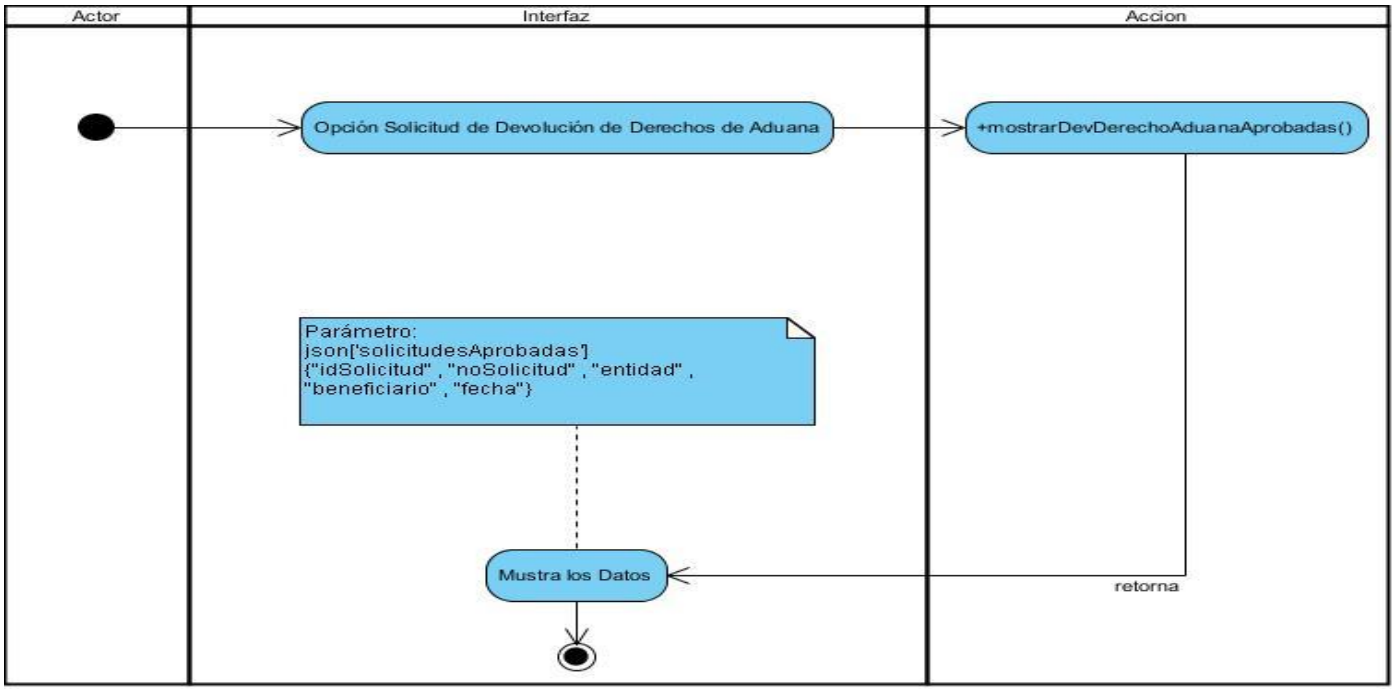

Figura 7: Diagrama de Secuencia Orientado a Actividades del Componente Visual correspondiente al requisito Aprobar/Rechazar solicitud de devolución de derechos de aduana.

2.3.4.2 Diagrama de Secuencia Orientado a Actividades del Negocio

Este tipo de diagrama es la respuesta del diseño a los requisitos funcionales capturados en fases anteriores y evidencia paso a paso la interacción entre las tres capas de la arquitectura y los pasos que debe seguir el programador para concluir satisfactoriamente la aplicación. Tiene un nivel superior de interacción con los datos del sistema.

En la **Figura 8** se presenta un ejemplo del diagrama de secuencia orientado a actividades del negocio del requisito funcional Registrar Declaración Jurada. Presenta un total de tres calles, las cuales constituyen las clases participantes, incluye también actividades que identifican llamadas a funcionalidades y pasos a

seguir para la implementación, así como bifurcaciones y ciclos. El objetivo de este diagrama es lograr realizar todas las validaciones del negocio exigidas para realizar el despacho de este documento complementario y luego la inclusión de los datos en la base de datos.

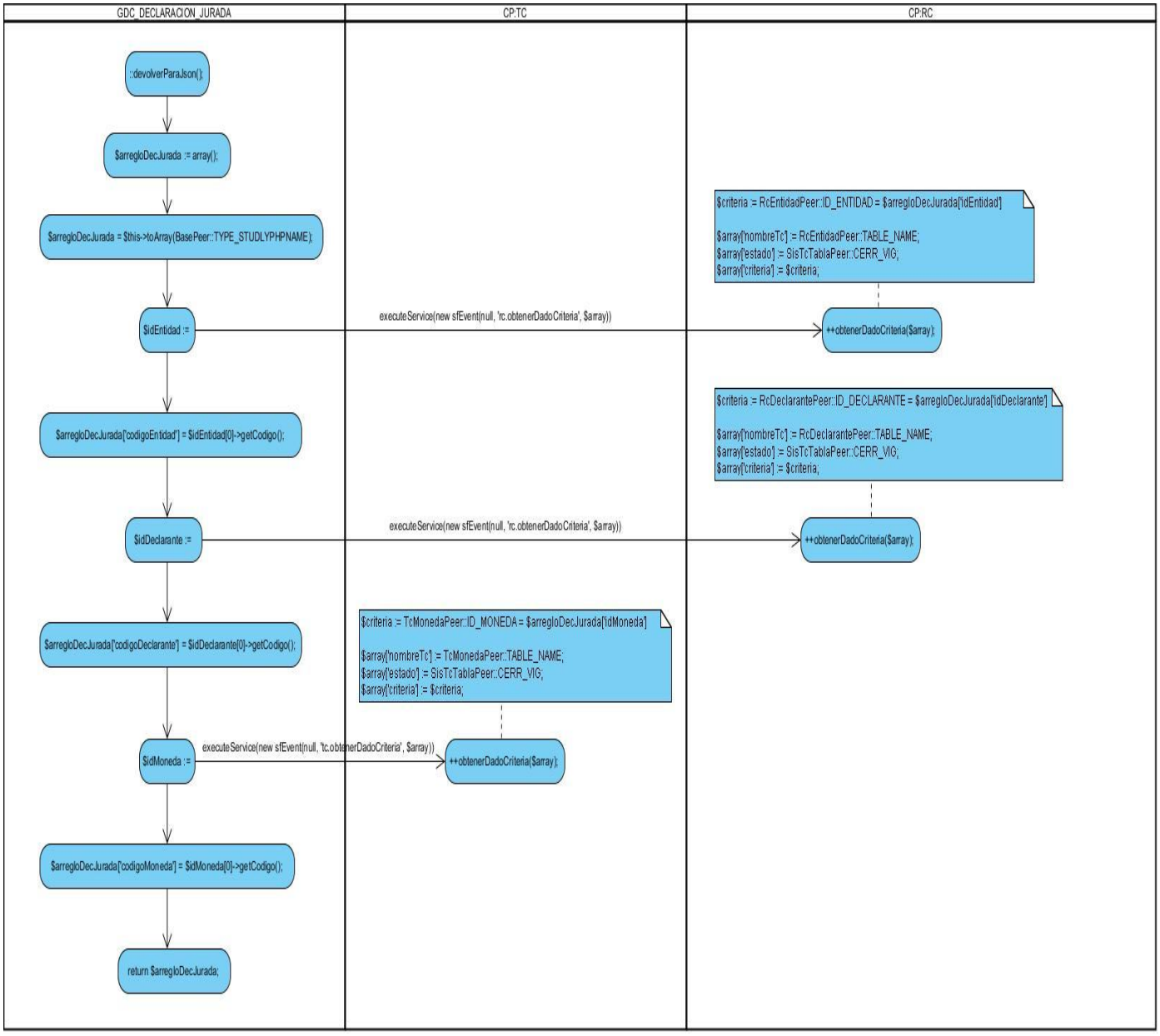

Figura 8: Diagrama de Secuencia Orientado a Actividades del Negocio correspondiente al requisito Registrar Declaración Jurada.

En la **Figura 9** se muestra de manera más específica un flujo de actividades a seguir y la relación que existe entre estas, incluyendo algunas llamadas a funcionalidades, lanzamiento de errores y condicionales.

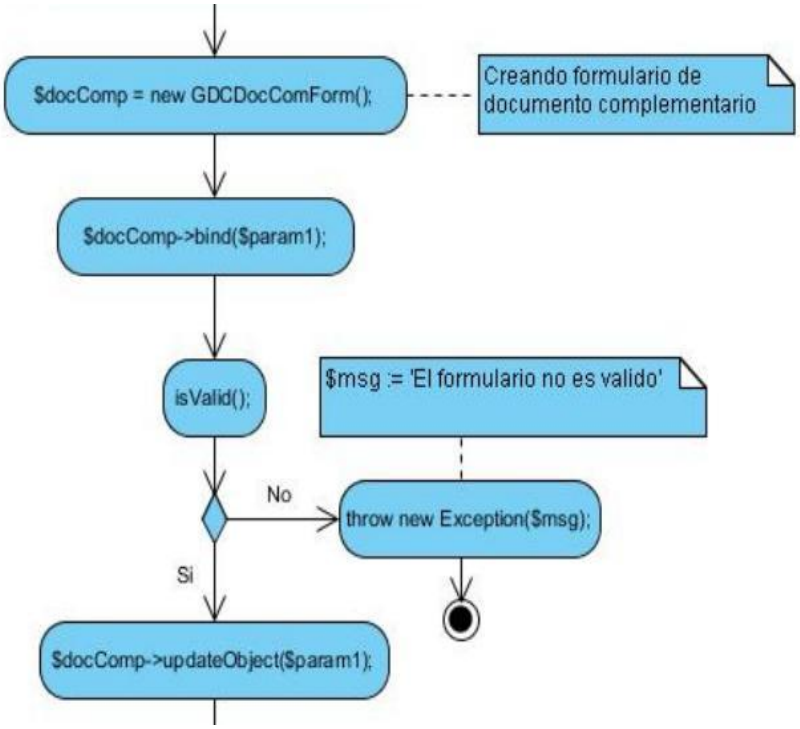

Figura 9: Flujo de actividades.

2.3.5 Diseño de la Base de Datos

Cómo se puede decidir qué relaciones debe tener una base de datos determinada o qué atributos deben presentar las relaciones, qué claves primarias y qué claves foráneas se deben declarar. La tarea de tomar este conjunto de decisiones recibe el nombre de diseñar la base de datos.

Una base de datos sirve para almacenar la información que se utiliza en un sistema determinado. Las necesidades y los requisitos de los futuros usuarios del sistema de información se deben tener en cuenta para poder tomar adecuadamente las decisiones anteriores.

En resumen, el diseño de una base de datos consiste en definir la estructura de los datos que debe tener la misma en un sistema de información determinado. En el caso relacional, esta estructura será un

conjunto de esquemas de relación con sus atributos, dominios de atributos, claves primarias, claves foráneas, etc. (Costa)

2.3.5.1 Modelo relacional

El modelo relacional **Figura 10**, generado para el sistema es el usado en el módulo Documentos Complementarios del Despacho Comercial contiene las tablas y las relaciones entre ellas que a su vez guardarán la información de dicho módulo.

Las tablas de color naranja son las más relevantes arquitectónicamente, dentro de las que se destacan GdcDocComp, GdcFactComProv, GdcFacturaEmpaque, GdcListaEmpaque, GdcCertificadoOr, GdcFacturaVenta, GdcDeclaracionJurada y GdcPermiso así como sus relaciones, debido a que en ellas se almacenan los datos más significativos para la gestión de los Documentos Complementarios.

Las restantes son nomencladoras y contribuyen al completamiento de la información de las tablas nombradas anteriormente a través de los servicios que brindan. De ellas las grises fuertes pertenecen al esquema TC de la base de datos, permitiendo el acceso desde todos los subsistemas que conforman el GINA a los metadatos necesarios para su correcto funcionamiento; las grises claras son TC que pertenecen al esquema GDC. Las blancas pertenecen a los esquemas Contacto y Persona respectivamente y por último las amarillas pertenece al esquema RC.

Todas en su conjunto conforman el modelo de datos relacional normalizado en Segunda Forma Normal (2FN) que plantea que una relación está en segunda forma normal si todas las relaciones poseen atributos atómicos (1FN), y además si todo atributo no primo de la relación depende funcionalmente o de manera total de la clave primaria de la misma

Figura 10: Modelo relacional.

2.3.6 Métricas para la validación del diseño

En la ingeniería de software se ha hecho recurrente la necesidad de comprobar si los artefactos que se generan en la búsqueda de una solución son factibles. Por lo tanto es importante aplicar técnicas que permitan obtener resultados cuantificados que evidencien si se ha tomado el camino correcto. Precisamente porque ese resultado numérico es lo que posibilita obtener respuestas efectivas. Con este objetivo se aplican las métricas que existen para el software que permiten centralizar las funcionalidades operativas de un programa.

El IEEE *"Standard Glossary of Software Engering Terms"* define el termino métrica como *"una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado"* (Negro, 2008)

2.3.6.1 Métricas Orientadas a Objeto

Las métricas OO[22] deben ajustarse a las características que distinguen el software OO. Estas hacen hincapié en el encapsulamiento, la herencia, complejidad de clases y polimorfismo. Se centran en métricas que se pueden aplicar a las características de encapsulamiento, ocultamiento de información, herencia y técnicas de abstracción de objetos que hagan única a esa clase.

Los objetivos principales de las métricas OO son los mismos que los existentes para las métricas surgidas para el software estructurado: (Negro, 2008)

- Evaluar mejor la calidad del producto.
- Estimar la efectividad del proceso.
- Mejorar la calidad del trabajo realizado en el nivel del proyecto.

2.3.6.1.1 Métricas Orientadas a Clases

La clase es la unidad principal de todo sistema OO, por consiguiente las métricas para una clase individual, las jerarquías de clases y las colaboraciones de clases resultaran sumamente valiosas para un ingeniero de software que tenga que estimar la calidad de un diseño. Se ha visto que la clase encapsula a las operaciones (procesamiento), y a los atributos (datos). La clase suele ser el "Predecesor" de las

[22] Terminología utilizada para referir Orientada a Objetos.

subclases (también denominadas *descendientes*) las cuales heredan sus atributos y operaciones. La clase suele colaborar con otras clases. Todas estas características se pueden utilizar como bases de las métricas abordadas a continuación. (Negro, 2008)

El Conjunto de Métricas CK

Uno de los conjuntos de métricas de software OO a los que se hace referencia más ampliamente es el propuesto por Chidamber y Kemmerer[23]. Estas métricas propuestas se refieren al diseño de las clases a las cuales suele aludirse con el nombre de conjunto de métricas CK para sistemas OO, ellas son: (Negro, 2008)

- Métodos ponderados por clase (MPC).
- Árbol de profundidad de herencia (APH).
- Número de descendiente (NDD).
- Acoplamiento entre clases objeto (ACO).
- Respuesta para una clase (RPC).
- Carencia de cohesión en los métodos (CCM).

Para la validación del diseño se toma en consideración:

- Carencia de cohesión en los métodos (CCM) o Lack of Cohesion in Methods (LCOM).
- Acoplamiento entre clases objeto (ACO) o Coupling between Object Classes (CBO).

CCM o LCOM

Plantea que todo método situado dentro de una clase C, accede a uno o más atributos (denominados también variables de instancia). LCOM es el número de métodos que acceden a uno o más de los mismos atributos. Es decir que, los métodos tienen acceso a atributos en común. Si ningún método accede a los mismos atributos, entonces LCOM será 0.

Sí LCOM es elevado, los métodos pueden estar acoplados entre sí a través de dichos atributos. Esto incrementará la complejidad del diseño de clases. En general unos valores elevados para LCOM implican

[23] Ingenieros estadounidenses que desde el año 1991 son la referencia mundial para aplicar métricas de diseño Orientado a Objeto.

que la clase podría diseñarse mejor descomponiéndola en dos o más clases distintas. Aun cuando existen casos en que es justificable un valor elevado de LCOM, es deseable mantener un elevado grado de cohesión, en otras palabras mantener un valor bajo para LCOM. (Negro, 2008)

Aplicando dicha métrica a la clase más significativa en el negocio:

GdcDocComp	
Atributos	**Identificador**
idDocComp	a
idTipoDocComp	b
idEstadoDoc	c
anno	d
numero	e
numeroOriginal	f
numeroVu	g
fechaRegistro	h
fechaEntrada	i
observaciones	j

Tabla 1. Atributos de la clase GdcDocComp.

GdcDocComp	
Métodos	**Atributos**
generarNumero	e
obtenerDocComp	0
devolverDocComp	0
anularDocComp	c
datosTiposDoc	0

Tabla 2. Métodos de la clase GdcDocComp.

Resultados obtenidos en la aplicación de la métrica:

Después de aplicar la métrica CCM o LCOM (**ver Tabla 1 y 2**) a una de las clases más importantes dentro del sistema en este caso **GdcDocComp**, se pudo concluir que dicha clase tomada como muestra para la aplicación de esta métrica, arrojó como resultado un nivel de CCM igual 0, ya que ninguno de los métodos de dicha clase accede a ningún atributo común, para este resultado representa un nivel óptimo según proponen los autores de esta métrica. Esto disminuye la complejidad del diseño de clases y por tanto favorece a una alta cohesión.

ACO o CBO

CBO mide el acoplamiento entre clases de objetos. Una clase esta acoplada a otra si los métodos de una clase usan métodos o atributos de la otra y viceversa. Dada esta definición, los usos pueden significar el tipo de miembro, el tipo de parámetro, variable de método local o el llamado a métodos de una clase. CBO es el número de clases con la que una clase dada esta acoplada. Incluye el acoplamiento basado en herencia (es decir, acoplamiento entre clases relacionadas vía herencia).

En esencia CBO es el número de colaboraciones[24] enumeradas para una clase en su tarjeta índice CRC[25]. A medida que los valores de CBO crecen, es probable que la reusabilidad de la clase vaya descendiendo. Valores altos de CBO complican también las modificaciones y la comprobación que se produce cuando se efectúan modificaciones. (Negro, 2008)

Aplicando dicha métrica a la clase GdcFactComProv en el negocio:

GdcFactComProv	
Responsabilidades	**Colaboradores**
devolverParaJson	sfRcIntegracionComun
obtenerContactos	GdcrContactoDoc
obtenerMercancias	GdcMerFactComProv
obtenerSubtotales	GdcrFactSubtotales
devolverHtml	-
insertarFactComProv	GdcrContactoDoc, GdcDocComp
generarTablaMercancia	-
generarSubtotales	-

Tabla 3: Tarjeta índice CRC para la clase GdcFactComProv.

[24] Son aquellas clases necesarias para proveer a una clase con la información necesaria para completar una responsabilidad.

[25] Un modelo clases-responsabilidades-colaboraciones CRC es realmente una colección de tarjetas índice estándar que representan clases. Las tarjetas están divididas en tres secciones. A lo largo de la cabecera de la tarjeta usted escribe el nombre de la clase. En el cuerpo se listan las responsabilidades de la clase a la izquierda y a la derecha los colaboradores.

Resultados obtenidos en la aplicación de la métrica:

Según plantean Sahraoui, Godin & Miceli en su artículo *"Can Metrics Help Bridging the Gap Between the Improvement of OO Design Quality and Its Automation?"* es deseable que el valor de CBO se encuentre por debajo de 14 para poder asegurar la mantenibilidad y reusabilidad del diseño propuesto. (Sahraoui, et al.) Teniendo en cuenta este criterio y contando que las múltiples colaboraciones con una misma clase son tratadas como un único acceso, se obtuvo un valor de CBO igual a 5, lo que permite afirmar que el diseño propuesto en esta investigación es válido según las métricas aplicadas.

Métricas propuestas por Lorenz y Kidd

Existe otro grupo de métricas muy conocidas también centradas en las clases como la base para la obtención de métricas especializadas. La gran diferencia consiste en las definiciones hacia las cuales fueron dirigidas en su concepción.

- Tamaño.
- Herencia.
- Valores Internos.
- Valores externos.

Para la validación del diseño se toma en consideración:

- Las orientadas al tamaño.

Métrica Tamaño de Clase (TC)

Esta métrica puede determinar el tamaño de una clase para conocer el grado de responsabilidad que existe entre cada una de ellas en un diseño específico. Pueden ser medidas a través de dos factores:

- El número total de operaciones (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase.
- El número de atributos (tanto atributos heredados como atributos privados de la instancia) que están encapsulados en la clase.

Si existen valores grandes de TC estos mostrarán que una clase puede tener demasiadas responsabilidades, lo cual reducirá la reusabilidad de la clase y complicará la implementación y la comprobación. Cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente. (Negro, 2008)

Utilizando los umbrales determinados por **Lorenz, M y Kidd, J.** para un diseño se genera como resultado:

Parámetros de calidad	Valores Grandes de TC
Reutilización	Reduce la reutilización de la clase
Implementación	Complica la implementación
Complejidad de las pruebas	Hace compleja las pruebas del sistema
Responsabilidad	La clase debe tener bastante responsabilidad

Tabla 4. Parámetros de calidad para valores grandes de TC. (Lorenz, y otros, 1994)

Medidas o umbrales aplicados:

Número de operaciones y/o atributos

TC	Umbral
Pequeño	<= 20
Medio	>20 y <=30
Grande	>30

Tabla 5. Umbrales para TC (Lorenz, y otros, 1994)

Medidas para las principales clases de la solución:

No.	Clases	Atributos	Operaciones	Tamaño
1	GdcDocComp	10	5	Pequeño
2	GdcFactComProv	11	8	Pequeño
3	GdcFactEmpaque	11	8	Pequeño
4	GdcListaEmpaque	11	5	Pequeño
5	GdcCertificadoOr	15	1	Pequeño
6	GdcPermiso	16	6	Pequeño
7	GdcDeclaracionJurada	4	5	Pequeño
8	GdcFacturaVenta	8	1	Pequeño
9	GdcMerFacCom	8	1	Pequeño
10	GdcMerFacEmp	12	1	Pequeño
11	GdcMerListEmp	8	1	Pequeño
12	GdcMerPermiso	11	1	Pequeño

13	GdcMerCerOrg	7	1	Pequeño
14	GdcMerDecJurada	12	1	Pequeño
15	GdcrFacturaSubtotal	5	2	Pequeño
16	GdcContrato	6	2	Pequeño
17	GdcBulto	9	2	Pequeño
18	GdcTotalDecJur	12	1	Pequeño
19	GdcrFacCont	3	0	Pequeño
20	GdcrFactEmpCont	3	0	Pequeño
21	GdcrListaEmpCont	3	0	Pequeño
22	GdcrPermisoCont	3	0	Pequeño
23	GdcrContactoDoc	3	2	Pequeño
24	GdcDatoMercancia	2	0	Pequeño
25	GdcrDocCompDatoMer	3	0	Pequeño

Tabla 6. Clases de la capa de negocio a las que se les aplicó la métrica TC.

Resultados obtenidos en la aplicación de la métrica:

Se le aplicó la métrica de TC a un total de 25 clases para un total de 196 atributos y un promedio de atributos de 7.84 y un total de 54 operaciones y un promedio de operaciones de 2.16. De las clases analizadas se tienen un total de 25 de tamaño pequeño, 0 de tamaño medio y 0 grandes.

Umbral	Tamaño	Cantidad de Clases
<=20	Pequeño	25
>20 y <=30	Medio	0
>30	Grande	0

Tabla 7. Clases por tamaño.

Se puede apreciar que todas las clases se clasifican en pequeñas, ninguna mediana, y ninguna grande, lo que implica un resultado positivo según los parámetros de calidad propuestos para esta métrica.

2.4 Modelo de Implementación

En la implementación se comienza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares.

El flujo de trabajo de diseño propone crear un plano del modelo de implementación, por lo que sus últimas actividades están vinculadas a la creación del modelo de despliegue. El flujo de trabajo de

implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue.

Los diagramas de despliegue y componentes, son artefactos generados en este flujo de trabajo y conforman lo que se conoce como un modelo de implementación al describir los componentes a construir, su organización y dependencia entre nodos físicos en los que funcionará a aplicación.

Afortunadamente la mayor parte de la arquitectura del sistema es capturada durante el diseño, siendo el propósito principal de la implementación desarrollar la arquitectura y el sistema como un todo.

Válido señalar que los diagramas de despliegue correspondientes a los módulos a desarrollar no serán contemplados en esta investigación ya que son generados a nivel de aplicación en este caso el GINA.

2.4.1 Estándar de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. (Guerra, 2011) En la actualidad existen diferentes estándares para cada uno de los lenguajes, su utilización permite una comunicación fluida y directa entre los programadores de manera que se favorece la reutilización y mantenimiento de los sistemas.

El Departamento de Soluciones para la Aduana de CEIGE presenta su propia propuesta de estándar de codificación para todas las aplicaciones a desarrollar. Comprende todo el código generado bajo la tecnología y el lenguaje PHP y que utilicen la arquitectura regida por la utilización del framework Symfony.

Como la implementación de las aplicaciones está basada en la utilización de un framework que en su totalidad está codificado en inglés, es imposible desligar la implementación completamente de este idioma. Por lo tanto la implementación se realizará teniendo en cuenta los idiomas de inglés, español, incluso la combinación de ambos al mismo tiempo, sin olvidar la utilización de los sufijos y prefijos **Set**, y **Get**.

Teniendo en cuenta los componentes de Symfony, las aplicaciones deben tener nombres que dejen reflejado claramente cuál es el propósito de la misma, ya sea en una palabra o siglas. Además de un

grupo de aspectos relevantes que presenta dicho documento: (Departamento de Soluciones para la Aduana, CEIGE., 2011)

Con respecto a las acciones (métodos o funcionalidades de la clase *nombreDelModuloAction.class.php*):

- Dentro de las especificaciones del framework está que cada una de estas acciones debe comenzar con la palabra ***execute***.
- Todos los nombres de acciones deben estar en la nomenclatura ***camelCase***[26] comenzando por la palabra ***execute***.
- Los nombres de las acciones deben especificar con la menor cantidad de palabras cuál es el objetivo de la acción, de ser posible estar en infinitivo. Debe especificar bien claro cuál es la acción que se pretende ejecutar pero sin especificar los parámetros que recibe.

Con respecto al nombre de las clases:

- Los nombres de las clases deben estar expresados en notación ***UpperCamelCase***[27].
- Se mantiene el uso de los sufijos ***Peer*** para las clases que se encargan de estas funciones en el modelo.
- No se deben utilizar guiones bajos en su nombre "_".
- Deben expresar con claridad cuál es el alcance y la responsabilidad de la clase.
- Los nombres de las clases no deben estar atados a las clases de las que se deriva, cada clase debe tener un significado por ella misma, no en dependencia de la clase de la que deriva.

Con respecto a los nombres de las funciones:

Todas las funciones definidas por los desarrolladores deben seguir la nomenclatura ***UpperCamelCase***, a no ser que para cierto ámbito se especifiquen características específicas.

Además deben cumplir con las siguientes disposiciones de forma general:

[26] Consiste en escribir frases o palabras compuestas eliminando los espacios intermedios y poniendo en minúscula la primera letra y en mayúscula las demás primeras letras de cada palabra contigua.

[27] Consiste en escribir frases o palabras compuestas eliminando los espacios intermedios y poniendo en mayúscula la primera letra de cada palabra incluyendo la primera letra de la frase.

- Los nombres de las funciones deben dejar reflejado claramente cuál es la acción que realiza la misma.
- Se debe apoyar en la utilización de sufijos que ayuden a identificar el resultado final de la ejecución de un método.
- Se debe apoyar en los prefijos para expresar la acción que realiza sobre un elemento determinado.

Con respecto a las variables:

- Los nombres deben expresar claramente el contenido de la misma.
- Pueden estar referidas en singular o plural.
- Se definen al principio de las estructuras donde son utilizadas.
- En caso de que no se le asigne un valor inicial se deben inicializar con un valor que indique el tipo de dato más general al que debe pertenecer.
- De esta nomenclatura se exceptúan las variables generadas por el framework.
- Los arreglos comenzaran con la palabra **arreglo** y luego el resto en ***camelCase***.

2.4.2 Tratamiento de Errores

Entre los aspectos más importantes a tener en cuenta durante el desarrollo de un software se encuentra el tratamiento de errores, esto se puede afirmar ya que entre las principales causas de errores están: el mal trabajo del usuario con la aplicación y los ataques contra la seguridad de la misma. Es por tanto necesario identificar cada posible error que pudiese producirse en el flujo de trabajo de cada capa del software.

En el módulo Despacho Comercial se gestiona información importante sobre las mercancías que se encuentran tanto en proceso de exportación como de importación, así como del propio proceso, la cual juega un rol fundamental en la toma de decisiones, por dichas razones se realizan varias validaciones.

En la capa de la vista se realizan validaciones en pos de asegurar el formato y tipo de los datos introducidos por el usuario de forma que se minimicen las validaciones de la capa de negocio, esto se lleva a cabo a través de funciones que brinda el framework ExtJS. En la capa de negocio, las validaciones son realizadas a través de los validadores de formularios que provee el framework Symfony. Estos formularios son utilizados para las inserciones o actualizaciones, ya que garantizan la limpieza de los

datos obtenidos de la capa de la vista. Ante la ocurrencia de errores en la capa de negocio se toma como estrategia notificar al usuario, garantizando que los mensajes sean amigables, precisos y brinden solo la información que necesite saber el usuario, dicha estrategia persigue no mostrar información que luego pueda ser explotada para vulnerar la aplicación.

2.4.3 Diagrama de Componentes

El diagrama que muestra la **Figura 11** presenta los principales componentes que se utilizan en el módulo Despacho Comercial. Se realizan las peticiones desde las interfaces de usuarios las cuales son atendidas por el controlador frontal y este a su vez se comunica con la clase *actions* que interactúan con las clases del negocio creadas por Propel. Los componentes de configuración están presentes en todas las acciones realizadas en el módulo.

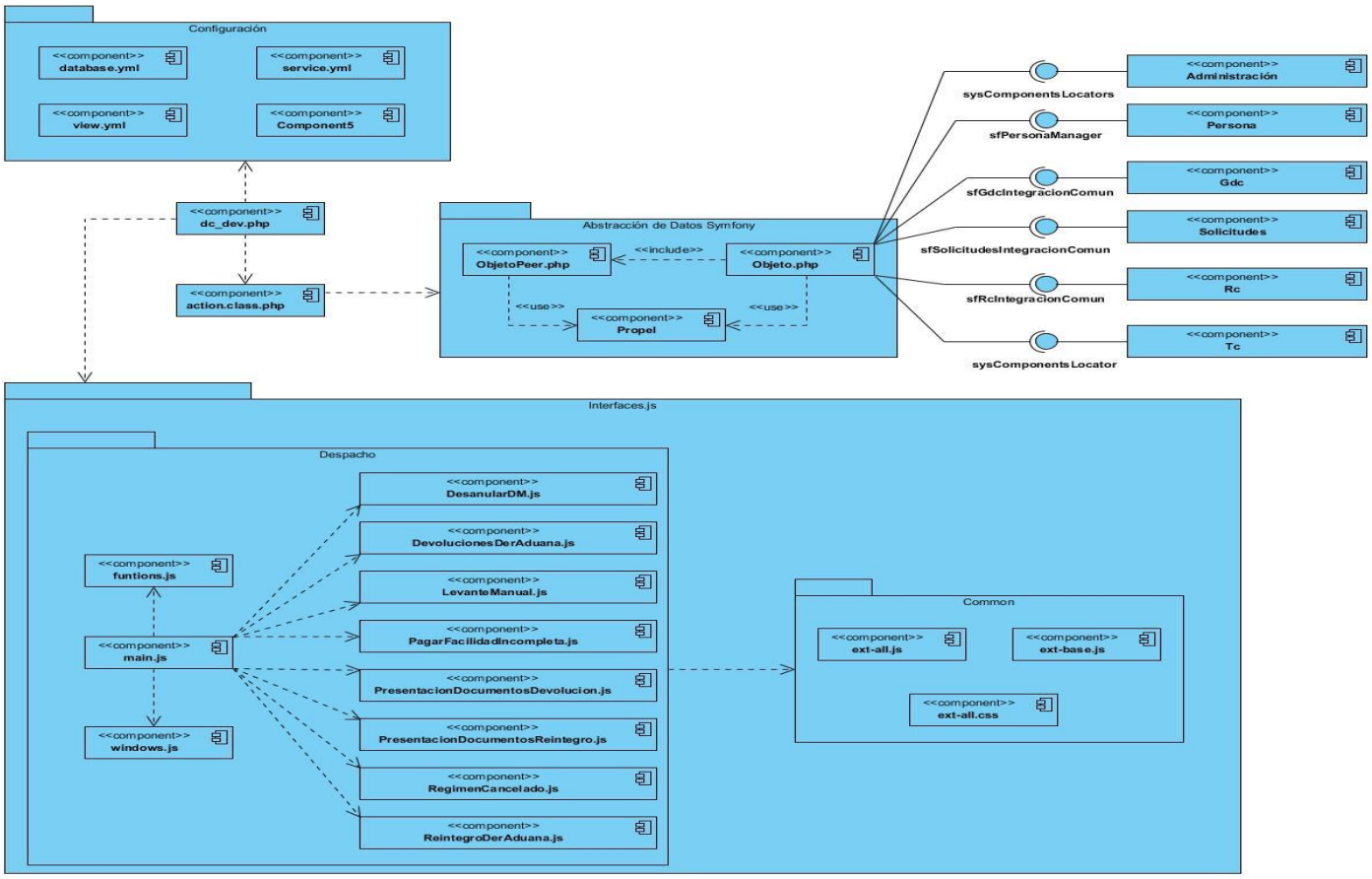

Figura 11: Diagrama de componente perteneciente al módulo Despacho Comercial.

2.5 Conclusiones parciales

Se obtuvieron como resultado, una serie de diagramas que fueron la base utilizada para dar cumplimiento de manera satisfactoria a la implementación del sistema. Se logró incorporar a la solución obtenida todos los requisitos, minimizando la complejidad de implementación de las clases y atribuyendo la responsabilidad equitativamente, logrando también un alto por ciento de reutilización. Se aprovechó al máximo las posibilidades brindadas por los framework utilizados, logrando un correcto tratamiento de errores y aplicación del estándar de codificación propuesto por el Departamento de Soluciones para la Aduana.

Capítulo 3. Constatación de la solución

3.1 Introducción

En este capítulo se realiza la validación de la solución propuesta con el objetivo evaluar los resultados obtenidos tras la implementación de las operaciones en las clases utilizadas para dar respuesta a los requisitos planteados por el cliente. Se presentarán los resultados alcanzados en la fase de pruebas correspondiente al Modelo de desarrollo del Departamento de Soluciones para la Aduana por el cual está regida esta investigación y que ratifique la validez de las funcionalidades del módulo de Despacho Comercial y Documentos Complementarios respectivamente.

3.2 Técnicas de Evaluación Dinámicas o Pruebas de Software

Estas técnicas generan entradas al sistema con el objetivo de detectar fallos cuando el sistema ejecuta dichas entradas. Los fallos se observan cuando se detectan incongruencias entre la salida esperada y la salida real. La aplicación de técnicas dinámicas es también conocida como pruebas de software o *testing* y se aplican generalmente sobre el código, único producto ejecutable del desarrollo. (Juristo, y otros, 2006)

3.2.1 Técnicas de prueba

Las técnicas de evaluación dinámica o prueba proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas. Estas técnicas se agrupan en:

- **Técnicas de caja blanca o estructurales:** se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
- **Técnicas de caja negra o funcionales:** realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar. (Juristo, y otros, 2006)

Para la validación de uno de los módulos desarrollados, en este caso el módulo DC se tuvo en cuenta la técnica de caja negra o funcional, la cual se detallará a continuación.

3.2.1.1 Pruebas de caja negra o funcional

También conocidas como pruebas de comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una "Caja Negra" cuyo comportamiento solo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas. No obstante, como el estudio de todas las posibles entradas y salidas de un programa sería impracticable, se selecciona un conjunto de ellas sobre las que se realizan las pruebas. Para seleccionar el conjunto de entradas y salidas sobre las que trabajar, hay que tener en cuenta que en todo programa existe un conjunto de entradas que causan un comportamiento erróneo en el sistema y como consecuencia producen una serie de salidas que revelan la presencia de defectos. Dado que el objetivo final de la prueba es encontrar una serie de datos de entrada cuya probabilidad de pertenecer al conjunto de entradas que causan dicho comportamiento erróneo sea lo más alto posible. (Juristo, y otros, 2006)

3.2.1.2 Estrategia de pruebas

La estrategia que se ha de seguir a la hora de evaluar dinámicamente un sistema software debe permitir comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Más concretamente, los pasos a seguir son:

1. Pruebas unitarias: Comienzan con la prueba de cada módulo.
2. Pruebas de integración: A partir del esquema del diseño, los módulos probados se vuelven a probar combinados para probar sus interfaces.
3. Prueba del sistema: El software ensamblado totalmente con cualquier componente hardware que requiere se prueba para comprobar que se cumplen los requisitos funcionales.
4. Pruebas de aceptación: El cliente comprueba que el software funciona según sus expectativas. (Juristo, y otros, 2006)

La estrategia de prueba que se utilizó para validar la solución del módulo GDC fueron las pruebas unitarias y serán detalladas a continuación.

3.2.1.3 Pruebas unitarias

La prueba de unidad es la primera fase de las pruebas dinámicas y se realizan sobre cada módulo del software de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional de un programa independiente, está correctamente codificado. En estas pruebas cada módulo será probado por separado y lo hará, generalmente la persona que lo creó. (Juristo, y otros, 2006)

Las pruebas unitarias se desarrollaron haciendo uso del framework Symfony, utilizado en la implementación del módulo. Este tipo de pruebas son altamente recomendadas llevarlas a cabo antes de realizar las pruebas funcionales, debido a que las pruebas unitarias permiten encontrar los errores más evidentes y fáciles de corregir, mientras en la etapa de pruebas funcionales el sistema debería estar bastante estable y con muy pocos errores críticos.

Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular, se encargan de un único caso cada vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto. Para más información, consultar el Capítulo 15. Pruebas unitarias y funcionales de la Guía definitiva de Symfony. (Potencier, 2008)

3.3 Aplicación de pruebas de caja negra al módulo DC

Como se menciona anteriormente una de las pruebas realizadas a la solución propuesta son las pruebas de funcionalidad que se encuentran contenidas dentro de las pruebas de caja negra. El objetivo principal de este tipo de pruebas es lograr localizar fallas funcionales dentro del sistema e identificar situaciones en las cuales la respuesta de este no se apega a las especificaciones establecidas. Estas pruebas se realizan a través de casos de pruebas insertando valores válidos e inválidos para verificar que:

- Se aplique apropiadamente cada regla del negocio.
- Los resultados esperados ocurran cuando se usen datos válidos.
- Sean desplegados los mensajes apropiados de error y precaución cuando se usan datos inválidos.

Los casos de pruebas son un conjunto de condiciones o variables con las que se determina si el requisito de una aplicación es parcial o completamente satisfactorio. Existen muchos casos de prueba para determinar que un requisito es satisfactorio, para poder comprobar que todos los requisitos de una aplicación fueron revisados debe existir un caso de prueba para cada requisito y si este tiene requisitos secundarios se debe hacer un caso de prueba para cada uno de estos. Durante la realización de las pruebas al módulo DC se realizaron 7 casos de pruebas (uno para cada requisito funcional) con cada uno de sus Escenarios de Prueba (EP), a continuación se muestran cada uno de estos casos de prueba:

1. Caso de Prueba: Cancelar temporalidad de Oficio.
    - EP 1.1 Cancelar la temporalidad de oficio.
    - EP 1.2 Adicionar segmento de artículo a cancelar de oficio.
    - EP 1.3 Modificar segmento de artículo a cancelar de oficio.
    - EP 1.4 Eliminar segmento de artículo a cancelar de oficio.
    - EP 1.5 Datos Incorrectos.
2. Caso de Prueba: Ajustar temporalidad.
    - EP 1.1 Ajustar temporalidad.
    - EP 1.2 Datos Incorrectos.
3. Caso de Prueba: Desanular declaración de mercancías.
    - EP 1.1 Desanular la declaración de mercancías.
    - EP 1.2 Datos incorrectos.
4. Presentar documentación correspondiente a la solicitud de devolución de derechos de aduana.
    - EP 1.1 Presentar documento de devolución de derechos de aduana.
    - EP 1.2 Datos incorrectos.
5. Presentar documentación correspondiente a la solicitud de reintegro de derechos de aduana.
    - EP 1.1 Presentar documento de reintegro de derechos de aduana.
    - EP 1.2 Datos incorrectos.
6. Aprobar/Rechazar devolución de derechos de aduana.
    - EP 1.1 Buscar solicitud de devolución de derechos de la aduana.
    - EP 1.2 Aprobar la solicitud de devolución de derechos de la aduana.
    - EP 1.3 Rechazar la solicitud de devolución de derechos de la aduana.

7. Aprobar/rechazar reintegro de derechos de aduana.
    - EP 1.1 Buscar solicitud de reintegro de derechos de la aduana.
    - EP 1.2 Aprobar la solicitud de reintegro de derechos de la aduana.
    - EP 1.3 Rechazar la solicitud de reintegro de derechos de la aduana.

Se realizó una primera iteración donde se pusieron en práctica cada uno de los casos de pruebas con sus respectivos escenarios, en esta iteración fueron identificadas un total de 3 no conformidades que se enfocan en la validación de los datos que son introducidos por parte del usuario, estas no conformidades fueron solucionadas. Luego se procedió a realizar una segunda iteración para identificar cualquier otra no conformidad que pudiese existir. En la **Figura 12** se pueden observar los resultados obtenidos en ambas iteraciones.

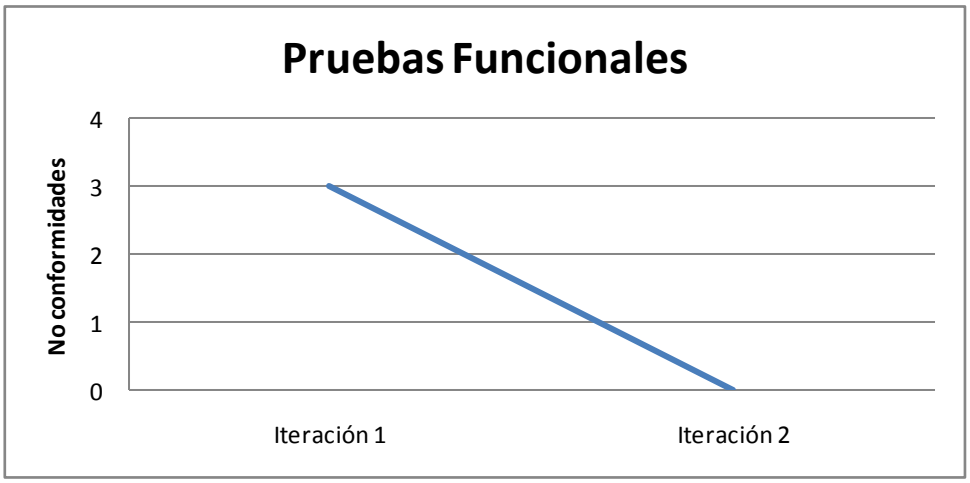

Figura 12: Resultado de las pruebas funcionales.

En la segunda iteración el sistema quedó en total funcionamiento y acorde a las necesidades funcionales requeridas por el cliente.

Ejemplo de un caso de prueba

A continuación en la **Tabla 8** se muestra el caso de prueba que fue aplicado al RF Presentar documentación correspondiente a la solicitud de reintegro de derechos de aduana. En los escenario de prueba del mismo se valora la respuesta del sistema para un posible valor de la variable en cuestión sea este válido (V) o incorrecto (I).

Escenario	Descripción	No. solicitud	Respuesta del sistema	Flujo central
EP 1.1 Presentar documento de reintegro de derechos de aduana	Se presenta documentación correspondiente a la solicitud de reintegro de derechos de aduana.	V 15	Muestra una pantalla con la información correspondiente a la solicitud.	1-El usuario selecciona la opción del menú “Presentación Documentos Reintegro". 2-Muestra la información correspondiente a la presentación de documentos de la solicitud de reintegro de derechos de aduana. 3-Introduce el número de la solicitud. 4-Selecciona la opción “Aceptar”.
EP 1.2 Datos incorrectos	Señala los errores cometidos al introducir los datos.	I 15et I vacio	Muestra un mensaje notificando el/los error(es) detectado(s) por el subsistema Solicitudes.	1-El usuario selecciona la opción del menú “Presentación Documentos Reintegro". 2-Muestra la información correspondiente a la presentación de documentos de la solicitud de reintegro de derechos de aduana. 3-Introduce el número de la solicitud incorrectamente. 4-Selecciona la opción “Aceptar” del mensaje mostrado.

Tabla 8: Caso de prueba del RF Presentar documentación correspondiente a la solicitud de reintegro de derechos de aduana.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	No. solicitud	Campo de texto	No	Número de la solicitud (cadena de caracteres)

Tabla 9: Descripción de las variables.

En el caso de prueba ejemplificado fueron probados posibles valores a insertar en correspondencia con los escenarios de prueba definidos, de esta misma forma se procedió con los 7 casos de pruebas definidos en las 2 iteraciones realizadas.

3.4 Aplicación de pruebas unitarias al módulo GDC

Las pruebas unitarias de Symfony son archivos PHP normales cuyo nombre termina en Test.php y que se encuentran en el directorio test/unit/ de la aplicación. Su sintaxis es sencilla y fácil de leer. Cada prueba unitaria consiste en una llamada a un método de la instancia de lime_test[28]. El último parámetro de estos métodos siempre es una cadena de texto opcional que se utiliza como resultado del método.

Para ejecutar el conjunto de pruebas, se utiliza la tarea test:unit desde la línea de comandos. El resultado de esta tarea en la línea de comandos es muy explícito, lo que permite localizar fácilmente las pruebas que han fallado y las que se han ejecutado correctamente.

Casi todos los métodos permiten indicar un mensaje como último parámetro. Este mensaje es el que se muestra como resultado de la prueba cuando esta tiene éxito (Potencier, 2008). Para realizar dicha validación se ejecutaron un total de 9 pruebas unitarias a diferentes funcionalidades por ser las más significativas para el funcionamiento de la solución:

Métodos o funcionalidades tomados como muestra para la aplicación de las pruebas unitarias:

Funcionalidaddes a probar	Descripción
obtenerDoc	Devuelve un arreglo con los datos del documento complementario.
generarNumero	Genera el número consecutivo para el documento complementario.
datosPorCodDocumento	Devuelve un arreglo con los datos del documento complementario.
anularDocumento	Cambia el estado del documento complementario por “ANULADO”.

[28] Objeto del framework de pruebas lime que utiliza Symfony para realizar sus pruebas unitarias.

obtenerMercancias	Devuelve un arreglo con los datos de las mercancías asociadas al documento complementario.
obtenerSubtotales	Devuelve un arreglo con los datos de los subtotales asociadas al documento complementario.
InsertarContactos	Inserta los contactos asociados a los documentos complementarios.
obtenerContactos	Devuelve un arreglo con los datos de los contactos asociados al documento complementario.
devolverHtml	Devuelve un arreglo con el HTML y los datos asociados al documento complementario.

Tabla 10: Funcionalidades a aplicarle pruebas unitarias.

A continuación se mostrará un ejemplo de prueba unitaria aplicada y el resultado arrojado.

La funcionalidad **obtenerDoc** permite devolver la información almacenada en base de datos correspondiente a los parámetros introducidos (año, tipo documento y número), en caso de que el documento no se encuentre registrado dicha funcionalidad retornará un arreglo de errores, sino devolverá el documento satisfactoriamente. Todos los requisitos que necesiten la obtención de un documento complementario podrán realizarlo mediante este método, dentro de ellos los encargados de registrar, modificar y anular de cada uno de los documentos de ahí la importancia que tiene probar su funcionamiento.

Prueba	obtenerDoc()
Entrada	Datos para la búsqueda del documento complementario (un arreglo, el cual contiene el año, el tipo de documento y el número).
Salida	Un arreglo con los datos del documento complementario encontrado.
Condición	Si la cadena devuelta coincide con lo esperado se considera satisfactoria.

Tabla 11: Parámetros aplicados a las pruebas unitarias realizadas.

```php
$test = new lime_test(1, new lime_output_color());
$arrayDocComp= array(
            'numero' => 1,
            'anno' => date('Y'),
            'idTipoDocComp' => 1
        );
$result = GdcDocComp::obtenerDoc($arrayDocComp['numero'],
                                 $arrayDocComp['anno'],
                                 $arrayDocComp['idTipoDocComp']);

$test->isa_ok($result, 'array','se obtuvo un documento ');
```

Figura 13: Fragmento del archivo 'pruebaDocComplementarioTest.php'.

```
my@debianmyPC:/media/Datos/server$ cd www/GINA
my@debianmyPC:/media/Datos/server/www/GINA$ php symfony test:unit pruebaDocComplementario
PHP Warning:  Module 'oci8' already loaded in Unknown on line 0
PHP Warning:  Module 'PDO_OCI' already loaded in Unknown on line 0
1..1
ok 1 - se obtuvo un documento
 Looks like everything went fine.
my@debianmyPC:/media/Datos/server/www/GINA$
```

Figura 14: Resultado de aplicar la prueba unitaria a la funcionalidad obtener documento complementario.

Posterior a la realización de las pruebas unitarias a las funcionalidades que se listan en la **Tabla 10**, se confirmó el correcto funcionamiento de las mismas, arrojando como resultado que la implementación realizada en el módulo GDC cumple con las especificaciones de los requisitos trazados.

3.5 Conclusiones parciales

La calidad del módulo implementado fue la premisa fundamental en el desarrollo de este capítulo. En ese sentido se obtuvieron los casos de prueba que ayudan a comprobar el correcto funcionamiento del producto desarrollado aplicando finalmente pruebas unitarias y de caja negra. En general, los resultados obtenidos fueron favorables, desde el punto de vista funcional todos los requisitos realizan las funcionalidades requeridas y responden a las necesidades del cliente. Además tras aplicar las pruebas

unitarias al sistema llevadas a cabo mediante el uso del framework Symfony, quedó demostrada la robustez del sistema implementado.

Conclusiones generales

Con la culminación del presente trabajo de diploma se desarrolló el módulo Despacho Comercial del Sistema de Gestión Integral de Aduana (GINA), contribuyendo a la agilización de los procesos aduanales para mejorar las operaciones importación y exportación en el proceso de Despacho Comercial. Los resultados alcanzados permiten concluir:

- Se elaboró el marco teórico de la investigación a partir del estado del arte; el mismo evidenció la carencia de una solución informática capaz de responder a las necesidades y requerimientos de la AGR. Por otra parte, se analizaron diferentes tecnologías, lenguajes y herramientas indispensables para llevar a cabo el desarrollo de la investigación.
- Se realizó el diseño y la implementación de los componentes, con el objetivo de erradicar los problemas del sistema en explotación actualmente en la AGR y fusionar sus mejores prácticas teniendo en cuenta las necesidades del cliente.
- Se aplicaron pruebas al software implementado, las cuales demostraron el cumplimiento de diferentes atributos de calidad, lo que demuestra que los componentes cumplen satisfactoriamente con los requisitos definidos por los clientes.

Recomendaciones

A lo largo de esta investigación fueron identificados algunos elementos que no fueron abordados al no ser parte del alcance inicial, pero se considera que son elementos que incrementarían la utilidad de este trabajo, se recomienda:

- Continuar realizando pruebas de calidad a los componentes para garantizar su buen funcionamiento y su adecuada certificación.
- Realizar el despliegue de la aplicación en varias entidades aduaneras para comprobar si cumple con las expectativas del cliente desde el punto de vista tecnológico.

Bibliografía

Netbeans.org. *Información del lanzamiento del IDE NetBeans 6.9.1.* [En línea] [Citado el: 22 de Febrero de 2012.] http://netbeans.org/community/releases/69/index_es.html.

A. Vizcaino, F. O. García y Caballero, I. 2008. *"Trabajando con visual paragm for uml".* s.l. : Universidad de Cantabria Fac. de Ciencias, 2008.

AGR. 2008. Aduana General de la República de Cuba. [En línea] 2008. [Citado el: 20 de Febrero de 2012.] http://www.aduana.co.cu/.

Alfaro, Econ. Félix Murillo. www.inei.gob.pe . *Herramientas CASE.* [En línea] http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf.

Asenjo, Jorge Sánchez. 2009. *Apuntes Completos Sistemas Gestores de Base de Datos.* 2009.

Bascón Pantoja, Ernesto. 2004. *El patrón de diseño modelo -vista - controlador (MVC) y su implementación el Java Swing. Vol. II.* 2004.

Berndtsson, Mikael. 2008. *Thesis Projects.* Pittsburgh : Springer, 2008. ISBN-13: 978-1-84800-008-7.

Bolivia, Aduana Nacional de. 2011. *A.N.B. Manual de Usuario SIDUNEA.* Bolivia : s.n., 2011.

BOOCH, Grady, RUMBAUGH, James y JACOBSON, Ivar. 2000. *"El lenguaje unificado de modelado. Manual de referencia".* s.l. : Addison Wesley, 2000.

CEIGE. 2012. *Modelo de Desarrollo de Software.* 2012.

Cobo Rodríguez, José Antonio. 2008. *Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas.* 2008.

Costa, Dolors Costal. *Introducción al diseño de bases de datos.*

Costilla, Carmen. 2002. *Características Objeto-Relacionales del Sistema de Gestión de Bases de Datos Oracle. .* 2002.

Departamento de Soluciones para la Aduana, CEIGE. 2011. *Propuesta de un Estándar de Codificación.* La Habana : s.n., 2011.

E.S.Taylor. 1959. *An Interim Report on Engineering Design.* Massachusetts Institute of Technology : s.n., 1959.

Eguíluz Pérez, Javier. 2009. *Introducción a CSS.* 2009.

—. 2008. *HTML y XHTML Introducción a XHTML.* 2008.

—. 2009. *Introducción a JavaScript.* 2009.

Eguiluz, Javier. 2008. www.symfony.es . [En línea] 2008. http://www.symfony.es/2008/02/page/2/.

Fabien Potencier, François Zaninotto. 2008. *Symfony 1.2, la guia definitiva.* 2008.

Foote, Johnson. 1988. *Designing Reusable Classes.* 1988.

Guerra, B.E.R. 2011. *Tránsito y Transferencia para el Despacho Comercial en la Aduana General de la República de Cuba.* Habana : Universidad de las Ciencias Informáticas., 2011.

Guerra, María y de la Fe López, María. 2006. *La automatización de los procesos aduaneros de los países miembros de la CAN en el marco del Proyecto Granadúa (UE-CAN): Caso Venezuela. VIII REUNIÓN DE ECONOMÍA MUNDIAL.* Venezuela : s.n., 2006.

ieee.org. 1990. standards.ieee.org. [En línea] 1990. [Citado el: 15 de Febrero de 2012.] http://standards.ieee.org/findstds/standard/610.12-1990.html..

Juristo, Natalia, Moreno, Ana M. y Vegas, Sira. 2006. *TÉCNICAS DE EVALUACIÓN DE SOFTWARE, Versión: 12.0.* 2006.

Lorenz, M y Kidd, J. 1994. *Object-Oriented Software Metric.* 1994.

Magna, Agenda. 2009. Agenda Magna. [En línea] 27 de Febrero de 2009. [Citado el: 2012 de enero de 14.] http://agendamagna.wordpress.com/2009/02/26/glosario-de-terminos-aduaneros/..

—. 2009. Agenda Magna. El sitio de soluciones justas. [En línea] 26 de Febrero de 2009. [Citado el: 2 de Diciembre de 2011.] http://agendamagna.wordpress.com/2009/02/26/glosario-de-terminos-aduaneros/.

Negro, Ing Pablo Ariel. 2008. *Umbrales para métricas Oientadas a Objeto.* 2008.

Potencier, Fabien. 2008. *Symfony la guía definitiva. Capítulo 15. Pruebas unitarias y funcionales.* 2008.

Real Academia Española. Diccionario de la lengua española - Vigésima segunda edición. [En línea] [Citado el: 2 de Diciembre de 2011.] http://buscon.rae.es/draeI/.

Reyero, Eusebio. 2009. Thespacer.net. [En línea] 15 de Enero de 2009. [Citado el: 19 de Febrero de 2012.] http://wwff.thespacer.net/blog/metodologias-de-diseno.

Rodríguez, , Gorge I.Arce. 2008. *La Importancia de las Aduanas en el Comercio Exterior.* San Marcos : s.n., 2008.

Sahraoui, Houari A., Godin, Robert y Miceli, Thierry. *Can Metrics Help Bridging the Gap Between the Improvement of OO Design Quality and Its Automation?* Montreal (QC), Canada : s.n.

Sommerville, Ian. 2005. *Ingeniería del Software (séptima edición).* Madrid (España) : Pearson Educación, SA,, 2005. 84-7829-074-5.

Torossi, Profesor: A.U.S. Gustavo Marcelo. *Diseño de Sistemas.* s.l. : Universidad Tecnológica Nacional - F.R.R.

Unido, A.R. Sistema HM Revenue & Customs. [En línea] [Citado el: 12 de Diciembre de 2011.] http://customs.hmrc.gov.uk..

Universidad Carlos III de Madrid. 2006. JAVASCRIPT. [En línea] Universidad Carlos III de Madrid., 26 de Abril de 2006. [Citado el: 22 de Febrero de 2012.] http://perso.wanadoo.es/javascript_12/.

Vazquez, Jose Antonio Gallego. 2003. *Desarrollo de web con PHP y MySQL.* 2003.

Zaninotto, Fabien Potencier y François. 2008. *Symfony la guia definitiva.* 2008.