

Universidad de las Ciencias Informáticas

Facultad 3



Versión 1.1 del subsistema Depósitos de Aduana

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

Autor: Yosvany Medina Carbó


Tutores: Ing. Liannis Soria Barreda

Ing. Iliana de la Rosa Zayas Frias

Ing. Yordani Cruz Segura

La Habana, Junio de 2013

Año 55 de la Revolución

A portrait of Fidel Castro, a man with a beard and mustache, looking slightly to the right. He is wearing a dark jacket. The background is the Cuban flag, featuring three horizontal stripes of blue, white, and red, with a white five-pointed star on the red stripe.

"El futuro de nuestra patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento, porque precisamente es lo que más estamos sembrando; lo que más estamos sembrando son oportunidades a la inteligencia (...)"

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

Declaración de autoría

Declaro ser el autor de la presente tesis y recomiendo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yosvany Medina Carbó

Firma del Autor

Ing. Liannis Soria Barreda

Firma del Tutor

Ing. Iliana de la Rosa Zayas Frias

Firma del Tutor

Ing. Yordani Cruz Segura

Firma del Tutor

Datos de contacto

Autor: Yosvany Medina Carbó

- Estudiante de la Universidad de las Ciencias Informáticas.
- Correo electrónico ycarbo@estudiantes.uci.cu.

Tutor(a): Ing. Liannis Soria Barreda.

- Ingeniera en Ciencias Informáticas graduada en el 2007 en la Universidad de las Ciencias Informáticas.
- Correo electrónico lsoria@uci.cu.

Tutor(a): Ing. Iliana de la Rosa Zayas Frias

- Ingeniera en Ciencias Informáticas graduada en el 2010 en la Universidad de las Ciencias Informáticas.
- Correo electrónico ilzayas@uci.cu.

Tutor: Ing. Yordani Cruz Segura.

- Ingeniero en Ciencias Informáticas. Graduado en el 2011 en la Universidad de las Ciencias Informáticas.
- Correo electrónico ysegura@uci.cu.

Definitivamente este trabajo no se habría podido realizar sin la colaboración de muchas personas que me brindaron su ayuda; siempre resultará difícil agradecer a todos aquellos que de una u otra manera me han acompañado en el desarrollo de este trabajo, porque nunca alcanza el tiempo, el papel o la memoria para mencionar o dar con justicia todos los créditos y méritos a quienes se lo merecen. Por tanto, quiero agradecerles a todos ellos cuanto han hecho por mí, para que este trabajo saliera adelante de la mejor manera posible.

Partiendo de esta necesidad y diciendo de antemano MUCHAS GRACIAS, primeramente deseo agradecer a mis padres por hacer de mí una mejor persona a través de su ejemplo de honestidad y entereza por lo que siempre han sido una guía a lo largo de mi vida.

En segundo lugar a mi hermana, a mi familia, a mis tutores, a mis amistades y a todas las personas que han formado parte de mi vida a las que me encantaría agradecerles su amistad, consejos, apoyo, ánimo y compañía en los momentos más difíciles de mi vida. Algunas están aquí conmigo y otras en mis recuerdos y en mi corazón, sin importar en donde estén quiero darles las gracias por formar parte de mí, por todo lo que me han brindado y por todas sus bendiciones.

Para ellos: Muchas gracias y que Dios los bendiga.

A mis padres, a mi hermana y a mi familia en general por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo perfectamente mantenido a través del tiempo.

A mi madre Belkis:

Por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada, por su amor.

A mi padre Jorge Luis:

Por los ejemplos de perseverancia y constancia que lo caracterizan y que me ha infundado siempre, por el valor mostrado para salir adelante y por su amor.

La informatización de los procesos aduaneros en Cuba trae consigo una reducción de costos y tiempo de procesamiento de información, además de mejorar la calidad del trabajo. La Aduana General de la República de Cuba se ha trazado como meta la informatización de la mayoría de los procesos que se realizan en sus instalaciones y uno de ellos es el de controlar las mercancías en los Depósitos de Aduana.

El sistema que se utiliza para llevar el control de las mercancías presenta varias dificultades debido a las nuevas resoluciones de la aduana y a que existe una cierta cantidad de información que es proporcionada mediante soportes digitales y otra parte de la misma de forma manual. El presente trabajo se planteó como objetivo desarrollar un sistema, que permita controlar las mercancías en los Depósitos de Aduana, integrado al sistema de Ventanilla Única a partir de las nuevas resoluciones.

Con este fin se utilizó como metodología de desarrollo el Modelo de desarrollo definido en el CEIGE para todos sus proyectos y el lenguaje de modelado UML para elaborar todos los artefactos. Las tecnologías que fueron utilizadas para su implementación son la herramienta de desarrollo NetBeans, el lenguaje de programación PHP (del inglés, *Hypertext Preprocessor*), la plataforma Symfony, la librería ExtJS de JavaScript y Oracle como sistema gestor de base de datos.

Palabras claves: Aduana, Depósitos, Mercancías

Introducción.....	3
Capítulo 1 Fundamentación teórica	8
1.1 Introducción	8
1.2 Generalidades de los Depósitos de Aduana en Cuba	8
1.3 Soluciones informáticas para la gestión y control de las operaciones en los Depósitos de Aduana.....	9
1.3.1 Soluciones extranjeras	9
1.3.2 Soluciones cubanas.....	13
1.4 Metodología, Lenguaje y Herramientas de modelado utilizadas para el desarrollo.	14
1.5 Ingeniería de requerimientos	18
1.5.1 Actividades de la Ingeniería de Requerimientos	19
1.6 Diseño de software.....	20
1.6.1 Patrones de Diseño	21
1.7 Conclusiones parciales.....	24
Capítulo 2: Descripción de la solución propuesta	25
2.1 Introducción	25
2.2 Modelo Conceptual	25
2.3 Especificación de los Requerimientos del Sistema.....	26
2.3.1 Técnicas para la Captura de Requerimientos.....	27
2.3.2 Requerimientos Funcionales	27
2.3.3 Técnicas para la Validación de Requerimientos.....	32
2.3.4 Requisitos No Funcionales	33
2.4 Patrones de diseño utilizados.....	33
2.4.1 Patrones GRASP	33
2.4.2 Patrones GoF.....	34
2.4.3 Patrón MVC según Symfony	35
2.5 Diseño del sistema	35
2.5.1 Diagrama de clases del Diseño con Estereotipos Web	35
2.5.2 Diagrama de Secuencia	37
2.5.3 Diagrama de clases persistentes.....	38

2.5.4 Diagrama de paquetes	39
2.5.5 Diseño de la base de datos	41
2.6 Métricas para la evaluación del diseño	42
2.6.1 Métricas propuestas por Lorenz y Kidd	42
2.6.2 Métrica de Tamaño Operacional de Clase (TOC)	43
2.6.3 Métrica Relación entre clases	45
2.7 Conclusiones Parciales	48
Capítulo 3: Implementación y pruebas al sistema	49
3.1 Introducción	49
3.2 Descripción general de la disciplina Implementación	49
3.3 Modelo de implementación	49
3.3.1 Estándar de codificación	49
3.3.2 Tratamiento de Errores.....	51
3.3.3 Diagrama de Despliegue.....	53
3.3.4 Diagrama de Componentes.....	53
3.4 Validación de la solución	55
3.4.1 Pruebas de Software	55
3.4.2 Aplicación de Pruebas de Caja Negra	55
3.5 Conclusiones Parciales	60
Conclusiones.....	61
Recomendaciones	62
Referencias Bibliográficas	63
Anexos	67
Glosario de términos.....	71

Introducción

La introducción y el uso de las Tecnologías de la Información y las Comunicaciones (TIC) ha significado a escala mundial un salto vertiginoso en el desarrollo científico técnico; desde su llegada a los escenarios nacionales, se han convertido en un elemento indispensable para establecer las líneas de desarrollo de la sociedad cubana, buscando dar solución a los problemas del hombre. Tomando en consideración esa realidad mundial y gracias a la firme voluntad política del gobierno cubano, se han desarrollado en el país múltiples programas encaminados a lograr la informatización de la sociedad; que están relacionados con el proceso de utilización ordenada y masiva de las TIC para satisfacer las necesidades de información y conocimiento de la sociedad lo que constituye una de las metas que tiene Cuba, en el presente y para los próximos años.

En el proceso de informatización de la sociedad cubana la Universidad de las Ciencias Informáticas (UCI) juega un papel importante porque en ella se forman los futuros profesionales del software. En esta institución se realizan proyectos de cooperación con entidades nacionales que traen consigo que el desarrollo del software cubano sea un paso importante para lograr independencia de soluciones informáticas de empresas privadas y contribuir al desarrollo del país.

La Aduana General de la República de Cuba es una de las entidades para las que la UCI realiza productos informáticos. Dicha entidad constituye un órgano de control en frontera y de la actividad interna vinculada al comercio exterior, que garantiza la seguridad y protección de la sociedad socialista y de la economía nacional, así como la recaudación fiscal y las estadísticas del comercio exterior, a través del cumplimiento de las políticas estatales de competencia aduanera para el tráfico internacional de viajeros, mercancías y medios de transporte.(1)

Una de las funciones de la aduana en Cuba es proteger a la sociedad socialista del tráfico ilegal de armamentos, explosivos, drogas, objetos del patrimonio cultural, especies protegidas y proteger a la economía nacional, mediante el control del cumplimiento de la política comercial. Para lograr sus funciones se plantea como objetivo ser una aduana moderna y efectiva en el enfrentamiento de ilícitos aduaneros que contribuya a una frontera infranqueable en aras de garantizar la seguridad de la sociedad socialista y de la economía, así como en un despacho ágil, seguro y transparente, como resultado de un actuar profesional, responsable e íntegro.(1)

La Aduana General de la República (AGR) apoyada en su Centro de Automatización de la Dirección y la Información (CADI), y la UCI, se encuentra trabajando en la informatización de la mayoría de sus procesos, constituyendo de notable importancia el control de las mercancías que entran o salen de los Depósitos de Aduana; entiéndase por estos, lugar o instalación autorizado por la Aduana para el almacenamiento de mercancías sujetas al régimen aduanero¹ de Depósito de Aduanas², el que puede ser público o privado.(2)

En la actualidad los Depósitos de Aduana poseen una solución informática para el control de las mercancías que entran o salen de la entidad, denominada DEPAD Sistema de Gestión de los Depósitos de Aduana. La misma cumple con las normativas o resoluciones aduaneras vigentes hasta el momento en que fue desarrollada y forma parte del sistema GINA, el cual tiene como misión informatizar todos los procesos aduaneros bajo las condiciones de trabajo y políticas establecidas por la aduana cubana. Esta solución recibe la información en ficheros XML, los cuales tienen que ser proporcionados de forma manual o mediante soportes digitales debido a que los parámetros que son de interés para la aduana y la estructura de los mismos ha cambiado, lo que trae consigo que en muchas ocasiones esta información contenga errores. Producto a esta situación la solución existente no responde a las necesidades actuales de la aduana por lo que para llevar a cabo un control efectivo de las mercancías es de suma importancia que la información contenida en los XML sea recibida de forma electrónica, y para lograrlo es necesario que el sistema Gestión Integral de Aduanas se integre al sistema Ventanilla Única. Este último constituye un nuevo sistema informático que se está desarrollando para garantizar la gestión, digitalización y recepción electrónica de la documentación y la información asociada a los procesos de la Aduana General de la República. No contar con la integración de estos sistemas trae consigo que el control de las mercancías por parte de los depositarios³ se convierta en un proceso engorroso e ineficiente en algunas ocasiones, que no se agilicen los procedimientos correspondientes a los depósitos y que parte de la información

¹ 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.

² Régimen aduanero mediante el cual las mercancías importadas se almacenan bajo control aduanero en un lugar designado a este efecto (Depósito de Aduana), sin el pago de los derechos de aduanas y en espera de que se le otorgue un nuevo régimen.

³ Persona natural o jurídica habilitada por la Aduana para operar Depósitos de Aduana o almacenes de mercancías bajo control aduanero.

no se tenga en el momento preciso, lo que dificulta la realización de las acciones a tomar en cuenta, en caso de ser necesario.

A partir de la problemática planteada se define como **problema a resolver**: el subsistema actual de Depósitos de Aduana no permite controlar las mercancías en los depósitos de aduana, integrado al sistema de Ventanilla Única a partir de las nuevas resoluciones de la Aduana General de la República.

Definiéndose como **objeto de estudio**: el control de las mercancías en los Depósitos de Aduana.

Según lo planteado anteriormente se establece como **campo de acción**: los procesos asociados al control de las mercancías en los Depósitos de Aduana.

Para dar solución al problema planteado se trazó como **Objetivo general**: Desarrollar un subsistema para el sistema GINA que permita el control de las mercancías en los depósitos de aduana, integrado al sistema de Ventanilla Única a partir de las nuevas resoluciones.

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

1. Definir el estado actual del proceso de control de las mercancías en los Depósitos de la Aduana para la elaboración del marco teórico-conceptual sobre el que abordará el presente trabajo.
2. Describir los requisitos del subsistema que se desarrolla para la realización de un control efectivo de las mercancías en los depósitos.
3. Realizar el diseño y la implementación de la solución propuesta para que el subsistema cumpla con las necesidades del cliente.
4. Validar la solución desarrollada para la obtención de un correcto funcionamiento del subsistema.

Se plantea la siguiente **Idea a defender**: Si se desarrolla un subsistema informático integrado al sistema de Ventanilla Única, permitirá que se gestionen los procesos asociados al control de las mercancías en los Depósitos de Aduana teniendo en cuenta las nuevas resoluciones.

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

1. Elaboración de la fundamentación teórica y la revisión bibliográfica.
2. Especificación de los requisitos.
3. Validación de los requisitos.
4. Diseño de la base de datos.

- 5 Descripción del modelo de datos.
- 6 Diseño e implementación de las interfaces de usuario.
- 7 Diseño del diagrama de clases del diseño.
- 8 Diseño de los diagramas de secuencia.
- 9 Implementación de los requisitos.
- 10 Diseño de los casos de prueba.
- 11 Aplicación de pruebas de caja negra al subsistema.
- 12 Documentación de los resultados de las pruebas al subsistema.

Los **Posibles resultados** son: obtener un sistema que, integrado al sistema de Ventanilla Única, permita controlar las mercancías en los depósitos de aduana teniendo en cuenta las nuevas resoluciones.

Los Métodos científicos de investigación utilizados son:

Métodos teóricos

Analítico-Sintético: El análisis permite descomponer un todo en sus partes y cualidades; la síntesis es la operación inversa que provoca la unión entre las partes previamente analizadas, lo que posibilita hallar relaciones y características generales de la realidad. Este método contribuye al análisis del proceso de control de las mercancías llevado a cabo en los depósitos de aduana, para así entender mejor sus características y funciones; facilitando el análisis y estudio de las partes fundamentales de este proceso.

Método modelación: Este método representa de manera simplificada la realidad, lo que permite crear modelos con vista a la realidad. Se aplica en la generación de los artefactos necesarios para el análisis y diseño del proceso de control de las mercancías en los depósitos de aduana.

Métodos empíricos

Entrevista: Este método facilita la captura de los requisitos; consiste en una conversación entre el investigador y el o los funcionarios para la obtención de los requisitos. En este caso fue de gran ayuda para la definición de los requerimientos pertinentes para el análisis y diseño del proceso de control de las mercancías en los depósitos de aduana.

El trabajo está estructurado en tres 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 los Depósitos de Aduana, a nivel internacional y nacional. Detalla la metodología, lenguaje y herramientas utilizadas en el

desarrollo de la solución, técnicas en la ingeniería de requerimientos, y los patrones de software más utilizados.

El **segundo capítulo** se refiere a las características del sistema y los temas correspondientes al diseño; se exponen las técnicas utilizadas para la captura y validación de los requerimientos. Además se muestra un listado de los requerimientos funcionales con una pequeña descripción, el modelo conceptual, se muestran los diagramas de secuencia orientado a actividades, de clases del diseño con estereotipos Web, de clases persistentes, diseño de la base de datos y las métricas para la evaluación del diseño.

El **tercer capítulo** aborda temas correspondientes al desarrollo de la solución, describe cómo se implementan los elementos del modelo de diseño, define la estructura del producto y la construcción de la solución. Se enfatiza en la validación del sistema, empleando pruebas de caja negra 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.

Capítulo 1 Fundamentación teórica

1.1 Introducción

En el capítulo se describen los elementos principales que fundamentan el contenido de este trabajo. Se realiza un estudio profundo de las soluciones informáticas nacionales e internacionales para la gestión de los Depósitos de Aduana; así como de algunas de las técnicas de captura y validación de los requisitos. Además, se describen aspectos importantes de los estilos arquitectónicos, patrones de desarrollo de software, herramientas, metodología y lenguaje para el desarrollo del producto.

1.2 Generalidades de los Depósitos de Aduana en Cuba

El Régimen de Depósito de Aduanas comenzó a utilizarse en Cuba en 1992, el cual logró un incremento considerable en el año 1996 donde el Decreto Ley No. 162, del 3 de abril de 1996, de Aduanas, establece en su artículo 156, que el régimen de depósito de aduanas es aquel con arreglo al cual las mercancías importadas se almacenan bajo control aduanero en un lugar designado a este efecto, sin el pago de los derechos de aduana y en espera de que se les otorgue un nuevo régimen. (3)

El Régimen de Depósito de Aduanas en Cuba se encuentra regulado por la Resolución No. 12, de 3 de julio de 2002, del Jefe de la Aduana General de la República, Normas para la aplicación del Régimen Aduanero de Depósito de Aduanas y para los Locales, Instalaciones y Áreas destinadas a depositar las mercancías sujetas a dicho régimen.(4)

En Cuba los depósitos se clasifican en públicos y privados, existiendo tres modalidades a las que pueden acogerse los comerciantes a quienes interese disfrutar de las facilidades que brinda este régimen. Estas son: (4)

- Depósitos de carácter público: destinados a depositar mercancías de cualquiera que ostente la titularidad del régimen, mediante el arrendamiento de espacios o la contratación de servicios de almacenamiento, entre la administración del depósito y los titulares del régimen.
- Bodega Pública: es un espacio dentro del Depósito de carácter público que, formando parte de éste, está destinado a prestar los servicios de recepción, almacenamiento, custodia y facturación de las mercancías de varios depositantes.
- Depósito de carácter privado: destinados al uso exclusivo del titular que disfruta del régimen, al que se autoriza para el almacenamiento de mercancías propias de su actividad.

La utilización del Régimen de Depósito de Aduanas en Cuba, reporta numerosos beneficios tanto para las empresas importadoras y exportadoras cubanas, como para los empresarios

extranjeros que realizan actividades de comercio exterior en el país, entre los que se encuentran:

- Facilita a las sucursales extranjeras acreditadas en Cuba la comercialización de sus productos dentro del territorio nacional.
- El almacenaje de las mercancías en el depósito permite la realización de actividades tales como: toma de muestras para su correcta clasificación, operaciones de conservación y mantenimiento, reempaque, formación de lotes y otros cambios con el objetivo de mejorar su apariencia y facilitar su transportación o comercialización siempre que no se realicen transformaciones sustanciales. Las mercancías que se encuentran bajo el régimen de depósito de aduanas pueden transferirse de un depósito a otro dentro de la misma o hacia distinta jurisdicción aduanera mediante el régimen especial de tránsito aduanero.
- Este régimen permite a las empresas importadoras cubanas verificar directamente la calidad, requisitos técnicos y marcas de las mercancías que desean adquirir, sin necesidad de trasladarse al país del proveedor para la inspección de los embarques.

1.3 Soluciones informáticas para la gestión y control de las operaciones en los Depósitos de Aduana

1.3.1 Soluciones extranjeras

1.3.1.1TICA

El proyecto de Tecnología de Información para el Control Aduanero (TICA) es un componente del Plan Estratégico del Servicio de Aduanas que busca modernizar la gestión aduanera mediante el uso intensivo de la tecnología. Sistema aduanero que funciona en la Aduana de Costa Rica y desarrollado por la misma; opera desde el 2007, es considerado un ejemplo de los beneficios de la modernización de los sistemas fiscales, elaborado con la herramienta de desarrollo GeneXus 9.0 y SQL Server 2008 como gestor de base de datos. (7)

Las principales características que tiene el nuevo modelo al cual responde el sistema de información TICA son las siguientes:

- Implementación de las mejores prácticas en los procedimientos aduaneros
- Utilización de un formato único de declaración aduanera (DUA)
- Automatización del proceso aduanero (recepción, validación, arancel integrado, pago, aceptación, selectividad y levante electrónico, entre otros)
- Autodeterminación y pago electrónico
- Un sistema de información centralizado (una sola base de datos, registro de todas las operaciones)

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

- Control centralizado, basado en inteligencia del negocio y en la aplicación eficiente del análisis de riesgo
- Conexión electrónica obligatoria con organismos públicos y privados (exenciones, notas técnicas, bancos)
- Eliminación de la presentación de papeles
- Operación del sistema las 24 horas y 365 días al año
- Adaptabilidad del sistema a las necesidades del comercio exterior (aplicación de convenios internacionales, tratados comerciales)

Este sistema introducirá cambios relacionados con los procedimientos aduaneros en los depósitos y estos están relacionados con:

- Asignación de número de inventario al ingreso de mercancías (mantiene relación con los documentos de ingreso y ubicación dentro del depósito)
- Mensaje de fin de tránsito (captura fecha y hora, consulta participación aduana, programa descarga)
- Reempaque y distribución (fraccionamiento y agrupamiento)
- Control de cambio de ubicación de mercancías con levante autorizado (tres días después de autorización de levante)
- Control de salidas de mercancías
- Control de plazos y cambio a mercancías en abandono.
- Despacho a bordo (autorización y control)

1.3.1.2 S4 tr@nsERP

Presenta distintos módulos de depósito que están basados en los requerimientos de cada una de las figuras aduaneras que gestionan, de modo que, además de los controles clásicos de un almacén, se incorporan las transmisiones EDI (siglas de **Electronic Data Interchange**, intercambio electrónico de datos) y los diferentes listados de control requeridos por la Aduana. Pueden funcionar como una aplicación independiente o integrada con el resto de sus módulos.

(5)

Módulos:

- Almacén de Depósito Temporal (ADT)
- Depósito Aduanero
- Depósito distinto del aduanero
- Depósito fiscal
- Depósito para restitución

Características técnicas:

- Multiempresa, Multidelegación

Parametrizable:

- En acceso y búsqueda de la información
- En trazabilidad
- En funcionalidad y modelización de datos
- En reportes y análisis
- Control del acceso a la información por módulos de seguridad
- Comunicación por mail fax e internet
- Mensajería electrónica integrada

Integración:

- Con su gestión de almacén
- Con su web, personalizado para clientes, usuarios y colaboradores
- Con otros módulos de S-4
- Construido con herramientas Microsoft, SQL Server como almacén de datos. Intercambio con XML (del inglés, ***Extensible Markup Language***) y herramientas de Office System de Microsoft para importación y exportación de información

Depósito Aduanero:

- Dos versiones: Público y Privado.
- Inclusión en Depósito Aduanero Privado (IDA)
- Registro de entradas y salidas
- Ventas en Depósito
- Transformaciones en depósito (RUN)
- Traspasos entre Depósitos
- Mensajes EDI
- Listado de existencias
- Contabilidad de existencias
- Trazabilidad
- Integración con módulo de ADT
- Integración con módulo de Aduanas
- Integración con facturación emitida y soportada (costes)
- Exportación de datos a Office System.

1.3.1.3 SIDUNEA

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

El Sistema Aduanero Automatizado (SIDUNEA) es la 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. (6)

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.(6)

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. (6)

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 (WCO) y por la Organización Internacional para la Estandarización (ISO). (6)

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
- Minimizar el contrabando
- Administrar efectivamente el proceso de despacho

Gracias a la tecnología JAVA, SIDUNEA permite el uso de tarjetas Inteligentes con procesadores y tecnología JAVA para controlar los accesos al sistema y los pagos electrónicos. Así mismo, su aplicabilidad vía Internet, permite acceder al sistema a través de dispositivos inalámbricos. Además cuenta con una interfaz de usuario amigable y 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)

1.3.2 Soluciones cubanas

1.3.2.1 Sistema Único de Aduanas (SUA)

La Aduana de Cuba presenta su propia solución informática denominada Sistema Único de Aduanas (SUA), el cual se comenzó a desarrollar en el año 2004 por especialistas del Centro de Automatización y Dirección de la Informatización (CADI) de la Aduana General de la República. El Sistema Único de Aduana tiene como objetivo automatizar el procesamiento informativo referente a todas las operaciones que conforman los diferentes procesos, ya sea de Medio de Transporte Internacional, Importaciones y Exportaciones con y sin carácter comercial, Bultos Postales y Viajeros, posee una base de datos única y centralizada, a la que acceden en línea todas las aduanas del país. Este sistema brinda al funcionario aduanero todo lo necesario para su buen desempeño donde todas las partes se retroalimenten.

El SUA es un sistema web multiplataforma desarrollado bajo las premisas del software libre. Es una aplicación que utiliza como gestor de Base de Datos Oracle, lenguaje de programación PHP y paradigma de programación estructurada.

Actualmente el sistema SUA presenta un módulo o subsistema para la gestión y control de las operaciones que se realizan en los Depósitos de Aduana, el cual se encarga de realizar todas las operaciones correspondientes a los depósitos y facilita el trabajo del personal encargado de operar en los mismos. Pero este subsistema presenta problemas en el control de las mercancías debido a que toda la información correspondiente a los procesos que gestiona la recibe una parte de forma manual y otra parte en soportes digitales, lo que posibilita que en algunas ocasiones esta información contenga errores.

Conclusión del estudio de las Soluciones informáticas

Después de haber analizado las soluciones informáticas para la gestión y control de las operaciones que se realizan en los diferentes depósitos bajo control aduanero, se puede concluir que a pesar de que brindan una amplia gama de funcionalidades, no son factibles para utilizar en la AGR, debido a diferentes argumentos como son:

- El TICA implementa las legislaciones y normas específicas de Costa Rica, desarrollado con herramientas privativas como son: la herramienta de desarrollo GeneXus 9.0 y SQL Server 2008 como gestor de base de datos.
- El sistema S4 tr@nsERP es construido con herramientas de Microsoft, es privativo por lo cual para poder utilizarlo es necesario comprar la licencia, utiliza como gestor de base

de datos SQL Server y el paquete Office de Microsoft para importación y exportación de información.

- El SIDUNEA es reconocido como el sistema más completo para las Aduanas y cumple con las normas establecidas por la Organización Internacional para la Estandarización (ISO), la Organización Mundial de Aduanas (OMA) y la Organización Mundial del Comercio (OMC). Pero no es una solución factible para utilizarla en la AGR debido a que para poder utilizarlo es necesario comprarlo (el costo varía según el tamaño del país (número de puestos fronterizos y destacamentos aduaneros); el volumen del comercio exterior; las condiciones de su infraestructura física; el nivel de destreza de los funcionarios de gobierno y aduaneros; y si se trata de un país de tránsito o sin litoral marítimo u otra circunstancia).
- El SUA a pesar de que cumple con las políticas establecidas en la AGR no es una solución factible debido a que se torna engorroso el proceso de mantenimiento del mismo, el cual debe ser realizado constantemente debido a los constantes cambios que se realizan en las legislaciones aduaneras con el objetivo de elevar la calidad de sus servicios. Por otra parte el propio subsistema utilizado para gestionar los depósitos se ha quedado obsoleto por los cambios ocurridos, lo que trae consigo que una parte de la información se gestione de forma manual.

1.4 Metodología, Lenguaje y Herramientas de modelado utilizadas para el desarrollo.

El Departamento de Soluciones para la Aduana perteneciente al centro CEIGE se encuentra desarrollando el sistema GINA, el mismo ya tiene definida la tecnología a emplear para la confección del mismo, por lo cual la metodología de desarrollo, el lenguaje y las herramientas a utilizar para desarrollar la Versión 1.1 del subsistema Depósitos de Aduana son las que se fundamentan a continuación.

1.4.1 Metodología de Desarrollo de Software

Las metodologías de desarrollo de software describen los pasos que se deben seguir para la producción de un determinado producto informático, a través de patrones, que surgieron gracias a la experiencia en un determinado campo del desarrollo de software, como es el caso de los patrones de diseño, los cuales sugieren una serie de pasos a la hora de realizar un diseño determinado.

En el presente trabajo de diploma será utilizada como metodología de desarrollo lo establecido en el documento CEIGE-Modelo de Desarrollo de Software v1.1, el cual especifica que los proyectos del CEIGE transitan por dos fases: Inicio o Estudio preliminar y Desarrollo.

Fases por donde transitan los proyectos del CEIGE

Inicio o Estudio preliminar: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel, la evaluación de la factibilidad del proyecto y su registro. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, y decidir si se ejecuta o no el proyecto. Los objetivos de la fase son: (8)

- Asegurar la factibilidad del proyecto.
- Establecer un plan para la ejecución del proyecto.

Hitos:

- Plan de desarrollo de software.
- Acta de inicio del proyecto firmada.

Desarrollo: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se refinan los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. El objetivo de esta fase es:

- Obtener un sistema que satisfaga las necesidades de los clientes y usuarios finales.

Hito:

- Producto liberado por entidad certificadora de calidad.

En esta fase se ejecutan las disciplinas Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de liberación.

1.4.2 Lenguaje Unificado de Modelado (UML) 2.0

Es uno de los lenguajes de modelado de procesos más conocido y utilizado en la actualidad. Está respaldado por el OMG (**Object Management Group**, Grupo Manejador de Objetos). El UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Además, ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. (9)

1.4.3 Herramienta CASE Visual Paradigm for UML 8.0 para el modelado

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo de desarrollo de software: análisis y diseño orientado a objetivos, construcción, prueba y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagrama y generar documentación. (10)

1.4.4 Lenguaje de Programación PHP 5.3

PHP es un lenguaje de programación generalmente usado en la programación de sitios web dinámicos y actualmente es casi el lenguaje de desarrollo de sitios más usado en todo el mundo. Fue originalmente creado por Rasmus Lerdorf para presentar su portafolio de trabajo en el año 1994. Originalmente fue desarrollado en Perl. PHP al principio significaba Personal Home Page pero con el tiempo como ya es desarrollado por otro grupo se llama PHP Hypertext Preprocesor. (11)

Características del lenguaje PHP

- PHP es un lenguaje interpretado, solo se necesita un navegador web para ejecutarlo.
- Es un lenguaje del lado del servidor, por lo que los script se ejecutan remotamente y el resultado aparece en la máquina cliente (local).
- Tiene soporte para muchos tipos de bases de datos, entre las principales están MySQL, PostgreSQL, SQLite, Oracle, entre otras.
- La sintaxis es parecida a la del lenguaje C (Que también tiene un parecido a Perl).
- Es embebido en código HTML.
- Soporte de orientación a objetos.

1.4.5 Marco de Trabajo (Framework) Symfony 1.2.8

Es un completo marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Está desarrollado completamente en PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. (12) Symfony se diseñó para que se ajustara a los siguientes requisitos: (12)

- Independiente del sistema gestor de bases de datos.

- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web. Preparado para aplicaciones empresariales, adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

1.4.6 Interfaz de usuario ExtJS 3.0

Es un framework diseñado para la creación de páginas web dinámicas del lado del cliente basado en lenguaje JavaScript, permite una gran reutilización de componentes, personalización de los mismos, haciendo uso del manejo de tecnologías como AJAX (del inglés, ***Asynchronous JavaScript And XML***) para el intercambio asincrónico de los datos entre el cliente y el servidor, además de lo ventajoso que puede representar la similitud de su aspecto con el de una aplicación de escritorio. Sus características principales son: gran desempeño, componentes de interfaz de usuario personalizables, con buen diseño y documentación. (13)

1.4.7 Entorno de Desarrollo Integrado Netbeans IDE 7.2

Es una herramienta para el desarrollo de programas. Soporta lenguajes de programación: Java, PHP, C/C++ entre los más importantes. Netbeans IDE 7.2 es una herramienta de programación integrada. Está enfocado al lenguaje de Programación Java, pero actualmente soporta PHP, C/C++, JavaScript, HTML entre otros. Viene integrado con servidores de aplicaciones GlassFish v3, Apache Tomcat y maneja Bases de Datos MySQL, PostgreSQL y cualquiera que se conecte con JDBC como Oracle, SQL Server y otros más.

1.4.8 Gestor de Base de Datos Oracle 11g

Es un sistema de gestión de base de datos objeto-relacional (ORDBMS por el acrónimo en inglés de ***Object-Relational Data Base Management System***), desarrollado por la empresa Oracle. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando:

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

- Su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS⁴ con licencia libre como PostgreSQL, MySQL o Firebird⁵. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux. Las principales funcionalidades aportadas por todo el SGBD Oracle son:
- Soporte y tratamiento de una gran cantidad de datos (Gbytes).
- 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.
- Conectividad entre las aplicaciones de los clientes en sus puestos de trabajo y el servidor de base de datos Oracle (estructura cliente/servidor).
- Conectividad entre bases de datos remotas (estructura de bases de datos distribuidas).
- Portabilidad.
- Compatibilidad.

1.5 Ingeniería de requerimientos

Existen diversas definiciones de requerimiento, por ejemplo Ian Sommerville presenta una definición de acerca de lo que es requerimiento que plantea: “Un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este.” (14)

Otra definición de requerimiento se puede encontrar en el glosario de Terminología de Ingeniería de Software de la IEEE y lo define como: “Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo.” (15)

“Una condición o capacidad que debe estar presente en un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formal.” (15)

La Ingeniería de Requerimientos cumple un papel primordial en el proceso de producción de software, ya que se enfoca en un área fundamental: definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedad, en forma consistente y compacta, las necesidades de los usuarios o

⁴ RDBMS (acrónimo en inglés de Relational DataBase Management Systems), son programas que permiten organizar datos en una o más tablas relacionadas.

⁵ Sistema de administración de bases de datos relacional de código abierto.

clientes; de esta manera, se pretende minimizar los problemas relacionados por la mala gestión de requisitos en el desarrollo de sistemas. (16)

Otros de los conceptos de la ingeniería de requerimientos son propuestos por Pressman el cual plantea. “Ingeniería de Requerimientos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajan. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactúan los usuarios finales con el software.” (17) En este trabajo se entiende por IR este concepto propuesto por Pressman.

La finalidad de la disciplina de requisitos es:

- Establecer y mantener un acuerdo con los clientes y otros interesados acerca de lo que debe hacer el sistema.
- Proporcionar desarrolladores de sistema con un buen conocimiento de los requisitos del sistema.
- Definir los límites del sistema (delimitarlo).
- Proporcionar una base para planificar el contenido técnico de las iteraciones.
- Proporcionar una base para la estimación del coste y del tiempo en que desarrollar el sistema.
- Definir una interfaz de usuario para el sistema, centrándose en las necesidades y los objetivos de los usuarios.

1.5.1 Actividades de la Ingeniería de Requerimientos

Las actividades de la Ingeniería de Requerimientos ayudan a reconocer la importancia que tiene realizar una especificación y administración adecuada de los requerimientos de los clientes o usuarios para el desarrollo de un proyecto de software. Esta disciplina explica cómo obtener las solicitudes de los interesados y transformarlas en un conjunto de productos de trabajo de los requisitos que cubran el ámbito del sistema que va a crearse y proporcionen requisitos detallados sobre lo que el sistema debe hacer.

En el proceso de definición de las necesidades del sistema se deben identificar los requisitos que el sistema debe cumplir para satisfacer las necesidades de los usuarios finales. Para esto no existe una técnica estandarizada y estructurada que brinde un marco de desarrollo que avale la calidad del resultado.

La ingeniería de requisitos cuenta con 5 actividades, de ellas fueron utilizadas en el presente trabajo solo 3 y son las que a continuación:

Captura de requisitos

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema. Por la complejidad que esta actividad puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa. (18)

Algunas de las técnicas de captura de requerimientos son:

Entrevistas: Las entrevistas se emplean para reunir información proveniente de personas o de grupos. El éxito de esta técnica, depende de la habilidad del entrevistador y de su preparación para la misma.

Lluvia de ideas (Brainstorm): Este es un modelo que se usa para generar ideas. La intención en su aplicación es la de generar la máxima cantidad posible de requerimientos para el sistema.

Especificación de Requerimientos

En esta fase se documentan los requerimientos acordados con el cliente, en un nivel aprobado de detalles. En la práctica esta actividad se va desarrollando en conjunto con el análisis debido a que en esta se plasman en documentos los requerimientos del cliente.

Para la actividad especificación de requisitos también existen varias técnicas, a continuación se mencionan las empleadas en este trabajo:

Plantillas o patrones: Se utilizan para describir los requisitos mediante el lenguaje natural, pero de una manera estructurada.

Validación de requisitos

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias. (18)

1.6 Diseño de software

En el diseño se modela el sistema y se encuentra la forma para que soporte todos los requerimientos, incluyendo los requisitos no funcionales, se asienta en el núcleo técnico del proceso de ingeniería del software y se aplica independientemente del paradigma de desarrollo utilizado. En la elaboración de la solución es preciso tener presente ciertos estándares, estilos y patrones que son muy útiles en la obtención de un diseño de software robusto. (17)

1.6.1 Patrones de Diseño

Dentro de las definiciones de patrones se encuentra la propuesta por Craig Larman: “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.” (19)

¿Qué son los patrones?

- Solucionan un problema: los patrones capturan soluciones, no solo principios o estrategias abstractas. (20)
- Son un concepto probado: capturan soluciones demostradas, no teorías o especulaciones. (20)
- La solución no es obvia: los mejores patrones generan una solución a un problema de forma indirecta. (20)
- Describen participantes y relaciones entre ellos: describen módulos, estructuras del sistema y mecanismos complejos. (20)

Los Patrones de Diseño no son más que soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan, 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.

Patrones GRASP

GRASP es un acrónimo que significa General Responsibility Assignment Software Patterns (en español, patrones generales de software para asignar responsabilidades) y representa la descripción de los principios fundamentales de la asignación de responsabilidades expresados como patrones. (21)

El nombre se eligió para indicar la importancia de captar (*grasping*) estos principios, si se quiere diseñar eficazmente el software orientado a objetos. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Tienen como motivación:

- La asignación de responsabilidades como la habilidad más importante en el análisis y diseño orientado por objetos. (21)

- Respetar los principios fundamentales como uno de los factores críticos, para obtener diseños reutilizables, mantenibles y extendibles. (21)

Patrones GoF

El grupo de GoF (*del inglés, **Gang of Four***) clasificaron los patrones en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. (22)

- Creacionales: los patrones creacionales se relacionan con las formas de crear instancias de objetos. El objetivo de estos patrones es abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados. (22)
- Estructurales: los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionales pueden ser incluso objetos simples u objetos compuestos. (22)
- Comportamiento: los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos. (22)

1.6.2 Arquitectura de software

Existen diversas definiciones formales referentes al término arquitectura de software como por ejemplo.

“La arquitectura del software de un programa o sistema de computación es la estructura o estructuras del sistema que comprende los elementos del software, las propiedades externamente visibles de esos elementos, y las relaciones entre ellos”. (23)

“La Arquitectura de Software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones”. (24)

Esta definición es un poco amplia, por lo que una de las más aplicadas y la que se toma como referencia en este trabajo es la establecida por la 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.”

1.6.2.1 Estilos Arquitectónicos

Los estilos arquitectónicos no son patrones pertenecientes a un tipo de vista concreto, que definen una serie de restricciones a los tipos de elementos y relaciones de la vista arquitectónica. (25)

Dentro de los estilos arquitectónicos existentes se encuentra el estilo de Llamada y Retorno y dentro de este estilo se encuentran los estilos:

- Model-View-Controller (MVC).
- Arquitecturas en Capas.
- Arquitecturas Orientadas a Objetos.
- Arquitecturas Basadas en Componentes.

1.6.2.2 Patrones de arquitectura

Los patrones de arquitectura están relacionados fundamentalmente con la estructura de un sistema de software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes. Describen un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. (26)

Patrón Modelo Vista Controlador (MVC)

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres capas o componentes distintos. El patrón MVC se utiliza frecuentemente en aplicaciones web, donde la vista es la página HTML (*del inglés, **Hypertext Markup Language***) que se visualiza en el navegador y el código que proporciona de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista, enviarlos al modelo y manejar también las respuestas del modelo y transmitirlos hacia la vista. (27)

Modelo

El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar. El modelo desconoce la existencia de las vistas y del controlador. Ese enfoque suena interesante, pero en la práctica no es aplicable pues deben existir interfaces que permitan a los módulos comunicarse entre sí. Esta es la recepción específica del dominio de la información sobre la cual funciona la aplicación. El modelo es una forma de llamar la capa de dominio. La lógica de dominio añade significados a los datos. (27)

Vista

Las vistas son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo. Una vista obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada vez que el modelo del dominio cambia por medio de notificaciones generadas por el modelo de la aplicación. (27)

Controlador

El controlador es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador. (27)

1.7 Conclusiones parciales

La investigación realizada trajo como resultados un estudio de los conceptos, normas y resoluciones principales sobre el régimen de depósito de aduanas, así como el funcionamiento, estructura y clasificación en Cuba.

El estudio de los sistemas extranjeros seleccionados que son utilizados para la gestión de los depósitos ayudó a definir que no son factibles porque en su totalidad son privativos, lo cual provoca gastos de grandes sumas de dinero por concepto de licencia, soporte y compra. Además de que no cumplen con las legislaciones y normativas aduaneras cubanas vigentes para el funcionamiento y control de los Depósitos de Aduana.

Por otra parte al analizar la solución cubana como se explica anteriormente se concluye que es una solución que cumple con las políticas cubanas pero no satisface las necesidades actuales existentes en la Aduana correspondiente a la gestión y control de las mercancías en los depósitos por lo que se necesita desarrollar subsistema que permita intercambiar información con todos los subsistemas del GINA.

Durante la ejecución de las disciplinas de desarrollo requisitos, análisis, diseño e implementación, las herramientas, técnicas, lenguajes y tecnologías a utilizar serán las definidas por el CEIGE.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Capítulo 2: Descripción de la solución propuesta

2.1 Introducción

En el capítulo se presenta una propuesta del sistema a desarrollar, para esto se muestra el modelo conceptual, los requerimientos funcionales definidos para el sistema y las técnicas de captura y validación de requerimientos utilizadas. También se describe el diseño del sistema, los diagrama de clases del diseño con estereotipos web, el diagrama de paquetes y los diagramas de secuencia de cada uno de los requisitos funcionales de manera que se garantice que el diseño esté lo más orientado posible a la programación y asegurar así una mayor rapidez y calidad en este proceso. Además se muestran los patrones utilizados en la solución, el diagrama de clases persistentes, el diseño de la base de datos y las métricas para la evaluación del diseño.

2.2 Modelo Conceptual

El modelo conceptual explica cuáles son y cómo se relacionan los conceptos relevantes en la descripción del problema. Es aquella representación del sistema por medio de definiciones organizadas en forma estructurada. (28)

El modelo conceptual muestra los conceptos, los atributos y las asociaciones entre los conceptos. (29)

A continuación se muestra el modelo conceptual del negocio, el cual presenta los conceptos, los atributos y las relaciones entre los conceptos que intervienen en el proceso de control de las mercancías en los depósitos de aduana.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

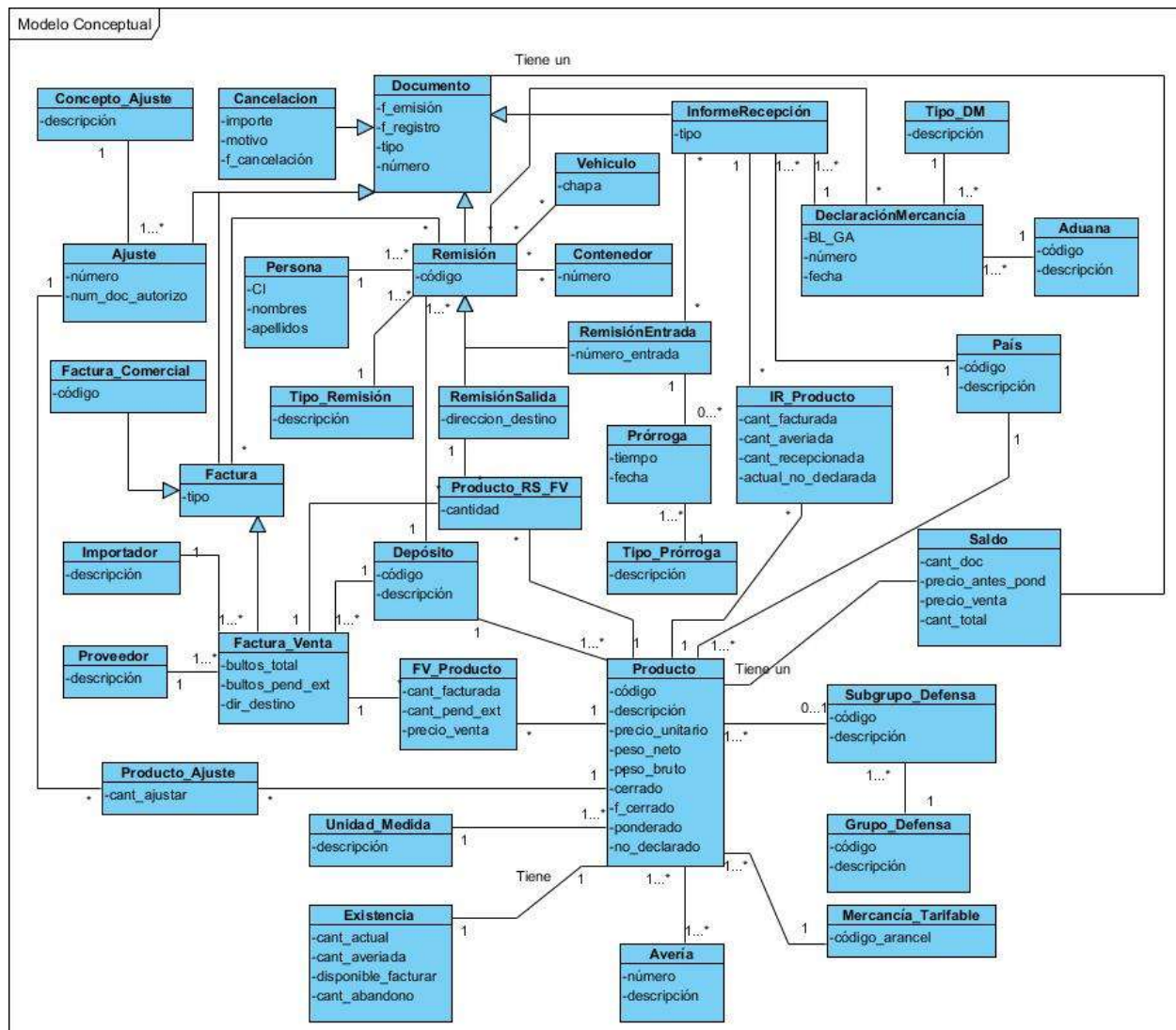


Figura 1 Modelo Conceptual de Depósitos de Aduana

2.3 Especificación de los Requerimientos del Sistema

Con la especificación de los requisitos de software se describe cada detalle a tener en cuenta para el desarrollo del sistema lo que ayuda a evitar errores en posteriores etapas; por lo que la utilización de técnicas para la captura y validación asegura la calidad de estos requisitos. A continuación se muestran las técnicas utilizadas y el listado de los requisitos funcionales definidos para el sistema.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

2.3.1 Técnicas para la Captura de Requerimientos

Para la recopilación y obtención de la información necesaria para la captura de requisitos, se utilizaron las siguientes técnicas: la entrevista, la observación y la tormenta de ideas.

La **entrevista** se utilizó con el fin de obtener un intercambio mediante preguntas con los especialistas, inspectores de los depósitos, funcionarios del Departamento de Técnicas Aduaneras de la AGR, Jefes de los depósitos y en visitas realizadas a las instalaciones para esclarecer con precisión el funcionamiento de todo el trabajo en los Depósitos de Aduana. La **observación** se llevó a cabo para percibir como se desarrollan las operaciones en los distintos depósitos, pudiendo captar directamente las particularidades de estos procesos en las visitas realizadas; así como la **tormenta de ideas**, donde en conjunto con la especialista de la Aduana que atiende los Depósitos de Aduana, inspectores, los jefes de los depósitos y el personal de técnica aduanera, se acumularon ideas para tener una perspectiva general de las necesidades del sistema.

2.3.2 Requerimientos Funcionales

Los requerimientos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer. (14)

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

Como resultado de haber aplicado las técnicas de captura de requerimientos abordadas en el sub epígrafe anterior se obtuvo como resultado un total de 39 requisitos funcionales, los cuales se encuentran agrupados en doce agrupaciones. A continuación se exponen cada una de las agrupaciones y los requisitos funcionales que pertenecen a cada una de ellas.

Agrupación 1 Procesar documentos

- **RF Procesar remisión de entrada:** el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.
- **RF Procesar informe de recepción:** el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

- **RF Procesar remisión de salida:** el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.
- **RF Procesar factura de venta:** el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.
- **RF Procesar información de ajuste:** el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.
- **RF Procesar información de cancelación:** el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.

Agrupación 2 Mostrar remisiones

- **RF Mostrar datos de la Remisión de Entrada:** el sistema debe mostrar todos los datos asociados a una Remisión de Entrada.
- **RF Mostrar datos de la Remisión de Salida:** el sistema debe mostrar todos los datos asociados a una Remisión de Salida.
- **RF Buscar Remisiones:** el sistema debe buscar todas las Remisiones.

Agrupación 3 Mostrar informes de recepción

- **RF Mostrar datos del informe de recepción:** el sistema debe mostrar todos los datos asociados a un Informe de Recepción.
- **RF Mostrar informes de recepción realizados:** el sistema debe mostrar los datos generales asociados a un Informe de Recepción.
- **RF Mostrar informes de recepción parciales:** el sistema debe agrupar los Informes de Recepción, de cada depósito, según la parcialidad de los mismos.
- **RF Mostrar informes de recepción por descripción de producto:** el sistema debe mostrar los Informes de Recepción que presenta un determinado producto.
- **RF Mostrar informes de recepción por criterio (Avería, Faltante, Sobrante o No Declarada):** El sistema debe mostrar de cada depósito, todos los Informes de Recepción que presenten mercancías con avería, faltante, sobrante o no declarada.
- **RF Mostrar informes de recepción por país de procedencia:** el sistema debe mostrar los Informes de Recepción que se han realizado según el país de procedencia.

Agrupación 4 Mostrar facturas

- **RF Mostrar datos de la factura de venta:** el sistema debe mostrar todos los datos asociados a una Factura de Venta.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

- **RF Buscar productos de la factura de venta:** el sistema debe mostrar los datos asociados a los productos y las Facturas de Venta a las que pertenecen los mismos.
- **RF Mostrar estado de la factura de venta:** el sistema debe mostrar el estado en que se encuentran las Facturas de Venta de cada depósito.

Agrupación 5 Mostrar ajustes

- **RF Mostrar datos de Ajuste:** el sistema debe ser capaz de mostrar los datos asociados a los Ajustes que se realizan en cada depósito.
- **RF Mostrar Ajustes por producto:** el sistema debe mostrar los Ajustes que se han realizado para un determinado producto.

Agrupación 6 Mostrar datos de la notificación de Averías

- **RF Mostrar datos de la notificación de averías:** el sistema debe ser capaz de mostrar los datos asociados a las Notificaciones de Avería que se realizan en cada depósito.

Agrupación 7 Mostrar documentos cancelados

- **RF Mostrar documentos cancelados:** el sistema debe mostrar los datos de los documentos que han sido cancelados en cada depósito.

Agrupación 8 Gestionar infracciones

- **RF Identificar infracciones:** el sistema debe ser capaz de identificar las infracciones en cada depósito.
- **RF Mostrar alertas de infracciones:** el sistema debe ser capaz de mostrar los datos asociados a las alertas de infracciones en cada depósito.

Agrupación 9 Gestionar inventarios

- **RF Mostrar reporte de inventario:** el sistema debe ser capaz de mostrar un reporte de inventario en cada depósito.
- **RF Mostrar histórico por referencia de producto:** el sistema debe ser capaz de mostrar los datos asociados al histórico por referencia de producto en cada depósito.
- **RF Mostrar saldo total por depósito:** el sistema debe ser capaz de mostrar el saldo total en cada depósito.

Agrupación 10 Gestionar prórrogas

- **RF Registrar prórroga:** el sistema debe ser capaz de registrar prórrogas en cada depósito.
- **RF Buscar prórrogas autorizadas:** el sistema debe ser capaz de buscar los datos asociados a las prórrogas autorizadas en cada depósito.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Agrupación 11 Mostrar incidencias

- **RF Mostrar remisión de entrada sin informe de recepción:** el sistema debe ser capaz de mostrar los datos asociados a las remisiones de entrada sin informe de recepción que se reciben en cada depósito.
- **RF Mostrar informe de recepción sin declaración de mercancías:** el sistema debe ser capaz de mostrar los datos asociados a los informes de recepción sin declaraciones de mercancías que se reciben en cada depósito.
- **RF Mostrar informe de recepción fuera de tiempo:** el sistema debe ser capaz de mostrar los datos asociados a los IR que estén fuera de tiempo en cada depósito.
- **RF Mostrar depósito con tiempo sin operar:** el sistema debe ser capaz de mostrar los datos asociados a los depósitos con tiempo sin operar.
- **RF Mostrar mercancías con tiempo sin operar:** el sistema debe ser capaz de mostrar los datos asociados a las mercancías con tiempo sin operar en cada depósito.
- **RF Mostrar mercancías en abandono y próximas al abandono:** el sistema debe ser capaz de mostrar los datos asociados a las mercancías en abandono y próximas al abandono en cada depósito.

Agrupación 12 Mostrar mercancías

- **RF36 Mostrar mercancías en existencia:** el sistema debe ser capaz de mostrar los datos asociados a las mercancías en existencia en cada depósito.
- **RF37 Mostrar mercancías por país de origen:** el sistema debe ser capaz de mostrar los datos asociados a las mercancías por país de origen en cada depósito.
- **RF38 Mostrar mercancías para la defensa:** el sistema debe ser capaz de mostrar los datos asociados a las mercancías para la defensa en cada depósito.
- **RF39 Mostrar mercancías no declaradas:** el sistema debe ser capaz de mostrar los datos asociados a las mercancías no declaradas en cada depósito.

Los requisitos fueron descritos usando una plantilla establecida por el centro CEIGE en su modelo de desarrollo y un ejemplo de la misma se puede ver en la Tabla 1 Descripción textual del requisito Mostrar datos de la remisión de entrada.

Tabla 1 Descripción textual del requisito Mostrar datos de la remisión de entrada.

Precondiciones	El usuario se ha autenticado correctamente.
Flujo de eventos	
Flujo básico	Mostrar datos de la remisión de entrada

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

1	Seleccionar del menú la opción Documentos – Remisiones – Remisión de entrada.
2	El sistema muestra una pantalla para realizar la operación que corresponde.
3	Filtrar la búsqueda según los datos que desee: código de la remisión de entrada, rango de fechas y código del depósito.
4	En el caso de que especifique el código del depósito, muestra su código y descripción.
5	Seleccionar la opción Buscar.
6	El sistema muestra el rango de fechas especificado, en el caso de que no especifique ninguno muestra como fecha inicial el primer día del año en curso y como fecha final el día en curso.
7	El sistema muestra los siguientes datos de cada remisión de entrada cuya fecha de remisión se corresponda con la especificada, agrupadas por depósito: número, fecha, tipo, aduana entrada salida, número de cancelación (en el caso de que hubiese sido cancelada).
8	Seleccionar una remisión de entrada.
9	El sistema muestra los siguientes datos de cada declaración de mercancías asociada a la remisión de entrada seleccionada: número, bl/ga, tipo y fecha de emisión.
10	El sistema muestra el número de cada factura comercial asociada a la remisión de entrada seleccionada.
11	El sistema muestra los siguientes datos de cada informe de recepción asociado a la remisión de entrada seleccionada: número, fecha de emisión y tipo.
12	El sistema muestra los siguientes datos del chofer asociado a la remisión de entrada seleccionada: nombre y apellidos, carnet de identidad, licencia de conducción y hora de llegada.
13	El sistema muestra los siguientes datos de cada vehículo asociado a la remisión de entrada seleccionada: número de la chapa, número del contenedor (en el caso de que hubiese).
14	Seleccionar un contenedor.
15	El sistema muestra el número de cada sello asociado al contenedor.
16	Concluye el requisito.
Pos-condiciones	
1	Se visualizan los datos de la remisión de entrada.
Validaciones	
1	Se validan los datos según lo establecido en el Modelo Conceptual.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Conceptos	Remisión	Visibles en la interfaz: Número Fecha Utilizados internamente: Tipo
	Remisión de Entrada	Utilizados internamente: Identificador
	Factura de venta	Visibles en la interfaz: Número
	Informe de recepción	Visibles en la interfaz: Número Tipo Fecha
	DM	Visibles en la interfaz: Número BL/GA Tipo Fecha
	Chofer	Visibles en la interfaz: Nombre y apellidos Licencia CI Hora
	Vehículo	Visibles en la interfaz: Chapa
	Contenedor	Visibles en la interfaz: Número
	Sello	Visibles en la interfaz: Número
	Aduana	Visibles en la interfaz: Descripción
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

2.3.3 Técnicas para la Validación de Requerimientos

En el procedimiento propuesto por el CEIGE se recomienda el uso de varias técnicas para la correcta validación de los requisitos, las cuales son: revisiones del documento de requerimientos y la construcción de prototipos.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Se realizaron diversas revisiones al documento de requerimientos para lograr una correcta interpretación de la información transmitida, los señalamientos planteados fueron recogidos y aplicados posteriormente. Además se efectuó la construcción de prototipos, los cuales fueron presentados por cada requerimiento de software, aclarando con la Especialista Principal de Sistemas Automatizados del CADI, si las necesidades del área de trabajo Depósitos de Aduana fueron cubiertas por el sistema.

2.3.4 Requisitos No Funcionales

Los requisitos no funcionales son los establecidos para el sistema GINA, debido a que se desarrolla un subsistema que forma parte de este, estos requisitos son descritos en el documento: CIG-ADU-N-GS Requisitos Software GINA, el cual se encuentra en el expediente del proyecto. Se encuentran descritos de acuerdo a su clasificación: usabilidad, fiabilidad, eficiencia y de soporte.

2.4 Patrones de diseño utilizados

El desarrollo del sistema empleando el framework Symfony proporciona ventajas significativas para los desarrolladores de software. El marco de trabajo mencionado es capaz de fusionar buenas prácticas de trabajo por sí mismo, de forma que los desarrolladores no tengan que preocuparse por implementar varios de los patrones de diseño y arquitectónicos más utilizados en la actualidad, ya que el mismo framework los maneja. A continuación se exponen varios patrones de diseño que han sido utilizados durante el desarrollo de la solución:

2.4.1 Patrones GRASP

- Creador: en la clase aduanaActions se localizan las acciones definidas para el módulo, en las cuales se crean los objetos de las clases que representan las entidades, confirmando que la clase es “creador” de dichas entidades.
- Controlador: Symfony implementa un controlador frontal para atender todas las peticiones web (sfActions), es el único punto de entrada de toda la aplicación en un determinado entorno. Cuando recibe una petición, utiliza el sistema de enrutamiento para enviar la acción al controlador responsable de la solución y se encarga de manejar la seguridad y ejecución de los filtros. Este patrón se evidencia en las clases sfFrontController, sfWebFrontController, sfContext, los “actions” y el index.php del ambiente.
- Alta Cohesión: Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

ello es la clase `adunaActions`, es responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades. Está formada por varias funcionalidades que están estrechamente relacionadas.

- Bajo Acoplamiento: la clase `adunaActions` hereda únicamente de `sfActions` para alcanzar un bajo acoplamiento de clases. 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.
- Experto: Symfony utiliza Propel para realizar su capa de abstracción en el modelo de, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos (Peer del Modelo) poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan.

2.4.2 Patrones GoF

- Singleton: la clase `sfRouting` presenta el método `getInstance()`, es utilizada por el controlador frontal (`sfWebFrontController`) y se encarga de enrutar todas las peticiones que se hagan a la aplicación. `sfRouting` garantiza la existencia de una única instancia y la creación de un mecanismo de acceso global a dicha instancia.
- Command: se encuentra presente en la clase `sfWebFrontController`, en el método `dispatch()`, es la encargada de establecer el módulo y la acción que se va a usar según la petición del usuario. Además se aplica en la clase `sfRouting`, que está desactivada por defecto y procede según las necesidades del administrador del sistema donde se aplique el framework, la cual se puede activar o desactivar.
- Decorator: la clase abstracta `sfView`, padre de todas las vistas, contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo `layout.php`, conocido como plantilla global, contiene el código HTML que es tradicional en todas las páginas del sistema, para no tener que repetirlo. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla.

2.4.3 Patrón MVC según Symfony

El patrón MVC obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el framework, todo lo relacionado con la interfaz de usuario se guarda en la vista, la manipulación de datos se guarda en el modelo y el procesamiento de las peticiones constituye el controlador. El empleo del patrón MVC en un sistema resulta bastante útil además de restrictivo.

Symfony está basado en el patrón de arquitectura conocido MVC, formado por tres niveles: el Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio, la Vista es la encargada de originar las páginas que son mostradas como resultado de las acciones y el Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. (12)

La implementación que realiza Symfony de la arquitectura MVC incluye varias clases como son:

- `sfController`: es la clase del controlador y se encarga de decodificar la petición y transferirla a la acción correspondiente.
- `sfRequest`: guarda todos los elementos que integran la petición (parámetros, cookies, cabeceras, entre otros).
- `sfResponse`: posee las cabeceras de la respuesta y los contenidos. El contenido de este objeto se convierte en la respuesta HTML que se remite al usuario.
- El singleton de contexto (que se obtiene mediante `sfContext::getInstance()`): guarda una referencia a todos los objetos que constituyen el núcleo de Symfony y puede ser accedido desde cualquier parte de la aplicación.

2.5 Diseño del sistema

En el desarrollo o ciclo de vida de un sistema informático, el diseño del sistema constituye un elemento fundamental del mismo. Se centra en proporcionar la funcionalidad del sistema a través de sus diferentes componentes. En esta disciplina se diseña una solución que convierte los requisitos del cliente en un sistema de información real.

El diseño debe proporcionar una completa idea de lo que es el software, enfocando los dominios de datos, funcional y comportamiento desde el punto de vista de la Implementación.

2.5.1 Diagrama de clases del Diseño con Estereotipos Web

Los estereotipos son el mecanismo de extensibilidad incorporado más utilizado dentro de UML. Un estereotipo representa una distinción de uso, puede ser aplicado a cualquier elemento de

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

modelado. Las extensiones UML para el diseño Web, exponen la solución para la modelación de diagramas de clases del diseño sobre tecnologías Web.

A continuación se presenta el diagrama de clases del diseño con estereotipos web correspondiente al RF Mostrar datos de la Remisión de Entrada. En este se puede observar que la página servidora `aduanasActrion.class.php` se encarga de atender las peticiones que realiza el controlador frontal; luego estas se redireccionan al layout o plantilla global, que es quien almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El layout carga en el cuerpo la plantilla o página cliente (`indexSuccess.php`) que importa todas las interfaces de usuario del módulo, estas interfaces se encuentran representadas dentro del paquete `InterfacesJS`. Es muy frecuente la existencia de una relación de agregación entre la página cliente y los formularios, que contienen los componentes para la entrada de datos que serán enviados al servidor y procesados por las funciones de la clase `aduanasActrion.class` con la utilización de las clases del paquete `Model`, que garantiza el registro y obtención de la información (ver Figura 2).

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

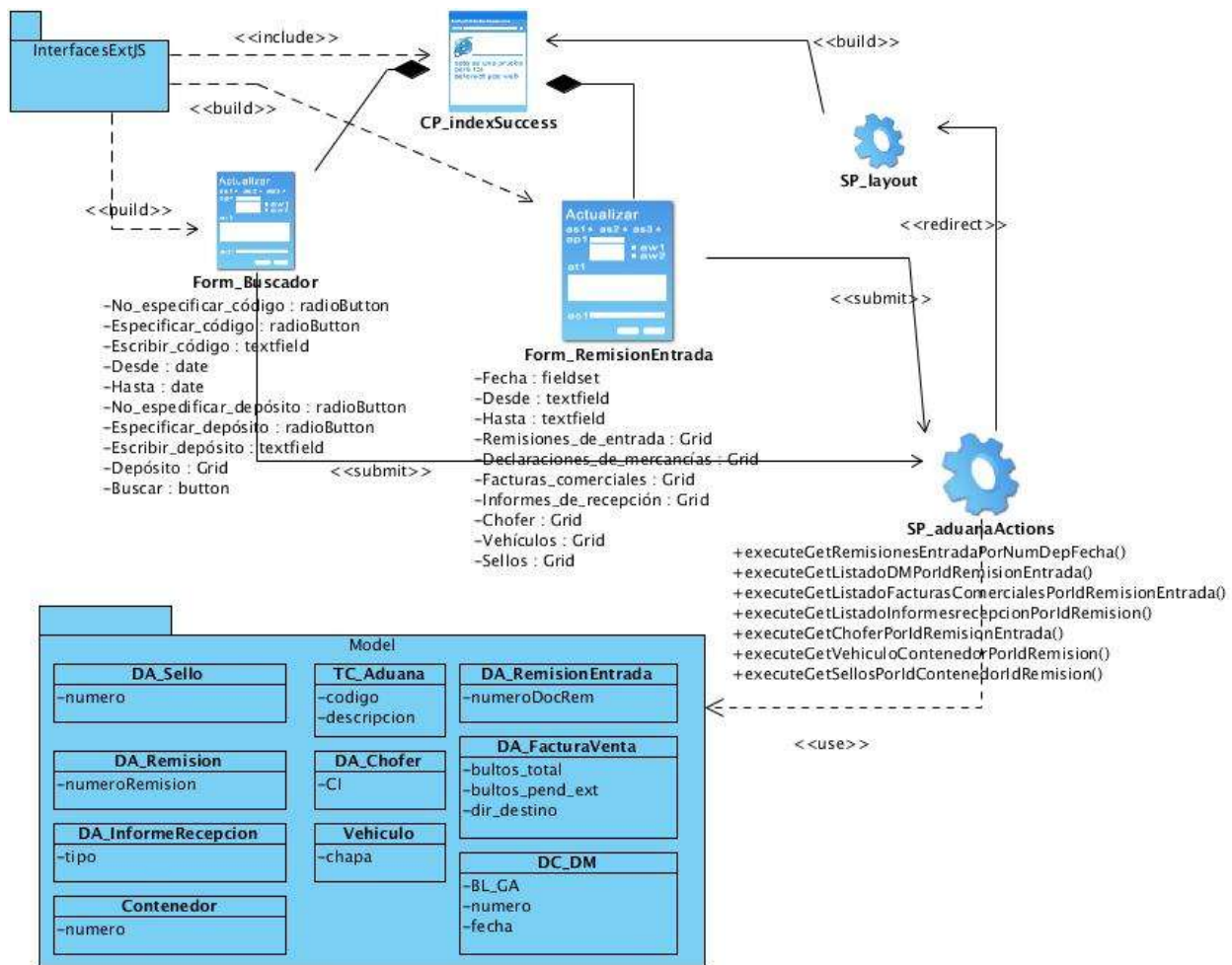


Figura 2 Diagrama de clases del diseño con estereotipos web del RF Mostrar datos de la remisión de entrada

2.5.2 Diagrama de Secuencia

El diagrama de secuencias, que se define en UML (Unified Modeling Language), es uno de los más utilizados para identificar el comportamiento de un sistema, por representar los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por una transacción del sistema. (30)

La figura 3 muestra un fragmento de un diagrama de secuencia (Ver Anexo II), el mismo pertenece al RF Mostrar datos de la remisión de entrada, como se puede observar se inicia cuando se selecciona del menú la opción Documentos – Remisiones – Remisión de entrada. Posterior a esto se muestra una pantalla para realizar la operación que corresponde y se filtra la búsqueda dando clic en el botón buscar del filtro. Luego son enviados los datos al servidor, los que son obtenidos por el controlador aduanaAction.class.php, el cual es el encargado de

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

realizar una llamada a la funcionalidad `GetRemisionesEntradaPorNumDepFecha($numRE, $desde, $hasta, $codDep)` que se encuentra en la clase `DaRemisionEntradaPeer.php` y se encarga de realizar las validaciones pertinentes y devolver un listado de Remisiones de entrada.

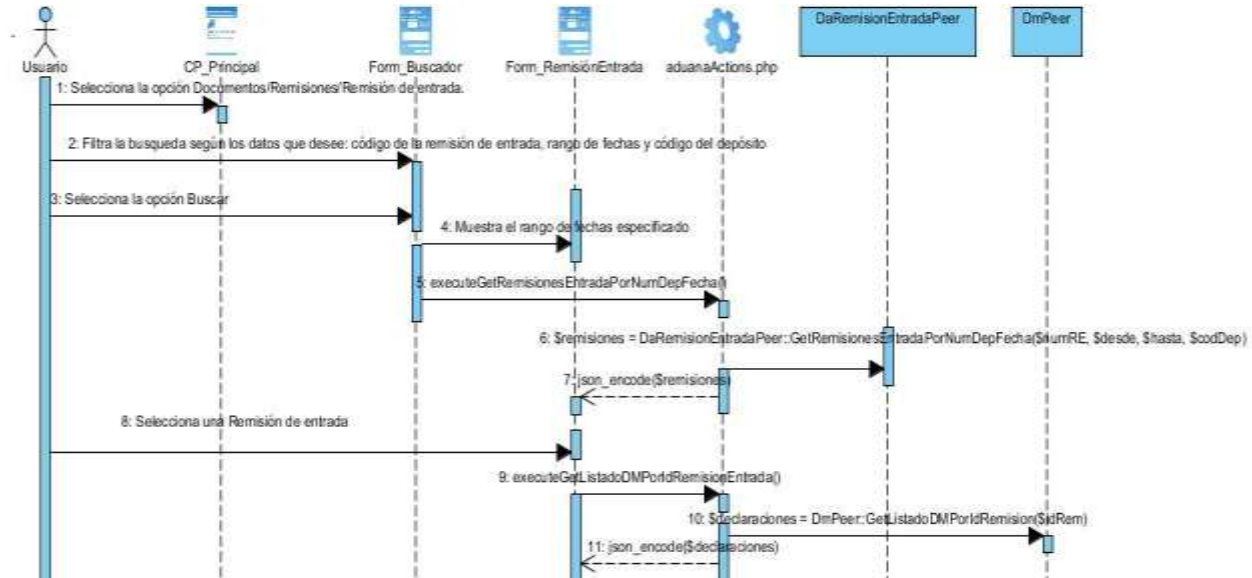


Figura 3 Fragmento del diagrama de secuencia del RF Mostrar datos de la remisión de entrada

2.5.3 Diagrama de clases persistentes

La figura 4 muestra el diagrama de diseño de las clases persistentes del sistema, sirve como una primera aproximación al diseño definitivo del modelo de datos.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

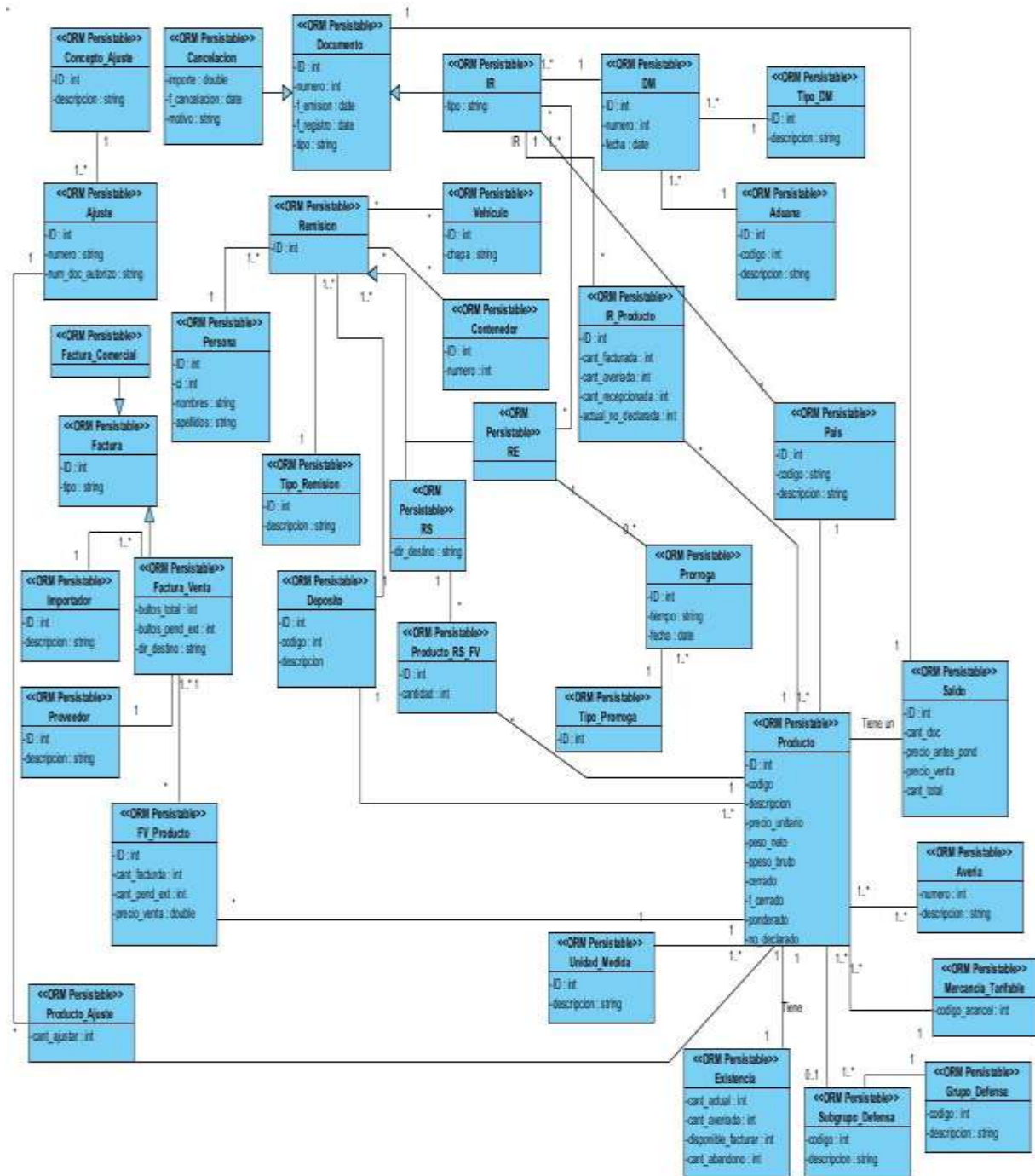


Figura 4 Diagrama de clases persistentes

2.5.4 Diagrama de paquetes

El lenguaje UML ofrece el mecanismo paquete que permite describir los grupos de elementos o subsistemas. Un paquete es un conjunto de cualquier tipo de elementos de un modelo: clases, casos de uso, diagramas de colaboración u otros paquetes (los anidados); es un mecanismo

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

utilizado para agrupar elementos de UML que permite organizar los elementos modelados con UML, facilitando de esta forma el manejo de los modelos de un sistema complejo. (31)

Los diagramas de paquetes se usan para reflejar la organización de los paquetes y sus elementos, y para proveer una visualización de sus correspondientes nombres de espacio. Entre sus características se tiene: (31)

- Permiten dividir un modelo para agrupar y encapsular sus elementos en unidades lógicas individuales.
- En general, pueden tener una interfaz (métodos de clases e interfaces exportadas) y una realización de éstas interfaces (clases internas que implementan dichas interfaces)
- Se pueden utilizar para plantear la arquitectura del sistema a nivel macro.

El diagrama de paquetes (Figura 5) de la presente solución cuenta con un paquete fundamental llamado Depósito constituido por 4 paquetes internos; entre los que se encuentran: el paquete Interfaces JS (hace referencia a las clases .js pertenecientes a las interfaces visuales de la aplicación), el paquete Controlador (contiene el controlador frontal, deposito_dev.php), el paquete Lib (contiene todas las clases del negocio de la solución) y el paquete Config (constituido por los ficheros propel.ini y database.yml). Este paquete está relacionado con 5 paquetes externos: el paquete TC (perteneciente al Subsistema Tablas de Control del Sistema GINA), el paquete Persona (perteneciente al Subsistema Persona del Sistema GINA), el paquete DC (representa al Subsistema Despacho Comercial del Sistema GINA), el paquete RC (perteneciente al subsistema Registro Central del Sistema GINA) y el paquete Symfony Lib (representa las librerías del Framework Symfony).

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

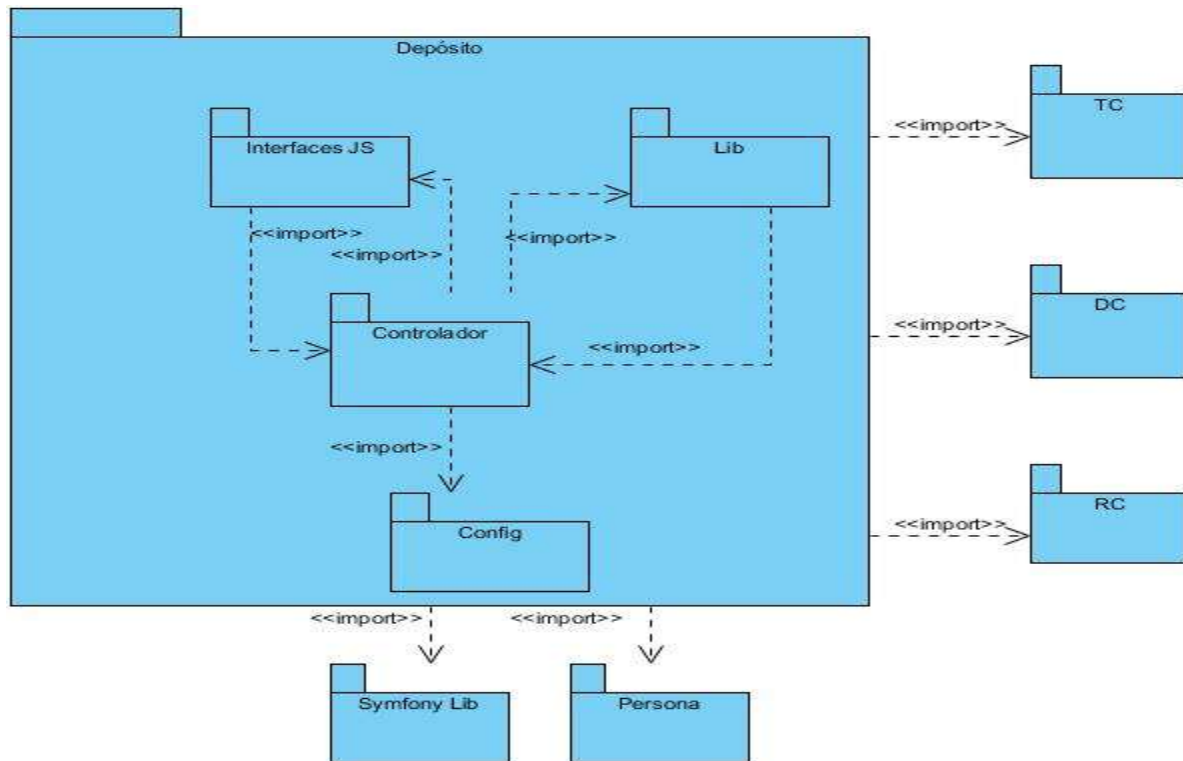


Figura 5 Diagrama de paquetes de la solución

2.5.5 Diseño de la base de datos

El modelo de datos (Ver Anexo III) está formado por un conjunto de conceptos que permiten describir la realidad que contiene las tablas y relaciones entre ellas que guardan la información referente a los depósitos.

En la Figura 6 se muestra un fragmento del modelo de datos desarrollado, el mismo está constituido por un total de 42 tablas, de ellas sólo 30 son propias del esquema del presente trabajo, representadas por el color gris. De las restantes tablas, 9 forman parte del Subsistema Tablas de Control, que es el encargado de llevar el control de los nomencladores del sistema GINA, mostradas con el color verde. También se cuenta con 3 tablas representadas con el color azul claro que pertenecen a otros subsistemas. La base de datos se encuentra en 3ra Forma Normal pues todas sus tablas contienen una llave primaria, los atributos son atómicos y no contiene dependencias funcionales transitivas ni parciales. De esta manera se evita la redundancia de los datos y los problemas que puedan ocurrir con las actualizaciones de estos en las tablas, protegiendo así la integridad de los mismos.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

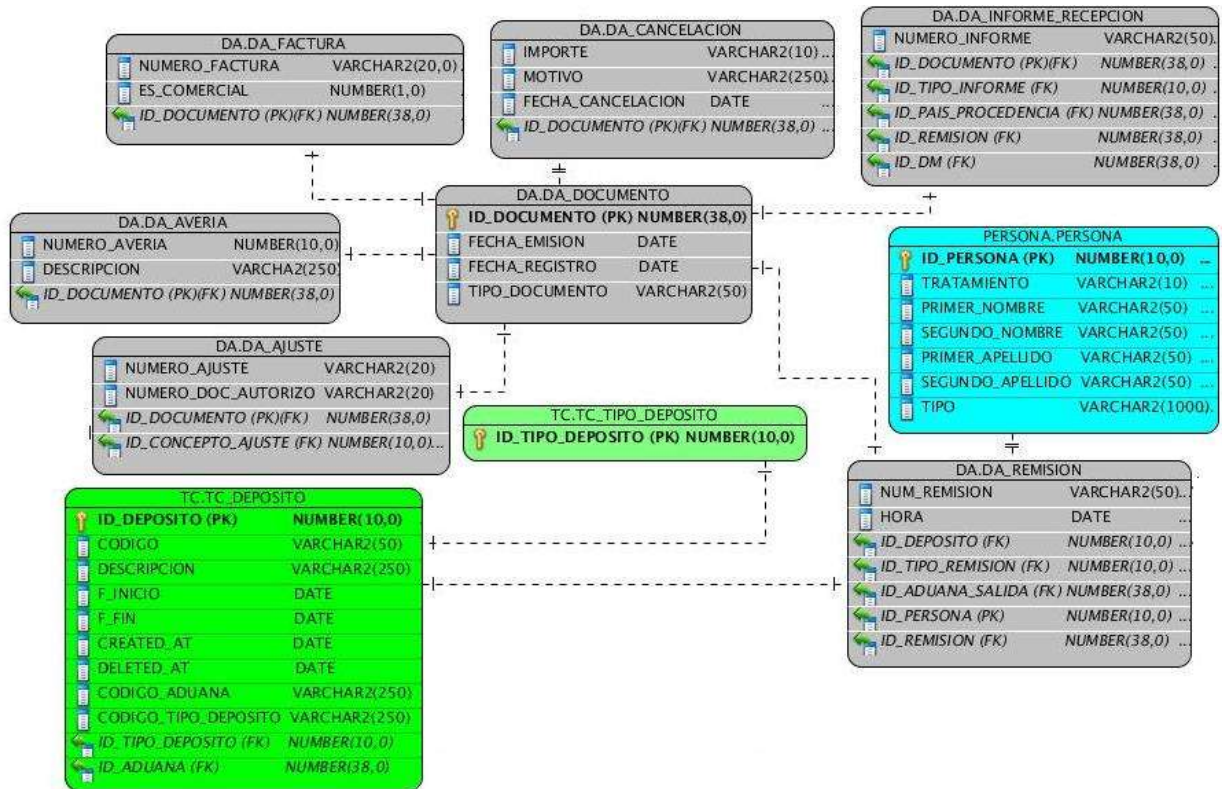


Figura 6 Fragmento del Modelo de Datos

2.6 Métricas para la evaluación del diseño

Aunque los términos medida, medición y métricas se utilizan a menudo indistintamente, existe gran confusión a la hora de referirse a ellos. Dentro del contexto de la ingeniería del software, una medida proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto. La medición es el proceso por el cual los números o símbolos son asignados a atributos o entidades en el mundo real tal como son descritos de acuerdo a reglas claramente definidas, también es definida como el acto de determinar una medida. (32)

2.6.1 Métricas propuestas por Lorenz y Kidd

Lorenz y Kidd⁶ 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, para luego

⁶ Lorenz y Kidd: especialistas autores del libro "Object-Oriented Software Metrics". Prentice Hall. 1994. Texto recomendado para introducirse en la medición de atributos de entidades desarrolladas con metodología orientada a objetos.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

promediar los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y asuntos relacionados con el código así como las métricas orientadas a valores externos examinan el acoplamiento y la reutilización. (33)

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

- Tamaño operacional de clase (TOC)
- Relación entre clases (RC)

2.6.2 Métrica de Tamaño Operacional de Clase (TOC)

Luego de analizar todas las métricas mencionadas anteriormente, se decidió utilizar para validar el diseño de la solución la de Tamaño Operacional de Clase (TOC). La misma se encarga de medir la calidad de acuerdo a los atributos Responsabilidad, Complejidad de Implementación y Reutilización de las clases.

La aplicación de la métrica TOC define los siguientes atributos de calidad:

- Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

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 del sistema y los mismos son reflejados en la siguiente tabla.

Tabla 2 Umbrales para los parámetros de calidad de la métrica TOC

Atributos de calidad	Categoría	Criterio
Responsabilidad	Baja	\leq Promedio
	Media	$>$ Promedio y $\leq 2 \times$ Promedio
	Alta	$> 2 \times$ Promedio

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Complejidad de implementación	Baja	\leq Promedio
	Media	$>$ Promedio y $\leq 2 \times$ Promedio
	Alta	$> 2 \times$ Promedio
Reutilización	Alta	\leq Promedio
	Media	$>$ Promedio y $\leq 2 \times$ Promedio
	Baja	$> 2 \times$ Promedio

En la figura 7 se muestra el comportamiento del nivel de procedimientos puesto en práctica en las clases del diseño realizado y a continuación en las figuras 8, 9 y 10 se representan los estados de los atributos de calidad: responsabilidad, complejidad de implementación y reutilización respectivamente.

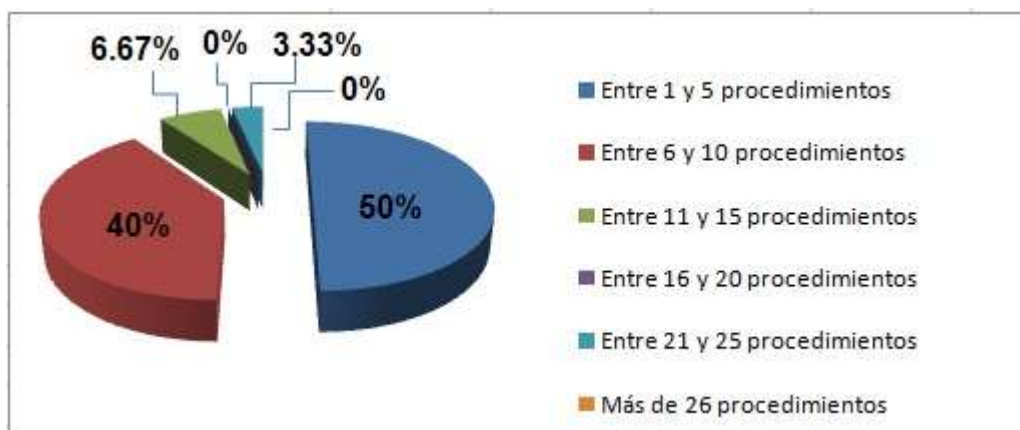


Figura 7 Representación del Nivel de Procedimientos de las Clases

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

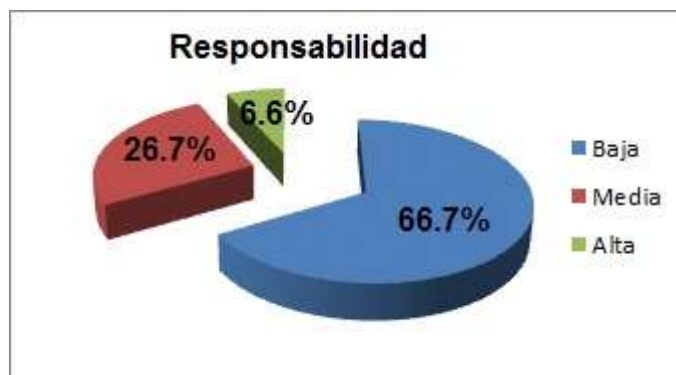


Figura 8 Representación de las clases según la responsabilidad.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA



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



Figura 10 Representación de las clases según la reutilización.

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 66.7% 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 66.7%, lo que hace que carezcan de mucha complejidad, por lo que se tornan mucho más reutilizables. Solamente un 6.6% 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.

2.6.3 Métrica Relación entre clases

La métrica relación entre clases consiste en el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

- Acoplamiento: Un aumento del RC implica un aumento del Acoplamiento de la clase.

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

- Complejidad de mantenimiento: Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- Reutilización: Un aumento del RC implica una disminución en el grado de reutilización de la clase.
- Cantidad de pruebas: Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Esta métrica fue aplicada a un total de 30 clases que pertenecen al diseño y se obtuvo tras aplicar la misma, valores promedio de 3,53 asociaciones de uso por clase.

La siguiente tabla muestra los umbrales utilizados para evaluar cada uno de los atributos de calidad que evalúa la métrica RC.

Tabla 3 Umbrales para los parámetros de calidad de la métrica RC

Atributos de calidad	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	\leq Promedio
	Media	$>$ Promedio y $\leq 2 \times$ Promedio
	Alta	$> 2 \times$ Promedio
Reutilización	Alta	\leq Promedio
	Media	$>$ Promedio y $\leq 2 \times$ Promedio
	Baja	$> 2 \times$ Promedio
Cantidad de pruebas	Baja	\leq Promedio
	Media	$>$ Promedio y $\leq 2 \times$ Promedio
	Alta	$> 2 \times$ Promedio

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

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA



Figura 11 Representación de la relación entre las clases según el acoplamiento

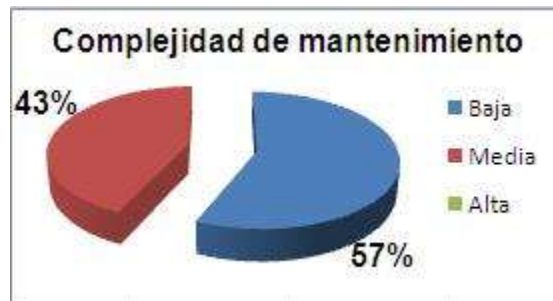


Figura 12 Representación de la relación entre las clases según la complejidad de mantenimiento



Figura 13 Representación de la relación entre las clases según la reutilización



Figura 14 Representación de la relación entre las clases según la cantidad de pruebas

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 57% de las clases tienen 3 o

CAPÍTULO 2 DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

menos dependencias unas de otras. Por su parte, el nivel de acoplamiento entre las clases se mantiene alto en un 53%, lo cual no 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 57% de forma positiva. Por otra parte se obtuvo un valor bajo de un 57% en el atributo cantidad de pruebas lo que implica una disminución de la cantidad de pruebas de unidad necesarias para probar una clase, mientras que la complejidad de mantenimiento alcanza un valor bajo en un 57% lo que implica que la complejidad de mantenimiento de la clase disminuya.

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.

2.7 Conclusiones Parciales

- La aplicación de las técnicas de captura de requerimientos permitió definir de manera clara los requisitos funcionales que debe cumplir el software para que el producto cumpla con las expectativas del cliente.
- El modelo conceptual permitió mostrar las entidades importantes que se encuentran en el ámbito del problema y sus relaciones.
- Con la elaboración de cada uno de los artefactos definidos se logró sentar las bases para la implementación del sistema.
- La aplicación de la métrica TOC permitió evaluar el diseño y determinar que el 66.7% 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 66.7%, lo que hace que carezcan de mucha complejidad, por lo que se tornan mucho más reutilizables.
- La aplicación de la métrica RC permitió determinar que el 57% de las clases tienen 3 o menos dependencias unas de otras, lo cual influye en que el acoplamiento entre las clases se mantenga alto en un 53%, la reutilización alcance niveles de 57% de forma positiva, la cantidad de pruebas se mantenga baja en un 57% y que la complejidad de mantenimiento alcance valores bajos en un 57%.

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS AL SUBSISTEMA

Capítulo 3: Implementación y pruebas al sistema

3.1 Introducción

En este capítulo se abordan los aspectos relacionados con la implementación del sistema y la validación del mismo, se realiza el diagrama de despliegue y de componentes y se enuncian los estándares de codificación tenidos en cuenta a la hora de la implementación. También se valida mediante pruebas, si el software realizado responde a las necesidades del cliente.

3.2 Descripción general de la disciplina Implementación

A partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares. Al reutilizar componentes de software ya implementados se lleva a cabo el desarrollo necesario para ajustar a los requisitos actuales y posteriormente realizar la integración de los componentes.

3.3 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes como ficheros de código fuente, ejecutables y similares. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o los lenguajes de programación utilizados y cómo dependen los componentes unos de otros. (34)

3.3.1 Estándar de codificación

El Departamento de Soluciones para la Aduana del Centro de Informatización y Gestión de Entidades elaboró un documento donde se definen los estándares de codificación y comprende todo el código generado bajo la tecnología y el lenguaje PHP y que utilicen la arquitectura regida por la utilización del Framework Arquitectónico Symfony.

En este documento se citan todos los aspectos relevantes a tener en cuenta a la hora de implementar una aplicación. A continuación se relacionan algunos de los aspectos relacionados con los componentes del framework.

En el caso de las Acciones Symfony trae su propia nomenclatura para las clases y sus funciones, pero no especifica claramente cuál es el nombre que se le debe poner a cada una de las funcionalidades del modelo que serán accedidas por el usuario [dígase acciones]. (35)

- Dentro de las especificaciones del Framework está que cada una de estas acciones debe comenzar con la palabra execute.

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS AL SUBSISTEMA

- Todos los nombres de acciones deben estar en la nomenclatura CamelCase⁷ comenzando por la palabra execute.
- En caso de ser acciones referentes a un módulo de CRUD de una tabla deben ser nombres específicos como executeNuevo, executeEditar, etc.
- Los nombres de las acciones deben especificar con la menor cantidad de palabras cual es el objetivo de la acción, de ser posible estar en infinitivo. Debe especificar bien claro cuál es la acción que se pretende ejecutar con la acción pero sin especificar los parámetros que recibe.

En el caso del nombre de las clases especifica que:

- Deben estar expresados en notación UpperCamelCase⁸.
- No se deben utilizar guiones bajos en su nombre “_”.
- Deben expresar con claridad cuál es el alcance y la responsabilidad de la clase.
- No deben estar atados a las clases de las que se deriva, cada clase debe tener un significado por ella misma, no en dependencia de la clase de la que deriva.
- En los nombres compuestos por más de tres palabras se debe revisar el diseño, no sea que se le estén dando a la clase más responsabilidades de las que realmente tiene.

En el caso del nombre de las funciones especifica que:

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

En el caso de las Variables especifica que:

- Los nombres de las variables deben expresar claramente el contenido de la misma.
- Pueden estar referidas en singular o plural.
- Se definen al principio de las estructuras donde son utilizadas.
- En caso de que no se le asigne un valor inicial se deben inicializar con un valor que indique el tipo de dato más general al que debe pertenecer.
 - a. Los tipos de datos cadena son definidos con comillas dobles (“”).

⁷ **CamelCase:** es un estilo de escritura que se aplica a frases o palabras compuestas

⁸ El *UpperCamelCase* es un estilo de escritura donde la primera letra de cada una de las palabras es mayúscula.

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS AL SUBSISTEMA

- b. Los tipos de datos de caracteres se definen con comillas simples (').
- c. En caso de que se espere almacenar tipos de datos diversos no se inicializa.
- De esta nomenclatura se exceptúan las variables generadas por el Framework.

En el caso de los Arreglos especifica que:

- Terminan con el sufijo “array” de la palabra arreglo en inglés.
- Los nombres de los arreglos deben estar en plural ya que son estructuras que contienen colecciones de datos.
- Cuando se itera sobre un arreglo la variable temporal que se utiliza para almacenar los elementos del arreglo debe tener un nombre en singular a fin con nombre del arreglo.

Para el caso de las llaves se colocará la primera en la misma línea del comienzo del bloque separada por un espacio y la llave de cierre del bloque en la última línea.

Para comentar una línea se utilizará el o los caracteres que dicta el lenguaje de programación utilizado, al igual que para los comentarios de bloque, es decir, que requiera más de un línea.

3.3.2 Tratamiento de Errores

Cuando se desarrolla un software es de vital importancia, para garantizar su correcto funcionamiento, identificar y controlar los posibles problemas que se pueden presentar a la hora de interactuar con el software.

Con el propósito de afrontar cualquier situación inesperada o excepcional que se presente mientras se ejecuta un programa, se utiliza como uno de los métodos para el control de excepciones las palabras claves try, catch (ver figura 15). Las acciones que pueden producir una excepción se ejecutan dentro del bloque try y al producirse una anomalía, el flujo de control salta inmediatamente al controlador de excepciones, utilizándose la palabra catch para definir dicho controlador. Con el objeto de excepción throw excepciones, conteniendo información detallada sobre el error.

```
try {
    $depoperteneciaaduana = SisTcTablaPeer::ejecutarServicio('tc',
        'tc.depositoPerteneceAduana', array(
            'param' => array('codAduana' => $codigoaduana,
                'codDeposito' => $codigodeposito,
                'codTipo' => $codigo_deposito_aduana,
                'fecha' => null)));
} catch (Exception $exc) {
    $errores[] = $exc->getMessage();
}
```

Figura 15 Segmento de código donde se manifiesta el tratamiento de errores mediante try, catch

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS AL SUBSISTEMA

Otro de los aspectos utilizado en el tratamiento de errores del sistema desarrollado es la utilización de los formularios generados por Symfony para insertar, eliminar o modificar valores de la base de datos (ver figura 16), utilizando cada uno de los validadores implementados de manera que se garantiza una mayor coherencia de los mismos. Se le asigna a cada elemento del formulario un nuevo valor que posteriormente mediante la utilización de la funcionalidad `bind(array('fechaEmision' => $fechaemision, 'fechaRegistro' => $fechahora, 'idDeposito' => $deposito[0]->getIdDeposito(), 'tipoDocumento' = "FACTURA VENTA"))` se validarán si los datos son correctos. En caso que lo sea se actualizan los valores utilizando `updateObject(array('fechaEmision' => $fechaemision, 'fechaRegistro' => $fechahora, 'idDeposito' => $deposito[0]->getIdDeposito(), 'tipoDocumento' = "FACTURA VENTA"))` y por último se realiza la salva utilizando `save()`. Si los datos no son válidos se muestra un mensaje de error y no permite la inserción o modificación de estos.

```
$idfact = $fac->getObject()->getIdFactura();
$docum = new DaDocumentoForm();

$docum->bind(array(
    'fechaEmision' => $fechaemision,
    'fechaRegistro' => $fechahora,
    'idDeposito' => $deposito[0]->getIdDeposito(),
    'tipoDocumento' => "FACTURA VENTA"
));

if (!$docum->isValid()) {
    $errores[] = $docum->getErrorSchema();
} else {
    $docum->updateObject(array(
        'fechaEmision' => $fechaemision,
        'fechaRegistro' => $fechahora,
        'idDeposito' => $deposito[0]->getIdDeposito(),
        'tipoDocumento' => "FACTURA VENTA"
    ));

    $docum->save();
}
```

Figura 16 Segmento de código donde se manifiesta el tratamiento de errores mediante los formularios de Symfony

Las validaciones del negocio se encargan de controlar el flujo de los datos recibidos en el controlador para evitar consecuencias alarmantes. Se llevan a cabo en las diferentes clases

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS AL SUBSISTEMA

con el uso de funciones propias del lenguaje. En caso de que exista algún error, es notificado al usuario para que pueda corregirlos.

3.3.3 Diagrama de Despliegue

Los diagramas de despliegue modelan la arquitectura en tiempo de ejecución de un sistema y muestran las relaciones físicas entre los componentes hardware y software en el sistema final. Estos describen la arquitectura física del sistema durante la ejecución en términos de: procesadores, dispositivos y componentes de software. (36)

En la figura 17 se muestra el diagrama de despliegue del presente trabajo. El mismo está compuesto por una PC Cliente la cual se encuentra conectada a una impresora mediante conexión USB y al Servidor Apache mediante el protocolo HTTPS. También cuenta con un Servidor Oracle, el cual está conectado con el Servidor Apache mediante el protocolo TCP/IP.



Figura 17 Diagrama de despliegue

3.3.4 Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones, y muestran las opciones de realización incluyendo código fuente, binario y ejecutable. (37)

En la figura 18 se muestra el diagrama de componentes de la solución. El mismo está constituido por el componente que representa el controlador frontal de la aplicación, el cual se relaciona con el *actions.class.php* teniendo entre sus funciones principales la comunicación entre las vistas y las clases del negocio. El *actions* como componente se relaciona con tres paquetes, uno de ellos es el de las interfaces .js constituido por todas las clases encargadas de recibir y mostrar los datos al usuario, otro paquete es la clase *sfAuxiliarTC* que por medio de eventos es capaz de obtener información de los diferentes plugins y subsistemas y el último paquete es el de la abstracción de datos de Symfony, constituido por los componentes *Objeto.php* y *ObjetoPeer.php* que a su vez utilizan Propel como ORM. El último paquete de

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS AL SUBSISTEMA

componentes mencionado se relaciona con el de configuración, constituido el mismo por el *databases.yml* que es el que contiene la información necesaria para la conexión a la base de datos y el *view.yml* encargado de definir las opciones entre las vistas. Además de relacionarse con el de configuración, también lo hace con cuatro componentes pertenecientes al Sistema GINA: RC (subsistema Registro Central), TC (subsistema Tablas de Control), Persona (subsistema Persona) y DC (subsistema Despacho Comercial). Además cuenta con un paquete común entre el de configuración y el de las interfaces.js integrado por el componente *ext-all.js* (contiene todas las clases del ExtJS), el *ext-base.js* (encargado de configurar el acceso a las librerías del ExtJS) y el *ext-all.css* (tiene como función principal el control de la apariencia).

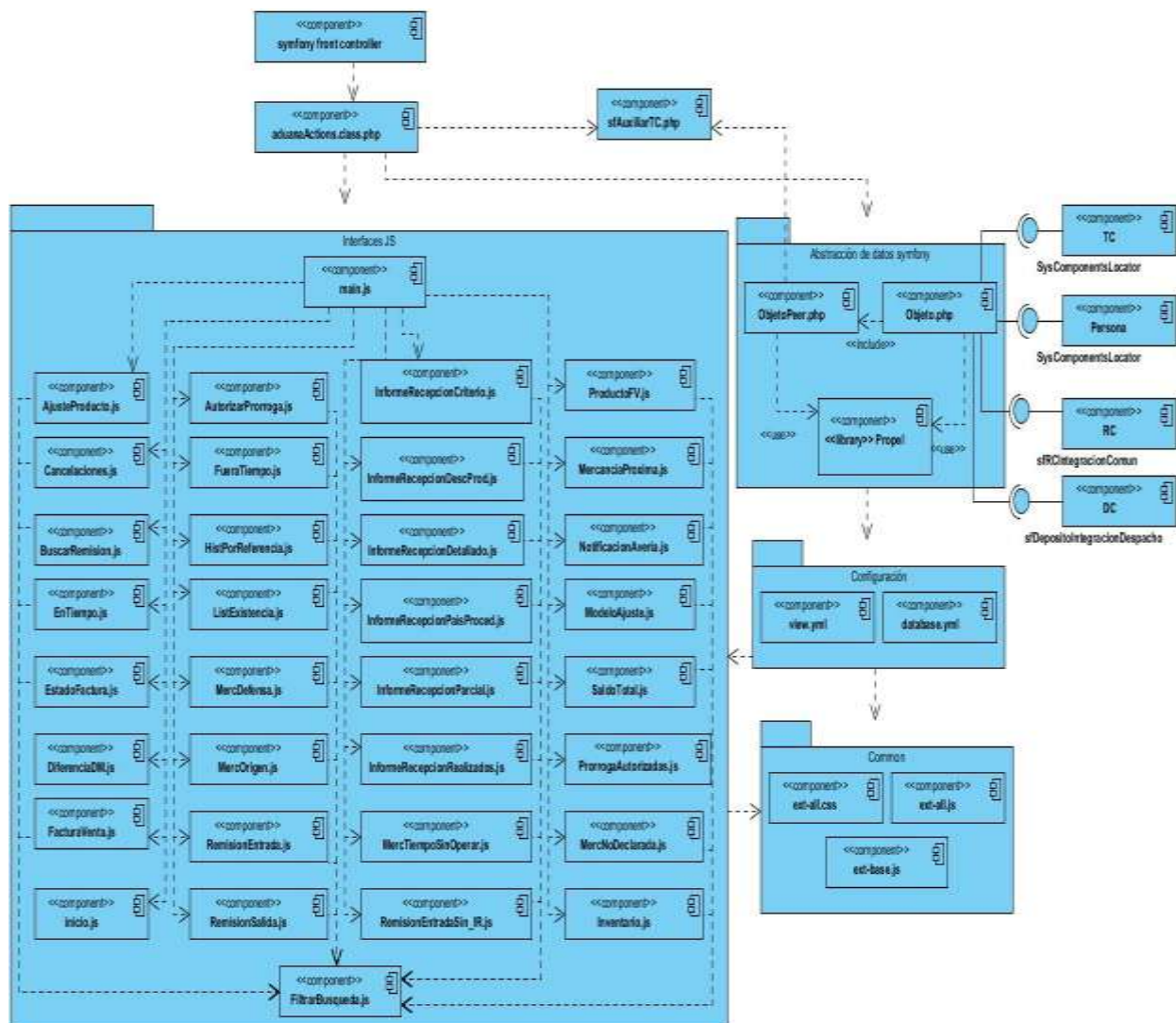


Figura 18 Diagrama de componentes de la solución

3.4 Validación de la solución

3.4.1 Pruebas de Software

En el proceso de pruebas de software se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. (38)

Las pruebas del software, conocidas también como técnicas de evaluación dinámica son un elemento crítico para la garantía de la calidad del sistema, representan una revisión final de las especificaciones del diseño y de la implementación. Su principal objetivo es diseñar pruebas que, sistemáticamente, saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y de esfuerzo. (17)

Las pruebas no pueden asegurar la ausencia de errores; sólo puede demostrar que existen defectos en el software. (17)

3.4.2 Aplicación de Pruebas de Caja Negra

Las Pruebas de Caja Negra son aquellas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. La misma se centra principalmente en los requisitos funcionales del software y permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todas las funcionalidades de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos del sistema y ejercitándolos. (17)

Los casos de prueba de la 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 estos tipos de errores:

- 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

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS AL SUBSISTEMA

Dentro de las pruebas de Caja Negra se incluyen las Pruebas de Funcionalidad que son las que se le aplicaron a la solución obtenida. Las Pruebas de Funcionalidad examinan si el sistema cubre sus necesidades de funcionamiento, acorde a las especificaciones del diseño. En ellas se debe verificar si el sistema lleva a cabo correctamente todas las funcionalidades requeridas y la validación de los datos, además se deben realizar pruebas de comportamiento ante distintos escenarios. Estas pruebas deben estar enfocadas a tareas, a límites del sistema, a condiciones planeadas de error y de exploración. (39)

Para la puesta en marcha de este tipo de pruebas se hace necesario la presencia de un buen diseño de casos de prueba, el cual es una técnica de particiones equivalentes que ha sido modificado y adaptado a los proyectos de la UCI. (40)

Los diseños de casos de pruebas recogen todas las combinaciones de escenarios posibles a ejecutar, contiene la descripción de todas las variables o datos de entrada al sistema. (40)

Con el propósito de comprobar que todos los requisitos de una aplicación son revisados, debe haber al menos un caso de prueba para cada requisito a menos que un requisito contenga requisitos secundarios.

A continuación se muestra como ejemplo el diseño de caso de pruebas para el RF Mostrar datos de la remisión de entrada.

Tabla 4 Descripción del requisito funcional a probar

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS AL SUBSISTEMA

Nombre del requisito	Descripción general	Escenarios de prueba	Descripción del escenario	Flujo central
Mostrar datos de la remisión de entrada.	El sistema debe mostrar las remisiones de entrada que cumplan con el criterio de búsqueda especificado.	EC 1.1 Mostrar los datos de las remisiones de entrada de Entrada.	Se muestran los datos de las remisiones de entrada que estén asociadas a los parámetros de búsqueda.	1- Seleccionar del menú la opción Documentos - Remisiones - Remisión de entrada. 2- Introducir el código de la remisión. 3- Especificar el rango de fecha. 4- Introducir el código del depósito. 5- El sistema busca y muestra el depósito especificado o un listado con los depósitos cuyo código sea similar al entrado como parámetro. 6- Seleccionar el depósito. 7- Se presiona el botón Buscar. 8- El sistema muestra un listado de las remisiones encontradas agrupadas por depósito.

Tabla 5 Caso de prueba del requisito funcional Mostrar datos de la remisión de entrada

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS AL SUBSISTEMA

Escenario	Descripción	Código	Desde	Hasta	Depósito	Respuesta del sistema	Flujo central
EC 1.1 Buscar las remisiones de entrada	Se muestran los datos de las remisiones de entrada registradas asociadas a los parámetros de búsqueda.	N/A	N/A	N/A	N/A	El sistema debe mostrar un listado con los datos de las remisiones agrupadas por depósito. En caso de no existir ninguna remisión que cumpla con los criterios especificados debe mostrar un mensaje.	1- Seleccionar del menú la opción Documentos - Remisiones - Remisión de entrada. 2- Introducir el código de la remisión. 3- Especificar el rango de fecha. 4- Introducir el código del depósito. 5- El sistema busca y muestra el depósito especificado o un listado con los depósitos cuyo código sea similar al entrado como parámetro. 6- Seleccionar el depósito. 7- Se presiona el botón Buscar. 8- El sistema muestra un listado de las remisiones encontradas agrupadas por depósito.
		vacío	vacío	vacío	vacío		
		V	V	V	V		
		00002	01/01/2012	18/05/2013	1516		
		I	N/A	N/A	V	El sistema debe mostrar un mensaje donde indique que no existen remisiones de entrada con ese código.	
		nuevo	vacío	vacío	1516		

Resultado de las pruebas

De esta forma se le realizaron las pruebas de funcionalidad a los requisitos del sistema en dos iteraciones. En la primera iteración se probaron los 39 requisitos funcionales, aplicándole un caso de prueba a cada uno, y cada caso de prueba con uno o más escenarios de pruebas. Como resultado se obtuvieron 10 no conformidades (Tabla 3.3), las mismas fueron sobre validaciones de campos en las interfaces, validaciones en la lógica de negocio y errores ortográficos. En la segunda iteración se verificó la resolución de las no conformidades detectadas en la iteración anterior, estas fueron resueltas en un cien por ciento, contribuyendo a la mejora del funcionamiento del sistema, con lo cual se concluyó la aplicación de estas pruebas.

Tabla 6 Resultado de los casos de prueba

No	Caso de prueba	Cantidad de escenarios	No conformidades
1	Procesar información de la Remisión de Entrada.	1	1
2	Procesar información del Informe de Recepción	1	1
3	Procesar información de la Remisión de Salida	1	1
4	Procesar información de la Factura de Venta	1	1

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS AL SUBSISTEMA

5	Procesar información de Ajuste	1	0
6	Procesar información de Cancelación	1	0
7	Mostrar datos de la Remisión de Entrada	1	1
8	Mostrar datos de la Remisión de Salida	1	1
9	Buscar remisiones	4	0
10	Mostrar datos del Informe de Recepción Detallado	1	1
11	Mostrar datos del Informe de Recepción Realizado	1	0
12	Mostrar Informes de Recepción parciales	1	0
13	Mostrar Informes de Recepción por descripción de producto	1	0
14	Mostrar Informes de Recepción por criterio (Avería, Faltante, Sobrante o No Declarada)	1	0
15	Mostrar Informes de Recepción por país de procedencia	1	0
16	Mostrar datos de la Factura de Venta	1	0
17	Buscar productos de la Factura de Venta	1	0
18	Mostrar estado de la Factura de Venta	1	0
19	Mostrar datos del Ajuste	1	0
20	Mostrar Ajustes por descripción de producto	1	0
21	Mostrar datos de la Notificación de Avería	1	0
22	Mostrar Documentos Cancelados	1	0
23	Identificar infracciones	1	0
24	Mostrar alertas de infracciones	1	0
25	Mostrar reporte de inventario	1	0
26	Mostrar histórico por referencia de producto	1	0
27	Mostrar saldo total por depósito	1	0
28	Registrar prórroga	1	1

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS AL SUBSISTEMA

29	Buscar prórrogas autorizadas	1	0
30	Mostrar RE sin IR	1	0
31	Mostrar IR sin DM	1	0
32	Mostrar IR fuera de tiempo	1	1
33	Mostrar depósito con tiempo sin operar	1	0
34	Mostrar mercancías con tiempo sin operar	1	1
35	Mostrar mercancías en abandono y próximas al abandono	1	0
36	Mostrar mercancías en existencia	1	0
37	Mostrar mercancías por país de origen	1	0
38	Mostrar mercancías para la defensa	1	0
39	Mostrar mercancías no declaradas	1	0
Total	39	42	10

3.5 Conclusiones Parciales

- El diseño y la implementación de las interfaces y clases del negocio permitieron obtener una aplicación que fuera capaz de gestionar los procesos asociados al control de las mercancías en los depósitos de aduana.
- Durante la fase de implementación se aprovechó al máximo las posibilidades brindadas por los marcos de trabajo utilizados, logrando un correcto tratamiento de errores.
- La aplicación de pruebas funcionales a la solución desarrollada permitió encontrar errores que afectaban el funcionamiento del sistema, dando la posibilidad de que pudieran ser corregidos a tiempo y que se obtuviera una aplicación que responde completamente a los requisitos funcionales definidos durante la etapa del análisis.
- En el capítulo se pudo demostrar la correcta realización de un software con las características definidas en un inicio, potente, flexible, seguro y eficaz, permitiendo demostrar que el objetivo propuesto ha sido cumplido.

Conclusiones

Luego de realizado el presente trabajo se llegó a las siguientes conclusiones.

- En la presente investigación se abordaron las problemáticas que dieron origen al objetivo del mismo, argumentadas por las dificultades presentadas en el proceso de control de las mercancías en los depósitos de aduana; lo cual permitió realizar un estudio de varios sistemas utilizados para este fin y llegar a la conclusión de que no son factibles para utilizarlos en la Aduana General de la República de Cuba.
- La identificación de un conjunto de sistemas informáticos que se encargan de gestionar solicitudes permitió analizar las características y el comportamiento de cada uno, determinándose que ninguno era factible para utilizarlo en la Aduana General de la República de Cuba por lo que no podían formar parte de la solución.
- La utilización de patrones de diseño unido a la elaboración de cada uno de los artefactos definidos por el Modelo de desarrollo del CEIGE, permitió que se sentaran las bases para la implementación del sistema deseado.
- El diseño y la implementación de las interfaces y las clases del negocio de la presente solución, permitieron que se obtuviera una aplicación que respondiera a los requisitos funcionales definidos durante la etapa de análisis
- La aplicación de pruebas de funcionalidad posibilitó la detección de errores en el funcionamiento de la solución obtenida permitiendo que los mismos pudieran ser resueltos.
- Finalmente el diseño y la implementación realizada en el presente trabajo permitieron que se obtuviera una aplicación que fuera capaz de satisfacer las necesidades de los clientes, dándole así cumplimiento al objetivo general propuesto en la presente investigación.

Recomendaciones

Los objetivos de este trabajo fueron alcanzados satisfactoriamente pero con el fin de contribuir a un mayor desarrollo de la presente investigación se realizan las siguientes recomendaciones:

- Realizar la liberación de la aplicación por el equipo de calidad de software de la universidad.
- Garantizar la exportación a PDF (formato de documento portátil o Portable Document Format) de cada uno de los reportes generados por el cliente.
- Extender el uso del sistema a todas las aduanas del país.

Referencias Bibliográficas

1. Aduana General de la República de Cuba. Sitio oficial de la Aduana cubana. [En línea] 1996. [Citado el: 02 de Noviembre de 2012.] <http://www.aduana.co.cu>.
2. Aduana General de la República de Cuba. Resolución No. 33/96 Glosario de Términos. Sitio Web Sistema de Órganos Aduaneros Aduana General de la República de Cuba. [En línea] 1996. [Citado el: 02 de Noviembre de 2012] <http://www.aduana.co.cu>
3. Asociación Latinoamericana de Integración - ALADI. Convenio internacional para la simplificación y armonización de los regímenes aduaneros. [En línea] 13 de Marzo de 2003 Comunidad Andina. [Citado el: 10 de Diciembre de 2012.] <http://intranet.comunidadandina.org/Documentos/DInformativos/SGdi498.doc>.
4. Aduana General de la República. Gaceta Oficial de la República de Cuba. Gaceta Oficial No. 37 Ordinaria de 12 de julio de 2002. Resolución No. 12/2002. ISSN 1682-7511. [En línea] 2008. [Citado el: 10 de Diciembre de 2010.] <http://www.gacetaoficial.cu>
5. Depósitos aduaneros. [Citado el: 10 de Diciembre de 2012] http://www.s-es.ms_Depositos.aspx
6. Ana Alejandra Febres Arellano. Conociendo al SIDUNEA - Sistema Aduanero Automatizado. [En línea] Marzo de 2004 [Citado el: 10 de Diciembre de 2012]. <http://www.gestiopolis.com/canales2/economia/sidunea.htm>
7. GENERALIDADES SOBRE EL TICA. [Citado el: 10 de Diciembre de 2012]. http://www.inteliagencia.net/infotica/TK_Generalidades.htm
8. Centro de Informatización de la Gestión de Entidades. Modelo de Desarrollo de Software v1.1. [En línea] 10 de Diciembre de 2012. [Citado el: 7 de Enero de 2013.] http://10.52.17.252:5700/Repo/AGR/LineasBases/LB_Gina/1-GIna_EstudioPreliminar/Documentos%20Generales/
9. Modelado de procesos. [En línea] 06 de Junio de 2010. [Citado el 7 de enero de 2013.] <http://otroblogmas.fullblog.com.ar/modelado-de-procesos.html>
10. Modelado de procesos con UML. [Citado el 7 de enero de 2013.] <http://modelado10.wordpress.com/>
11. Flores, Antonio. Introducción al Lenguaje de Programación PHP. Madrid: s.n., 2008. [Citado el: 10 de Enero de 2013.]
12. Potencier, Fabien and Zaninotto, François. Symfony 1.2, la guía definitiva. [PDF] [En línea] 30 de Diciembre de 2008. [Citado el: 10 de Enero de 2013.] http://www.dtic.co.cu/FTP/libros/symfony_1_2_guia_definitiva.pdf

REFERENCIAS BIBLIOGRÁFICAS

13. Bullock, Christian. 2002. Ext JS. Comunidad en Español. [En línea] 2002. [Citado el: 10 de Enero de 2013.] <http://extjises.com/>.
14. Sommerville, Ian. Ingeniería del Software (séptima edición). Madrid (España): Pearson Educación, SA, 2005. ISSN 84-7829-074-5. [Citado el: 10 de Enero de 2013.]
<http://es.scribd.com/doc/66701088/Ian-Sommerville-Ingenieria-de-Software-Septima-Edicion>
15. IEEE. 1990. 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology. [En línea] 1990. [Citado el: 10 de Enero de 2013.]
<http://standards.ieee.org/findstds/standard/610.12-1990.html>.
16. Arias Chaves, Michael. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. Universidad de Costa Rica: Revista InterSedes, 2005, Vol. VI. ISSN 1409-4746. [Citado el: 15 de marzo de 2013.]
<http://revistas.ucr.ac.cr/index.php/intersedes/article/download/790/851>
17. Pressman, Roger S. Ingeniería de Software: Un enfoque práctico. México DF: MacGraw Hill, 2006. Sexta Edición. [Citado el: 15 de marzo de 2013.]
<http://www.slideshare.net/jdbq16/ingenieria-de-software-un-enfoque-prctico-pressman-5th-ed>
18. José Escalona, María and Koch, Nora. Ingeniería de Requisitos en Aplicaciones para la Web - Un estudio comparativo. [PDF] Universidad de Sevilla: Departamento de Lenguajes y Sistemas Informáticos, 2002. [Citado el: 15 de marzo de 2013.]
<https://www.lsi.us.es/docs/informes/LSI-2002-4.pdf>
19. Olivares Rojas, Juan Carlos. Patrones de Diseño. [PDF] [Citado el 17 de marzo de 2013.]
<http://antares.itmorelia.edu.mx/~jcolivares/courses/dp07a/patrones.pdf>
20. Patrones de Alexander a la Tecnología de objetos. [PDF] [Citado el 17 de marzo de 2013.]
<http://zarza.usal.es/~fgarcia/doc/patrones1.pdf>
21. Cortés, Gloria and Casallas, Rubby. Introducción a los Patrones de Diseño. [PDF] Dpto. de Ingeniería de Sistemas y Computación. Universidad de los Andes. Material de base, 2008. [Citado el: 17 de marzo de 2013.]
<http://sistemas.uniandes.edu.co/~isis2603/dokuwiki/lib/exe/fetch.php?media=principal:isis2603-introduccionpatronesdiseño.pdf>
22. Capítulo 2: Marco Teórico. [PDF] [Citado el 17 de marzo de 2013.]
http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/sanchez_r_ma/capitulo2.pdf
23. Reynoso Billy, Carlos. Introducción a la Arquitectura de Software. [PDF] Buenos Aires: Universidad de Buenos Aires, 2004, Vol. I. [Citado el: 18 de marzo de 2013.]

REFERENCIAS BIBLIOGRÁFICAS

- <http://carlosreynoso.com.ar/archivos/arquitectura/Introduccion.PDF>
24. Clements, Paul C. Software Architecture. [PDF] Estados Unidos: Software Engineering Institute, 2002. ISBN: 15213-3890. [Citado el: 18 de marzo de 2013.]
<http://www.lucas.lth.se/events/2002/clements020306.pdf>
25. Serrano, Juan Manuel. Arquitectura del software. [PDF] [Citado el 18 de marzo de 2013.]
<http://zenon.etsii.urjc.es/grupo/docencia/as/material/tema1.pdf>
26. Veloso Hernández, Pedro. Uso de Patrones de Arquitectura. [PPT] s.l: UMP, 2009. [Citado el: 18 de marzo de 2013.] <http://ijegonzalezf.files.wordpress.com/2009/07/uso-de-patrones-de-arquitectura-capitulo-4.ppt>.
27. Bascón Pantoja, Ernesto. El patrón de diseño modelo -vista - controlador (MVC) y su implementación el Java Swing. [PDF] 4, 2004, Vol. II. [Citado el: 18 de marzo de 2013.]
http://tesis.pucp.edu.pe/repositorio/bitstream/handle/123456789/944/SANCHEZ_MERCADO_ALVARO_SISTEMA_INFORMATIZADO_HOSPITAL.pdf
28. Modelos Conceptuales. Abril, 2011. [Citado el 18 de marzo de 2013.]
<http://www.buenastareas.com/ensayos/Modelos-Conceptuales/1979709.html>
29. El Modelo Conceptual. [PDF] [Citado el 18 de marzo de 2013.]
http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/MCONCEP_A4.pdf
30. Zapata, Carlos Mario and Garcés, Gilma Liliana. Generación del diagrama de secuencias de UML desde esquemas preconceptuales. [PDF] [Citado el: 18 de marzo de 2013.]
<http://revista.eia.edu.co/articulos10/art7.pdf>.
31. Gutierrez, Demián. UML Diagramas de Paquetes (UML ilustrado). [PDF] Universidad de los Andes, Septiembre 2009. [Citado el: 21 de marzo de 2013.]
http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf.
32. Peña Lemus, Yudisleidys and Hernández Díaz, Yuniersy. SIMETSE–Sistema de METricas para evaluar el diseño. Ciudad de la Habana: Universidad de la Ciencias Informáticas, 2007. [Citado el: 21 de marzo de 2013.]
33. Lorenz, M. y Kidd, J. Object-Oriented Software Metrics. [PDF] [En línea] 1994. [Citado el: 21 de marzo de 2013.] <http://www.cc.uah.es/drg/b/RodHarRama00.pdf>
34. Jacobson Ivar, Booch Grady y Rumbaugh James. El Proceso Unificado de Desarrollo de Software. [Citado el: 22 de marzo de 2013.]
<http://es.scribd.com/doc/30251931/El-Proceso-Unificado-de-Desarrollo-de-Soft-Jacobson>
35. Departamento de Soluciones para la Aduana, CEIGE. Propuesta de un Estándar de Codificación. [Citado el: 22 de marzo de 2013.]

REFERENCIAS BIBLIOGRÁFICAS

36. Martínez Zurita, Alberto. Diagrama de despliegue. [Citado el: 2 de mayo de 2013.]
<http://www.slideshare.net/albertomartinezzurita/diagrama-de-despliegue-17693724>.
37. MODULO 2. Tema 12: Modelo de implementación: Diagramas de Componentes y Despliegue. [PDF] [Citado el: 2 de mayo de 2013.]
<http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r90203.PDF>.
38. PRUEBASDESOFTWARE Inicio: Cursos y servicios de gestión de calidad de software. Las pruebas de software. 2005. [Citado el: 5 de mayo de 2013.]
<http://pruebasdesoftware.com/laspruebasdesoftware.htm>.
39. Álvarez, José Luis Moreno. UDLAP. Aplicación de un Sistema Experto para el desarrollo de Sistema Evaluador del modelo Capability Maturity Model (CMM) niveles dos y tres. . [En línea] 17 de mayo de 2004. [Citado el: 18 de mayo de 2011.]
http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/moreno_a_jl/.
40. Peña, Ing. Yadira Machado. Planificación, Diseño y Ejecución de Pruebas Funcionales. Habana: Calisoft, marzo-2011. [Citado el: 18 de mayo de 2011.]

Anexos

Anexo I: Prototipo de interfaz de usuario del requisito funcional Mostrar datos de la remisión de entrada.

☒ No especificar código
☐ Especificar código

Desde:
Hasta:

☒ No especificar depósito
☐ Especificar depósito

Depósito	Descripción

Página
de 1

Buscar

Remisiones de entradas

Fecha

Desde:

Hasta:

Remisiones de entrada

Número	Tipo	Fecha	Aduana	Cancelación

Declaraciones de mercancías

Número	BL/A	Tipo	Fecha

Facturas comerciales

Número

Informes de recepción

Número	Tipo	Fecha

Chofer

Nombre y Apellidos	CI	Licencia	Hora

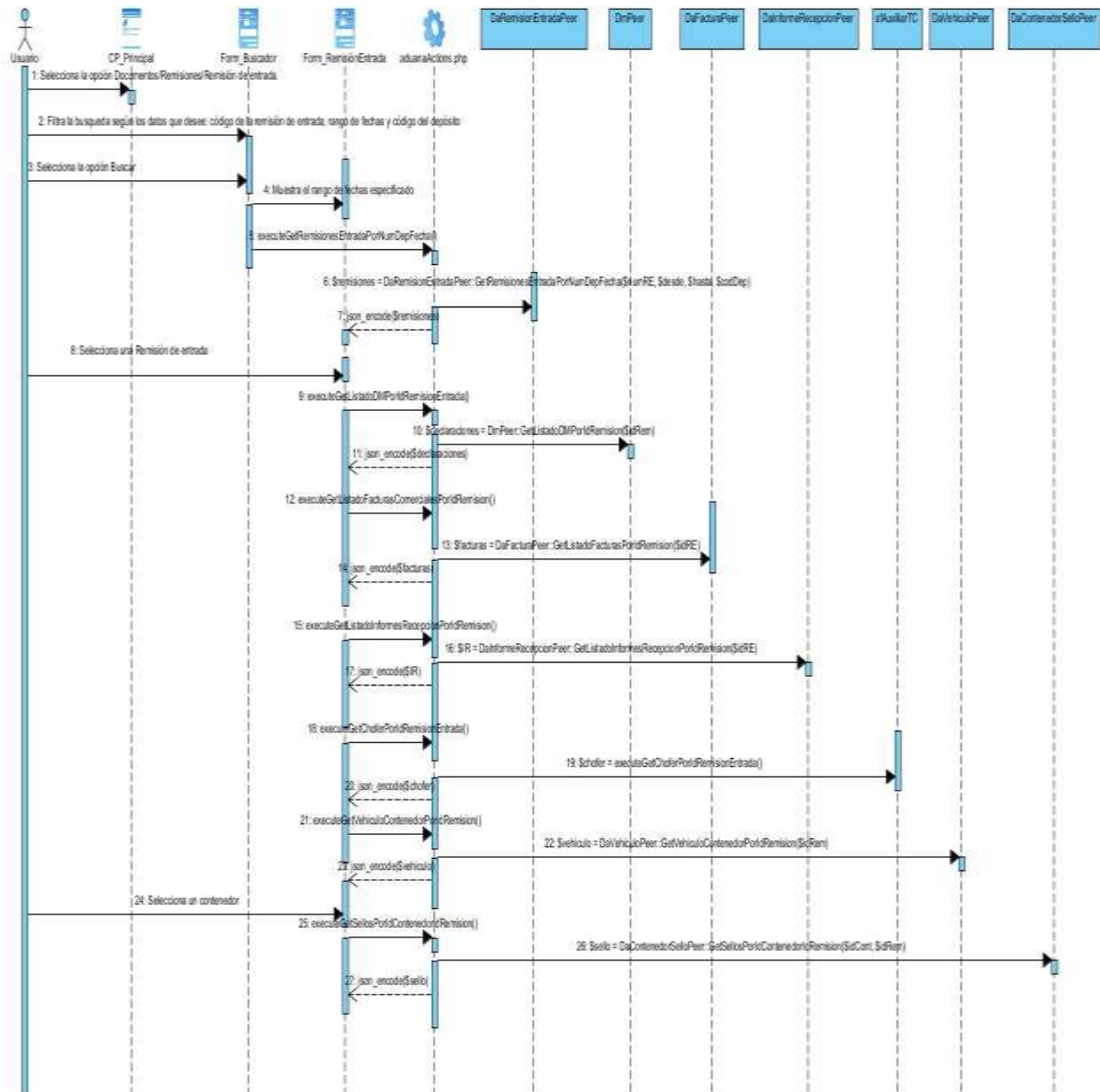
Vehículos

Chapa	Contenedor

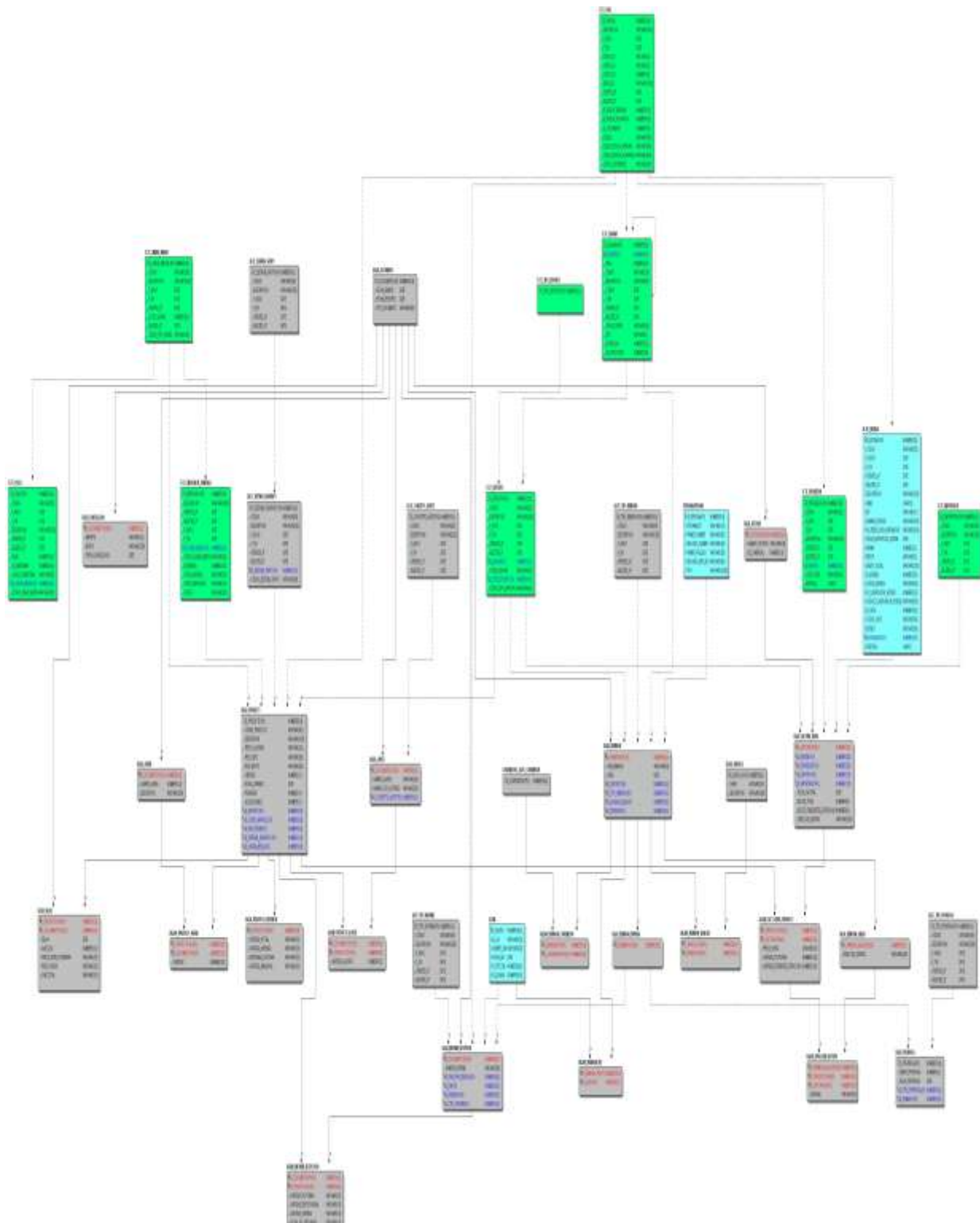
Sellos

Número

Anexo II: Diagrama de secuencia del RF Mostrar datos de la remisión de entrada.



Anexo III: Modelo de datos de la solución.



ANEXOS

Anexo IV Premio obtenido en la presentación del presente trabajo en la VIII Peña Tecnológica del MININT.



CLOSARIO DE TÉRMINOS

Glosario de términos

Aduana: La aduana es la oficina pública y/o fiscal que, a menudo bajo las órdenes de un estado o gobierno político, se establece en costas y fronteras con el propósito de registrar, administrar y regular el tráfico internacional de mercancías y productos que ingresan y egresan de un país.

Aduana General de la República: Es la encargada de regular el control aduanero aplicable a la entrada, el tránsito, el cabotaje, el trasbordo, el depósito y la salida del territorio nacional de mercancías, viajeros y sus equipajes, bienes y valores sujetos a regulaciones especiales, incluidas la flora y la fauna protegidas, así como los medios en que se transporten, además forma parte de la Administración del Estado y se subordina al Consejo de Ministros.

Artefacto: pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar actividades.

CASE: Computer Aided Software Engineering, en su traducción al español significa Ingeniería de Software Asistida por Computación, es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo de software, su objetivo es acelerar el proceso de automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas.

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

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

Framework: un framework, es una 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.

Java: Lenguaje de programación orientado a objetos desarrollado por *Sun Microsystems* a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++.

JavaScript: Es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

JDBC: (del inglés, Java Database Connectivity) es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java,

CLOSARIO DE TÉRMINOS

independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

JSON: (JavaScript Object Notation) Notación de Objetos JavaScript, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

Mercancías: todos los bienes corporales muebles de comercio o no, con la sola excepción de los Efectos Personales de los viajeros.

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

OMA (Organización Mundial de Aduanas): es un organismo internacional dedicado a ayudar a los países miembros (normalmente representado por las respectivas aduanas) a cooperar y estar comunicados entre ellos en materia aduanera.

OMC (Organización Mundial del Comercio): es la organización internacional que se ocupa de las normas que rigen el comercio entre los países.

ORM: (*Object-Relational Mapping*) Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional.

Reempaque: se realiza con frecuencia en los Depósitos de Aduana, es la repetición de la acción y efecto de empacar la mercancía.

Régimen Aduanero: tratamiento aplicable a las mercaderías sometidas al control de la aduana, de acuerdo con las leyes y reglamentos aduaneros, según la naturaleza y objetivos de la operación.

Tecnologías de la Información y las Comunicaciones (TIC's): Se denominan Tecnologías de la Información y las Comunicaciones (TIC's) al conjunto de tecnologías que permiten la adquisición, producción, almacenamiento, tratamiento, comunicación, registro y presentación de informaciones, en forma de voz, imágenes y datos contenidos en señales de naturaleza acústica, óptica o electromagnética.

UML: es el Lenguaje de Modelado Unificado para detallar, construir, visualizar y documentar las partes o artefactos de un software.

GLOSARIO DE TÉRMINOS

WEB: Es un medio de comunicación de texto, gráficos y otros objetos multimedia a través de Internet, es decir, la web es un sistema de hipertexto que utiliza Internet como su mecanismo de transporte o desde otro punto de vista, una forma gráfica de explorar Internet.

XML: es la abreviatura de lenguaje de marcas extensible, está diseñado para estructurar, transportar y almacenar información. XML no es un reemplazo de HTML sino un complemento a este.