Universidad de las Ciencias Informáticas

Facultad 3

Título: Diseño e implementación del módulo Importación de los Depósitos Temporales de Frontera

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

Autores: Yaksel Duran Rivas

Daylin Sanz Mantilla

Tutores: Ing. Lianet Pineda de la Nuez

Ing. Yuniel Rodríguez Bello

La Habana, Junio 2011

"Año 53 de la Revolución."

DECLARACIÓN DE AUDITORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de Junio del año 2011

_____	_____
Firma del autor	Firma del autor
Yaksel Duran Rivas	Daylin Sanz Mantilla
_____	_____
Ing. Yuniel Rodríguez Bello	Ing. Lianet Pineda de la Nuez
Firma del tutor	Firma del tutor

"La gloria y el tiempo no son más que un estímulo al cumplimiento del deber."

José Martí

La vida nos enfrenta constantemente a impredecibles obstáculos y retos. Para el ser humano lo más importante no es vencerlos, sino comprender lo placentero que es vivirlos. Aún hay recuerdos que perduran en nuestra mente de estos 5 años de universitario y pertenecen a aquellas personas con las que compartimos nuestros días. Cada uno es algo especial y cada uno de ellos forma parte de nuestra historia. Gracias a todas esas personas por estar ahí, gracias a nuestros amigos por estar siempre presentes. La Universidad es la escuela de la vida, no hay obstáculos que con interés y decisión no se destruyan. ¿Y los retos? ... Los retos son pruebas que nos pone la vida.

Graduarnos fue el gran reto que guardó para nosotros.

Daylin y Yaksel.

AGRADECIMIENTOS

Yaksel

A mis queridos padres Marlene Rivas Barzaga y Luis Manuel Duran Marrón por haberme traído al mundo, por confiar siempre en mí, por su ejemplo, su amor, su cariño, su apoyo incondicional, y por la educación que en mí inculcaron.

A mi hermano Yunior Duran Rivas por apoyarme en cada momento, por ser mi paradigma como persona y como profesional.

A toda mi familia por estar día a día brindándome todo su apoyo.

A mis tutores, y de manera muy especial a Lianet, por su gran ayuda, sin ella hubiera sido imposible la realización de este trabajo y graduarme como ingeniero en ciencias informáticas.

A mis compañeros y amigos del proyecto: JJ y Elizabeth, quienes supieron ayudarme desinteresadamente en cada momento.

A mis compañeros de estudio y amistades en estos 5 años maravillosos de universidad, a los que están y los que estuvieron, a mis amistades en Venezuela, quienes contribuyeron con mi formación como persona y trabajador.

A mis compañeros en este último año de universitario, mis sinceros y fieles amigos, lo que han estado presente en cada momento, los que nunca olvidaré por soportarme, quererme y ayudarme tanto: Bonilla, Pablo, Ivian, Raquel y Cervela.

A mi compañera de tesis y amiga Daylin Sanz Mantilla, sin la cual hubiera sido imposible realizar este sueño, por su empeño y consagración en cada momento; y a su hermana Daylenes por su contribución y ayuda desinteresada.

A mi hermano Dayán, con quien he compartido grandes momentos en la universidad, quien ha sido siempre importante y especial para mí y lo seguirá siendo, formando parte de mi vida, de mis pensamientos, sentimientos, decisiones y emociones. No podría quedarte alguna duda de lo que significas

para mí ni de tu lugar en mi ser. Solo espero que esta amistad llegue muy lejos, se mantenga y quede para la historia.

A la Revolución que me permitió estudiar en Universidad de las Ciencias Informáticas y a todos mis profesores por su dedicación y por contribuir a mi formación profesional y revolucionaria.

A todos los que de una manera u otra contribuyeron al desarrollo de esta investigación lleguen mis agradecimientos más sinceros.

A todos, Gracias

AGRADECIMIENTOS

Daylin

Quiero agradecer haber llegado tan lejos a mis padres Zoraima y Camilo, los que me han brindado todo su amor y fortaleza para salir adelante cada día y enfrentar cada reto que me impone la vida. Por su desmedido amor y las horas de sacrificio dando de ellos siempre lo mejor para que saliera adelante. Son mi gran resorte para lograr alcanzar mis metas en la vida.

A: mi hermana Daylenes Sanz Mantilla que me ha apoyado y extendido las dos manos siempre que lo he necesitado. Aunque no lo digo con mucha frecuencia eres muy importante y espero que eso nunca cambie.

A: mis abuelas Ofelia y Zoila que me brindan su infinito cariño en todo momento, a mis padrinos Osvaldo y Gisela que siempre han estado velando y preocupándose por mi bienestar. A mi tía Zoelia, que aunque lejos siempre me acompaña, a todos mis primos que me apoyan incondicionalmente.

A: mis tutores, muy especialmente a Lianet, ya que hicieron derroche de profesionalidad en todos los sentidos para ayudarnos a salir adelante y cumplir este reto tan importante.

A: mi amigo y compañero de tesis Yaksel Duran Rivas, sin el cual no hubiese podido llegar hasta aquí, enfrentando cada día, los traspiés que nos hacían ver este día tan lejano.

A: la Revolución por darme la oportunidad de ser quien soy, al MININT que ha sido una escuela dentro de otra, a la universidad que ha hecho de mí una mejor persona, a todos los profesores que me han brindado parte de su conocimiento.

A: los amigos, que estuvieron dispuestos siempre a ayudarme y a apoyarme en especial Dulce, Omar, Yunior, Yeiciel y los Chicos y a todos los que he ido descubriendo y conservando a lo largo de mi vida y en especial, en estos cinco años universitarios, los de mi grupo desde primer año y los del actual, pues me han tenido que aguantar mis malos y buenos momentos, haciendo mi estancia en la UCI placentera la mayor parte del tiempo.

A Dayán, mi amigo incondicional, que con algunos ha presentado desavenencias, pero a mí no ha conseguido dejar de hablarme ni un segundo, y ha tenido que aguantar mis perretas y alegrías, siempre

con buena cara, ayudándome cada vez que lo necesité y sabiendo que siempre ha podido y puede contar con mi amistad.

A Manolito que siempre tuvo que soportar mi pedidera de favores, brindándome en todo momento su mano amiga y haciéndome reír con sus locuras; a Ivian, Pablo, Bonilla, con los cuales compartí gran parte de mi vida universitaria.

A: Raquel y Rowe por estar presentes cuando las necesité, por brindarme su confianza y porque a pesar de las desavenencias en algunas ocasiones, supimos limar asperezas y mantener la amistad.

A: todas aquellas personas amigas que me apoyaron en Venezuela y no tuvieron reparos en ayudarme siempre, muy especialmente a Yanexy, Llyvis y Dariel.

A todos y todas aquellas personas que siempre confiaron en mí.

A todos muchas gracias por existir y por haberme dado la oportunidad de conocerlos.

DEDICATORIA

Yaksel

Hoy que mi sueño ya está cumplido quiero dedicarles especialmente a mis padres Luis Manuel Duran Marrón y Marlene Rivas Barzaga cada letra de este Trabajo de Diploma.

A mi hermano Yunior Duran Rivas, a quien le seguí los pasos como profesional e ingeniero.

Daylin

Dedico el presente trabajo de diploma a mi familia, muy especialmente a:

Mis padres Zoraima L. Mantilla Sosa y A. Camilo Sanz Marcheco, que han sido los mejores del mundo, dándome su cariño incondicional y sirviéndome de guía para ser la persona que soy hoy.

RESUMEN

La informatización de procesos en la actualidad brinda ventajas y beneficios de orden económico, social, y tecnológico, asegurando una mejora en la calidad del trabajo, con una reducción de costos y tiempo de procesamiento de información. En este proceso se encuentra inmersa la Aduana General de la República que se ha propuesto la informatización de sus procesos. Una de estas áreas es depósitos temporales, la cual posibilita que las mercancías objeto de operaciones aduaneras sean depositadas provisionalmente, en recintos cerrados, silos o áreas ubicadas en espacios geográficos próximos a las oficinas aduaneras. El trabajo de diploma tiene como objetivo diseñar e implementar un sistema que informatice los procesos asociados a la importación de mercancías en los depósitos temporales de frontera de la Aduana General de la República.

En el desarrollo de la solución se utiliza como metodología de desarrollo de software RUP [1] con adaptaciones propias del Departamento de Soluciones para la Aduana, haciendo uso de UML [2] como lenguaje de modelado en la creación de los distintos artefactos correspondientes al diseño y la implementación: diagrama de paquetes, diagramas de clases del diseño con estereotipos web, diagramas de secuencia y diagrama de componentes. Se valida el diseño obtenido haciendo uso de diferentes métricas. Se implementa el sistema y se comprueba la solución con la realización de pruebas de caja negra.

La utilización de este sistema le permitirá a la Aduana General de la República poseer un software que permita gestionar los procesos de importación de mercancías en los depósitos temporales de frontera.

Palabras Claves: informatización, mercancías, procesos, software.

[1] *Rational Unified Process* (Proceso Unificado de Racional). Es un proceso de desarrollo de software constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

[2] Lenguaje Unificado de Modelado. Conjunto de notaciones y diagramas estándar utilizados para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan.

Tabla de Contenidos

INTRODUCCIÓN

El comercio internacional implica necesariamente la circulación entre fronteras de mercancías y personas, aumentando gradualmente las exportaciones e importaciones en los últimos años. La habilidad que tenga un país de transportar bienes o productos de un lugar a otro de forma eficiente, segura y a un costo razonable, puede brindarle ventajas notables en cuanto al aumento de su mercado potencial, las relaciones con los demás países y la fortaleza de su respectiva economía.

A pesar de estos beneficios, el tráfico internacional en ocasiones puede traer consigo diversos riesgos y comprometer el desarrollo de la economía nacional, la salud y la seguridad de un país, siendo el papel de las aduanas muy importante, con la misión de detección y enfrentamiento al narcotráfico, terrorismo y contrabando, así como la protección de la población, la industria nacional y el medio ambiente.

Las mercancías que circulan hacia nuestro país desde diferentes regiones, al llegar a Cuba permanecen temporalmente en almacenes autorizados por la Aduana General de la República (AGR), hasta continuar su curso al destino final, lo que implica que la AGR necesita llevar un control de las operaciones de importación que se realizan en los depósitos temporales, debido al volumen de mercancías que estos almacenan.

Un depósito temporal es un lugar o instalación autorizado por la AGR, para el almacenamiento de mercancías de toda clase, cualquiera que sea su naturaleza, cantidad, país de origen, destino o procedencia; que esperan un destino final. (1)

Los **depósitos temporales**, según su ubicación, se clasifican en:

De Frontera: están ubicados en áreas de los puertos y aeropuertos.

- Portuarios: situados dentro del área del recinto portuario.
- Aeroportuarios: situados dentro del área del recinto aeroportuario.

Interiores: están ubicados en áreas fuera de los puertos y aeropuertos.

INTRODUCCIÓN

En el Centro de Automatización para la Dirección y la Información de la AGR (CADI) se desarrolló un módulo dentro del Sistema Único de Aduanas (SUA), denominado Almacén. Este módulo informatiza solamente la extracción y recepción de mercancías bajo los regímenes aduaneros [3] de tránsitos [4] y transferencias[5] entre depósitos temporales, pero no implementa todos los requerimientos necesarios para la gestión y control de la totalidad de los procesos de importación que se efectúan dentro del depósito.

Este sistema utiliza el paradigma de la programación estructurada, complejizando su mantenimiento y reutilización; además es obsoleto e insuficiente y no se acopla con la arquitectura definida para el Sistema de Gestión Integral de la Aduana (GINA) basada en el patrón MCV (Modelo-Vista-Controlador). Entre las desventajas de esta solución está el paradigma de programación utilizado, el cual es inapropiado y poco robusto. El principal inconveniente de este método de programación, es que se obtiene un único bloque de programa, que cuando se hace demasiado grande puede resultar problemático su manejo. Teniendo en cuenta la poca factibilidad de esta solución, la dirección de la Aduana General de la República decide informatizar la totalidad de los procesos en los depósitos temporales de frontera.

Debido a la incapacidad del módulo Almacén para gestionar la totalidad de los procesos de importación que se llevan a cabo en los depósitos temporales de frontera, así como para integrarse al modelo arquitectónico establecido en el proyecto Aduana y a la obtención previa de los requisitos por parte de un grupo de analistas, el presente trabajo de diploma se propone resolver el siguiente **problema**: ¿ Cómo satisfacer los requisitos para informatizar los procesos asociados a la importación de mercancías en los depósitos temporales de frontera de la Aduana General de la República?

[3] Tratamiento aplicable a las mercancías sometidas al control de la Aduana, de acuerdo con la Normativa Aduanera, según la naturaleza y objetivos de la operación.

[4] Régimen aduanero bajo el cual son transportadas las mercancías de una aduana a otra en territorio nacional bajo control aduanero.

[5] Régimen aduanero bajo el cual son transportadas las mercancías de un depósito a otro de la misma aduana en territorio nacional bajo control aduanero

El **objeto de estudio** está enmarcado en: los procesos de diseño e implementación en el desarrollo de software de gestión aduanera y el **campo de acción** en**:** el diseño e implementación de los procesos de importación de mercancías en los depósitos temporales de frontera de la Aduana General de la República.

Con el propósito de dar solución al problema expuesto se persigue como **objetivo general**: diseñar e implementar un sistema informático que gestione los procesos asociados a la importación de mercancías en los depósitos temporales de frontera de la Aduana General de la República.

Se plantea la siguiente **idea a defender:** diseñar e implementar un sistema informático permitirá la informatización de los procesos de importación de mercancías en los depósitos temporales de frontera de la Aduana General de la República.

Del objetivo general se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico relativo a las soluciones de software que gestionan la importación de mercancías en los depósitos bajo control aduanero y el diseño de sistemas informáticos.
- Modelar el diseño del sistema.
- Implementar el sistema.
- Realizar pruebas del sistema.

Con el fin de dar cumplimiento a los objetivos del presente trabajo, se concibieron las **siguientes tareas de investigación:**

- Estudio del proceso de análisis del módulo depósitos temporales de frontera.
- Análisis del estudio del estado del arte de las soluciones de software que gestionen y controlen las operaciones de los depósitos de las aduanas para establecer una posición al respecto.
- Análisis de la metodología de desarrollo de software, lenguajes, herramientas de modelado y tecnologías utilizadas en el proyecto para contribuir a la confección del marco teórico.
- Modelado de los artefactos del diseño: diagramas de clases, diagramas de secuencia, diagrama de despliegue y modelo de datos.

- Validación del diseño obtenido.
- Implementación del código fuente del sistema.
- Diseño de casos de prueba.
- Registro de resultados de las pruebas.

Los **resultados esperados** con el desarrollo de la investigación constituyen los siguientes:

- Documentación del estado del arte.
- Modelo de datos.
- Diagrama de clases del diseño.
- Diagramas de secuencia.
- Diagrama de despliegue.
- Código fuente del sistema.
- Casos de Prueba.

Para la obtención de estos resultados se plantea la necesidad del uso de diferentes métodos científicos, como estrategia general, que orienta y permite organizar globalmente la actividad hacia el conocimiento y la solución al problema.

El método **histórico – lógico** es empleado para la revisión de las soluciones informáticas existentes con el fin de familiarizarse con el problema planteado, el proceso de desarrollo de software, así como las técnicas de diseño más usadas. El **analítico – sintético** contribuye al análisis sistemático del problema y la comprensión de los procesos aduaneros para mejor entendimiento. La **modelación** ayuda a comprender y analizar el sistema, así como conceptualizar la solución a través de la creación de modelos. La **implementación** es el desarrollo de la solución propuesta al problema planteado, siendo de vital importancia el uso de buenas prácticas de desarrollo de software para garantizar la validez y fiabilidad del trabajo.

El trabajo de diploma está constituido por 3 capítulos en los que se encuentra el contenido distribuido de la siguiente forma:

El **primer capítulo** muestra la fundamentación teórica de la investigación: el estado del arte de algunas de las soluciones informáticas existentes para la gestión de inventario y el control de las mercancías de importación en los depósitos temporales de frontera, así como un estudio de los elementos de diseño de software y las metodología, lenguajes y herramientas de modelado utilizadas para el desarrollo en el proyecto.

El **segundo capítulo** contiene los diagramas de clases del diseño con estereotipos web, los diagramas de secuencia y el modelo físico de datos, finalizando con la validación del diseño. En la fase de implementación se abordan temas correspondientes al desarrollo de la solución, describe cómo se implementan los elementos del modelo de diseño y define la estructura del producto y la construcción de la solución.

En el **tercer capítulo**, se enfatiza en la validación del sistema, empleando pruebas de caja negra[6] mediante la realización de casos de prueba, con el objetivo de detectar no conformidades existentes en la aplicación y poder solucionarlas para obtener un software que responda a los requisitos definidos por el cliente.

[6] Pruebas a un software desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

INTRODUCCION

En el capítulo se realiza un estudio del estado del arte de las soluciones informáticas nacionales e internacionales para la gestión y control de las operaciones de los depósitos temporales de las aduanas. También se detallan aspectos importantes de los estilos arquitectónicos, patrones de diseño, patrones de arquitectura, herramientas, lenguajes y tecnologías utilizadas en el proyecto.

1.1 Generalidades de los Depósitos Temporales de Frontera.

Hoy en día la comercialización entre las naciones ha alcanzado gran auge. La exportación e importación de mercancías son de vital importancia para países que están en desarrollo, contribuyendo al impulso de la economía de un país. Por estas razones se vuelve imprescindible, gestionar adecuadamente los detalles relativos a estos procesos mientras estén bajo responsabilidad de la Aduana.

Los depósitos temporales de frontera constituyen lugares autorizados por la Aduana, que surgen para almacenar mercancías de toda clase, cualquiera que sea su naturaleza, cantidad, país de origen, destino o procedencia, que esperan un destino final. Velar por que las legislaciones sean cumplidas es responsabilidad de los especialistas, inspectores y jefes de Aduana que laboran en los depósitos temporales de frontera de la Aduana General de la República. Entre las principales operaciones que intervienen en los procesos de importación y exportación se encuentran: carga, descarga, clasificación, extracción y almacenamiento de mercancías.

Estas operaciones consisten respectivamente en:

Carga: operación que consiste en colocar en un medio de transporte, las cargas; desde el muelle, la pista, almacén de depósito o desde otro medio de transporte, que serán transportadas por éste, para ser llevadas a su destino final.

Descarga: operación que consiste en extraer de un medio de transporte las cargas transportadas por éste, para ser depositadas sobre el muelle, la pista, almacén de depósito o sobre otro medio de transporte.

Clasificación: conjunto de operaciones dirigidas a identificar y separar las cargas.

Almacenamiento: operación que consiste en colocar las mercancías en el local del depósito temporal, atendiendo a las normas establecidas según su naturaleza, características y marcas.

Extracción: acción y efecto de retirar las mercancías del recinto portuario o aeroportuario para su traslado hacia el destino final.

En las operaciones de los depósitos temporales intervienen:

Operadores portuarios y aeroportuarios: persona natural o jurídica que en virtud de la autorización que le ha sido otorgada, administra y opera un almacén de depósito bajo control aduanero. Se compromete mediante la firma de un contrato a prestar servicios de carga, descarga, recepción, clasificación y entrega de las mercancías.

Inspector de Aduana: controla las actividades de importación y exportación que puede incluir: chequeo integral de todas las operaciones realizadas a fin de asegurarse del cumplimiento de las formalidades aduaneras.

Declarante (Agente de Aduanas o Apoderado): toda persona natural o jurídica que hace una declaración en Aduana o en nombre de la cual esta declaración es hecha. **Agente de Aduanas o Apoderado** que se ocupa de realizar los trámites aduaneros para poder obtener las mercancías y llevarlas hasta su destino final en la importación o ser exportadas. (2)

1.2 Soluciones informáticas para la gestión y control de las operaciones en los depósitos temporales de frontera de la Aduana General de la República.

Actualmente, debido al volumen de mercancías que se maneja en los depósitos temporales de las Aduanas en los diferentes países, es muy común poseer un sistema que sirva de apoyo al control de las mercancías dentro de los depósitos, para evitar pérdidas innecesarias y desvío de valores.

A continuación se muestra un análisis sobre algunas soluciones extranjeras y cubanas, que informatizan la gestión de las operaciones de los depósitos bajo control aduanero.

1.2.1 Soluciones Extranjeras.

S4 tr@nsERP

Los distintos módulos de depósito disponibles en **S4 tr@nsERP** están basados en los requerimientos de cada uno de los módulos que gestionan, de modo que, además de los controles clásicos de un almacén, se incorporan las transmisiones del Intercambio electrónico de datos (EDI) y los diferentes listados de control requeridos por la Aduana. Pueden funcionar como una aplicación independiente o integrada con el resto de los módulos de **S4 tr@nsERP.** Está construido con herramientas Microsoft y SQL Server como almacén de datos, intercambio con Lenguaje de Marcas Extensible (XML)[7], y herramientas de Office System de Microsoft. (3)

El módulo de la aplicación: **Almacén de Depósito Temporal,** gestiona los siguientes procesos:

- Cambios de ubicación.
- Recepción de tránsitos.
- Registro de entradas y salidas.
- Control de vencimiento de declaraciones.
- Contabilidad de existencias.
- Certificados de recepción.

EV Sys: Sistema de Gestión de Recintos Aduaneros y Control de Depósitos Aduaneros Industriales.

El régimen especial de Depósito Aduanero Industrial, permite operaciones de industrialización en los recintos aduaneros, que pueden ser instalaciones portuarias y aeroportuarias, con derecho a la gestión de

[7] Lenguaje de Etiquetado Extensible utilizado en el intercambio de una gran variedad de datos. Su función principal es describir datos. Sirve para estructurar, almacenar e intercambiar información.

impuestos en el proceso de importación. En casos de operaciones de industrialización en áreas aduaneras, se exige que la empresa esté habilitada previamente como "Depósito Aduanero".

El sistema EVSys, desarrollado por Softway[8], fue creado al tomar como base el ADE Coana/Cotec 02/2003 [9], que especifica el funcionamiento del sistema informatizado exigido y permite controlar totalmente el Depósito Aduanero Industrial.

Esta misma solución está preparada para atender también toda la operación del régimen DAC (Depósito Aduanero Certificado), que considera la mercancía como exportada a partir del momento en que se la admita en este régimen, incluso si permanece en territorio nacional, y brinda un sin fin de ventajas fiscales y logísticas a la operación. (4)

CLIP™

A Cotecna[10] se le ha concedido la gestión de depósitos de aduanas en varios países. Trabajando como un facilitador de servicios eficaz e independiente, Cotecna actúa como socio de:

- **Aduanas**: proporcionando una garantía eficaz y fiable de la integridad del sistema.
- **Usuarios:** como proveedor de servicios y a través de una gestión de envíos ágiles, segura e integrada.

Según los requisitos de control aduanero, Cotecna ha desarrollado su propia solución informatizada de gestión de depósitos de aduanas llamada CLIP™. Este servicio no sólo proporciona una imagen a tiempo real de los niveles del inventario, sino que además asegura que se apliquen controles aduaneros continuos y medidas de seguridad a lo largo de la cadena logística del depósito. CLIP™ utiliza un sistema

[8] Consultora orientada a proveer soluciones informáticas prácticas y confiables para la gestión y administración, para dinamizar el trabajo de la organización, aumentar su eficacia y productividad, y acompañar su crecimiento dando respuesta a sus necesidades crecientes en un mercado altamente competitivo.

[9] Legislación que establece una nueva sistemática para auditoría y suministro de datos, prevé que los recintos aduaneros deban cumplir una serie de controles.

[10] Empresa internacional líder en servicios de inspección comercial, seguridad y certificación.

de gestión de depósitos, logística y transporte. Es por lo tanto una solución que une en una sola plataforma los depósitos y la logística, asegurando así la coexistencia óptima de la gestión del espacio, los informes y las funciones de control aduanero. (5)

SIDUNEA

El Sistema Aduanero Automatizado (SIDUNEA) es una herramienta informática para el control y administración de la gestión aduanera, desarrollada por la Conferencia de las Naciones Unidas sobre el Comercio y el Desarrollo (UNCTAD), y que actualmente es usada con éxito en más de 80 países. SIDUNEA permite realizar un seguimiento automatizado de las operaciones aduaneras y controlar efectivamente la recaudación de los impuestos aduaneros, porque este sistema verifica automáticamente los registros, calcula los impuestos y contabiliza todo lo relativo a cada declaración, con la mínima intervención del factor humano subjetivo.

Al ser un sistema multidisciplinario, está especializado en cada área del trabajo aduanero para ser la herramienta de trabajo de todos los clientes de la aduana, sean usuarios internos o externos, privados o públicos. De este modo se convierte en un único lenguaje, seguro y comprensible para todos los actores del proceso. SIDUNEA se puede configurar de acuerdo a las características nacionales de cada régimen aduanero, al arancel nacional y a la legislación de cada país, además de implementar los estándares internacionales para procesar los datos de comercio exterior ya acordados por la Organización Mundial de Aduanas (OMA) y por la Organización Internacional para la Estandarización (ISO).

Entre las ventajas que se pueden obtener con la aplicación del Sistema Aduanero Automatizado se encuentran:

- Optimizar los tiempos y recursos del proceso aduanero.
- Monitorear el pago de los impuestos.
- Evitar la evasión de impuestos.
- Administrar efectivamente el proceso de despacho.

Gracias a la tecnología JAVA, permite el uso de tarjetas inteligentes con procesadores, controlar los accesos al sistema y los pagos electrónicos. Así mismo, su aplicabilidad vía Internet, permite acceder al sistema a través de dispositivos inalámbricos. También es resistente a las caídas de las telecomunicaciones, es compatible con la mayoría de los sistemas Administradores de Bases de Datos Relacionales (RDBMS), *Java Database Connectivity (JDBC[11])* e independiente de las plataformas y de los sistemas de base de datos. Además cuenta con:

- Interfaz de usuario amigable.
- Funciones especiales como multilenguaje, gestión, propiedad de documentos y auditoría.

SIDUNEA permite trabajar con mayor comodidad para elaborar recaudos cuando más convenga, dentro de los tiempos permitidos, con o sin conexión a la red, así como la posibilidad de revisar los datos ingresados tantas veces como sea necesario, sin pérdida de tiempo ni recursos. Garantiza la transparencia en los procesos, rapidez en el control de las operaciones en tiempo real, reducción de trámites y tiempo de almacenamiento, sustitución del papel por documentos electrónicos, así como el pago electrónico de los impuestos. (6)

MODSHD (Módulo de Depósito Temporal): permite a las almacenadoras, la generación e impresión del formato del Pase de Salida en el Sistema Aduanero Automatizado SIDUNEA, en el que se integra el código de barra que requiere el circuito de inspección de rayos x, así como la descongestión de impresión de los mismos, lo que repercutirá positivamente en la rapidez de las operaciones; así como realizar los procedimientos de localización o relocalización de las mercancías, sobrantes y faltantes y la emisión del Acta de Recepción. (7)

1.2.3 Soluciones Cubanas

SUA (Sistema único de Aduanas)

En el año 2004, especialistas del Centro de Automatización y Dirección de la Informatización de la Aduana General de la República (CADI), comenzaron el desarrollo de una solución de software para gestionar los

[11] Java Database Connectivity, API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

procesos aduanales en el país, denominado Sistema Único de Aduanas (SUA), uniéndoseles posteriormente, el esfuerzo de estudiantes y profesores de la Universidad de las Ciencias Informáticas, para modernizar las tecnologías que utilizan para gestionar los procesos de la Aduana.

El SUA es un sistema web multiplataforma desarrollado bajo las premisas del uso de software libre. Dentro del SUA existe el subsistema denominado Almacén, que gestiona solamente una parte de los procesos que se realizan en los depósitos temporales, tales como transferencias y tránsitos de mercancías de importación y exportación. Es una aplicación que utiliza como gestor de Base de Datos: Oracle, lenguaje de programación PHP y paradigma de programación estructurada.

Conclusiones del estudio:

De las soluciones extranjeras analizadas anteriormente, que se emplean para gestionar las operaciones que se realizan en los diferentes depósitos bajo control aduanero, no se encontró ninguna factible para utilizar por la Aduana General de la República. Las mismas implementan normas y legislaciones propias de sus países, excepto el Sistema Aduanero Automatizado (SIDUNEA). Todas son herramientas privativas y esto conlleva un alto costo en inversiones por concepto de licencias y soporte. En el caso de la solución cubana las razones de no adecuación, están sustentadas además de la tecnología sobre la cual está desarrollada y paradigma de programación utilizados, en su incapacidad para gestionar una gran cantidad de procedimientos y operaciones comunes que ocurren en los depósitos temporales de frontera. (2)

1.3 Diseño de Software.

El proceso de desarrollo de software es aquel en el que las necesidades del usuario son traducidas en requisitos de software, transformados en diseño y el diseño implementado en código. (8)

Cuando se diseña se debe tener conocimientos sobre los clientes y sus necesidades, crear prototipos, realizar pruebas, definir requisitos y tomar decisiones sobre estrategia de la empresa.

El éxito de un software también depende de:

- Facilidad de uso.
- Agradable interfaz de usuario.

- Funcionalidades.
- Contenidos adecuados.

Un software en desarrollo debe presentar ciertos estándares para su construcción; estilos, patrones y tendencias que son muy útiles en la obtención de un diseño de software que sea fiel y tenga robusta representación del sistema final.

1.3.1 Arquitectura de Software.

La arquitectura de software surge como una imperiosa necesidad para garantizar la eficacia y constituye una importante área de especialización. Esta materia cubre los aspectos más relevantes de la arquitectura de software, incluyendo atributos de calidad, patrones de diseño, middlewares[12], SOA[13], otras tendencias y paradigmas actuales. (9)

La Arquitectura de software consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. Además establece los fundamentos para que analistas, diseñadores, programadores, entre otros, trabajen en una misma línea que permita alcanzar los objetivos propuestos, cubriendo todas las necesidades.

Una arquitectura de software se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solo se tienen en cuenta los de tipo funcional, sino también otros como la adaptabilidad, flexibilidad e interacción con otros sistemas. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información.

La arquitectura de software, está muy vinculada al diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma

[12] Software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas.

[13] Arquitectura orientada a servicio (*Service Oriented Architecture*).

adecuada para satisfacer los requerimientos de desempeño de un sistema, así como los requerimientos no funcionales.

Una de las definiciones más aplicadas es la ofrecida en el documento IEEE STD 1471-2000 (*Software Engineering Standards Committee of the IEEE Computer Society, 2000*): “La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.” (10)

1.3.2 Estilos Arquitectónicos.

Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante composición de los estilos fundamentales. Los estilos conjugan elementos, componentes, conectores, restricciones y configuraciones. La descripción de un estilo se puede formular en lenguaje natural o en diagramas, pero lo mejor es hacerlo en un lenguaje de descripción arquitectónica o en lenguajes formales de especificación.

A diferencia de los patrones de diseño, que son centenares, los estilos se ordenan en seis o siete clases fundamentales y unos veinte ejemplares, como máximo. Es importante señalar el esfuerzo por incluir todas las formas existentes de aplicaciones en un conjunto de dimensiones tan modestas. Las arquitecturas complejas o compuestas resultan del agregado o la composición de estilos más básicos. (10)

1.3.3 Patrones.

Un patrón es una solución a un problema en un contexto, que codifica conocimiento específico acumulado por la experiencia en un dominio.

Alexander Christopher plantea: ¨ Cada Patrón es una regla de tres partes, la cual expresa una relación entre un cierto contexto, un problema y una solución.¨ (11)

Craig Larman define: “Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos sobre cómo aplicarlo en nuevas situaciones, o sea, un patrón es una descripción

de un problema bien conocido que suele incluir: descripción, escenario de uso, solución concreta, consecuencias de utilizar el patrón, ejemplos de implementación y lista de patrones relacionados." (12)

Características de los patrones:

- Proporcionan soluciones concretas y técnicas.
- Capturan soluciones, no sólo principios o estrategias abstractas.
- Se utilizan en situaciones frecuentes.
- Favorecen la reutilización de código.
- Documentan experiencias de diseño existentes y bien probadas.
- Proveen un vocabulario común y comprensible.
- Son una forma de documentar la arquitectura del software.

Patrones de Arquitectura.

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular recurrente que surge en un cierto contexto. Este esquema se especifica describiendo los componentes, con sus responsabilidades, relaciones, y las formas en que colaboran.

Patrón Modelo-Vista-Controlador (MVC).

Patrón MVC (Modelo-Vista-Controlador) se encarga de separar interfaces, fue creado con Smalltalk-80[14].

El modelo es la capa del dominio, la vista es la capa de presentación y el controlador son los objetos de flujo de trabajo.

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 componentes distintos. El patrón de llamada y retorno MVC (según CMU[15]), se

[14] Lenguaje de programación reflexivo orientado a objetos, de tipo dinámico.

ve frecuentemente en aplicaciones web, donde la vista es la página HTML[16] y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Descripción del patrón:

Modelo: representa la información con la que trabaja la aplicación y se encarga de acceder a los datos.

Vista: transforma la información obtenida por el modelo en las páginas web a las que acceden los usuarios.

Controlador: es el encargado de coordinar todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista.

Patrones de Diseño.

Los patrones de diseño son fundamentales para buscar soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Los patrones de diseño son descripciones de clases cuyas instancias colaboran entre sí y brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares.

Un patrón de diseño es una solución a un problema de diseño y para ser considerada patrón, debe poseer ciertas características:

- Se debe haber comprobado su efectividad, dándole solución a problemas similares con anterioridad.
- Debe ser reusable, es decir, aplicable a diversos problemas de diseño en variadas circunstancias.

Los patrones de diseño se clasifican en:

---

[15] Universidad Carnegie Mellon (en inglés: *Carnegie Mellon University, CMU*), ubicada en la ciudad de Pittsburgh (Pensilvania) y es uno de los más destacados centros de investigación superior de los Estados Unidos en el área de informática y robótica.

[16] *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto).

Fundamentales: patrones de diseño básicos, ya que son muy usados para la representación de otros patrones.

Creación: abstraen comportamientos referentes a la forma en que se deben crear los objetos.

Descomposición: los patrones de descomposición proveen ayuda para descomponer actores y conceptos complejos en múltiples clases.

Estructurales: describen las formas comunes en que distintos tipos de objetos pueden ser organizados para trabajar y colaborar entre ellos.

Comportamiento: se utilizan para administrar y combinar ciertos comportamientos.

Patrones GRASP.

***General Responsibility Assignment Software Patterns* (GRASP):** son patrones generales de software empleados para asignar responsabilidades. Se considera que más que patrones, son un conjunto de buenas prácticas de aplicación recomendables en el diseño de software, es decir, describen los principios fundamentales del diseño de objetos para la asignación de responsabilidades. Es una solución a un problema de diseño no trivial que es efectiva y reusable. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.

A continuación se presentan algunos de ellos:

Experto: la responsabilidad de realizar una labor es de la clase que tiene o puede tener los atributos.

Controlador: despacha operaciones. Permite asignar la responsabilidad de manejar los mensajes eventos del sistema a una clase que puede representar el sistema total o la organización.

Alta cohesión: indica cuán relacionadas están las responsabilidades de una clase. Es un patrón evaluativo: entre más alta cohesión presenta, más fácil de entender, de cambiar y reutilizar.

Bajo acoplamiento: permite que el diseño de clases sea más independiente. Reduce el impacto de los cambios y aumenta la reutilización.

Creador: un objeto tiene la responsabilidad de crear objetos de una clase determinada.

Patrones GoF.

Gang-of-Four traducido al español significa ¨Pandilla de los Cuatro¨. Describe clases y objetos que se comunican entre sí, y se adapta para resolver un problema general de diseño en un contexto particular.

Estos patrones de acuerdo a su **propósito** se clasifican en:

- Creacionales.
- Estructurales.
- Comportamiento.

De acuerdo a su **ámbito** se clasifican en:

- Clases: relaciones estáticas entre clases.
- Objetos: relaciones dinámicas entre objetos. (13)

En el libro "*Design Patterns: Elements of Reusable Object Oriented Software*" (Gama 1995) se definieron una serie de patrones. A continuación se listan algunos de ellos:

- **Fábrica**: su propósito es crear objetos, permitiendo al sistema identificar que clase se debe instanciar en tiempo de ejecución.
- **Instancia única**: garantiza que una clase solo tenga una instancia y proporciona un punto de acceso global a la misma.
- **Comando**: encapsula una petición en un objeto, permitiendo así entre otras cosas parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las mismas.
- **Registro**: permite compartir datos y objetos en la aplicación sin hacer uso de variables globales.
- **Adaptador**: convierte la interfaz de una clase en otra distinta, que es la que esperan los clientes.

- **Compuesto**: combina objetos en estructuras de árboles para representar jerarquías de parte - todo.
- **Decorador**: añade dinámicamente nuevas responsabilidades a un objeto.
- **Proxy**: proporciona un sustituto o representante de otro objeto para controlar el acceso a este.
- **Mediador**: define un objeto que encapsula cómo interactúan un conjunto de objetos. (14)

Métricas para validar el diseño.

Las métricas de calidad describen muchos y variados casos de medición, siendo una métrica una medida estadística que se aplica a todos los aspectos de calidad de software, los cuales deben ser medidos desde diferentes puntos de vista como el análisis, construcción, funcional, documentación, métodos, proceso, usuario, entre otros. Se caracterizan por ser simples y fáciles de calcular, empírica e intuitivamente persuasivas, consistentes y objetivas, independientes del lenguaje de programación.

Las métricas para diseño proporcionan al diseñador una mejor visión interna, ayudan a que el diseño evolucione a un nivel superior de calidad. Deben permitir comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado.

Métricas orientadas a clases.

Una clase es la unidad principal de todo sistema orientado a objeto (OO). Por lo que las medidas y métricas para una clase individual, la jerarquía de clases, y las colaboraciones de clases resultan sumamente valiosas para un ingeniero de software que necesite estimar la calidad de un diseño.

Métricas propuestas por Lorenz y Kidd[17]

Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas al tamaño para las clases se centran en el recuento de atributos y operaciones para cada clase individual, y los valores promedio para el sistema orientado a objetos como un todo. Las métricas basadas en la herencia se enfocan en la forma en que las

[17] *Lorenz, M. et al, 1994] Lorenz, M. & Kidd, J. "Object-Oriented Software Metrics".* Prentice Hall. 1994. Texto recomendado para introducirse en la medición de atributos de entidades desarrolladas con metodología orientada a objetos.

operaciones se reutilizan en la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y los aspectos orientados al código, mientras que las métricas orientadas a valores externos, examinan el acoplamiento y la reutilización.

Familia de métricas propuestas por Chidamber & Kemerer[18].

Los conjuntos de métricas propuestos por Chidamber y Kemerer han sido uno de los más referenciados, las que son conocidas normalmente como la serie de métricas CK. Chidamber y Kemerer proponen seis métricas, siendo la métrica Número de Descendientes la más completa, que permite determinar el tamaño, el acoplamiento y la cohesión, a diferencia del resto de ellas, que permiten obtener estas mismas características por separado.

- Métodos ponderados por clase (MPC): Tamaño y complejidad.
- Profundidad árbol de herencia (PAH): Tamaño.
- Número de descendientes (NDD): Tamaño, acoplamiento y cohesión.
- Acoplamiento entre clases (ACO): Acoplamiento.
- Respuesta para una clase (RPC): Comunicación y complejidad.
- Carencia de cohesión en los métodos (CCM): Cohesión interna (15)

1.4 Metodología, lenguajes y herramientas de modelado utilizadas para el desarrollo.

En la Universidad de las Ciencias Informáticas 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 de Informatización de la Gestión de Entidades (CEIGE), específicamente en el Departamento de Soluciones para la Aduana se encuentra desarrollándose el producto GINA que constituye la nueva solución integrada para las operaciones aduanales.

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

[18] Shyam R. Chidamber y Chris F. Kemerer enunciaron en el año 1994 un total de seis métricas para la medición de la calidad en el desarrollo de aplicaciones orientadas a objetos.

1.4.1 Metodología de desarrollo de software RUP.

RUP (del inglés: ***Rational Unified Process***), es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. (16)

RUP es un conjunto de metodologías adaptables al contexto a todas las necesidades que pueda tener cada entidad. RUP, como proceso de desarrollo de software define **quién** hace **qué**, **cómo** y **cuándo**, representando esto a través de cuatro elementos:

Los trabajadores que responden a la pregunta ¿Quién?: se definen los comportamientos y las responsabilidades de un individuo o grupo de estos, sistemas automatizados o máquinas, que trabajan en equipo, realizando las actividades y siendo propietarios de los elementos.

Los artefactos que responden a la pregunta ¿Qué?: son productos tangibles del proyecto, ya sean elementos dentro del modelo o código fuente, que son producidos, modificados y usados por las actividades.

Las actividades que responden a la pregunta ¿Cómo?: tarea con un claro objetivo, realizada por un trabajador y manipula elementos.

Los flujos de trabajo de las disciplinas que responde a la pregunta ¿Cuándo?: es la secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

Ciclo de vida

El ciclo de vida RUP fue creado ensamblando los elementos en secuencias semi-ordenadas, y organiza las tareas en fases e iteraciones. Divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones, haciéndose en cada una, mayor o menor hincapié en las distintas actividades.

Inicio: en esta fase las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de requisitos, además es donde se realiza un plan de fases, se identifican los principales casos de uso, los riesgos y se define el alcance del proyecto.

Elaboración: en esta segunda fase se lleva a cabo la construcción del producto por medio de una serie de iteraciones, las cuales se orientan al desarrollo de la línea base de arquitectura, abarcando el flujo de trabajo requerimientos, el modelo del negocio, análisis, diseño y una parte de la implementación. En esta etapa se realiza un plan de proyecto, se tienen en cuenta los casos de uso y se eliminan los riesgos.

Construcción: en esta fase se lleva a cabo la construcción del producto por medio de una serie de iteraciones. Para cada iteración se seleccionan algunos casos de uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Es donde se concentra la elaboración de un producto totalmente operativo, eficiente y el manual de usuario.

Transición: en esta última fase del proceso de desarrollo se hace entrega del producto al cliente, se instala y se realiza la adecuada capacitación a los usuarios que lo van a emplear. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

Flujos de trabajo:

• **Modelado del negocio:** describe los procesos del negocio e identifica los trabajadores que participarán en el proceso y qué actividades se desarrollarán.

• **Requerimientos:** define qué debe realizar el sistema, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.

• **Análisis y diseño:** describe cómo el sistema será desarrollado a partir de la funcionalidad prevista y los requisitos, indicando lo que debe ser programado.

• **Implementación:** define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes, así como la estructura de capas de la aplicación.

• **Prueba (Testeo):** busca los defectos y errores a lo largo del ciclo de vida.

• **Instalación:** produce la liberación del producto y realiza actividades (instalación, asistencia a usuarios, entre otros.) para entregar el software a los usuarios finales.

• **Administración del proyecto:** involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

• **Administración de configuración y cambios:** describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, entre otros.

• **Ambiente:** contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Adecuaciones de RUP en el Departamento de Soluciones para la Aduana.

Actualmente en el Departamento de Soluciones para la Aduana se usa como metodología de desarrollo de Software RUP con adecuaciones, ya que es un proceso de desarrollo de software configurable que puede ser adaptado a las características propias del proyecto. En vez de la utilización del modelo de casos de uso del negocio y los diagramas de actividades, se propone el uso del Modelo de Procesos de Negocio con la notación BPMN (Notación para el Modelado de Procesos de Negocio). Para los conceptos del negocio, en vez del Modelo de Objeto del Negocio, se propone confeccionar el Modelo Conceptual. Se debe corresponder cada requisito funcional como un caso de uso. (17)

Se sustituye los diagramas de secuencia tradicionales por los diagramas de secuencia orientada a actividades, representado por calles, comentarios, actividades y sus relaciones, que permiten modelar y representar más detalladamente el flujo de las operaciones. Se encuentra especificado en el modelo de diseño del proyecto Aduana.

1.4.2 Notación para el modelado de procesos de negocio: BPMN

El modelado del negocio se realiza mediante *Business Process Modeling Notation* (BPMN o Notación para el Modelado de Procesos de Negocio), que es una notación gráfica que describe la lógica de los pasos de un proceso de negocio, siendo esta notación diseñada para coordinar la secuencia de los procesos, y la comunicación que fluyen entre los participantes de las diferentes actividades. El modelado con BPMN es muy importante debido a:

- Es un estándar internacional de modelado de procesos aceptado por la comunidad.
- Es independiente de cualquier metodología de modelado de procesos.

- Permite modelar los procesos de una manera unificada y estandarizada permitiendo un entendimiento a todas las personas de una organización.

Los elementos gráficos en BPMN, están clasificados en cuatro categorías:

Objetos de flujo: son los principales elementos gráficos que definen el comportamiento de los procesos. Dentro de los cuales se encuentran:

- Eventos: sucede en el transcurso de un proceso de negocio, afectando el flujo del proceso, y teniendo una causa y un resultado.
- Actividades: representan el trabajo ejecutado dentro de un proceso de negocio. Las actividades pueden ser tareas o subprocesos.
- Compuertas: elementos del modelado utilizados para controlar la divergencia y la convergencia del flujo. Existen cinco tipos de compuertas: (exclusiva, basada en eventos, paralela, inclusiva y compleja).

Objetos de conexión: son elementos usados para conectar dos objetos del flujo dentro de un proceso.

Existen tres tipos:

- Línea de secuencia.
- Asociaciones.
- Línea de mensaje.

Canales: elementos empleados para organizar las actividades de flujo en variadas categorías visuales, que pueden representar roles, responsabilidades o las áreas funcionales.

Artefactos: se utilizan para proveer información adicional sobre un proceso. Existen tres tipos:

- Objetos de datos.
- Grupos.

- Anotaciones.

1.4.3 Lenguaje de modelado UML 2.0

El Lenguaje Unificado de Modelado es un lenguaje que permite especificar, construir, visualizar y documentar los artefactos de un sistema orientado a objetos, describiendo la semántica esencial del significado de estos diagramas y símbolos. (18)

El UML puede usarse para modelar distintos tipos de sistemas, ya sean de software, de hardware, y organizaciones del mundo real, ofreciendo nueve diagramas en los cuales modelar sistemas:

• **Diagramas de Casos de Uso**: permite modelar los procesos.

• **Diagramas de Secuencia**: se emplea para modelar el paso de mensajes entre objetos.

• **Diagramas de Colaboración**: se utiliza para modelar interacciones entre objetos.

• **Diagramas de Estado:** es para modelar el comportamiento de los objetos en el sistema.

• **Diagramas de Actividad:** se usa para modelar el comportamiento de los Casos de Uso, objetos u operaciones.

• **Diagramas de Clases:** permite modelar la estructura estática de las clases en el sistema.

• **Diagramas de Objetos:** es utilizado para modelar la estructura estática de los objetos en el sistema.

• **Diagramas de Componentes:** con él se puede modelar componentes.

• **Diagramas de Implementación:** empleado para modelar la distribución del sistema.

1.4.4 Herramienta CASE para el modelado: Visual Paradigm for UM 6.4

Visual Paradigm es una herramienta profesional que permite crear diagramas UML, utilizando un lenguaje estándar para todo el equipo de desarrollo, que facilita la comunicación entre éstos. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. También ayuda a construir rápidamente, aplicaciones de calidad siendo mejores y con un

coste menor. Permite modelar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (19)

Visual Paradigm posee una interfaz de usuario fácil de utilizar y está compuesta por varios productos como: Visual Paradigm for UML, Business Process Visual ARCHITECT, Teamwork Server, entre otros.

1.4.5 Lenguaje de programación: PHP 5.3

PHP (*HyperText Preprocessor***)** es un lenguaje scripting [19] que permite generar dinámicamente los contenidos en un servidor web, principalmente el Apache. Se utiliza para crear páginas web dinámicas y permite la conectividad entre diversos servidores de base de datos. Se caracteriza por tener muy buena potencia, alto rendimiento, facilidad de aprendizaje y escasez de consumo de recursos. Es multiplataforma y libre, por lo que se presenta como una alternativa de fácil acceso para todos y permite las técnicas de programación orientada a objetos.

La versión utilizada de PHP es la 5.3 que se centra en mejorar la estabilidad e incluye varias ventajas:

- Mejoras de rendimiento.
- Manejo de excepciones.
- Mejoras con la implementación con Oracle. (20)

1.4.6 Marco de trabajo (*Framework*): *Symfony* 1.2.8

Symfony es un marco de trabajo que se utiliza para crear aplicaciones PHP, permitiendo aumentar la productividad y calidad de trabajo. Ha sido diseñado para optimizar el desarrollo de las aplicaciones web. Tiene variadas características como por ejemplo: separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. (21)

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

[19] Lenguaje de programación interpretado. Son interpretados, suelen ser de alto nivel y orientados a tareas sencillas. Por lo general son cortos y se desarrollan rápidamente.

completo a los aspectos específicos de cada aplicación. Symfony es muy escalable y seguro, ya que permite controlar hasta el último acceso a la información e incluye por defecto protección contra ataques XSS[20] y CSRF[21]. Es un marco de trabajo muy documentado y se publica bajo licencia MIT[22], con la que se puede desarrollar aplicaciones web comerciales, gratuitas y/o de software libre. Permite su integración con librerías desarrolladas por terceros y sigue la mayoría de mejores prácticas y patrones de diseño para la web. Symfony es compatible con la mayoría de gestores de bases de datos.

1.4.7 Interfaz de usuario: ExtJS 3.0

ExtJS es una librería Javascript que permite construir aplicaciones complejas en Internet. Esta librería incluye:

- Modelo de componentes extensibles.
- Componentes de interfaz de usuario personalizables.
- Un API[23] fácil de usar.
- Licencias de código abierto y comerciales.

Su empleo tiene grandes ventajas ya que permite crear aplicaciones complejas. Está diseñado para la creación de páginas web dinámicas del lado del cliente. Permite reutilizar y personalizar componentes, además es de gran ventaja la similitud de su aspecto con el de una aplicación de escritorio y utiliza componentes predefinidos. (22)

[20] Cross-site scripting: tipo de inseguridad informática o agujero de seguridad basado en la explotación de vulnerabilidades del sistema de validación de HTML incrustado.

[21] Cross-site request forgery (CSRF) o falsificación de petición en sitios cruzados, también conocido como XSRF.

[22] Licencia de software que ha empleado el Instituto Tecnológico de Massachusetts (MIT, *Massachusetts Institute of Technology*).

[23] Interfaz de programación de aplicaciones (del inglés *application programming interface*) conjunto de funciones y procedimientos, ofrece cierta biblioteca para ser utilizado por otro software.

1.4.8 Gestor de base de datos: Oracle 11g

Oracle (RDBMS o *Relational Data Base Management System*), es un sistema de gestión de base de datos relacional, desarrollado por Oracle Corporation. Es una herramienta cliente/servidor para la gestión de bases de datos. Considerado como uno de los sistemas de bases de datos más completos por su:

- Soporte de transacciones.
- Estabilidad.
- Soporte multiplataforma.

Entre las principales funcionalidades de Oracle se encuentran:

- Soporte de una gran cantidad de usuarios accediendo concurrentemente a los datos.
- Seguridad de acceso a los datos, restringiendo dicho acceso a las necesidades de cada usuario.
- Integridad referencial en su estructura de base de datos.
- Portabilidad y compatibilidad.
- Conectividad entre las aplicaciones de los clientes en sus puestos de trabajo y el servidor de base de datos Oracle (estructura cliente/servidor). (23)

CONCLUSIONES

En el capítulo se realiza un estudio de las soluciones extranjeras que se emplean para gestionar los procesos que se efectúan en los depósitos temporales de frontera, comprobándose que no existe una que satisfaga las necesidades de la AGR, ya que no cumplen con las legislaciones y normativas aduaneras cubanas vigentes para el funcionamiento y control de los depósitos temporales, además de ser privativas; y la solución cubana es inadecuada pues no gestiona la totalidad de los procesos de los depósitos temporales de frontera, y es incapaz de ajustarse a la nueva arquitectura del proyecto Aduana.

Se explican metodologías, lenguajes y herramientas que se deben emplear para desarrollar el software que le dé solución al problema presentado, garantizando una gestión más completa y eficiente de las operaciones que se realizan en dichos depósitos.

CAPÍTULO 2: DISEÑO E IMPLEMENTACION DEL SISTEMA

INTRODUCCIÓN

En el capítulo se detalla la modelación del sistema mediante el diseño y su validación, factores que permiten contribuir a una arquitectura estable, sólida y que soporten los requisitos definidos, siendo un punto de partida en las actividades de la implementación. Se muestran una serie de artefactos de ambos flujos de trabajo, patrones de diseño utilizados, el código fuente de algunas clases y pantallas de la aplicación.

2.1. Propuesta del sistema

Se propone el desarrollo del sistema usando una arquitectura cliente-servidor, que permite al cliente realizar peticiones al servidor y recibir respuestas del mismo, facilitando la integración distribuida del sistema en red, con los recursos, y aplicaciones que, definidos modularmente en los servidores, administran, ejecutan y atienden las solicitudes de los clientes. De esta manera permite una comunicación transparente entre los elementos que conforman la estructura.

Entre las principales características de la arquitectura Cliente/Servidor, se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo. (24)

El sistema propuesto se integra al producto GINA (Gestión Integral Aduanera) como un subsistema del mismo y permite su implementación y comunicación con otros subsistemas, a fin de lograr la integridad del producto final.

2.2. Requerimientos del sistema

Se precisa de un buen diseño que permita junto a la implementación del sistema, satisfacer un conjunto de requisitos definidos previamente, los cuales se detallan a continuación:

No	Requisito	Descripción
1	Gestionar lo real descargado.	Se registra, modifica y anula lo real descargado de una aeronave en el depósito temporal de frontera correspondiente.
2	Gestionar la recepción de mercancías.	Se registra, modifica y anula la recepción de mercancías provenientes de otro depósito.
3	Registrar la extracción de mercancías.	Se registra la extracción de mercancías efectuada en el depósito.
4	Confirmar la extracción de mercancías.	Se confirma la extracción de mercancías efectuada en el depósito.
5	Modificar la extracción de mercancías.	Se modifica la extracción de mercancías efectuada en el depósito.
6	Anular la extracción de mercancías.	Se anula la extracción de mercancías efectuada en el depósito.
7	Gestionar las averías en los bultos de importación.	Se registra, modifica y anula las averías producidas en los bultos de importación.
8	Gestionar el desagrupe de bultos.	Se registra, modifica y anula los resultados del desagrupe de mercancías en los bultos.
9	Gestionar la prórroga en mercancías de importación.	Se registra y anula la prórroga debidamente autorizada en mercancías de importación.
10	Declarar el abandono legal.	Se registran las mercancías declaradas en abandono legal.
11	Registrar Declaración de Abandono Legal.	Se registra el documento Declaración de Abandono Legal correspondiente a un abandono legal efectuado.
12	Anular el abandono legal.	Se anula una declaración de abandono legal registrada previamente.
13	Revocar el abandono legal.	Se registra la revocación del abandono, una vez presentada la Carta de Revocación de Abandono aprobada.

Tabla 1: *Requisitos funcionales del subsistema depósitos temporales.*

2.3. Patrones de diseño utilizados.

Cuando se diseña, muchas veces se presentan problemas que deben ser resueltos para que el software funcione correctamente. A lo largo de los años, los programadores han encontrado problemas en el diseño

de sus programas. De ahí surge la necesidad de utilizar patrones como solución a problemas en un contexto determinado.

Los **patrones de diseño** son un conjunto de soluciones a problemas que generalmente se encuentran en el diseño de un programa. Cada patrón explica cómo resolver un determinado problema, bajo determinadas circunstancias, y las ventajas y desventajas de su uso. (25)

Para la implementación con Symfony se utilizan varios patrones, situándolos en las capas del modelo y el controlador.

2.3.1. Patrones GRAPS implementados

Alta cohesión: cada elemento dentro del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos. Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.” (25)

Se evidencia en el diseño de las clases, agrupándolas por funcionalidades que son fácilmente reutilizables. Cada elemento del diseño realiza una labor única dentro del sistema. Un ejemplo es la clase Actions, formada por varias funcionalidades estrechamente relacionadas, siendo la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones e instanciar objetos.

Bajo acoplamiento: este patrón asigna la responsabilidad de mantener el control sobre el flujo de eventos del sistema, a clases específicas. Es muy utilizado para mantener organizadas todas las funcionalidades, posibilitando agrupar los procedimientos semejantes, para hacer menos engorroso el proceso de validación y la seguridad. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y recurre a ellas. (25)

La clase Actions hereda de sfActions únicamente para alcanzar un bajo acoplamiento. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen

asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

Creador: este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental es encontrar un creador que se conecte con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento". (25)

En la clase Action se encuentran las acciones definidas para el sistema y se ejecutan en cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que la clase Actions es "creador" de dichas entidades.

Experto: la responsabilidad de realizar una labor es de la clase que tiene o puede tener los atributos. Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. (25)

Es uno de los patrones más utilizados en Symfony. La librería Propel, para mapear la Base de Datos y para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y genera las clases con todas las funcionalidades comunes de las entidades. Las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan.

Controlador: es un evento generado por actores externos. Se asocian con operaciones del sistema, como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos. Normalmente un controlador delega en otros objetos el trabajo que se necesita hacer, coordina o controla la actividad. (25)

Se emplea la página deposito_dev.php, que se encarga de tramitar todas las peticiones que se realizan a través de ella para direccionarla al resto de las plantillas. Además todas las peticiones Web son manipuladas por un solo controlador frontal (sfActions), que es el punto de entrada único de toda la aplicación en un entorno determinado.

2.3.2. Patrones GOF implementados.

Instancia única (*singleton*): admite exactamente una instancia de una clase. Los objetos necesitan un único punto de acceso global.

Es el caso del controlador frontal, donde hay una llamada a la función sfContext::getInstance() que garantiza que siempre se acceda a la misma instancia.

Comando (*comand*): este patrón permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación, ni el receptor real de la misma. Para ello se encapsula la petición como un objeto. (25)

Se observa en la clase sfWebFrontController, en el método dispatch(). Esta clase está por defecto y es la encargada de establecer el módulo y la acción que se va a usar según la petición del usuario.

Fábrica (*factory*): su propósito es crear objetos, permitiendo al sistema identificar que clase se debe instanciar en tiempo de ejecución. Asegura que sólo exista una instancia de una clase específica en un sistema a desarrollar y la creación de un mecanismo de acceso global a dicha instancia.

Cuando el marco de trabajo necesita crear un nuevo objeto, por ejemplo, busca en la definición el nombre de la clase que se debe utilizar para esta tarea.

Registro (*registry)*: este patrón es muy útil para los desarrolladores en la Programación Orientada a Objetos, siendo un medio sencillo y eficiente de compartir datos y objetos en la aplicación sin la necesidad de preocuparse por conservar numerosos parámetros o hacer uso de variables globales.

Este patrón se aplica en la clase sfConfig, que es la encargada de acumular todas las variables de uso global en el sistema.

2.4. Modelo del diseño.

El diseño de sistemas tiene gran importancia en el desarrollo de una aplicación informática, posibilita la calidad y permite materializar con precisión los requerimientos del cliente. Es un conjunto de pasos que permiten al diseñador describir todos los aspectos del sistema a construir. El diseño debe proporcionar una completa idea de lo que es el software, es la última acción de la ingeniería correspondiente dentro de

la actividad del modelado, la cual establece una plataforma para la construcción (generación de código y prueba).

2.4.1. Diagrama de paquetes

El objetivo de este diagrama es obtener una visión del sistema, organizándolo en subsistemas y agrupando los elementos y las relaciones de dependencia entre ellos.
Estos diagramas contienen dos tipos de elementos:

Paquetes: un paquete es una agrupación de elementos, bien sea casos de uso, clases o componentes. Los paquetes pueden contener a su vez otros paquetes anidados que en última instancia contendrán alguno de los elementos anteriores.

Dependencias entre paquetes: existe una dependencia cuando un elemento de un paquete requiere de otro que pertenece a un paquete distinto.

El sistema propuesto es un subsistema del producto GINA (Gestión Integral Aduanera), el cuál presenta interacción con otros subsistemas. A continuación se muestra el diagrama de paquetes del subsistema depósitos temporales (Figura 1); el mismo relaciona diferentes paquetes: WEB para la capa de presentación, el CONTROLADOR y el LIB contiene los formularios y clases del negocio. Se evidencia la interacción con el subsistema TC (Tablas de Control), que permite administrar las clases comunes y utilizadas por los el resto de ellos. El subsistema PERSONA provee todo lo referente a las personas relacionadas con la Aduana General de la República. Una integración importante es con las librerías de Symfony y con el núcleo del marco de trabajo.

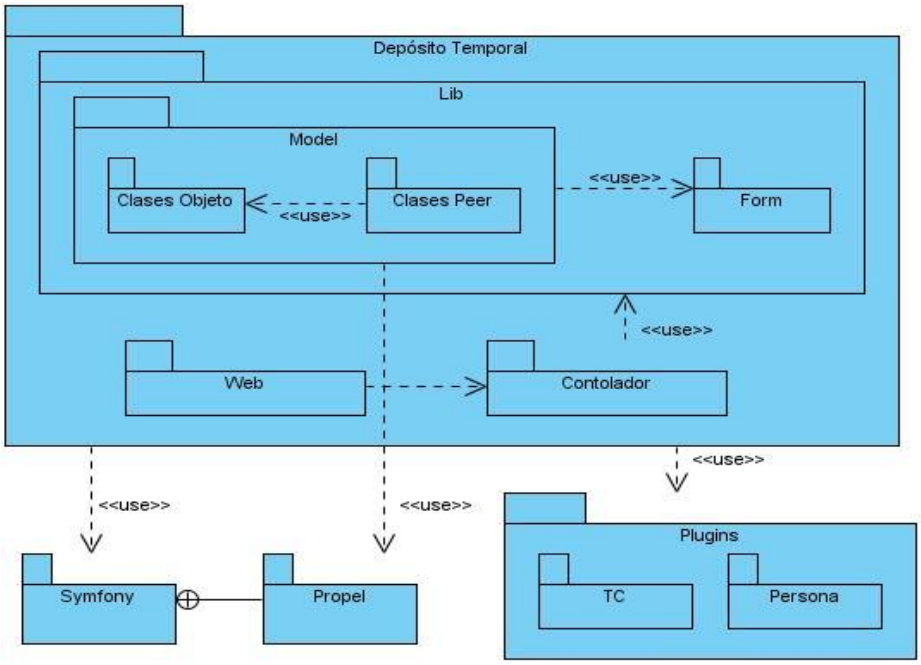

Figura 1. *Diagrama de paquetes del subsistema depósitos temporales de frontera.*

2.4.2. Diagramas de clases del diseño

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación, así como la estructura de un sistema, mostrando sus clases y las relaciones entre sus atributos. Contiene las definiciones de las entidades del software en vez de conceptos del mundo real, así como la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de los atributos.
- Navegabilidad.
- Dependencias. (24)

A continuación se muestra el diagrama de clases del diseño del subsistema depósitos temporales de frontera. (Figura 2)

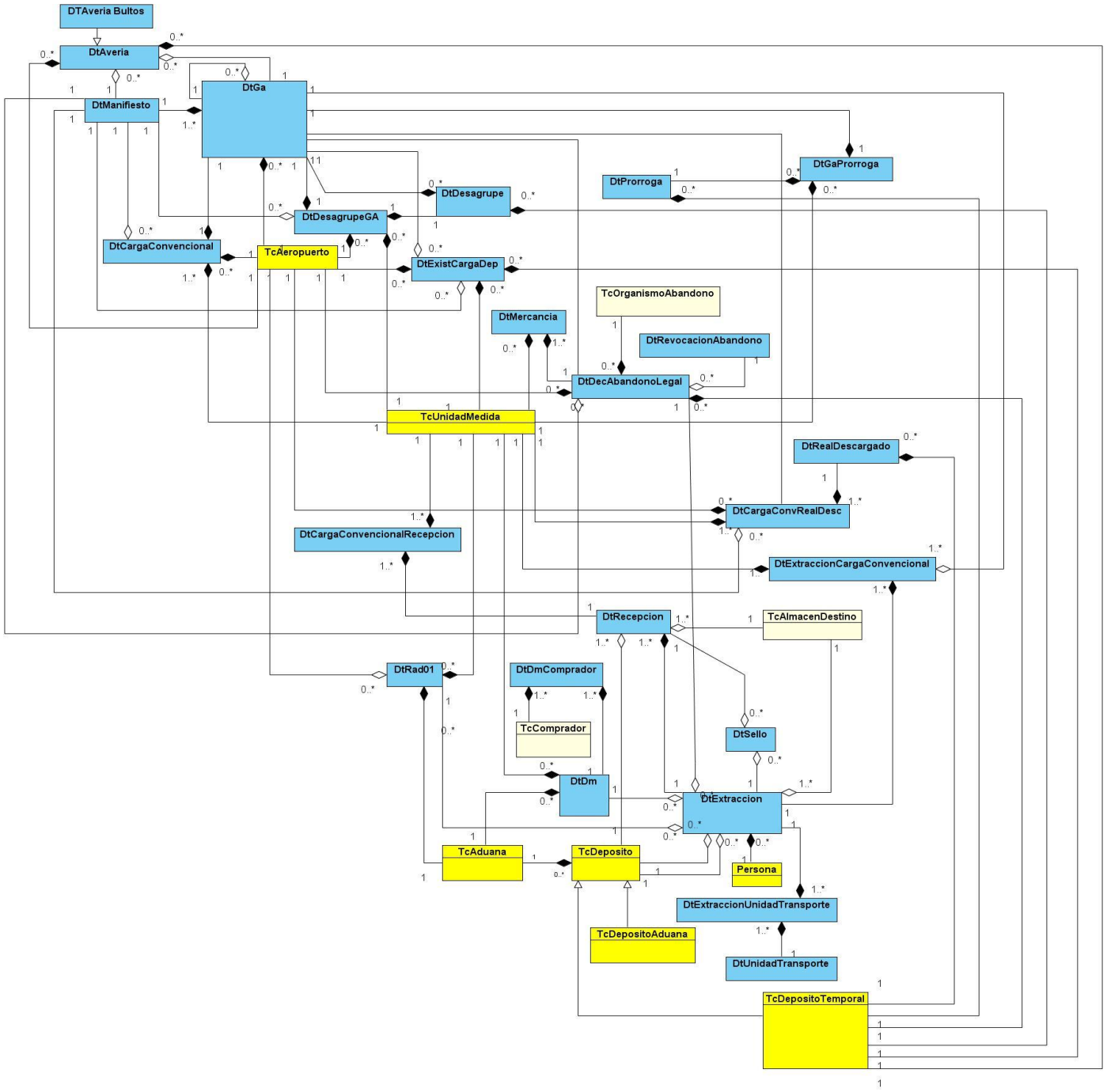

Figura 2. *Diagrama de clases del diseño del subsistema depósitos temporales de frontera.*

La utilización del marco de trabajo Symfony y el uso de las librerías ExtJS permiten desarrollar un diseño con una estructura similar para varios casos de uso. La página servidora Accions.class se encargará de las peticiones que realiza el controlador frontal; luego estas se redireccionan al Layout. El Layout carga en el cuerpo la plantilla (en el presente caso por ser la plantilla una página en blanco no se muestra en el diagrama), luego se construye la página cliente que importa todas las interfaces de usuario del subsistema. Estas interfaces están definidas en el fichero de configuración view.yml y se encuentran representadas dentro del paquete Interfaces JS.

A continuación se muestra el diagrama de clases del diseño (Figura 3) que describe el proceso de gestión de descarga de bultos de importación en los depósitos temporales de frontera. Se presenta una relación de agregación entre la página cliente (CP_PRINCIPAL) y el formulario (ResultadoDescarga), que contiene todos los componentes necesarios para la solución. Luego los datos registrados son enviados al servidor y recibidos por las funciones de la clase Action.class, la cual usa las clases del paquete de Acceso a Datos y la clase SfAuxiliarTcDepositosTemporales.class. Esta última se encarga de acceder al Núcleo GINA, y es accedida desde el Action o por las clases del modelo. Todo este proceso permite registrar y obtener toda la información necesaria.

Núcleo GINA: Representa la parte del GINA dedicada al intercambio de información entre subsistemas, los cuales son expertos en el manejo de información específica. El subsistema depósitos temporales posee intercambio de datos con el subsistema que maneja los datos propios de las personas así como las tablas de control, encargadas de proveer servicios y administrar clases usadas por los demás subsistemas.

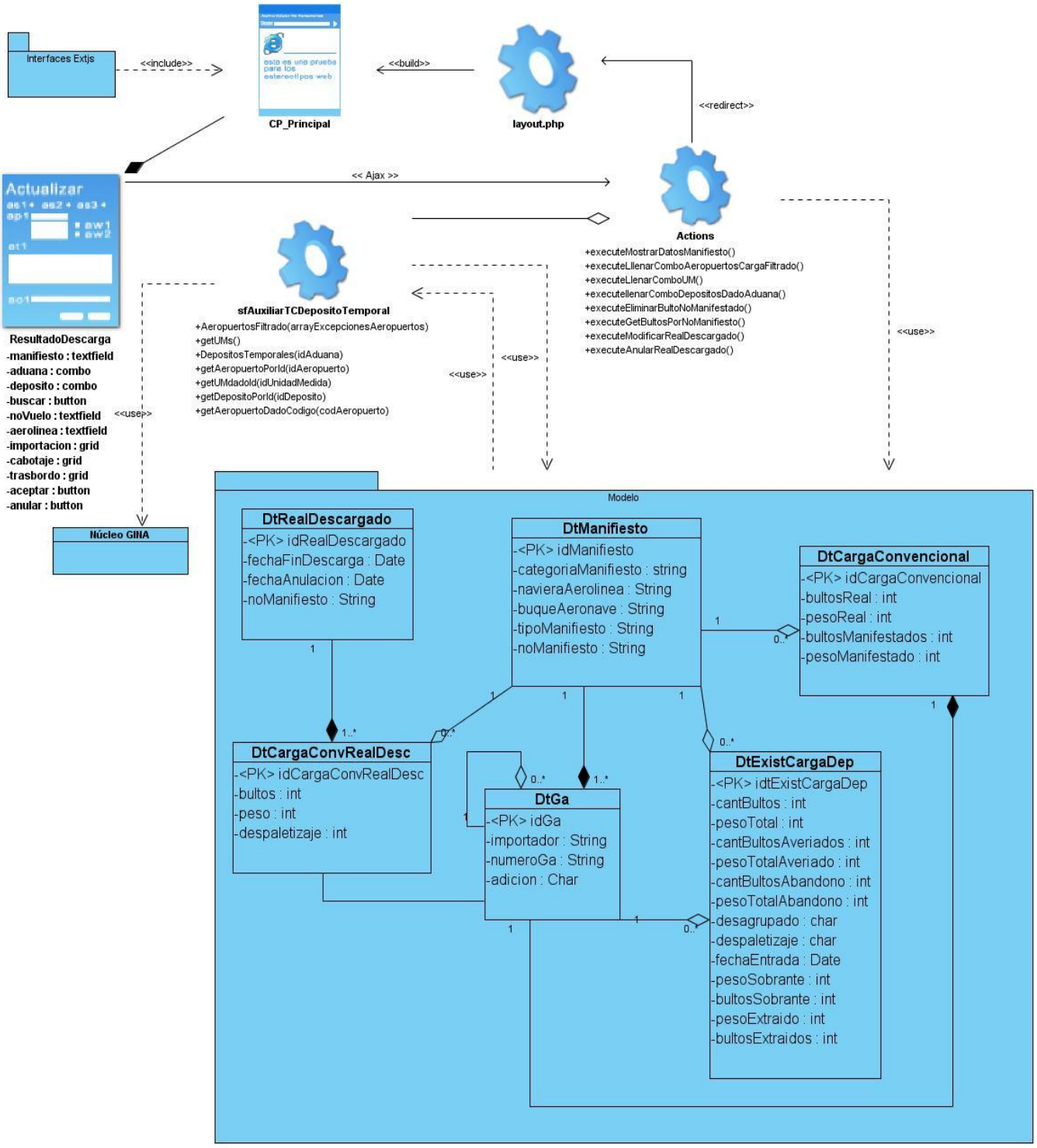

Figura 3. *Diagrama de clases con estereotipos web del requisito funcional gestionar real descargado.*

2.4.3. Diagrama de interacción del diseño

Los diagramas de Interacción del diseño modelan el comportamiento dinámico del sistema, el flujo de control de una operación y describen la interacción entre objetos.

Existen dos tipos de diagramas de interacción:

- Diagramas de Secuencia (dimensión temporal).

- Diagramas de Colaboración (dimensión estructural).

Diagramas de secuencia del diseño.

Un diagrama de secuencia muestra una interacción ordenada según la secuencia temporal de eventos, los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes. Tiene como objetivo describir el comportamiento dinámico del sistema de información, haciendo énfasis en la secuencia de los mensajes intercambiados por los objetos. Se pueden colocar etiquetas que pueden ser descripciones de acciones, restricciones de tiempo, entre otros.

Diagrama de secuencia orientado a actividades del negocio

En un diagrama de secuencia tradicional cuando el cumplimiento de una condición determina comportamientos mutuamente excluyentes, se generan distintos mensajes según la condición cumplida. Esto indica que se deben realizar varios diagramas, uno por cada condición. Otra de sus desventajas es que una representación de un diagrama de secuencia demasiado largo, puede ser difícilmente entendida por alguien ajeno al sistema.

El departamento de Soluciones para la Aduana evita el uso de los diagramas de secuencia tradicional y propone una nueva solución: los diagramas de secuencia orientada a actividades. Con este tipo de diagramas se definen las relaciones entre las clases y los usuarios así como el flujo de cada una de las actividades. Esta nueva forma de modelar el flujo de las operaciones, se representa por calles, comentarios, actividades y sus relaciones más explícitamente. Posee grandes ventajas y facilidades para el posterior desarrollo de la implementación, además de garantizar mayor calidad y entendimiento a los programadores.

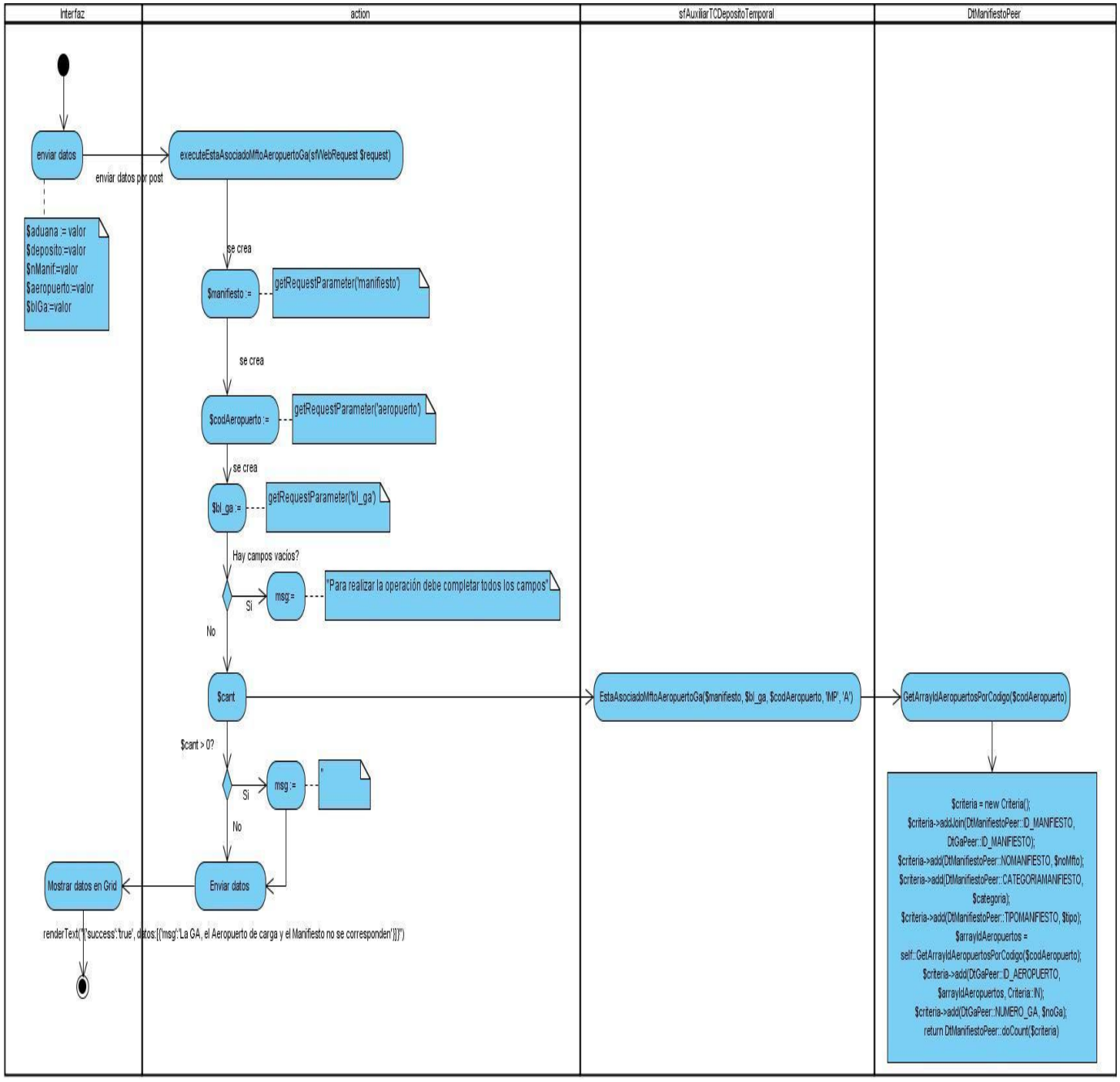

Figura 4. *Diagrama de actividades con calles de la funcionalidad buscar averías, en el requisito funcional gestionar las averías.*

El anterior diagrama de secuencia orientado a actividades (Figura 4), del requisito funcional gestionar averías, incluye las actividades y llamadas a funcionalidades, así como bifurcaciones, con un total de 4

calles que representan cada uno de los objetos que involucra la funcionalidad. A continuación parte de un flujo de actividades de la misma funcionalidad buscar averías, donde se evidencian llamadas a funcionalidades, bifurcaciones y captura de errores.

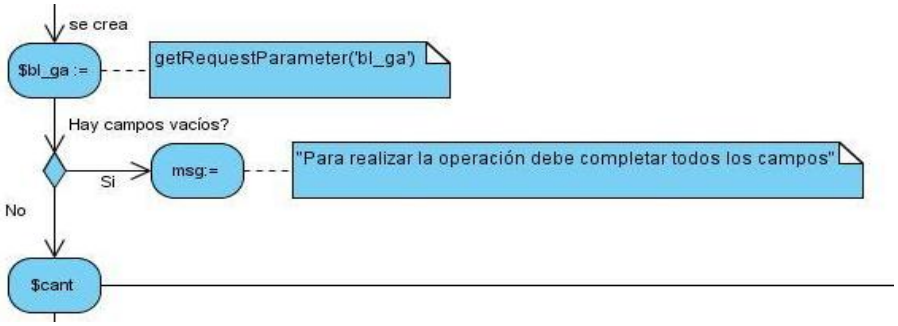

Figura 5. *Fragmento ampliado del diagrama de secuencia orientado a actividades, de la funcionalidad buscar averías.*

2.4.4. Diseño del modelo de datos

El modelado y diseño de la base de datos o esquema de base de datos es un factor importante para el desarrollo de un sistema. El objetivo del diseño es generar un conjunto de esquemas de relaciones que permitan almacenar la información con un mínimo de redundancia, pero que a la vez faciliten la recuperación de la información.

Modelo de datos:

Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos. Básicamente consiste en una descripción de algo conocido como “contenedor de datos”, así como de los métodos para almacenar y recuperar información.

Un modelo de datos consiste en:

- Objetos (entidades que existen y se manipulan).
- Atributos (Características básicas de estos objetos).
- Relaciones (forma en que se enlazan los distintos objetos entre sí).

A continuación se muestra el Modelo Físico de Datos, del subsistema depósitos temporales de frontera. (Figura 6)

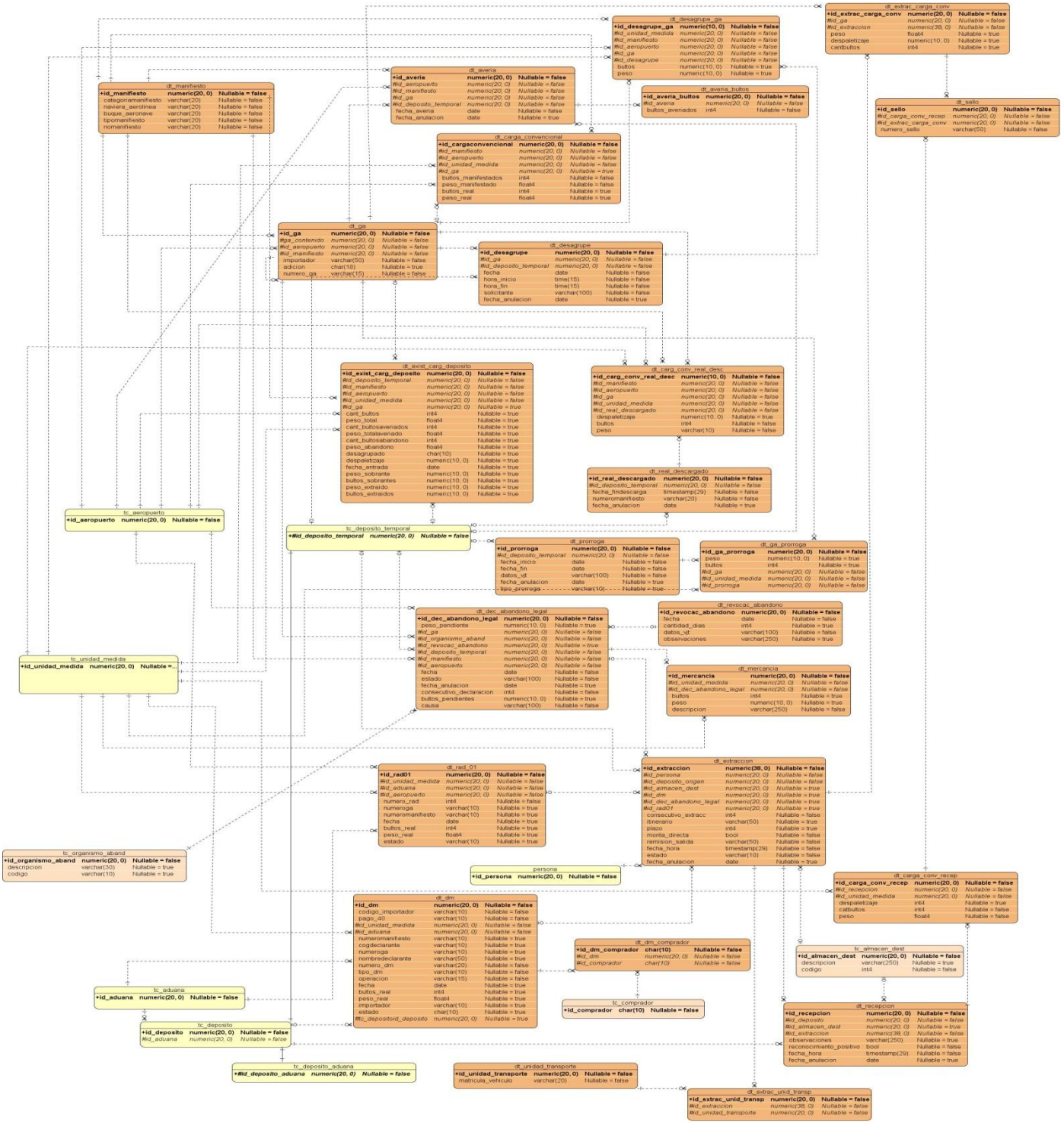

Figura 6. *Modelo de datos del subsistema depósitos temporales de frontera.*

2.5. Validación del diseño.

La medición es esencial para cualquier disciplina, y la ingeniería del software no es una excepción. Medir permite tener una visión más clara y profunda, y proporciona un mecanismo para la evaluación objetiva (24).

A continuación se aplican un conjunto de métricas de diseño orientado a objeto y se analizan los resultados para determinar la calidad del diseño planteado.

Métricas orientadas a clases.

Se sabe que la clase es la unidad principal de todo sistema orientado a objetos. Esto implica que las medidas y métricas para una clase individual, la jerarquía y las colaboraciones sean sumamente valiosas para un ingeniero de software que tenga que estimar la calidad de un diseño.

Métricas propuestas por Lorenz y Kidd.

Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones para una clase individual, y promedian los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y asuntos relacionados con el código, y las métricas orientadas a valores externos examinan el acoplamiento y la reutilización. (26)

Del conjunto de métricas planteadas por Lorenz y Kidd se aplicaron al diseño propuesto:

- **Tamaño operacional de clase (TOC)**
- **Relaciones entre clases (RC)**

Tamaño operacional de clase (TOC): esta métrica es muy usada por diseñadores de software, con una amplia documentación y literatura, fácil de calcular y bastante efectiva. Se basa en el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

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

***Tabla 2**: Evaluación de los atributos de calidad, métrica tamaño operacional de clase.*

Para un total de 30 clases que pertenecen al diseño, se obtuvieron tras aplicar la métrica, valores promedio de 9,63 operaciones por clase (Tabla 3).

Total de Clases	Promedio de operaciones
30	9,63

***Tabla 3.** Cantidad de clases y promedio de operaciones.*

Para evaluar las métricas son necesarios los valores de los umbrales para los parámetros de calidad. Algunos especialistas plantean umbrales para esta métrica basándose en el promedio de operaciones por clases obtenidos, estos valores fueron los aplicados en el diseño de este sistema. (Tabla 4).

	Categoría	Criterio	
Responsabilidad	Baja	<=Promedio	<= 9,63
	Media	>Promedio y <= 2*Promedio	>9,63 y <=19.26
	Alta	>2*Promedio	>19.26
Complejidad	Baja	<=Promedio	<= 9,63
	Media	>Promedio y <= 2*Promedio	>9,63 y <=19.26
	Alta	>2*Promedio	>19.26
Reutilización	Baja	>2*Promedio	>19.26
	Media	>Promedio y <= 2*Promedio	>9,63 y <=19.26

	Alta	<=Promedio	<= 9,63

Tabla 4. *Valores de los umbrales para la métrica: tamaño operacional de clase.*

Al representar las clases en correspondencia con el umbral especificado, se detectó que 25 de ellas son de tamaño pequeño, 1 mediana y 4 grandes. (Figura 7)

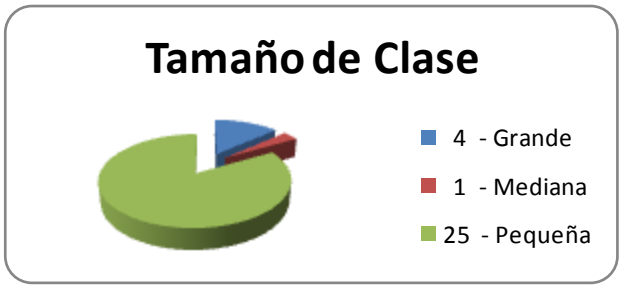

Figura 7: *Tamaño de clases del subsistema depósitos temporales de frontera.*

A continuación se muestran los resultados de la evaluación de la métrica.

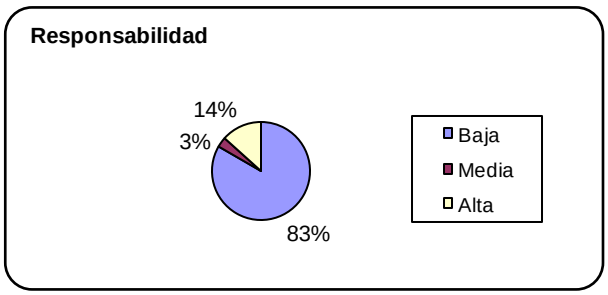

Figura 8. *Representación de las clases según la responsabilidad.*

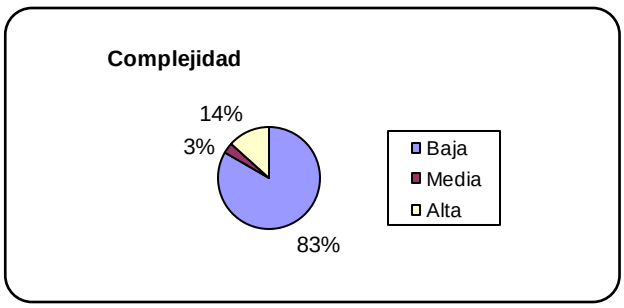

Figura 9. *Representación de las clases según la complejidad.*

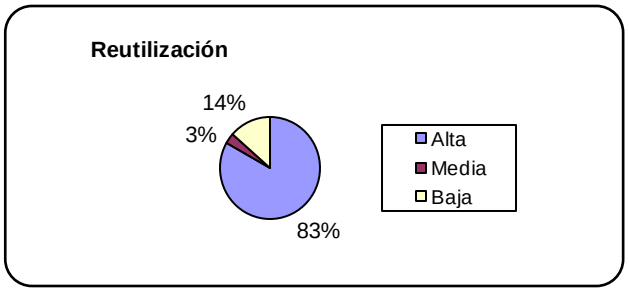

Figura 10. *Representación de las clases según la complejidad.*

Tras un análisis de los resultados arrojados por la evaluación bajo los instrumentos de medición de la métrica TOC, se demuestra que se alcanzaron buenos valores para cada uno de los atributos de calidad evaluados, puesto que, como se puede observar, el 83% de las clases del software contienen un número menor que el promedio de procedimientos por clases, lo cual influye positivamente en el hecho de que predomine una responsabilidad baja de las clases en un 83%, y hace que carezcan de mucha complejidad, por lo que se tornan mucho más reutilizables. Solamente un 14% de las clases tienen pocas posibilidades de reutilización y una gran complejidad de implementación. Estos valores demuestran que los indicadores de reutilización, complejidad y responsabilidad no se ven afectados.

Relaciones entre clases (RC): está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Atributo de Calidad	Modo en que lo afecta
Acoplamiento	El aumento del RC implica el aumento del acoplamiento de la clase.
Complejidad de mantenimiento	El aumento del RC implica el aumento de la complejidad del mantenimiento de la clase.
Reutilización	El aumento del RC implica la disminución en el grado de reutilización de la clase.

Tabla 5*: Evaluación de los atributos de calidad, métrica relaciones entre clases.*

Para un total de 30 clases que pertenecen al diseño, se obtuvieron tras aplicar la métrica, valores promedio de 2,66 asociaciones de uso por clase. (Tabla 6)

Total de Clases	Promedio asociaciones de uso
30	3,66

Tabla 6. *Cantidad de clases y promedio asociaciones de uso.*

Los valores de los umbrales para esta métrica que se basan en el promedio de asociaciones de uso por clases, estos valores fueron los aplicados en el diseño de este sistema.

	Categoría	**Criterio**	
Acoplamiento	Ninguno	0	
	Bajo	1	
	Medio	2	
	Alto	>2	
Complejidad de mantenimiento	Bajo	<=Promedio	<=3,66
	Medio	> Promedio y <= 2*Promedio	>3,66 y <=7,32
	Alto	> 2*Promedio	>7,32
Reutilización	Bajo	>2*Promedio	>7,32
	Medio	> Promedio <= 2*Promedio	>3,66 y <=7,32
	Alto	<=Promedio	<=3,66

Tabla 7. *Valores de los umbrales para la métrica relaciones entre clases.*

A continuación se muestran los resultados de la evaluación de la métrica.

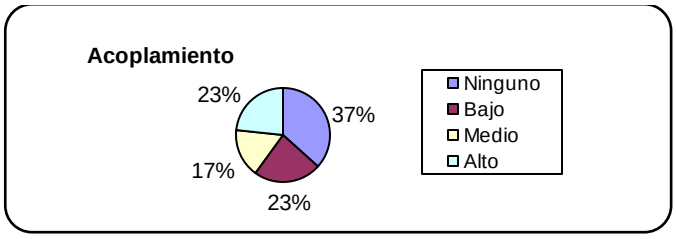

Figura 11. *Representación de las clases según el acoplamiento.*

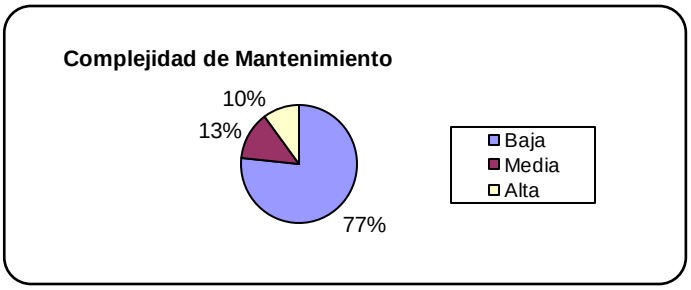

Figura 12. *Representación de las clases según complejidad de mantenimiento.*

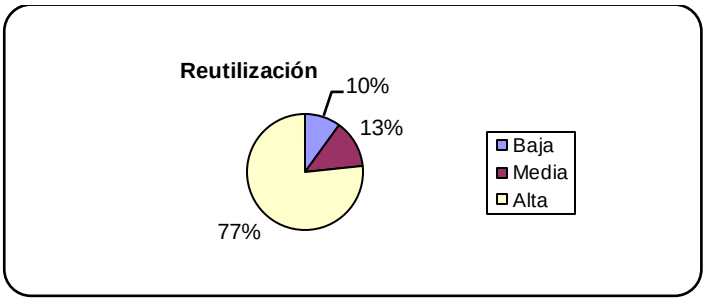

Figura 13. *Representación de las clases según la reutilización.*

La evaluación de los resultados obtenidos durante la aplicación de la métrica RC, demuestra que el diseño asumido tiene una calidad aceptable, puesto que, el 77% de las clases tienen 2 o menos dependencias unas de otras. Por su parte, el nivel de acoplamiento entre las clases se mantiene entre bajo y medio en un 77%, lo cual es un factor aceptable desde el punto de vista de la implementación del software, mientras que la capacidad de reutilización del software alcanza niveles de 77% de forma positiva.

Estos resultados de los atributos de calidad se suman a los obtenidos por las pruebas de TOC y demuestran el uso de un buen diseño de software.

2.6. Modelo de implementación.

La implementación comienza luego del resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente y ejecutables. Gran parte de la arquitectura del sistema es definida durante el diseño, siendo el propósito fundamental de la implementación el desarrollar la arquitectura y el sistema como un todo. Se implementan las clases encontradas durante el diseño. El modelo de implementación es una correspondencia directa de los modelos de diseño y de despliegue.

El modelo de implementación describe cómo los elementos del diseño, las clases, se implementan en términos de componentes o código fuente. Además describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados.

2.6.1. Modelo de despliegue.

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema. El diagrama de despliegue es un tipo de diagrama empleado para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. Muestra las relaciones físicas de los distintos nodos que componen un sistema. A continuación se muestra el modelo de despliegue correspondiente al subsistema depósitos temporales de frontera. (Figura 14)

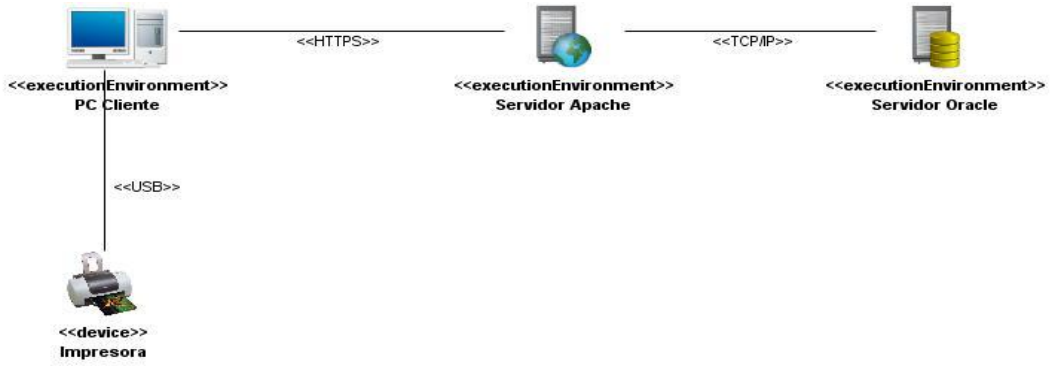

Figura 14. *Diagrama de despliegue subsistema depósitos temporales.*

2.6.2. Diagrama de componentes.

Un diagrama de componentes muestra el sistema dividido por componentes, así como la relación entre estos. Un componente es una parte física de un sistema (módulo, base de datos, programa o ejecutable), es decir, la materialización de una o más clases.

Elementos del diagrama de componentes

- Componentes.
- Interfaces.
- Relaciones de dependencia, generalización y asociación.
- Paquetes o subsistemas.

En el siguiente diagrama se representan los componentes del subsistema depósitos temporales (Figura 15), donde se evidencian las relaciones de los componentes, desde el acceso a través del controlador frontal, la clase actions.php, cada una de las interfaces de la aplicación, las clases del negocio y su interacción con otros subsistemas, así como los elementos de configuración.

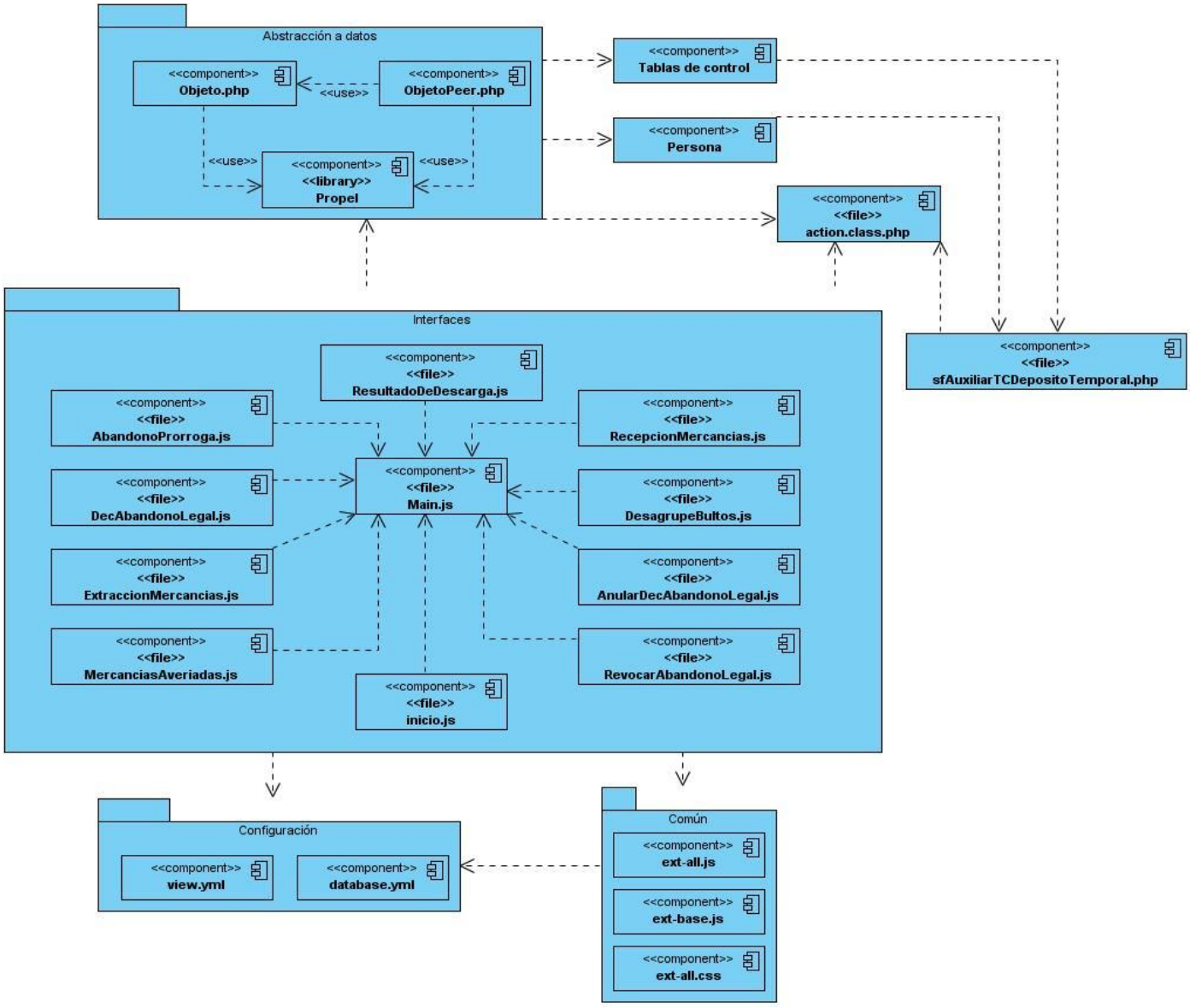

Figura 15. *Diagrama de componentes subsistema depósitos temporales de frontera.*

2.6.3. Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armónico, la legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. Posibilita que un equipo de programadores mantenga un código de calidad sobre el que se efectuarán luego revisiones del código. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento y mantenimiento del software.

La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta el final del proyecto de software. No es práctico adoptar un estándar de codificación una vez iniciado el trabajo. En el caso del Departamento de Soluciones para la Aduana, el estándar de codificación fue definido por la dirección del proyecto en sus inicios y se encuentra registrado en el documento "Propuesta de un estándar de codificación". (27)

Todos los nombres de acciones deben estar en la nomenclatura "CamelCase[24]" que inicia por la palabra **execute,** el nombre de las clases expresado en notación UpperCamelCase[25], además del uso "**Peer**" como sufijo. Los nombres de las variables deben expresar claramente el contenido de la misma. (27)

2.6.4. Las capas del Modelo-Vista-Controlador en Symfony.

El patrón Modelo-Vista-Controlador separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Modelo: es la representación específica del dominio de la información, sobre la cual funciona la aplicación.

[24]**CamelCase**: 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.

[25]**UpperCamelCase**: 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.

Vista: presenta el modelo en un formato adecuado para interactuar con el usuario, usualmente un elemento de la interfaz, como puede ser un formulario.

Controlador: responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. (21)

A continuación se muestra cómo se evidencia la aplicación del Modelo-Vista-Contolador, en la petición **Mostrar Datos del Manifiesto**, correspondiente al requisito funcional: gestionar real descargado.

La *Vista* se encarga de obtener y enviar la información deseada, desde las interfaces hacia el controlador, a través de peticiones.

```
Ext.Ajax.request({
    url: 'temporal/mostrarDatosManifiesto',
    params: {
        escala: Ext.getCmp('idnoManifiestoRD').getValue()
    },
```

Figura 16. *Petición Ajax desde la interfaz ResultadoDescarga.js (Vista)*

Luego el C*ontrolador* se encarga de obtener los datos enviados desde la *Vista* y según la petición, realizar la acción correspondiente, la cual accediendo al *Modelo* devolverá la respuesta determinada.

```
public function executeMostrarDatosManifiesto() {
    $escala = $this->getRequestParameter('escala');
    $manifiestos = DtManifiestoPeer::GetManifiestosDadoNoManifiesto($escala);
    $cant = count($manifiestos);
    if ($cant == 0) {
        $json = "{ success:true, datos:[";
        return $this->renderText($json . "]}");
    } else {
        $json = "{ success:true, datos:[";
        foreach ($manifiestos as $manifiesto) {
            $json = $json . "{'idManifiesto':'{$manifiesto->getIdManifiesto()}', 'cant':'{$cant}'
            if ($cant-- != 1) {
                $json = $json . ',';
            }
        }
        return $this->renderText($json . "]}");
    }
```

Figura 17. *Acción correspondiente a la petición (Controlador).*

```
public static function GetManifiestosDadoNoManifiesto($mumeroManifiesto) {
    $criteria = new Criteria();
    $criteria->add(DtManifiestoPeer::NOMANIFIESTO, $mumeroManifiesto);
    $manifiesto = DtManifiestoPeer::doSelect($criteria);
     if (isset($manifiesto))
         return $manifiesto;
     return null;
}
```

Figura 18. *Función GetManifiestoDadoNoManifiesto (Modelo).*

2.6.5. Interfaces del sistema

Para la realización de la aplicación se desarrollaron interfaces amigables y seguras, acorde con el personal de la Aduana General de la República que trabajará con la misma. A continuación se presenta la interfaz principal de la aplicación (Figura 19). En esta se encuentran los vínculos a todas las funciones con que debe cumplir la misma.

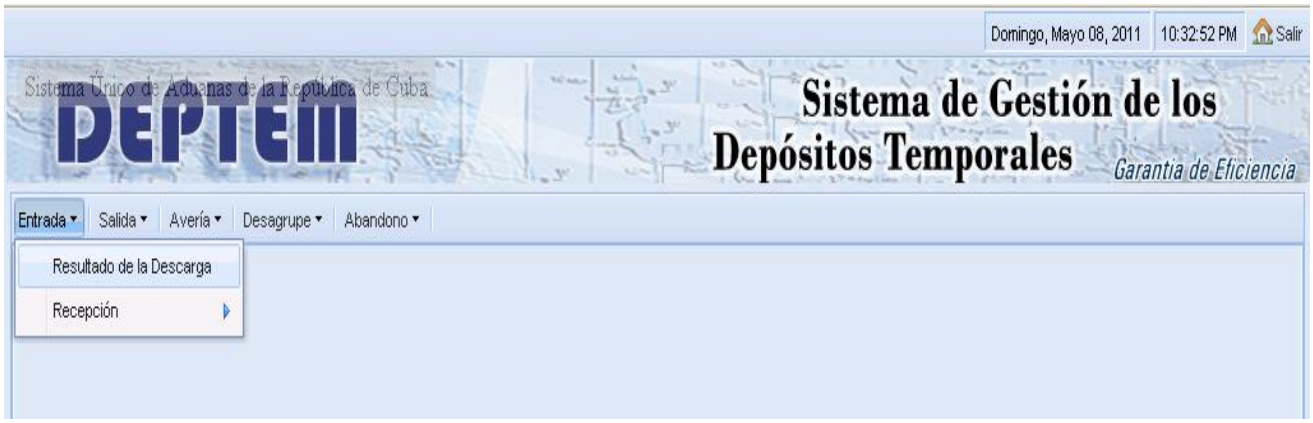

Figura 19. *Interfaz principal del subsistema depósitos temporales de frontera.*

Al seleccionar en el menú “Resultado de la Descarga”, se ejecuta el siguiente código que permite mostrar la pestaña correspondiente para la gestión de la descarga de mercancías de importación. (Figura 20)

```
new Viewport({
    bannerCls: 'banner_deptemp',
    panel: {
        frame: true,
        activeItem:0,
        id: 'mainPanelExt',
        layout: 'card',
        tbar: [{
            text: 'Entrada',
            menu: [{
                text: 'Resultado de la Descarga',
               handler: function(){
                    Ext.getCmp('mainPanelExt').getLayout().setActiveItem(0);
                    Ext.getCmp('mainPanelExt').add({
                        xtype: 'resultadoDeDescarga',
                        id: 'ResultadoDeDescarga'
                    });
                Ext.getCmp('mainPanelExt').doLayout();
                Ext.getCmp('mainPanelExt').getLayout().setActiveItem('ResultadoDeDescarga');
```

Figura 20. *Segmento de código fuente de la interfaz principal.*

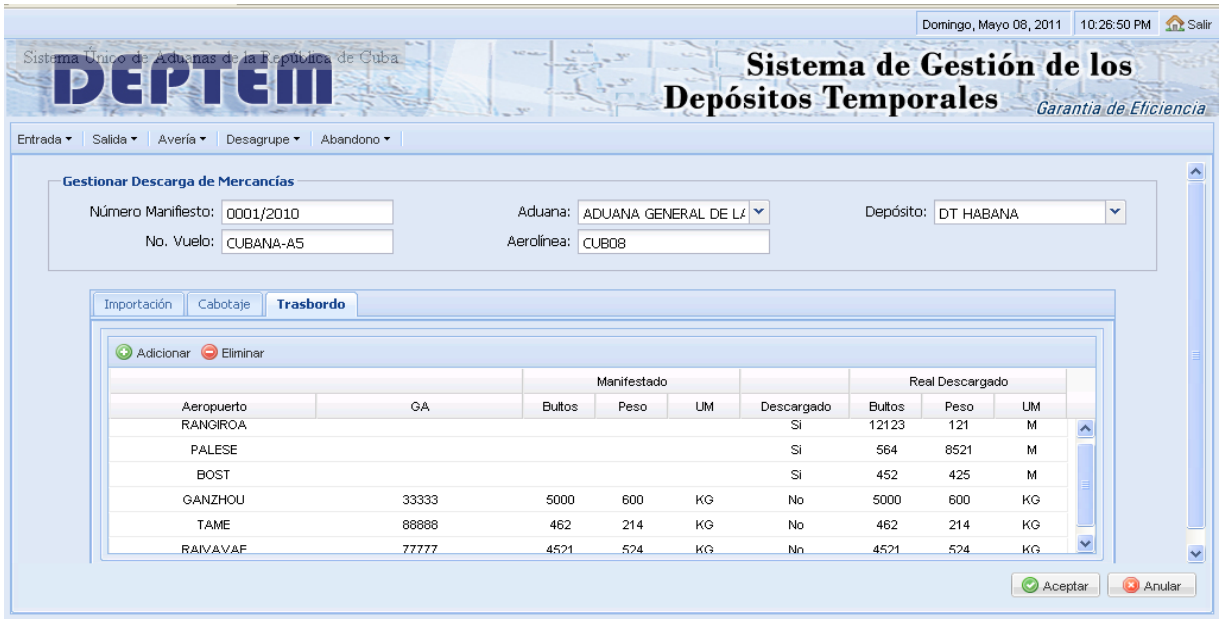

Figura 21. *Interfaz del requisito funcional "gestionar real descargado".*

Al introducir los datos referentes para el registro de la descarga: número de manifiesto, aduana y depósito, se muestran las cargas asociadas a la búsqueda, facilitando registrar, modificar o anular la descarga correspondiente. (Figura 21)

A continuación parte del código PHP que posibilita esta funcionalidad. (Figura 22)

```
public function executeGetCargasPorNoManifiestoDeposito() {
    $mumeroEscala = $this->getRequestParameter('escala');
    $codDeposito = $this->getRequestParameter('coddeposito');
    $tipoManifiesto = $this->getRequestParameter('tipoManifiesto');
    $realDescargado = DtRealDescargadoPeer::getRealDescargado($mumeroEscala, $codDeposito);
    $total = array();
    if (count($realDescargado) == 0) {
        $json = "{ success:true, datos:[";
        if ($tipoManifiesto == 'importacion') {
            $bultos = DtCargaConvencionalPeer::getCargasImportacion($mumeroEscala);
        } else {
            if ($tipoManifiesto == 'cabotaje') {
                $bultos = DtCargaConvencionalPeer::getCargasCabotaje($mumeroEscala);
            } else {
                $bultos = DtCargaConvencionalPeer::getCargasTrasbordo($mumeroEscala);
            }
        }
```

Figura 22. *Segmento de código, funcionalidad* ***GetCargasPorNoManifiestoDeposito().***

CONCLUSIONES

En este capítulo se mostraron los resultados del diseño e implementación, presentándose los artefactos generados en ambas fases: los diagramas de clases del diseño, diagramas de secuencia orientada a actividades, patrones de diseño utilizados, así como el modelo de la base de datos. Se presentaron los resultados de la validación del diseño propuesto, evidenciándose a través de la utilización de las métricas Tamaño Operacional de Clases y Relaciones entre Clases, buenos valores para los atributos de calidad, predominando una baja responsabilidad, complejidad y acoplamiento entre las clases. Se presentó el modelo de despliegue y parte del código fuente obtenido, a fin de obtener un sistema que satisfaga los requisitos definidos, y que permita gestionar los procesos de importación de mercancías en los depósitos temporales de frontera de la Aduana General de la República.

CAPÍTULO 3: VALIDACIÓN DEL SISTEMA

INTRODUCCIÓN

Desde el inicio de la construcción de un software pueden evidenciarse errores al definirse los objetivos, así como en las siguientes fases del diseño e implementación, y posteriormente a estas. Es por ello que el proceso de desarrollo debe estar acompañado de alguna actividad que garantice su calidad. Durante el desarrollo del capítulo, se valida mediante pruebas si el sistema implementado ha cumplido con el objetivo inicial y se presenta la validación de la solución propuesta en el capítulo anterior, de manera que se pueda comprobar las funcionalidades utilizadas para dar respuesta a los distintos requisitos planteados por el cliente.

3.1. Pruebas.

Al desarrollar sistemas informáticos se corre un alto riesgo de que se produzcan errores producto al fallo humano. Estos pueden ocurrir desde el comienzo del proceso, ya sea en la definición de los objetivos, el diseño, la implementación o en otras fases.

Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad, con el objetivo de medir el grado en que el software cumple con los requisitos. (28)

La fase de pruebas añade valor al producto que se maneja, ya que todos los programas pueden tener errores, y en esta fase deben ser detectados, contribuyendo así a la calidad del software. La prueba de software es un conjunto de herramientas, técnicas y métodos que hacen la excelencia del desempeño de un programa. En esta etapa se realizan los casos de prueba, que intentan ocasionar errores en el software, realizando todo tipo de operaciones que pudieran ocasionar un mal funcionamiento del mismo, para detectar problemas que el sistema pudiese presentar. Las pruebas no pueden confirmar la ausencia de errores del software, pero sí demostrar que tiene defectos.

3.2. Diseño de casos de prueba.

Cuando se diseñan casos de prueba se tienen las siguientes características:

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.

- Un buen caso de prueba es aquel que tiene una alta probabilidad de descubrir un error no encontrado hasta entonces.

Un producto de ingeniería puede ser probado de las siguientes maneras:

- **Pruebas de caja blanca:** permiten desarrollar pruebas que aseguren que la operación interna se ajusta a las especificaciones, y que todos los componentes internos se han probado de forma adecuada. Se realiza un examen minucioso de los detalles procedimentales, comprobando los caminos lógicos del programa, los bucles y condiciones, y examinado el estado del programa en varios puntos. (28)
- **Pruebas de caja negra:** realizan pruebas que verifican que cada función es operativa y que la entrada y salida se producen de forma correcta. Estas pruebas se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa.

Los casos de prueba de caja negra pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Las pruebas de caja negra pretenden encontrar los tipos de errores siguientes:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Generación de casos de prueba: la generación de cada caso de prueba debe ir acompañada del resultado que ha de producir el software al ejecutar dicho caso para detectar un posible fallo en el programa. Los casos de prueba determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Cada técnica de pruebas proporciona unos criterios distintos para generar estos casos o datos de prueba.

Después de un análisis de los diferentes métodos se ha llegado a la conclusión de que es más factible aplicar las pruebas de caja negra para comprobar la validez en las respuestas del programa ante las acciones del usuario y la calidad de las salidas en dependencia de las entradas.

Para validar el software desarrollado, fue sometido a pruebas de caja negra por el grupo de desarrollo del proyecto Aduana, haciendo uso de 13 casos de prueba correspondientes a los requisitos funcionales, siendo evaluadas todas sus funcionalidades. Luego de una detallada revisión se encontraron 5 no conformidades, en la primera iteración: 1 de validación y 4 de interfaz.

A continuación se enumeran:

1. En la pantalla gestionar prórrogas, cuando se daba clic en anular prórroga sin haberla seleccionado, el sistema mostraba incorrectamente el mensaje: “Prórroga anulada correctamente”.
2. No se tenía presente que al adicionar un no manifiesto en el requisito gestionar real descargado, no aceptara caracteres extraños en la cantidad de bultos.
3. En la pantalla de gestionar averías en bultos de importación, antes de realizar la búsqueda, mostraba el grid de averías y los botones del mismo habilitados.
4. En la pantalla de extracción de mercancías, cuando se registraba una extracción, el sistema mostraba el grid referente a los sellos habilitado. Solamente debía estar habilitado a la hora de confirmar la extracción de mercancías.
5. Para generar una declaración de abandono legal, en la pantalla que permite esta operación, no se tenía en cuenta validar que la cantidad de bultos no superara lo que realmente estaba en abandono en el depósito.

Las no conformidades fueron corregidas, comprobándose la eliminación de las mismas en una segunda iteración, obteniéndose como resultado de la misma 0 no conformidades.

A continuación se muestra el diseño del caso de prueba correspondiente el requisito funcional: gestionar real descargado.

Nombre del Requisito: Gestionar real descargado

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1: Gestionar real descargado.	EC 1.1: Buscar descarga.	Se hace una búsqueda de la descarga.
	EC 1.2: Adicionar no manifestado.	Se adicionan cargas no manifestadas.
	EC 1.3: Eliminar no manifestado.	Se eliminan cargas no manifestadas.
	EC 1.4: Registrar real descargado.	Se registra la descarga en el depósito.
	EC 1.5: Modificar real descargado.	Se modifica la descarga en el depósito.
	EC 1.6: Anular real descargado.	Se anula la descarga en el depósito.

Tabla 8: *Secciones a probar en el requisito.*

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	No. Manifiesto	Campo de texto	No	4 dígitos / 4 dígitos. #### / #### Ejemplo: 0210/2041
2	Aduana	Campo de selección	No	Seleccionar una de las opciones.
3	Depósito	Campo de selección	No	Seleccionar una de las opciones.
4	Aeropuerto	Campo de selección	No	Seleccionar una de las opciones.
5	Bultos	Campo de texto	No	Dígitos (número entero).
6	Peso	Campo de texto	No	Dígitos (número entero o decimal).
7	UM	Campo de selección	No	Seleccionar una de las opciones.

Tabla 9: *Descripción de variables.*

Matriz de datos

SC 1: Gestionar real descargado

Escenario	No. Manifiesto	Aduana	Depósito	Aeropuerto	Bultos	Peso	UM	Respuesta Esperada	Resultado de la Prueba	Flujo Central
EC 1.1: Buscar descarga.	I Aa21 /21as	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	Se muestra un mensaje indicando que no son válidos los elementos del formulario.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos incorrectos o correctos. Clic en el botón Buscar.
	V 0001/ 2011	I Sin seleccionar	NA	NA	NA	NA	NA	El sistema no permite la búsqueda	Satisfactorio	
	V 0001/ 2011	V AGR – CADI	I Sin seleccionar	NA	NA	NA	NA	El sistema no permite la búsqueda	Satisfactorio	
	V 0001/ 2011	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	El sistema permite la búsqueda correspondiente.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos válidos, que existen. Clic en el botón Buscar
	V 0001/ 2015	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	El sistema muestra un mensaje indicando que el número de manifiesto no existe.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos válidos, que no existen. Clic en el botón Buscar.

Escenario	No. Manifiesto	Aduana	Depósito	Aeropuerto	Bultos	Peso	UM	Respuesta Esperada	Resultado de la Prueba	Flujo Central
EC 1.2: Adicionar no manifestado	V 0001/2011	V AGR - CADI	V Importador Bayamo	I	V 24	V 52	V KILOGRAMO	Se muestra un mensaje indicando que no son válidos los elementos del formulario.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos correctos para la búsqueda. Clic en el botón Buscar. Clic en el botón adicionar del grid (importación, cabotaje o trasbordo). Introducir los datos del no manifestado. Clic en Aceptar.
	V 0001/2011	V AGR - CADI	V Importador Bayamo	V BOST	I 52a	V 50	V KILOGRAMO		Satisfactorio	
	V 0001/2011	V AGR - CADI	V Importador Bayamo	V BOST	V 26	I 5gf0	V KILOGRAMO		Satisfactorio	
	V 0001/2011	V AGR - CADI	V Importador Bayamo	V BOST	V 26	V 62	I		Satisfactorio	
	V 0001/2011	V AGR - CADI	V Importador Bayamo	V BOST	V 26	V 62	V KILOGRAMO	El sistema añade un no manifestado correctamente.	Satisfactorio	

Escenario	No. Manifiesto	Aduana	Depósito	Aeropuerto	Bultos	Peso	UM	Respuesta Esperada	Resultado de la Prueba	Flujo Central
EC 1.3: Eliminar no manifestado	V 0001/2011	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	El sistema muestra un mensaje indicando que debe seleccionar la carga a eliminar.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos correctos para la búsqueda. Clic en el botón Buscar. Clic en el botón eliminar del grid (importación, cabotaje o trasbordo), sin estar seleccionada una fila del grid.
	V 0001/2011	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	El sistema elimina el no manifestado.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos correctos para la búsqueda. Clic en el botón Buscar. Clic en el botón eliminar del grid (importación, cabotaje o trasbordo), estando seleccionada una fila del grid, correspondiente a un no manifestado.
	V 0001/2011	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	El sistema muestra un mensaje indicando que no se puede eliminar un manifestado.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos correctos para la búsqueda. Clic en el botón Buscar. Clic en el botón eliminar del grid (importación, cabotaje o trasbordo), estando seleccionada una fila del grid, correspondiente a un manifestado.

Escenario	No. Manifiesto	Aduana	Depósito	Aeropuerto	Bultos	Peso	UM	Respuesta Esperada	Resultado de la Prueba	Flujo Central
EC 1.4: Registrar real descargado	V 0001/2011	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	El sistema muestra un mensaje indicando que no se puede eliminar un manifestado, debe seleccionar la(s) carga a descargar.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos correctos para la búsqueda. Clic en el botón Buscar. Se seleccionan las cargas a descargar, y(o) se adicionan no manifestados. Clic en el botón aceptar, sin haber seleccionado ninguna carga a descargar.
	V 0001/2011	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	El sistema registra el real descargado.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos correctos para la búsqueda. Clic en el botón Buscar. Se seleccionan las cargas a descargar, y(o) se adicionan no manifestados. Clic en el botón aceptar, seleccionado alguna carga a descargar.

Escenario	No. Manifiesto	Aduana	Depósito	Aeropuerto	Bultos	Peso	UM	Respuesta Esperada	Resultado de la Prueba	Flujo Central
EC 1.5: Modificar real descargado	V 0001/2011	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	El sistema muestra un mensaje indicando que para modificar debe seleccionar la(s) carga correspondientes.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos correctos para la búsqueda. Clic en el botón Buscar. Se seleccionan las cargas a descargar, y(o) se adicionan no manifestados. Clic en el botón aceptar, sin seleccionar alguna carga a descargar.
	V 0001/2011	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	El sistema registra el real descargado.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos correctos para la búsqueda. Clic en el botón Buscar. Se seleccionan las cargas a descargar, y(o) se adicionan no manifestados. Clic en el botón aceptar, seleccionado alguna carga a descargar.

Escenario	No. Manifiesto	Aduana	Depósito	Aeropuerto	Bultos	Peso	UM	Respuesta Esperada	Resultado de la Prueba	Flujo Central
EC 1.6: Anular Real descargado	V 0001/2011	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	El sistema no permite dar clic en el botón anular, por lo que el sistema no anula la descarga		Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos correctos para la búsqueda. Clic en el botón Buscar. Se seleccionan las cargas a descargar, y(o) se adicionan no manifestados. Clic en el botón anular, cuando no existe un real descargado.
	V 0001/2011	V AGR - CADI	V Importador Bayamo	NA	NA	NA	NA	El sistema anula el real descargado satisfactoriamente.	Satisfactorio	Ir a Menú Entrada. Seleccionar Resultado Descarga. Se introducen datos correctos de la búsqueda. Clic en el botón Buscar. Se seleccionan las cargas a descargar, y(o) se adicionan no manifestados. Clic en el botón anular, cuando existe un real descargado.

Tabla 10: *Matriz de datos, sección: gestionar real descargado.*

CONCLUSIONES

En el desarrollo del presente capítulo, se valida el sistema desarrollado mediante pruebas de caja negra, realizando casos de prueba a todas las funcionalidades del mismo en dos iteraciones. Este proceso permite detectar la mayor cantidad de no conformidades que este presentaba para darle solución a las mismas. Luego de una segunda iteración no se encontraron no conformidades, obteniendo una aplicación que satisfaga los requisitos definidos.

CONCLUSIONES GENERALES

Luego de una exhaustiva investigación y la elaboración de la aplicación que permite satisfacer los requisitos para informatizar los procesos asociados a la importación de mercancías en los depósitos temporales de frontera de la Aduana General de la República, se concluye lo siguiente:

- Se analizaron las soluciones existentes que gestionan la importación de mercancías en los depósitos temporales, comprobándose que las extranjeras no eran factibles por ser privativas, costosas, y no adecuarse a las legislaciones de la Aduana General de la República; mientras la solución cubana SUA, es ineficiente por su incapacidad para gestionar todas las operaciones y procedimientos efectuados en los depósitos temporales.

- Se analizó la metodología de desarrollo de software, lenguajes, herramientas de modelado y tecnologías utilizadas en el proyecto Aduana con el objetivo de ser empleadas en el diseño e implementación del sistema.

- Se generaron los artefactos necesarios para documentar el diseño del sistema: diagramas de clases, diagramas de secuencia y modelo de datos, permitiendo materializar con precisión los requerimientos expuestos por el cliente.

- Se validó el diseño del sistema aplicando métricas que arrojaron resultados positivos obteniéndose buenos valores de reutilización, implementación y responsabilidad.

- Se implementó la solución haciendo uso de los estándares de codificación existentes por el Departamento de Soluciones para la Aduana, que permitió obtener un código armónico y legible para un mejor entendimiento de los desarrolladores.

- Se realizaron las pruebas de caja negra, obteniendo como resultado de los casos de prueba 5 no conformidades en la primera iteración, de ellas una fue de validación y cuatro de interfaz, siendo corregidas todas en la segunda iteración.

RECOMENDACIONES

Aún con el cumplimiento de los objetivos, el diseño e implementación del sistema para la informatización de los procesos de importación en los depósitos temporales de frontera de la Aduana General de la República, se necesita incorporar recomendaciones que se consideran fundamentales:

- Implementar funcionalidades asociadas a la obtención de reportes sobre el inventario de mercancías de importación en el depósito.
- Vincular la solución propuesta con los correspondientes subsistemas del producto GINA: Administración, Auditoría, Despacho Comercial, Despacho no Comercial, MTI y Asuntos Legales.
- Implementar los procesos asociados a la exportación de mercancías en los depósitos temporales de frontera de la Aduana General de la República.

GLOSARIO DE TÉRMINOS

Aduana: oficina pública, establecida generalmente en las costas y fronteras, para registrar, en el tráfico internacional, los géneros y mercaderías que se importan o exportan, y cobrar los derechos que adeudan.

AGR: Aduana General de la República.

GINA: Gestión Integral Aduanera.

Framework: estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Metodología: proceso de software detallado que define con precisión los artefactos, roles y actividades involucradas.

Clase: descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

Diagrama: representación gráfica de un conjunto de elementos. Visualiza un sistema desde diferentes perspectivas.

Especificación de requisitos: captura los requisitos de software para el sistema completo o una porción del mismo.

Middleware: software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red).

REFERENCIAS BIBLIOGRÁFICAS

1. AGR. Resolución 33/96. *Glosario de Términos Aduaneros.* [En línea] 1996.

2. **Arias, Roxana Fonseca y Sol, Ariel Calzadilla del.** *Modelado de un sistema informático para los Depósitos.* Ciudad de la Habana : s.n., 2010.

3. **S-4.** S-4. [En línea] [Citado el: 15 de Febrero de 2011.] http://www.s-4.es/ms_Depositos.aspx..

4. Softway. Soluciones de Software. [En línea] http://www.softcomex.com.br/site_espanhol/produtos_ev.shtml.

5. Gestión de depósitos de aduanas. [En línea] http://www.cotecna.com/COM/ES/bonded_warehouse_management.aspx.

6. Conociendo al SIDUNEA - Sistema Aduanero Automatizado. [En línea] http://www.gestiopolis.com/canales2/economia/sidunea.htm.

7. MODSHD(Módulo de Depósito Temporal). [En línea] http://www.seniat.gob.ve/portal/page/portal/MANEJADOR_CONTENIDO_SENIAT/08SISTEMASLINEA1/DESCARGA_SIDUNEA/MODSHD.pdf.

8. **S, PRESSMAN R.** *Ingeniería del software, un enfoque práctico.* Madrid : s.n., 2002. 84-481-3214-9.

9. **Reynoso, Carlos Billy.** Introducción a la Arquitectura de Software. [En línea] 3 de 2004. http://www.willydev.net/descargas/prev/IntroArq.pdf.

10. *IEEE STD 1471 (Software Engineering Standards Committee of the IEEE Computer Society.* 2000.

11. **CHRISTOPHER A., ISHIKAWA S., SILVERSTEIN M.** *A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series) .* New York : s.n., 1977. ISBN 0195019199.

12. **C, LARMAN.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : s.n., 1999. ISBN 970-17-0261-1.

13. **Prieto, Félix.** Patrones de diseño. [En línea] 2009. http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.

14. **Rodríguez, José Antonio Cobo.** *Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas.* La Habana : s.n., Junio, 2008.

15. **Galán, Darwin Daniel Hernández.** Scribd. [En línea] 15 de Marzo de 2010. http://es.scribd.com/doc/37117063/Metric-As.

16. **CORPORATION, R. S.** *Ayuda extendida de RUP (Rational Unified Process) Version 2003.06.00.65.* 2003.

17. **Martínez, Jenni Manso.** *Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE.* Habana : s.n., 2010.

18. **LARMAN.** *C. UML y Patrones. Introducción al análisis y diseño orientado a objetos. ISBN 970-17-0261-1.* México : s.n., 1999.

19. Visual Paradigm for UML. [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/ .

20. Tutorial de php. [En línea] http://www.phpya.com.ar/temarios/descripcion.php?cod=23.

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

22. Extjs. [En línea] http://www.extjses.com/.

23. **Velasco, Roberto Hernando.** El SGBDR Oracle. [En línea] 20 de Enero de 2011. http://www.rhernando.net/modules/tutorials/doc/bd/oracle.html.

24. **Herrera, L.H Tamayo y R.P.** *Información Adelantada de Pasajeros: Análisis y Diseño.* Habana : s.n., 2009.

25. EcuRed. [En línea] http://www.ecured.cu/index.php/Patrones_en_Symfony.

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

27. **CEIGE, Dpto Aduana.** *Propuesta de un Estándar de Codificación.* 2011.

28. **Calás, Alexey Talavera.** *Sistema Calculador de Métricas para evaluar la calidad de un software en el ERP- Cuba.* La Habana : s.n., 2010.

BIBLIOGRAFÍA

1. AGR. Resolución 33/96. *Glosario de Términos Aduaneros.* [En línea] 1996.

2. **Arias, Roxana Fonseca y Sol, Ariel Calzadilla del.** *Modelado de un sistema informático para los Depósitos.* Ciudad de la Habana : s.n., 2010.

3. **S-4.** S-4. [En línea] [Citado el: 15 de Febrero de 2011.] http://www.s-4.es/ms_Depositos.aspx..

4. Softway. Soluciones de Software. [En línea] http://www.softcomex.com.br/site_espanhol/produtos_ev.shtml.

5. Gestión de depósitos de aduanas. [En línea] http://www.cotecna.com/COM/ES/bonded_warehouse_management.aspx.

6. Conociendo al SIDUNEA - Sistema Aduanero Automatizado. [En línea] http://www.gestiopolis.com/canales2/economia/sidunea.htm.

7. MODSHD(Módulo de Depósito Temporal). [En línea] http://www.seniat.gob.ve/portal/page/portal/MANEJADOR_CONTENIDO_SENIAT/08SISTEMASLINEA1/DESCARGA_SIDUNEA/MODSHD.pdf.

8. **S, PRESSMAN R.** *Ingeniería del software, un enfoque práctico.* Madrid : s.n., 2002. 84-481-3214-9.

9. **Reynoso, Carlos Billy.** Introducción a la Arquitectura de Software. [En línea] 3 de 2004. http://www.willydev.net/descargas/prev/IntroArq.pdf.

10. *IEEE STD 1471 (Software Engineering Standards Committee of the IEEE Computer Society.* 2000.

11. **CHRISTOPHER A., ISHIKAWA S., SILVERSTEIN M.** *A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series) .* New York : s.n., 1977. ISBN 0195019199.

12. **C, LARMAN.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : s.n., 1999. ISBN 970-17-0261-1.

13. **Prieto, Félix.** Patrones de diseño. [En línea] 2009. http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.

14. **Rodríguez, José Antonio Cobo.** *Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas.* La Habana : s.n., Junio, 2008.

15. **Galán, Darwin Daniel Hernández.** Scribd. [En línea] 15 de Marzo de 2010. http://es.scribd.com/doc/37117063/Metric-As.

16. **CORPORATION, R. S.** *Ayuda extendida de RUP (Rational Unified Process) Version 2003.06.00.65.* 2003.

17. **Martínez, Jenni Manso.** *Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE.* Habana : s.n., 2010.

18. **LARMAN.** *C. UML y Patrones. Introducción al análisis y diseño orientado a objetos. ISBN 970-17-0261-1.* México : s.n., 1999.

19. Visual Paradigm for UML. [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/ .

20. Tutorial de php. [En línea] http://www.phpya.com.ar/temarios/descripcion.php?cod=23.

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

22. Extjs. [En línea] http://www.extjses.com/.

23. **Velasco, Roberto Hernando.** El SGBDR Oracle. [En línea] 20 de Enero de 2011. http://www.rhernando.net/modules/tutorials/doc/bd/oracle.html.

24. **Herrera, L.H Tamayo y R.P.** *Información Adelantada de Pasajeros: Análisis y Diseño.* Habana : s.n., 2009.

25. EcuRed. [En línea] http://www.ecured.cu/index.php/Patrones_en_Symfony.

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

27. **CEIGE, Dpto Aduana.** *Propuesta de un Estándar de Codificación.* 2011.

28. **Calás, Alexey Talavera.** *Sistema Calculador de Métricas para evaluar la calidad de un software en el ERP- Cuba.* La Habana : s.n., 2010.

29. **Bullock, Christian.** Ext JS. Comunidad en Español. [En línea] 2002. http://extjses.com/..

30. Introducción a los patrones de Software. [En línea] 15 de Septiembre de 2009. http://sistemas.uniandes.edu.co/~isis2701/dokuwiki/lib/exe/fetch.php?media=isis2701-patronesgrasp.pdf.

31. Maestría Tecnológica. [En línea] http://www.palermo.edu/ingenieria/maestria_tecno_informatic/descrip_cursos.html.

32. **Centro de Gobierno Electrónico (CEGEL), Facultad 3.** *Guía Metodológica para el Desarrollo de los Trabajo de Diploma en el Pregrado. Un consenso de tutores en el Centro de Gobierno Electrónico (CEGEL).* La Habana : s.n.

33. **Chidamber, S.R, Darcy, D.P y Kemerer, C.F.** *Managerial Use od Metrics for Object-Oriented Software: An Explicatory Analysis.* 1998.

34. **Cornejo, José Enrique González.** El lenguaje de modelado unificado. [En línea] 15 de Enero de 2011. http://www.docirs.cl/uml.htm.

35. **Fenton, Norman E.** *Software Metrics.* 1991.

36. **Flores, Antonio.** *Introducción al Lenguaje de Programación PHP.* Madrid : s.n., 2008.

37. **Johnson, R. E. y Helm, R.** *Design Patterns: Elements of Reusable Object-Oriented Software.* 1995.

38. **Mena, Aldas y Mena, Daniel Ernesto, Andrade Cadena, Maritza Alejandra.** *Tésis_Guía_práctica_Aplicación de buenas prácticas de programación para garantizar un software seguro.* Quito, Ecuador : s.n., 2010.

39. **Pérez, Ing. en Ciencias Informáticas Mileidy Magalys Sarduy.** *Propuesta de modelo de desarrollo.* La Habana : s.n., 2009.

40. **Potencier, Fabien.** *El tutorial, Jobeet.* 2009.

41. **Rodríguez, José Antonio Cobo.** *Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas.* La Habana : Universidad de las Ciencias Informáticas, 2008.

42. **Xavier Ferré Grau, María Isabel Sánchez Segura.** *Desarrollo Orientado a Objetos con UML.* 2008.