Universidad de las Ciencias Informáticas

Facultad 3

Título: Modelación e implementación de los módulos Alertas y Prórrogas del subsistema Depósitos de Aduana.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(a): Elizabeth Álvarez Marín

Tutor: Ing. Lianet Pineda de la Nuez

Ing. Yuniel Rodríguez Bello

Ciudad de la Habana, junio de 2011.

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Departamento de Soluciones para la Aduana de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

Elizabeth Álvarez Marín

_____

Firma del Autor

Lianet Pineda de la Nuez

_____

Firma del Tutor

Yuniel Rodríguez Bello

_____

Firma del Tutor

Agradecimientos

Quiero agradecer primeramente a mami Luzma por ayudarme en todo y darme lo mejor de sí.

A Carri por ser para mi más que un padre y apoyarme como tal.

A mi tía Esperanza, tío Juan Enrique, a Gelo y a Ana por ayudarme y apoyarme en todo.

A mis amigos Juan, Yoan, Adri, Ricardo y Ángel por ayudarme y estar en todo momento.

A mis amigas Dana, Gretter, Aylin, Daneisy, Yusney, Yenny y Yuri por su incondicionalidad.

A mis compañeros de grupo por compartir conmigo durante estos 5 años.

Y a mis tutores por su ayuda.

Dedicatoria

Dedico este trabajo de diploma a mami Luzma y a Carri por su amor incondicional.

A mis abuelitos Mila, Eliecer e Icha por brindar tanto cariño.

A mi hermano por quererme mucho.

A mis tíos Esperanza, Juan Enrique, Ana y Gelo por quererme como su hija.

A mi papá Julio que, aunque no tengamos muy buenas relaciones, sé que en algún momento se va a sentir orgulloso de mí.

Resumen

La informatización de los procesos aduaneros en Cuba como parte del desarrollo tecnológico que precisa actualmente, demanda una elevada capacidad tecnológica y operativa, lo que trae consigo la utilización de herramientas computacionales para el procesamiento de la información. La Aduana General de la República de Cuba se ha planteado la informatización de la mayoría de los procesos que se realizan en sus instalaciones, siendo uno de sus principales propósitos mantener un control estricto del movimiento de inventario así como detectar las infracciones cometidas por los depositarios en los Depósitos de Aduana. El presente trabajo tiene como objetivo desarrollar los módulos Alertas y Prórroga del subsistema Depósito de para facilitar la detección de infracciones en la normativa aduanera y llevar a cabo el control de las prórrogas.

En el desarrollo de la solución se utilizó como metodología de desarrollo de software Rational Unified Process (RUP) con adaptaciones propias del Departamento de Soluciones para la Aduana, valiéndose del lenguaje de modelado Unified Modeling Language (UML) para generar los artefactos. Para lograr un buen entendimiento del negocio se realizó el modelado de procesos del negocio utilizando la notación Business Process Modeling Notation (BPMN) y se definieron los requisitos teniendo en cuenta las técnicas de captura y validación de requisitos. También se generaron los artefactos correspondientes al diseño y la implementación como diagramas de clases del diseño con estereotipos web y diagrama de componentes, además se validó la solución con la realización de pruebas de caja blanca y caja negra.

***Palabras claves*:** Proceso Aduanero, Alerta, Prórroga, Infracciones.

Índice

AGRADECIMIENTOS III
DEDICATORIA IV
RESUMEN V
INTRODUCCIÓN 9
FUNDAMENTACIÓN TEÓRICA 13
1.1 Introducción 13
1.2 Características de los Depósitos de Aduana 13
1.3 Sistemas informáticos que gestionan las operaciones en los Depósitos de Aduana 15
1.3.1 Depósito Aduanero 15
1.3.2 S4 tr@nsERP 16
1.3.3 SIDUNEA 17
1.3.4 CLIP™ 19
1.4 Ingeniería de Requerimientos 20
1.4.1 Técnicas de captura de requerimientos 20
1.4.1 Técnicas de validación de requisitos 22
1.5 Diseño de Software 22
1.5.1 Arquitectura de Software 22
1.5.2 Estilos arquitectónicos 22
1.5.3 Patrones 23
1.6 Metodología, lenguajes y herramientas de modelado utilizadas para el desarrollo 29
1.6.1 Metodología de desarrollo de software 29
1.6.2 Notación para el Modelado de Procesos de Negocio 31
1.6.3 Leguaje de Modelado 32
1.6.4 Herramienta CASE para el modelado 33
1.6.5 Lenguaje de programación 33
1.6.6 Marco de trabajo (Framework) 33

1.6.7 Interfaz de usuario 34
1.6.8 Gestor de Base de Datos 34
1.7 Conclusiones 35
ANÁLISIS Y DISEÑO 36
2.1 Introducción 36
2.2 Modelado de los procesos del Negocio 36
2.2.1 Descripción del subproceso Identificar incidencias 37
2.2.2 Descripción del subproceso Otorgar Prórrogas 38
2.3 Modelo conceptual 38
2.4 Especificación de los Requerimientos de Software 39
2.4.1 Técnicas para la Captura de Requerimientos 39
2.4.2 Requerimientos Funcionales 40
2.4.3 Técnicas para la Validación de los Requerimientos 41
2.5 Diagrama de Paquetes 41
2.6 Diagrama de clases del diseño 42
2.6.1 Diagramas de clases del diseño con estereotipos Web 43
2.7 Diagramas de Vistas por escenarios 44
2.8 Patrones utilizados 45
2.9 Arquitectura MVC según Symfony 47
2.10 Diseño del modelo de datos 48
2.11 Conclusiones 50
IMPLEMENTACIÓN Y PRUEBA 51
3.1 Introducción 51
3.2 Implementación 51
3.3 Estándares de codificación utilizados 51
3.4 Elementos utilizados para el desarrollo de la solución 52

3.4.1 Tareas de Symfony 52
3.4.2 Variables de Symfony 53
3.5 Diagrama de Componente 53
3.6 Diagrama de despliegue 55
3.7 Validación de la solución 55
3.7.1 Tipo de prueba a realizar 55
3.7.2 Aplicación de la Prueba de Caja Blanca o Estructural 56
3.7.3 Aplicación de la Prueba de Caja Negra o Funcional. 61
3.8 Conclusiones 64
CONCLUSIONES 65
RECOMENDACIONES 66
GLOSARIO DE TÉRMINOS 67
REFERENCIAS BIBLIOGRÁFICAS 68
BIBLIOGRAFÍA 70

Introducción

La Aduana General de la República de Cuba (AGR) es una organización que controla el tráfico internacional de medios de transporte, mercancías, viajeros y bultos postales, contribuyendo a la protección nacional e internacional del medio ambiente. La capacidad de circular artículos a través de fronteras internacionales en forma rápida, segura y a un costo razonable, puede otorgar grandes ventajas al país e impulsar el desarrollo socioeconómico del mismo.

En este contexto la aduana juega un rol crítico, no solo para lograr las metas gubernamentales, sino también la efectividad de los controles que aseguren las recaudaciones, el cumplimiento de la legislación nacional, la protección y seguridad de la sociedad.

La AGR se ha planteado la informatización de la mayoría de sus procesos y la digitalización de los documentos que se utilicen en ellos, es de gran importancia la gestión y control de los procesos en los Depósitos de Aduana; los que se definen como cualquier lugar o instalación autorizado por la Aduana para el almacenamiento de mercancías sujetas al régimen aduanero[1] depósito de aduanas[2], el que puede ser público o privado.

Actualmente se benefician del régimen de aduanas los importadores comerciales, las sucursales y los agentes de sociedades mercantiles extranjeras. La normativa aduanera vigente establece plazos en los que los depositarios[3] deben entregarle información relativa a las entradas, salidas, movimientos de su inventario, sin embargo la Aduana no posee un software que permita recepcionar la información proveniente de los sistemas informáticos de los depositarios con el objetivo de vincularla al proceso de despacho aduanero[4].

[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 aduanas), 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.*

[4] *Cumplimiento de las formalidades aduaneras necesarias para exportar, importar o para colocar las mercancías bajo otro régimen aduanero.*

Por tal motivo los depositarios proporcionan la información de interés para la Aduana mediante soportes digitales o documentos generados en el formato tradicional, motivo que provoca gastos de recursos, tales como: combustible, materiales de oficina, entre otros. El procesamiento de dicha información se realiza de manera manual, razón por la cual el trabajo se dificulta, y se vuelve un proceso engorroso e ineficiente y en muchas ocasiones la información no se tiene en el momento preciso para la toma de decisiones oportuna y las acciones correctivas necesarias.

Por consiguiente la detección de infracciones cometidas en la normativa aduanera que está vigente, tales como: depósitos que incumplan en la entrega de la información, sobrepasen el tiempo regulado sin realizar operaciones de entrada y salida de mercancías, contengan mercancías que deban ser puestas a favor del estado cubano por incumplimientos en su extracción o que sobrepasen el tiempo normado sin que se realice ninguna transacción de las mismas; se complejiza y en muchas ocasiones no se detectan en el momento oportuno para que la Aduana pueda aplicar la medida que corresponda.

La Aduana otorga prórrogas a los depósitos para que en ocasiones excepcionales puedan aplazar la entrega de la información, actualmente el control de las mismas se realiza de forma manual por lo que este proceso es lento y no satisface las expectativas trazadas.

A partir de la problemática planteada se definió como **problema a resolver:** ¿cómo gestionar los procesos asociados al control de las prórrogas y las infracciones cometidas por los depositarios en los Depósitos de Aduana? Definiéndose como **objeto de estudio**: el proceso de desarrollo de software en la gestión aduanera.

Según lo planteado anteriormente se establece como **campo de acción**: modelación e implementación de los procesos asociados al control de las prórrogas y las infracciones cometidas por los depositarios en los Depósitos de Aduana.

Una vez planteada la problemática se define como **objetivo general**: desarrollar un sistema informático que gestione los procesos asociados al control de las prórrogas y las infracciones cometidas por los depositarios en los Depósitos de Aduana.

De ahí se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico relativo a las soluciones de software que gestionan los Depósitos de Aduana y el diseño de sistemas informáticos.
- Modelar los procesos asociados al negocio, objeto de informatización.

- Especificar los requisitos funcionales.
- Modelar el diseño del sistema para definir las clases y componentes necesarios para su implementación.
- Implementar un sistema que gestione el control de las prórrogas e infracciones en los depósitos de aduana.
- Realizar pruebas del sistema.

Para darle cumplimiento a los objetivos específicos se definieron las siguientes tareas:

- Análisis del estado del arte de las soluciones de software que gestionan los depósitos de aduana.
- Análisis de la metodología de desarrollo de software, lenguajes, herramientas de modelado y tecnologías utilizadas en el proyecto.
- Análisis de la normativa aduanera vigente correspondiente al control de las infracciones cometidas y las peticiones de prórrogas en los depósitos de aduana.
- Definición de los requisitos funcionales del sistema a desarrollar.
- Modelación de los artefactos del diseño: diagrama de clases, diagrama de vistas por escenarios, y modelo de datos.
- Implementación de la solución propuesta.
- Diseño de casos de prueba.
- Registro de los resultados de las pruebas.

Luego del desarrollo de las tareas anteriores se definieron como **posibles resultados:**

- Documentación del Estado del Arte.
- Modelo de procesos del negocio.
- Descripción de los procesos del negocio.
- Especificación detallada de los requisitos funcionales.
- Modelo de datos.
- Diagrama de clases del diseño.

- Diagrama de secuencia.
- Diagrama de despliegue.
- Código fuente del sistema, que permitirá el control de las infracciones cometidas por los depositarios en los depósitos de aduana y las prórrogas solicitadas por los mismos.

El documento está estructurado de la siguiente manera:

El primer capítulo (Fundamentación teórica) incluye un estado del arte sobre la gestión de Depósitos de Aduana a nivel internacional y nacional para la solución del problema que se enfrenta en la actualidad, detalla la metodología, lenguaje y herramientas de modelado para el desarrollo utilizadas en el proyecto, y muestra un análisis de los patrones de desarrollo de software, el segundo capítulo (Análisis y Diseño) expone una explicación detallada del problema y describe el flujo actual de los procesos para el control de las prórrogas y las infracciones cometidas por los depositarios, también plasma el diagrama de procesos correspondiente así como el modelo conceptual referente a dichos procesos, los requisitos funcionales del software y una descripción de los mismos. También describe la arquitectura del subsistema, muestra los diagramas de clases del diseño, de vistas por escenarios, así como la descripción de los patrones de diseño utilizados en la solución y el tercer capítulo (Implementación y prueba), enfoca la construcción de la solución explicando los aspectos principales de la implementación como es la utilización de los framework de desarrollo. Además se presenta el diagrama de componentes correspondiente a los módulos Alertas y Prórrogas y se muestran los casos de prueba realizados.

Fundamentación Teórica

1.1 Introducción

En el capítulo se realiza un estudio de las soluciones existentes para la gestión de las operaciones en los Depósitos de Aduana, además se caracterizan brevemente las técnicas para la captura y validación de requisitos, así como algunos aspectos de los estilos y patrones arquitectónicos, patrones de diseño, herramientas y tecnologías empleadas en el proyecto.

1.2 Características de los Depósitos de Aduana

Se entiende por Depósito Aduanero el régimen especial mediante el cual las mercancías extranjeras, nacionales o nacionalizadas, son depositadas en un lugar destinado a este efecto, bajo control y potestad de la aduana, sin estar sujetas al pago de impuestos de importación y tasa por servicios de aduana, para su venta en los mercados nacionales e internacionales, previo cumplimiento de los requisitos legales. (1)

En nuestro país 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:

Depósitos Públicos

Los depósitos públicos son aquellas instalaciones debidamente autorizadas en las que varios depositantes autorizados a utilizar el régimen depósito de aduana almacenan mercancías, siendo el depositario una persona jurídica distinta a los depositantes[5]. (2)

La instalación de los mismos estará subordinada a la existencia de un tráfico comercial importante o a la satisfacción de necesidades relacionadas con el comercio exterior.

[5] *Persona que deposita o a nombre de quien se depositan las mercancías.*

Depósitos Privados

Los depósitos de aduana privados son los reservados al uso exclusivo del titular que disfruta del régimen para el almacenaje de mercancías propias de su actividad. Los titulares de estos depósitos ostentarán simultáneamente la condición de depositario y depositante. (2)

Los locales destinados a depósitos privados deberán reunir los requisitos que garanticen el control aduanero y la seguridad fiscal.

Bodega Pública

Es un espacio dentro del depósito público, que formando parte de este está destinado a prestar los servicios de recepción, almacenaje, custodia y facturación de las mercancías de varios depositantes. (2)

La utilización del Régimen de Depósito de Aduana 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 nuestro país. Este régimen aduanero facilita a las sucursales extranjeras acreditadas en Cuba la comercialización de sus productos dentro del territorio nacional. Estas entidades pueden importar a los depósitos productos que pretenden comercializar en Cuba sin pagar derechos de importación, los que serán abonados por la entidad importadora que los adquiera una vez declarada la importación definitiva de la mercancía.

Otra ventaja es que 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 declaradas a este régimen pueden ser reexportadas bajo el régimen 3071[6], constituyendo esto una alternativa para los casos en los cuales no se logra concretar la importación o

[6] *Régimen aduanero que se utiliza en la Importación y Exportación de Mercancías para la reexportación procedente de provisiones de abordo.*

venta en el país, entonces terceros países pueden aparecer como posibles mercados, incluso, el destino podría ser el mismo país del proveedor, por ejemplo, en caso de devolución por incumplimiento de especificaciones técnicas acordadas en el contrato de compraventa suscrito entre las partes. También puede darse cuando se venza el plazo de permanencia en el depósito y no tengan rotación.

1.3 Sistemas informáticos que gestionan las operaciones en los Depósitos de Aduana

Actualmente muchas Aduanas en el mundo utilizan aplicaciones informáticas que gestionan las operaciones en depósitos bajo régimen aduanero, seguidamente se realiza un análisis de algunas de ellas.

1.3.1 Depósito Aduanero

Aplicación especialmente diseñada y desarrollada para la explotación de Depósitos Aduaneros en cualquiera de sus facetas:

- Depósito Aduanero (Privado o Público).
- Depósito Distinto del Aduanero.
- Almacén de Avituallamiento.
- Almacén de Depósito Temporal (A.D.T.).
- Local Autorizado para Mercancías de Exportación (L.A.M.E.).
- Almacenaje Mercancías Nacionales. (3)

Aplicación "Abierta".- aunque, en principio, la aplicación se contempla como estándar, existe la posibilidad de efectuar las modificaciones o ampliaciones en la medida en que los usuarios requieran para la perfecta integración con la aplicación general de la Empresa mediante la importación o exportación de ficheros con la información necesaria. (3)

Gestión telemática de todos los documentos E.D.I.[7] (Vía Internet): el envío y recepción de documentos EDI (Intercambio electrónico de datos) es totalmente transparente para el usuario. (3)

[7] *Intercambio electrónico de datos: intercambio entre sistemas de información, por medios electrónicos, de datos estructurados de acuerdo con normas de mensajes acordadas.*

1.3.2 S4 tr@nsERP

Dispone de distintos módulos de depósitos basados en los requerimientos de cada una de las figuras aduaneras que gestionan, estos además de los controles clásicos de un almacén incorporan las transmisiones EDI, los diferentes listados de control requeridos por la Aduana, y pueden funcionar como una aplicación independiente o integrada al resto de los módulos de S4 tr@nsERP.

Módulos:

- Almacén de Depósito Temporal (ADT).
- Depósito Aduanero.
- Depósito Distinto del Aduanero.
- Depósito Fiscal.
- Depósito para Restitución.(4)

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 lenguaje de marcas extensible por sus siglas en inglés XML y herramientas de Office System de Microsoft para importación y exportación de información. (4)

Características del módulo depósito aduanero

- Dos versiones: Público y Privado.
- Inclusión en Depósito Aduanero Privado (IDA).
- Registro de entradas y salidas.
- Control por referencias de producto.
- Ventas en Depósito.
- Transformaciones en depósito.
- 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. (4)

1.3.3 SIDUNEA

Es una herramienta informática utilizada en varios países, con el objetivo de mejorar el comercio internacional, a través de medidas que incluyen reformas a la práctica administrativa ya existente. Este emplea códigos internacionales y estándares desarrollados por la Organización Internacional para la Estandarización (ISO). Además puede ser configurado para adaptarse a los requerimientos operacionales de cualquier administración aduanera, pues su sistema de configuración permite a una aduana definir la información opcional, condicional y obligatoria que considere necesaria.

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

- Optimiza los tiempos y recursos del proceso aduanero.

- Aplica la ley con toda justicia.
- Cobra correctamente los impuestos y tasas.
- Detecta los errores en los valores de la declaración.
- Monitorea el pago de los impuestos.
- Evita la evasión de impuestos.
- Minimiza el contrabando.
- Crea incentivos para el declarante.
- Administra efectivamente el proceso de despacho, poniendo en práctica un esquema de garantía con la modalidad de pago anticipado, para facilitar el comercio y asegurar el cobro de los derechos aduaneros.
- Controla la ruta de comercio por medio de las oficinas de despacho de mercancía de cada aduana. (5)

Beneficios del Sistema SIDUNEA

- Evita la imposición de patrones organizacionales en la administración de la aduana.
- El SIDUNEA actualiza permanentemente las tablas, acuerdos y normas por lo que el usuario realizará declaraciones de acuerdo a las legislaciones vigentes.
- El usuario podrá ver detalles de sus declaraciones almacenadas localmente en su estación de trabajo y simplemente actualizar los datos.
- Posee un módulo de selectividad que permite a la aduana acelerar el proceso de despacho y mejorar su capacidad de control. Selecciona declaraciones para control empleando criterios aleatorios de selección local.
- Permite una auditoría completa de los archivos que contienen el registro de una transacción comercial (importación/exportación). (5)

¿Cómo está constituido el sistema SIDUNEA?

SIDUNEA es un programa cliente-servidor, que en términos prácticos permite trabajar con o sin conexión a la red. Como servidor, tiene una configuración de red Ethernet para que un gran número de usuarios ingresen y procesen información sin perder rendimiento, genera datos estadísticos sobre comercio exterior y permite el intercambio electrónico de datos entre comerciantes y aduana. El

software continuamente se mejora y se actualiza según las experiencias en el despacho de mercadería y estadísticas de la aduana internacional. (5)

Características tecnológicas

SIDUNEA permite el uso de tarjetas Inteligentes con procesadores y tecnología JAVA para controlar los accesos al sistema y los pagos electrónicos, su aplicabilidad vía Internet, permite acceder al sistema a través de dispositivos inalámbricos.

El Módulo de Aduanas trabaja principalmente con la Declaración de Mercancías de Exportación (DME), su ingreso al sistema, su verificación local y remota, registro, aplicación del resultado de selectividad y validación. Adicionalmente, contiene opciones de reporte para verificar el estado de bienes declarados bajo regímenes suspensivos, tales como el Depósito de Aduanas. (5)

1.3.4 CLIP™

Esta solución fue creada por la empresa Cotecna para la gestión de depósitos de aduanas, según los requisitos de control aduanero, que proporciona una imagen a tiempo real de los niveles del inventario y asegura que se apliquen controles aduaneros continuos y medidas de seguridad a lo largo de la cadena logística del depósito. CLIP™ es muy eficaz porque une en una sola plataforma los depósitos y la logística, asegurando así la coexistencia óptima de la gestión del espacio, los informes y las funciones de control aduanero.

Los servicios de gestión de depósitos de aduanas comprenden diez funciones principales:

- Recepción de mercancías en depósito.
- Gestión y logística.
- Almacenaje.
- Consolidación, desconsolidación, entre otras.
- Seguridad y protección.
- Controles de inventario en tiempo real hasta su cesión.
- Gestión del pago de impuestos y aranceles.
- Documentación.

- Seguro.
- Informes sobre la gestión.
- Otros servicios (garantías). (6)

Las soluciones informáticas para la gestión de los procesos en los depósitos de aduanas analizadas anteriormente no son viables porque a excepción del SIDUNEA, que es adaptable, los demás implementan normas y legislaciones de sus países de origen, además son herramientas privativas por lo que la obtención de su licencia y soporte puede resultar muy costoso. También muchas de estas soluciones están más enfocadas al crecimiento de las exportaciones que al cumplimiento de la normativa aduanera.

1.4 Ingeniería de Requerimientos

Es una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal. Una representación documentada de una condición o capacidad de un sistema. (7)

La Ingeniería de Requerimientos en sí cumple un papel primordial en el proceso de construcción de un software. Este estará basado en función de las necesidades planteadas por los clientes en un nivel muy general, donde se documentan, analizan y se definen los servicios o componentes del producto a desarrollar, además de las restricciones que tendrá dicha aplicación. Su principal tarea consiste en la definición del proceso a seguir en la construcción de un software, y de facilitar la comprensión de lo que el cliente requiera. La obtención correcta de los requerimientos puede llegar a describir con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento de un sistema.

1.4.1 Técnicas de captura de requerimientos

La captura de requisitos es uno de los problemas que se presentan durante el desarrollo de un software, mediante este proceso se especifican y validan las funciones que debe proveer el sistema, así como las restricciones sobre las que se deberá operar. Esta fase es de gran importancia, ya que los errores más comunes y más costosos de reparar, así como los que más tiempo consumen se deben a una inadecuada ingeniería de requisitos.

Al proceso de especificación de requisitos pertenecen las actividades: captura de requisitos y validación de requisitos. A continuación se presentan brevemente algunas técnicas clásicas para la realización de las mismas.

Captura de requisitos

La captura de requisitos es de vital importancia debido a que permite gestionar las necesidades del proyecto en forma estructurada, esta mejora la capacidad de predecir cronogramas de proyecto proporcionando un punto de partida para controlar actividades específicas, mejora la calidad del software pues si se cumple con todos los requisitos el software poseerá lo que el cliente desea por lo tanto tendrá buena calidad, evita rechazo de usuarios finales debido a que obliga a los usuarios a considerar sus requerimientos cuidadosamente.

Este proceso puede resultar complejo, principalmente si el medio donde se está llevando a cabo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él. A continuación se presentan un grupo de técnicas que han sido utilizadas para esta actividad en el proceso de desarrollo de todo tipo de software.

Entrevistas: Permiten al analista de software tomar conocimiento del problema y comprender los objetivos de la solución buscada. Las entrevistas son una de las formas de comunicación más naturales entre personas por lo que resulta casi inevitable su empleo durante el desarrollo del software.

Tormenta de ideas: Consiste en la acumulación de ideas y/o información sin evaluar las mismas. Como técnica de captura de requisitos es sencilla de usar y de aplicar.

Observación: Permite investigar el fenómeno en su manifestación externa, llevando a cabo un registro visual de lo que ocurre en una situación real. Observar cómo se hacen las cosas es una buena manera de entender lo que estas requieren.

Cuestionarios y Listas de comprobación: Se redacta un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario.

Comparación de terminología: Es utilizada en forma complementaria para obtener consenso respecto a la terminología a ser usada en el proyecto de desarrollo. (8)

1.4.1 Técnicas de validación de requisitos

Una vez definidos los requisitos estos deben ser validados, ya que los mismos tienen como objetivo demostrar que su captura y definición responde a las necesidades del sistema que el cliente realmente desea. Existen algunas técnicas que se pueden emplear para esta actividad:

Prototipos: Son simulaciones del posible producto, permitiendo conseguir una importante opinión en cuanto a si el sistema diseñado a partir de los requerimientos obtenidos le permite al usuario realizar su trabajo de la mejor manera.

Revisiones: Consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos.

Matrices de trazabilidad: Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos. (8)

1.5 Diseño de Software

1.5.1 Arquitectura de Software

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema. Consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema informático. Esta establece los fundamentos para que analistas, diseñadores, programadores trabajen en una línea común que permita alcanzar los objetivos del software, cubriendo todas las necesidades.

1.5.2 Estilos arquitectónicos

Los estilos arquitectónicos definen las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo numeroso y variado de sistemas de software. Determina el vocabulario de componentes y conectores que pueden ser utilizados en instancias de este estilo, con un conjunto de limitaciones en las descripciones arquitectónicas. Los estilos sirven para sintetizar y tener un lenguaje que describa la estructura de las soluciones, definen los patrones posibles de las aplicaciones y permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos no funcionales.

Estilos de Flujo de Datos.

- Tubería y filtros.

Estilos Centrados en Datos.

- Arquitecturas de Pizarra o repositorio.

Estilos de llamada y retorno.

- Modelo - Vista - Controlador (MVC).
- Arquitectura en capas.
- Arquitecturas orientadas a objetos.
- Arquitecturas basadas en componentes.

Estilos de código móvil.

- Arquitectura de máquinas virtuales.

Estilos heterogéneos.

- Sistemas de control de procesos.
- Arquitecturas basadas en atributos.

Estilos Peer-to-Peer.

- Arquitecturas basadas en eventos.
- Arquitecturas orientadas a servicios.
- Arquitecturas basadas en recursos. (9)

1.5.3 Patrones

En el desarrollo de un software ocurren problemas los cuales pueden ser solucionados con el uso de patrones, pues con el transcurso de los años se ha demostrado que son soluciones eficientes a las diferentes dificultades que se han presentado.

Según la definición de 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. (10)

Características de los patrones

- **Son soluciones concretas:** proponen soluciones a problemas concretos, no son teorías genéricas.
- **Son soluciones técnicas:** indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje.
- **Se utilizan en situaciones frecuentes:** ya que se basan en la experiencia acumulada al resolver problemas reiterativos.
- **Favorecen la reutilización de código:** ayudan a construir software basado en la reutilización (a construir clases reutilizables). Los propios patrones se reutilizan cada vez que se vuelven a aplicar.
- **El uso de un patrón no se refleja en el código:** al aplicar un patrón, el código resultante no tiene por qué delatar el patrón o patrones que lo inspiró. No obstante, últimamente hay múltiples esfuerzos enfocados a la construcción de herramientas de desarrollo basados en los patrones y frecuentemente se incluye en los nombres de las clases el nombre del patrón en que se basan, facilitando así la comunicación entre desarrolladores.
- **Es difícil reutilizar la implementación de un patrón:** al aplicar un patrón aparecen clases concretas que solucionan un problema concreto y que no será aplicable a otros problemas que requieran el mismo patrón. (10)

1.5.3.1 Patrones de Asignación de Responsabilidades. (GRAPS)

Los patrones GRASP (General Responsabilty Assignment Software Patterns), permiten la asignación de responsabilidades en parámetros útiles para el diseño del producto.

Entre los más útiles se encuentran:

Patrón Experto: asignar una responsabilidad al experto en la información, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto provee un bajo nivel de acoplamiento.

Patrón Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.

Patrón Controlador: el controlador es un intermediario entre la interfaz de usuario y el núcleo de las clases donde reside la lógica de la aplicación. El controlador no realiza mucho trabajo por sí mismo; más bien coordina la actividad de otros objetos. Asignar la responsabilidad del manejo de mensajes de los eventos del sistema a una clase que represente alguna de las siguientes opciones:

- El sistema global.
- La empresa u organización global.
- Algo activo en el mundo real que pueda participar en la tarea.
- Un manejador artificial de todos los eventos del sistema de un caso de uso.

Patrón Bajo Acoplamiento: una clase con bajo acoplamiento no depende de “muchas otras” clases. Las clases con alto acoplamiento recurren a muchas clases y no es conveniente. Son más difíciles de mantener, entender y reutilizar.

Patrón Alta Cohesión: es la meta principal que ha de buscarse en todo momento. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Su alto grado de funcionalidad, combinada con una reducida cantidad de operaciones, también simplifica el mantenimiento y los mejoramientos. La ventaja que significa una gran funcionalidad también soporta un aumento de la capacidad de reutilización. (10)

Los patrones Pandilla de los cuatro (Gang-of-Four ó GOF) se clasifican en tres grandes grupos:

1.5.3.2 Patrones de creación

Los patrones de creación están vinculados al proceso de creación de los objetos, entre los que se delegan los procesos de creación. A continuación se caracterizan algunos de ellos:

Abstract Factory (Fabricación Abstracta): trabaja con objetos de distintas familias de manera que no se mezclen entre sí, haciendo transparente el tipo de familia concreta que se esté usando.

Builder (Constructor Virtual): facilita la abstracción de un proceso de creación de un objeto complejo, centralizándolo en un único punto.

Factory Method (Método de Fabricación): centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado.

Prototype (Prototipo): crea nuevos objetos clonándolos de una instancia ya existente.

Singleton (Instancia Única): garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a la misma.

1.5.3.3 Patrones Estructurales

Los patrones estructurales tratan la composición de clases u objetos, para lo cual hacen uso de recursos, dependiendo del tipo de composición. A continuación se caracterizan algunos de ellos:

Adapter (Adaptador): adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.

Bridge (Puente): desacopla una abstracción de su implementación.

Composite (Objeto Compuesto): maneja objetos compuestos como si se tratase de uno simple.

Decorator (Envoltorio): facilita la adición de funcionalidad a una clase dinámica.

Facade (Fachada): provee una interfaz unificada y simple para acceder a una interface o grupo de interfaces de un subsistema.

Flyweight (Paso Ligero): reduce la redundancia en circunstancias en las que se producen grandes cantidades de objetos que poseen idéntica información.

Proxy: mantiene un representante de un objeto.

1.5.3.4 Patrones de Comportamiento

Los patrones de comportamiento caracterizan la forma en que las clases y objetos interactúan y se distribuyen la responsabilidad. Este tipo de patrones usan la herencia en las clases para describir algoritmos y flujos de control, y los objetos describen como cooperan un grupo de objetos para realizar una tarea que ninguno podría efectuar por sí solo. A continuación se caracterizan algunos de ellos:

Chain of Responsibility (Cadena de Responsabilidad): establece la línea que deben llevar los mensajes para que los objetos ejecuten la tarea indicada.

Command (Orden): encapsula una operación en un objeto, permitiendo ejecutarla sin necesidad de conocer el contenido de la misma.

Interpreter (Interprete): define una gramática para un lenguaje dado, así como las herramientas necesarias para interpretarlo.

Iterator (Iterador): realiza recorridos sobre objetos compuestos independientemente de la implementación de estos.

Mediator (Mediador): define un objeto que coordine la comunicación entre otros de distintas clases, pero que funcionan como un conjunto.

Observer (Observador): define una dependencia de uno a muchos entre objetos, de forma que cuando uno cambie de estado se notifique y actualicen automáticamente todos los que de él dependen.

Strategy (Estrategia): define una familia de algoritmos, encapsulados e intercambiables. La estrategia permite al algoritmo variar independientemente de los clientes que lo usan.

Template Method (Método de la Plantilla): redefine subclases siguiendo ciertos pasos de un algoritmo sin variar la estructura del mismo.

Visitor (Visitante): representa un funcionamiento a ser utilizado por los elementos de la estructura del objeto, además permite definir un nuevo funcionamiento sin variar las clases de los elementos en que opera.

1.5.3.5 Patrones Arquitectónicos

Layers (Capas): permite estructurar aplicaciones que se pueden descomponer en grupos de sub-tareas, donde cada grupo está en un determinado nivel de abstracción.

Pipes & Filtres (Tuberías y Filtros): provee una estructura para sistemas que procesan un flujo de datos. Cada etapa del proceso es encapsulada como un filtro. Los datos se pasan entre filtros adyacentes mediante “Pipes”. Recombinando filtros se obtienen familias de sistemas relacionados.

Blackboard (Pizarra): Útil para sistemas en que no se conoce una solución o estrategia determinada. Varios subsistemas especializados ensamblan su conocimiento para construir una posible solución parcial.

Broker (Corredor): Permite estructurar sistemas distribuidos desacoplados que interaccionan mediante invocación de servicios remotos. Un componente Broker es responsable de coordinar la comunicación, así como de transmitir resultados y excepciones.

Model-View-Controller (Modelo-Vista-Controlador): Divide una aplicación interactiva en tres componentes, el modelo que contiene la información y funcionalidad principal, la vista que se encarga de mostrar la información al usuario, y las controladoras que gestionan la entrada de usuario. Un mecanismo de propagación de cambios asegura la consistencia entre el modelo y la interfaz de usuario.

Presentation-Abstraction-Control (Presentación-Abstracción-Control): Estructura una aplicación interactiva como una jerarquía de agentes que cooperan. Cada agente es responsable de un determinado aspecto de la funcionalidad y consta de tres componentes: Presentación, Abstracción y Control, que separan la interacción con el usuario de la funcionalidad central y la comunicación con otros agentes.

Reflection (Reflexión): Proporciona un mecanismo para cambiar la estructura y el comportamiento del sistema dinámicamente. La aplicación se divide en dos partes: un meta-nivel y un nivel-base. El meta-nivel hace al software autoconsciente.

1.6 Metodología, lenguajes y herramientas de modelado utilizadas para el desarrollo

En la Universidad de las Ciencias Informáticas el trabajo en los proyectos productivos se organiza por facultades, dentro de las cuales se distribuyen los Centros de Desarrollo. El Centro de Informatización para la Gestión de Entidades (CEIGE), incluye el Departamento de Soluciones para la Aduana, este actualmente desarrolla el producto GINA para la gestión de los distintos procesos aduanales, a continuación se relacionan algunas características de la metodología, lenguajes, herramientas y tecnologías que se utilizan en el proyecto.

1.6.1 Metodología de desarrollo de software

Rational Unified Process (RUP), es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye una metodología para el análisis, implementación y documentación de sistemas orientados a objetos.

RUP es un proceso que se caracteriza por:

***Dirigido por casos de uso*:** los casos de uso describen los requisitos funcionales del sistema desde la perspectiva del usuario y se usan para determinar el alcance de cada iteración y el contenido de trabajo de cada persona del equipo de desarrollo.

Centrado en la arquitectura: la arquitectura permite ganar control sobre el proyecto para manejar su complejidad y controlar su integridad. Hace posible la reutilización a gran escala y provee una base para la gestión del proyecto.

Iterativo e incremental: se divide en cuatro fases: Inicio, Elaboración, Construcción y Transición, y cada una de ellas se divide en iteraciones. En cada iteración se trabaja en un número de disciplinas haciendo énfasis en algunas de ellas. Las disciplinas propuestas por RUP son:

Modelado del negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas, entre otras. Cada iteración añade funcionalidades al producto de software o mejora las existentes. (11)

RUP se divide en 4 fases, dentro de las cuales se realizan varias iteraciones según el proyecto y en las que se hace mayor o menos esfuerzo en las distintas actividades:

- Fase de Inicio (Inspección y Concepción): se hace un plan de fases, donde se identifican los principales casos de uso y se identifican los riesgos. Se concreta la idea, la visión del producto, como se enmarca en el negocio, el alcance del proyecto.
- Fase de Elaboración: se realiza el plan de proyecto, donde se completan los casos de uso y se mitigan los riesgos. planifica las actividades necesarias y los recursos requeridos, especificando las características y el diseño de la arquitectura.
- Fase de Construcción: se basa en la elaboración de un producto totalmente operativo y en la elaboración del manual de usuario. Construir el producto, la arquitectura y los planes, hasta que el producto está listo para ser enviado a la comunidad de usuarios.
- Fase de Transición: se realiza la instalación del producto en el cliente y se procede al entrenamiento de los usuarios. Realizar la transición del producto a los usuarios, lo cual incluye: manufactura, envío, entrenamiento, soporte y mantenimiento del producto, hasta que el cliente quede satisfecho, por tanto en esta fase suelen ocurrir cambios. (11)

Con estas fases se logra ejecutar un conjunto de mejores prácticas, como lo son:

- Desarrollar software iterativamente.
- Modelar el software visualmente.
- Gerenciar los requerimientos.
- Usar arquitecturas basadas en componentes.
- Verificación continua de la calidad.
- Gerenciar los cambios.

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

Actualmente el Departamento de Soluciones para la Aduana se usa la metodología de desarrollo de Software RUP con modificaciones, ya que es un proceso de desarrollo de software configurable que puede ser adaptado a las características propias de cada proyecto. En el proyecto en vez de la utilización del modelo de casos de uso del negocio y los diagramas de actividades, se propone el uso del Modelo de Proceso de Negocio con la notación BPMN (Notación para el Modelado de Procesos de

Negocio). Así como para los conceptos del negocio, en vez del Modelo de Objeto del Negocio, se propone confeccionar el Modelo Conceptual. Se corresponde cada requisito funcional a un caso de uso. Se pasará directamente al diseño ya que los analistas no tienen los conocimientos necesarios de los framework que se utilizan en el proyecto, por lo que los artefactos que se generen no ayudarían a los diseñadores, constituyendo una pérdida de tiempo. (12)

En el proyecto se propone el uso de un nuevo diagrama llamado Diagrama de vistas por escenarios en vez de utilizar el diagrama de secuencias, ya que este presenta algunas desventajas:

- Para modelar de una vifurcación debe crearse un nuevo diagrama.

- Una representación de un diagrama de secuencia demasiado largo, puede ser difícilmente entendido por alguien ajeno al sistema.

1.6.2 Notación para el Modelado de Procesos de Negocio

El modelado de procesos del negocio se realiza mediante la notación estándar Business Process Modeling Notation (BPMN) desarrollada por Business Process Management Initiative (BPMI). El objetivo principal de los esfuerzos de BPMN era dar una notación fácil de entender a todas las personas que gestionan y monitorizan los procesos. BPMN define un Business Process Diagram (BPD), que se basa en una técnica de grafos de flujo para crear modelos gráficos de operaciones de procesos de negocio. Un modelo de procesos de negocio, es una red de objetos gráficos, que son actividades (trabajo) y controles de flujo que definen su orden de rendimiento. Entre los objetivos del desarrollo de BPMN está crear un mecanismo simple para realizar modelos de procesos de negocio, y al mismo tiempo que sea posible gestionar la complejidad inherente en dichos procesos. Un BPD está formado por un conjunto de elementos gráficos. Estos elementos habilitan el fácil desarrollo de diagramas simples que serán familiares para la mayoría de analistas de negocio. Los elementos fueron elegidos para ser distinguibles los unos de los otros y para usar formas familiares para la mayoría de modeladores. Cuenta con cuatro categorías básicas de elementos:

- Objetos de Flujo: Actividades, Eventos y Compuertas (Gateway).
- Objetos de Conexión: Flujo de Secuencia, Flujo de Mensaje y Asociación.
- Bandas: Piscina (Pool) y Lienzo (Lane).

- Artefactos: Objetos de Datos (Data Object), Grupo (Group) y Anotaciones (Annotation). (13)

1.6.3 Leguaje de Modelado

UML 2.1 (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos. Actualmente es el más estandarizado por la industria, debido a que ha sido elaborado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores. (14)

Entre sus principales ventajas se encuentran:

- UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.
- Diagramas de Casos de Uso para modelar los procesos 'business'.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.
- UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. (14)

1.6.4 Herramienta CASE para el modelado

Visual Paradigm 3.4 para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este provee una construcción de las aplicaciones con mayor calidad. Permite modelar todos los tipos de diagramas de clases, generar código inverso, código desde diagramas y documentación. También proporciona abundantes tutoriales, demostraciones interactivas y proyectos UML.

1.6.5 Lenguaje de programación

PHP 5.3 es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas Web dinámicas, este es un lenguaje de código abierto que resulta muy útil para diseñar aplicaciones Web dirigidas a bases de datos, es orientado a objetos y está ampliamente difundido en todo el mundo, presentando grandes volúmenes de documentación que son de gran ayuda para los desarrolladores.

1.6.6 Marco de trabajo (Framework)

Symfony 1.2.8 es un framework desarrollado en lenguaje PHP que facilita el desarrollo de aplicaciones web encargándose de todos los aspectos comunes de las aplicaciones, dejando que el programador se dedique a aportar valor desarrollando las características únicas de cada proyecto.

Symfony aumenta exponencialmente la productividad y ayuda a mejorar la calidad aplicando las buenas prácticas y patrones de diseño que se han definido para la web.

Entre sus características se encuentran:

- Fácil de instalar y configurar en sistemas Windows, Mac y Linux.
- Funciona con todas las bases de datos comunes.
- Compatible solamente con PHP 5.
- Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Es utilizado en aplicaciones empresariales, por su adaptabilidad a las políticas y arquitecturas propias de cada empresa u organización.

- Publicado bajo licencia MIT[8] de software libre y apoyado por una empresa comprometida con su desarrollo.
- Traducido a más de cuarenta idiomas y fácilmente traducible a cualquier otro idioma. (15)

1.6.7 Interfaz de usuario

Ext JS 3.0 es una librería JavaScript para la creación de aplicaciones enriquecidas del lado del cliente y permite construir aplicaciones complejas y agradables para los usuarios. Sus características principales son: gran desempeño, componentes de interfaz de usuario personalizables, con buen diseño y documentación. Tiene como ventaja además que una vez que el código JavaScript haya sido interpretado por el navegador, el usuario se sentirá trabajando en el sistema como si fuera una aplicación de escritorio, porque la transacción de información se realiza a partir de comunicaciones asincrónicas.

1.6.8 Gestor de Base de Datos

Oracle 11g es un Sistema Gestor de Bases de Datos (SGBD) con características objeto-relacionales, que pertenece al modelo evolutivo de SGBD. Sus características principales son las siguientes:

- Entorno cliente/servidor.
- Gestión de grandes bases de datos.
- Usuarios concurrentes.
- Alto rendimiento en transacciones.
- Sistemas de alta disponibilidad.
- Disponibilidad controlada de los datos de las aplicaciones.
- Adaptación a estándares de la industria, como SQL-92.
- Gestión de la seguridad.
- Autogestión de la integridad de los datos.
- Opción distribuida.
- Portabilidad.

[8] *Licencia de Software Libre originaria del Instituto de Tecnología de Massachusetts. Es una licencia permisiva, lo que significa que permite la reutilización de la propiedad del software, compatible con la licencia GPL*

- Compatibilidad.
- Conectividad.
- Replicación de entornos. (16)

1.7 Conclusiones

En la construcción de un software resulta de gran importancia un buen entendimiento de los procesos que se quieren informatizar, y ajustarse al entorno donde se aplicará el mismo.

Tomando como base el análisis realizado anteriormente de las soluciones existentes en el mundo para la gestión de las operaciones en los Depósitos de Aduana, es significativo destacar que resulta poco factible utilizar los sistemas ya existentes porque son privativos y la mayoría no cumple con las legislaciones y normas establecidas por la Aduana General de la República de Cuba, además no se adecuan a la arquitectura del sistema GINA que se está desarrollando en la Universidad de las Ciencias Informáticas. También se debe resaltar que en Cuba no existe una solución de software que gestione las operaciones de este tipo de Depósitos, por lo que es de gran importancia el desarrollo de una aplicación eficaz basada en las buenas prácticas que ofrece el framework de PHP Symfony.

Análisis y Diseño

2.1 Introducción

Este capítulo describe la solución propuesta, mostrando el flujo principal de los procesos de negocio, el modelado de los mismos, así como su descripción; además expone los requisitos funcionales definidos para los módulos Alertas y Prórrogas por considerarse funciones fundamentales que debe garantizar el sistema; y además explica las técnicas de captura y validación de requerimientos utilizadas. Sumado a todo lo anterior, en este capítulo, se encuentran el modelo conceptual, los diagramas de clases del diseño con estereotipos Web, vistas por escenarios correspondientes a los requisitos funcionales "Insertar prórroga" y "Mercancías con tiempo sin operar" y el diagrama de clases del sistema; cuenta también con una breve descripción de los patrones de diseño utilizados en la solución de la aplicación y se presenta el modelo de datos.

2.2 Modelado de los procesos del Negocio

A continuación se describen los principales procesos que se llevan a cabo en los Depósitos de Aduana, referentes a la identificación de incidencias y el control de las prórrogas, para lograr un mejor entendimiento del negocio.

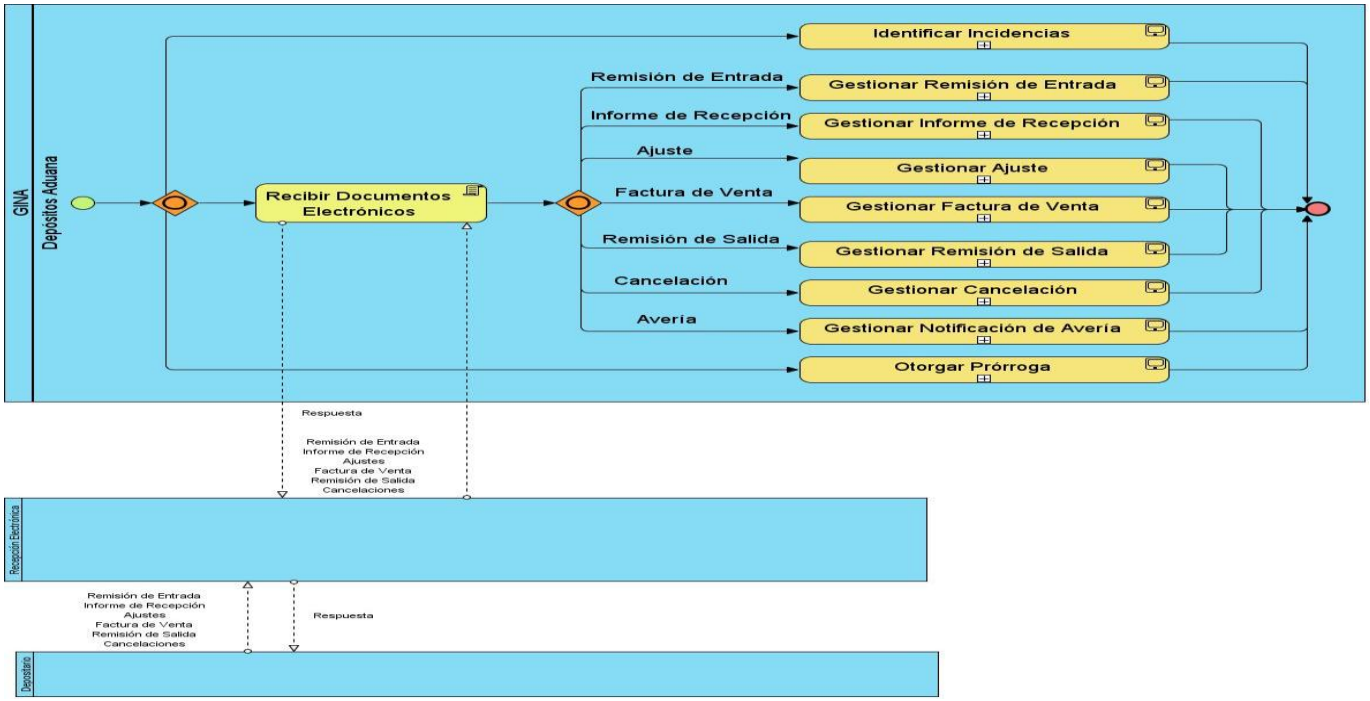

Fig. 1 Proceso Gestionar Depósitos de Aduana

El diagrama mostrado anteriormente representa un Mapa de Procesos[9] donde se muestran todos los procesos asociados a los Depósitos de Aduana, de los cuales se describen Identificar incidencias y Otorgar prorrogas.

2.2.1 Descripción del subproceso Identificar incidencias

El subproceso se inicia diariamente realizando un chequeo de los Depósitos y los documentos registrados en los mismos como las Remisiones de Entrada, los Informes de Recepción, y Facturas de venta, en el caso de que el sistema detecte alguna Remisión de Entrada para la cual no se haya presentado el Informe de Recepción correspondiente y se haya superado el tiempo límite normado para ello, o algún Informe de Recepción para el que no se haya presentado la Declaración de Mercancías (DM) correspondiente, o algún Depósito sin operar y este haya superado el tiempo límite normado, o alguna Factura de Venta que tenga DM presentada, bultos pendientes de extracción y esta haya superado el tiempo de extracción y/o el tiempo de aproximación al abandono, entonces emite la alerta correspondiente al inspector de la aduana notificando la violación.

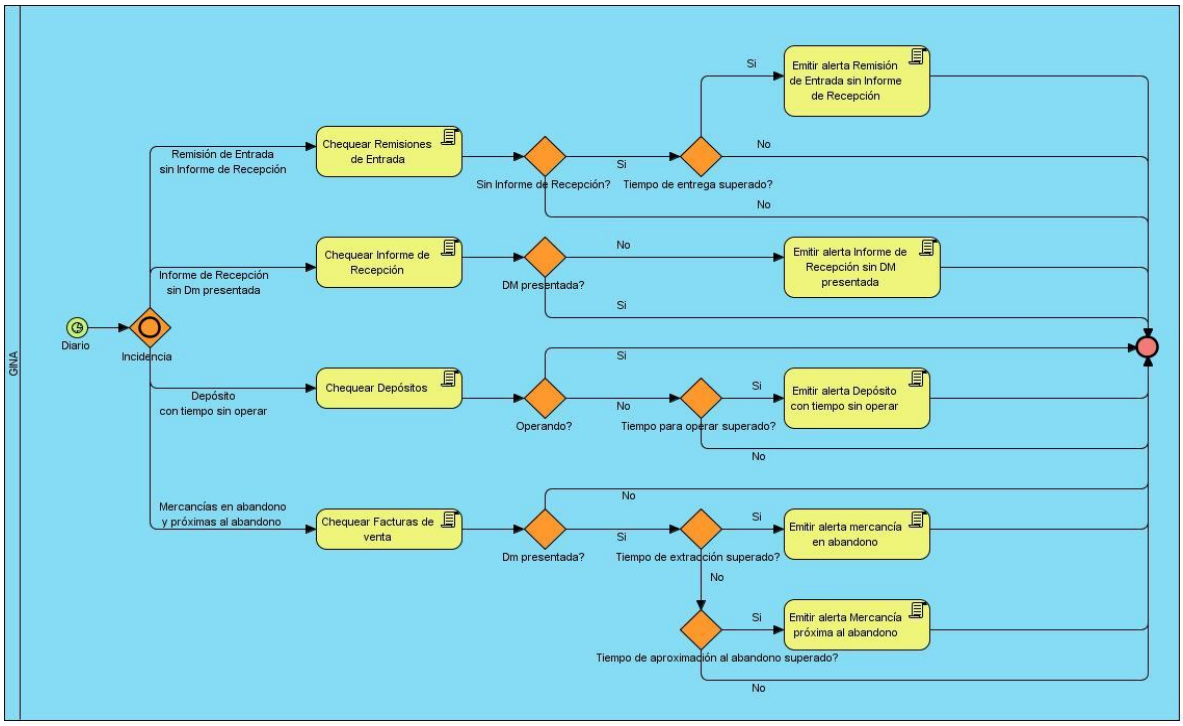

Fig. 2 Subproceso Identificar incidencias

[9] *El mapa de procesos ofrece una visión general del sistema de gestión. EN el se representan los procesos que componen el sistema así como sus relaciones principales. Dichas relaciones se indican mediante flechas y registros que representan los flujos de información.*

2.2.2 Descripción del subproceso Otorgar Prórrogas

El proceso comienza cuando el depositario se comunica por teléfono con la aduana de control correspondiente o se presenta ante el inspector de Aduana solicitando la prórroga, luego el inspector de Aduana valora la gravedad de la situación y decide si acepta la prórroga o la rechaza, en el caso de que la prórroga sea aceptada se registran los datos de la misma en el sistema, en caso contrario notifica al depositario que la prórroga no fue aceptada.

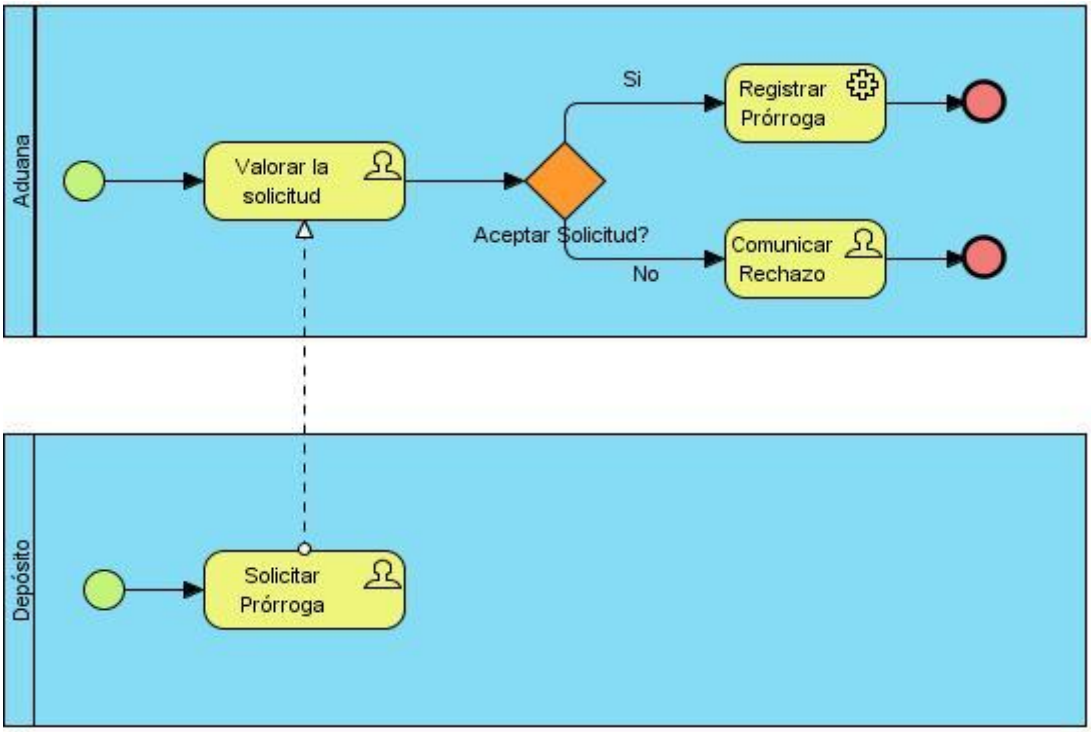

Fig. 3 Subproceso Otorgar prórrogas

2.3 Modelo conceptual

A continuación se muestran los conceptos relacionados con los procesos que se llevan a cabo en los Depósitos de Aduana mediante el Modelo conceptual:

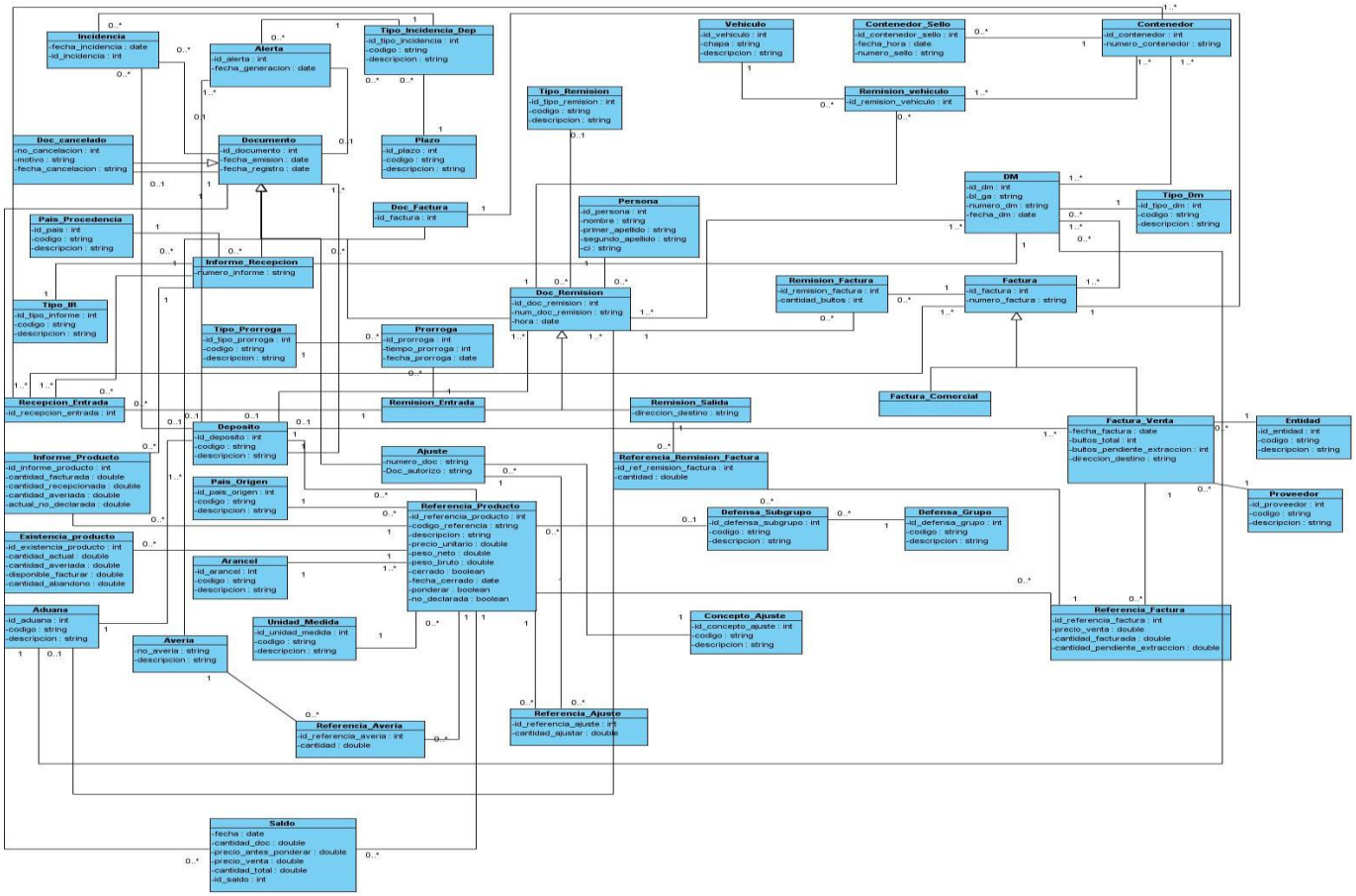

Fig. 4 Modelo conceptual

2.4 Especificación de los Requerimientos de Software

La especificación de los requerimientos del software es de gran importancia porque en la misma se encuentra cada detalle a tener en cuenta para el desarrollo del sistema y puede evitar errores en etapas posteriores, por lo que se deben tener presentes las técnicas para la captura y validación de estos requisitos. A continuación se muestran las técnicas utilizadas y un listado de los requisitos funcionales.

2.4.1 Técnicas para la Captura de Requerimientos

Para realizar la captura de requisitos se utilizaron las siguientes técnicas, las **entrevistas** que se le realizaron a Yamilka Felipe Bringas Especialista principal del Departamento Técnicas Aduaneras de la Jefatura de la Aduana General de la República, funcional con gran conocimiento sobre el tema tratado, por lo que contribuyó a que el analista de software comprendiera los procesos de negocio existentes.

También se auxilió la captura en la técnica de **tormenta de ideas**, con el objetivo de acumular información entre todos los especialistas de la Aduana y la analista de software, la cual fue útil usar porque ofreció una visión general de las necesidades del sistema informático a desarrollar.

2.4.2 Requerimientos Funcionales

No	Nombre	Descripción
1	Mostrar Remisiones de Entrada sin Informe de Recepción.	El sistema debe mostrar por cada depósito la relación de todas las Remisiones de Entrada sin Informe de Recepción y con Informes de Recepción con incumplimientos en su entrega.
2	Mostrar Informes de Recepción sin DM presentadas.	El sistema debe mostrar por cada depósito, los Informes de Recepción emitidos antes de la correspondiente Declaración de Mercancías de régimen 7100 que lo ampara y los que no tengan dicha DM presentada.
3	Mostrar Informe de Recepción fuera de tiempo de realización y de entrega.	El sistema debe mostrar por cada depósito los Informes de Recepción que estén presentados fuera de la fecha de realización y de entrega.
4	Mostrar depósitos con tiempo sin operar.	El sistema debe mostrar los depósitos que hayan superado el tiempo límite para estar sin operar.
5	Mostrar mercancías con tiempo sin operar.	El sistema debe mostrar por cada depósito las mercancías que en el rango de fechas definido no hayan realizado ninguna operación (solo se consideran como operaciones las entradas y salidas).
6	Mostrar mercancías en abandono y próximas al abandono.	El sistema debe mostrar los productos que están en abandono y próximos al abandono en cada depósito.
7	Autorizar prórrogas.	El sistema debe permitir autorizar prórrogas en la terminación o entrega de los informes de recepción de cada depósito.

8	Mostrar prórrogas autorizadas.	El sistema debe mostrar un listado de todas las prórrogas autorizadas de cada depósito.
9	Identificar infracciones de la normativa aduanera.	El sistema debe ser capaz de identificar infracciones cometidas a la normativa aduanera vigente.
10	Mostrar alertas de infracciones.	El sistema debe alertar sobre infracciones cometidas a la normativa aduanera vigente.

Tabla 2. 1 Requerimientos Funcionales del Sistema

2.4.3 Técnicas para la Validación de los Requerimientos

Para la validación de los requerimientos se utilizó la técnica de **prototipos de interfaz de usuario**, mediante la cual se verificó que el sistema cubre las necesidades planteadas por el cliente, viendo reflejadas cada una de las funcionalidades en los prototipos, además se realizaron **revisiones** técnicas a las funcionalidades por parte de los especialistas de la aduana y los analistas del departamento.

2.5 Diagrama de Paquetes

A continuación se muestran los paquetes que se relacionan con el subsistema Depósito de Aduana mediante el diagrama de paquetes.

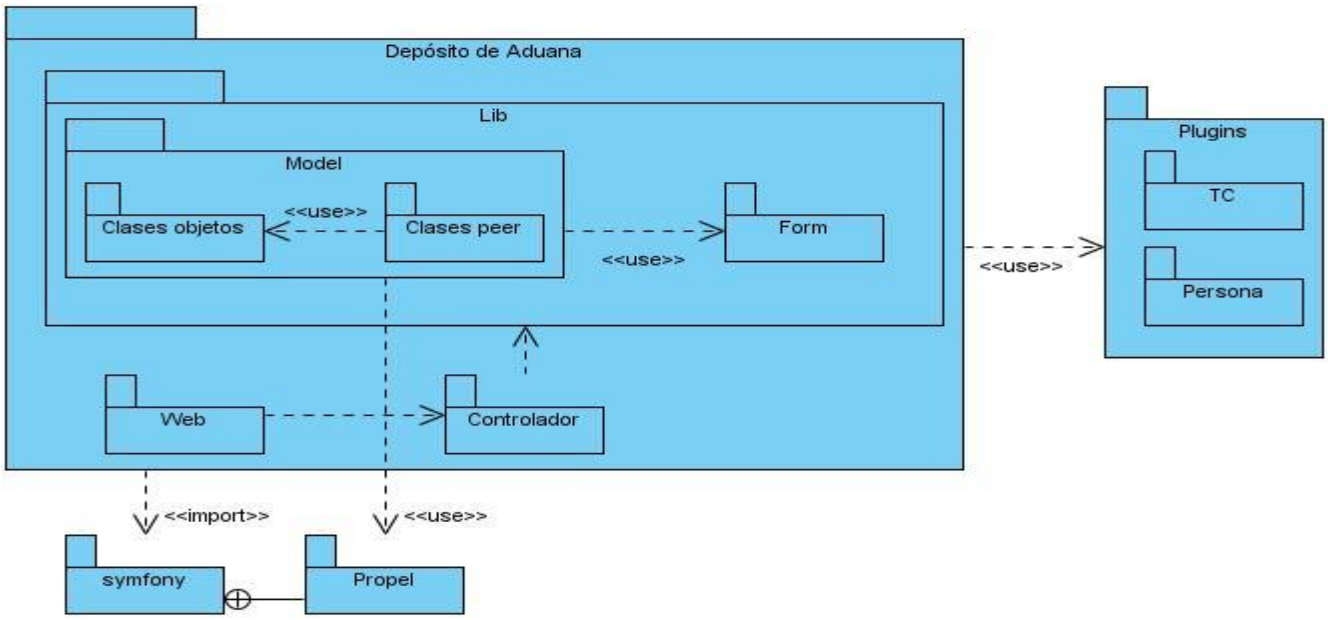

Fig. 5 Diagrama de paquetes de los módulos Alertas y Prórrogas del subsistema Depósito de Aduana.

2.6 Diagrama de clases del diseño

Un diagrama de clases presenta las clases del sistema y las relaciones que existen entre ellas, las cuales pueden ser asociativas, de herencia y de agregación. El mismo está compuesto por los siguientes elementos:

Clase: atributos, métodos y visibilidad.

Relaciones: Herencia, Composición, Agregación y Asociación.

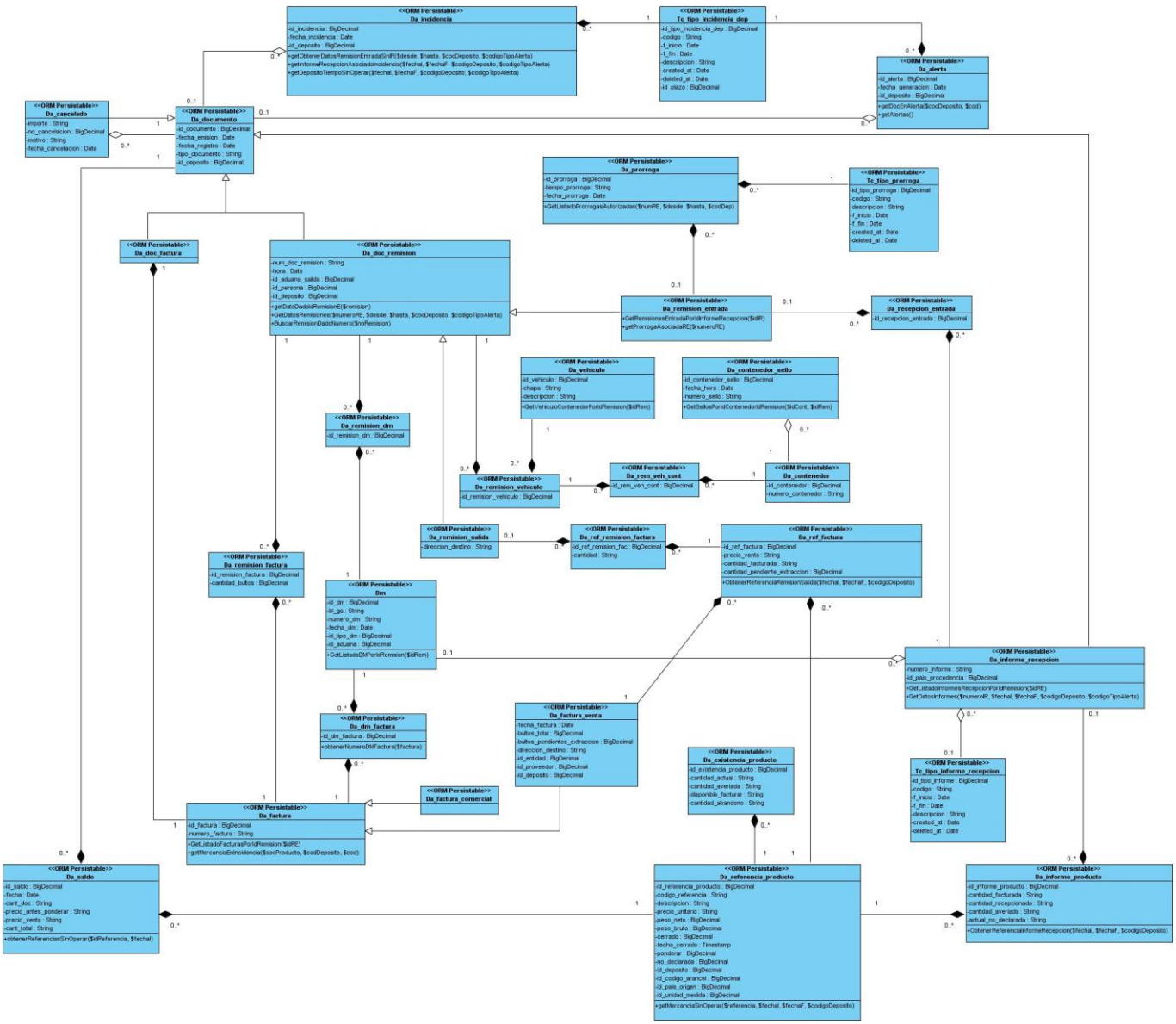

Fig. 6 Diagrama de clases del diseño

2.6.1 Diagramas de clases del diseño con estereotipos Web

En el siguiente diagrama se muestra la interrelación entre las páginas clientes, las servidoras, los formularios y las clases del modelo. El procedimiento se realiza de forma similar en los demás diagramas de clases con estereotipos web.

El controlador principal que está ubicado en el archivo index.php recibe todas las peticiones ya que su principal función es la de ser el único punto de entrada de peticiones, desde el cual se deriva luego al controlador específico que corresponda, según la URL accedida.

El controlador redirecciona hacia el layout, este carga el cuerpo de la plantilla y se construye una página cliente contenedora de todas las interfaces de usuario de la aplicación las cuales se encuentran definidas en el archivo de configuración view.yml, esta hace un link a la página cliente Insertar Prórroga que contiene un formulario por lo que se ve representada una relación de composición entre ambos, siendo este último responsable de enviar los datos al controlador para que sean procesados por las funciones que esta posee usando las clases del modelo.

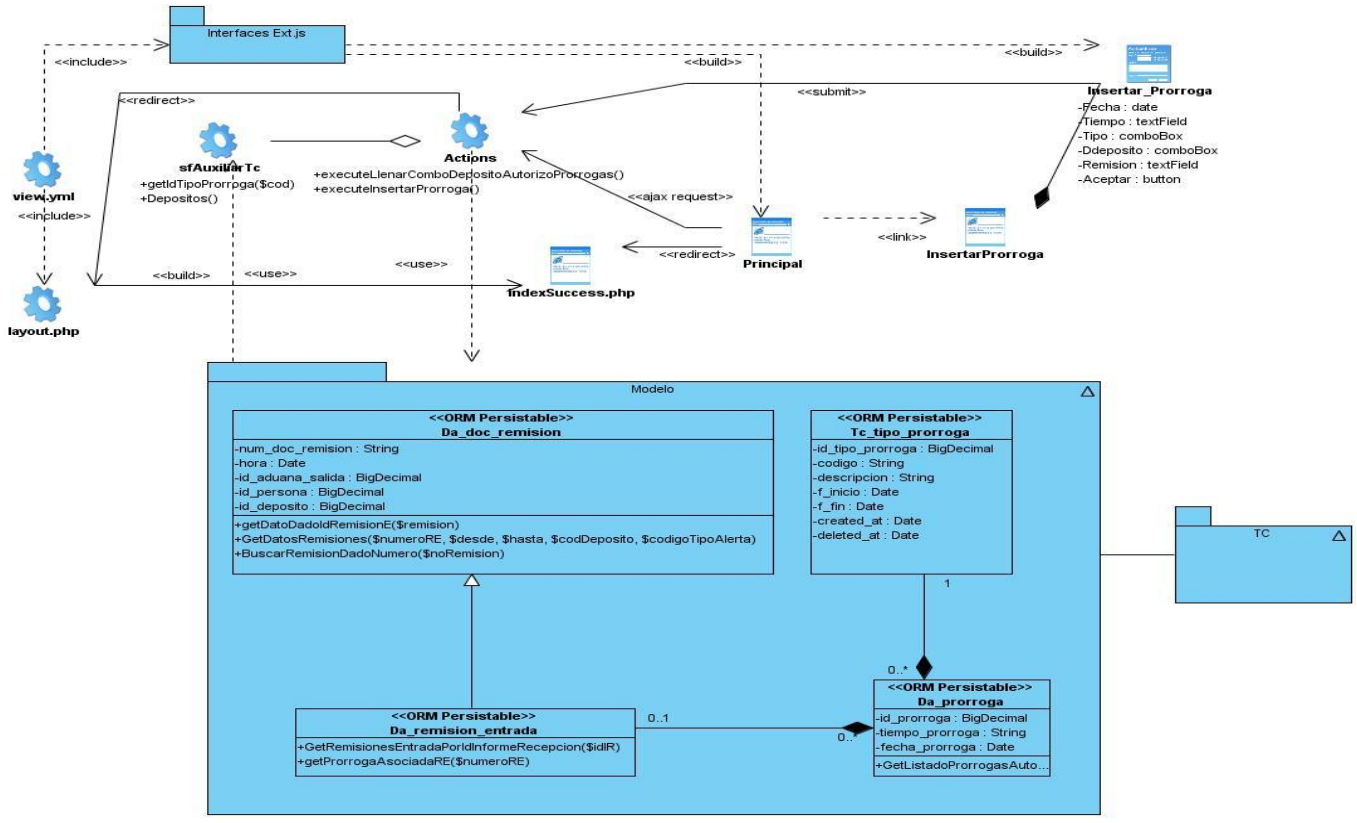

Fig. 7 Diagrama de clases del diseño Autorizar Prórroga

En este diagrama se relacionan cuatro clases DaDocRemision, DaRemisionEntrada, DaProrroga y TcTipoProrroga, un documento de remisión puede ser una Remisión de entrada o una Remisión de salida, en este caso se representa la remisión de entrada a la cual se le pueden otorgar varias prórrogas que pueden ser de realización o de entrega.

2.7 Diagramas de Vistas por escenarios

Los diagramas de Vistas por escenarios son utilizados por el Departamento de Soluciones para la Aduana, con el objetivo de lograr un mayor entendimiento en la etapa de implementación. El mismo muestra el flujo de actividades que se realizan desde que el usuario interactúa con la aplicación por primera vez hasta que el sistema emite una respuesta y donde cada escenario responde a un requisito funcional.

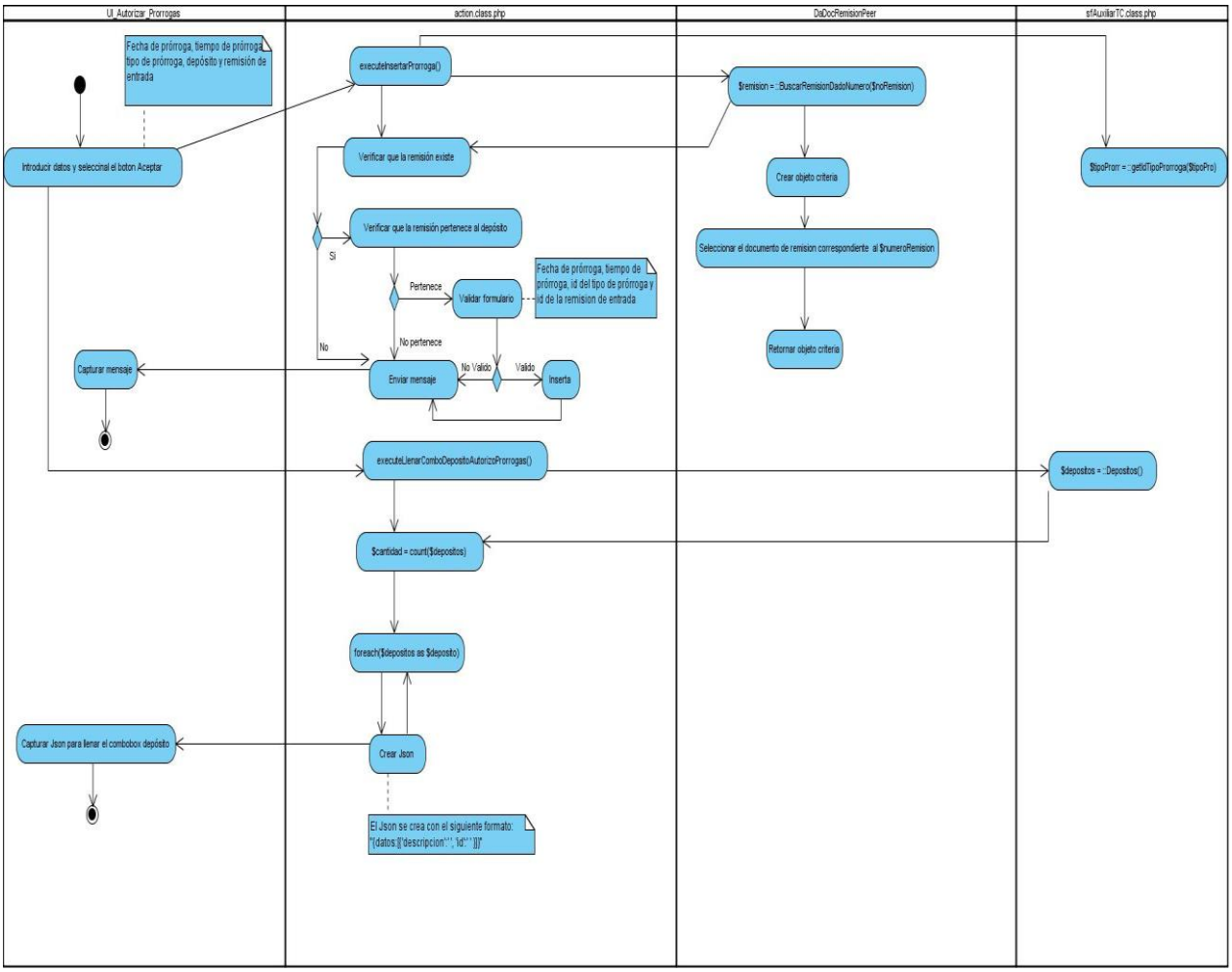

Fig. 8 Autorizar Prórrogas

En el diagrama anterior cada una de las actividades muestra los pasos a seguir durante la implementación para realizar la inserción de una prórroga, donde intervienen cuatro entidades, como son: IU_Autorizar_Prorrogas, action.class.php, DaDocRemisionPeer y sfAuxiliarTc.class.php; mediante la interfaz de usuario IU_Autorizar_Prorrogas se envían los datos a la clase controladora action.class.php que valida los datos a insertar, esta a su vez accede a los objetos de la entidad DaDocRemision a través de la clase de DaDocRemisionPeer que pertenece al modelo del sistema y es capaz de obtener los objetos desde la base de datos. Por otra parte la clase sfAuxiliarTc fue creada con el objetivo de agrupar las funcionalidades que acceden a los datos de otros esquemas mediante servicios.

2.8 Patrones utilizados

Fachada

El patrón fachada se evidencia al relacionar dos aplicaciones, un ejemplo de esto es la clase sysComponent que es utilizada en los servicios para acceder a otros modelos donde funciona como interfaz.

Singleton

El controlador frontal proporciona un punto de acceso global a la aplicación. En una acción, el método getContext() devuelve el mismo singleton, el cual es un objeto que guarda una referencia a todos los objetos del núcleo de Symfony relacionados con una petición dada.

Decorador

La plantilla no es un documento XHTML válido porque le faltan la definición del DOCTYPE y las etiquetas <html> y <body>. El motivo es que estos elementos se encuentran en el archivo llamado Layout.php que contiene el Layout de la página, el cual también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación y en él se integra el contenido de la plantilla, actuando como un decorador de ésta.

Bajo acoplamiento

Las funcionalidades que pueden ser reutilizadas desde cualquier parte del componente, son implementadas de forma tal que el acoplamiento con otras clases sea mínimo. Tal es el caso de la clase sfAuxiliarTc, que contendrá las funcionalidades que obtienen datos de otros esquemas a través de los servicios.

Experto

Es uno de los patrones que más se utiliza cuando se trabaja con Symfony, con la inclusión de la librería Propel para mapear la Base de Datos. Symfony utiliza esta librería para realizar su capa de abstracción en el modelo, 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.

Creador

La clase action.class.php actúa como un creador ya que utiliza específicamente los objetos del modelos y tiene los datos de inicialización que serán transmitidos a estos cuando sean creados.

Alta cohesión

Symfony agrupa las clases por funcionalidades que son reutilizables, bien por su uso directo o por herencia.

Controlador

Symfony implementa para cada vista de la aplicación un controlador que se encarga de atender el evento generado por la vista.

Controlador Frontal (Front Controller)

Symfony aplica además el patrón Controlador frontal, por lo que posee una estructura bien organizada de controladores. El controlador principal que está ubicado en el archivo index.php recibe todas las

peticiones ya que su principal función es la de ser el único punto de entrada de peticiones, desde el cual se deriva luego al controlador específico que corresponda, según la URL accedida.

2.9 Arquitectura MVC según Symfony

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC, que está formado por tres niveles:

- El modelo representa su lógica de negocio.
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, entre otros). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.(16)

Para realizar una aplicación web por más sencilla que sea se necesitan los siguientes componentes:

- La capa del Modelo.
  - Abstracción de la base de datos.
  - Acceso a los datos.
- La capa de la Vista.
  - Vista.

- Plantilla.
- Layout.

➢ La capa del Controlador.

- Controlador frontal.
- Acción.(17)

En total son siete scripts, lo que parecen muchos archivos para abrir y modificar cada vez que se crea una página. Afortunadamente, Symfony simplifica este proceso, toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. (17)

El controlador frontal y el layout son comunes para todas las acciones de la aplicación. Se pueden tener varios controladores y varios rayos, pero solamente es obligatorio tener uno de ellos. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática. Además las clases de la capa del modelo también se generan automáticamente, en función de la estructura de datos de la aplicación. La librería Propel se encarga de esta generación automática, ya que crea el esqueleto o estructura básica de las clases y genera automáticamente el código necesario. Cuando Propel encuentra restricciones de claves foráneas (o externas) o cuando encuentra datos de tipo fecha, crea métodos especiales para acceder y modificar esos datos, por lo que la manipulación de datos se convierte en un juego de niños. La abstracción de la base de datos es completamente transparente para el programador, ya que se realiza de forma nativa mediante PDO (PHP Data Objetos). Así, si se cambia el sistema gestor de bases de datos en cualquier momento, no se debe reescribir ni una línea de código, ya que tan sólo es necesario modificar un parámetro en un archivo de configuración; y la lógica de la vista se puede transformar en un archivo de configuración sencillo, sin necesidad de programarla. (17)

2.10 Diseño del modelo de datos

A continuación se muestra el modelo de datos, el cual permite describir la estructura física de la base de datos, es decir, el tipo de los datos de las columnas y la forma en que se relacionan las tablas, las

restricciones de integridad referencial, que son más que las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada, y las operaciones de manipulación de los datos.

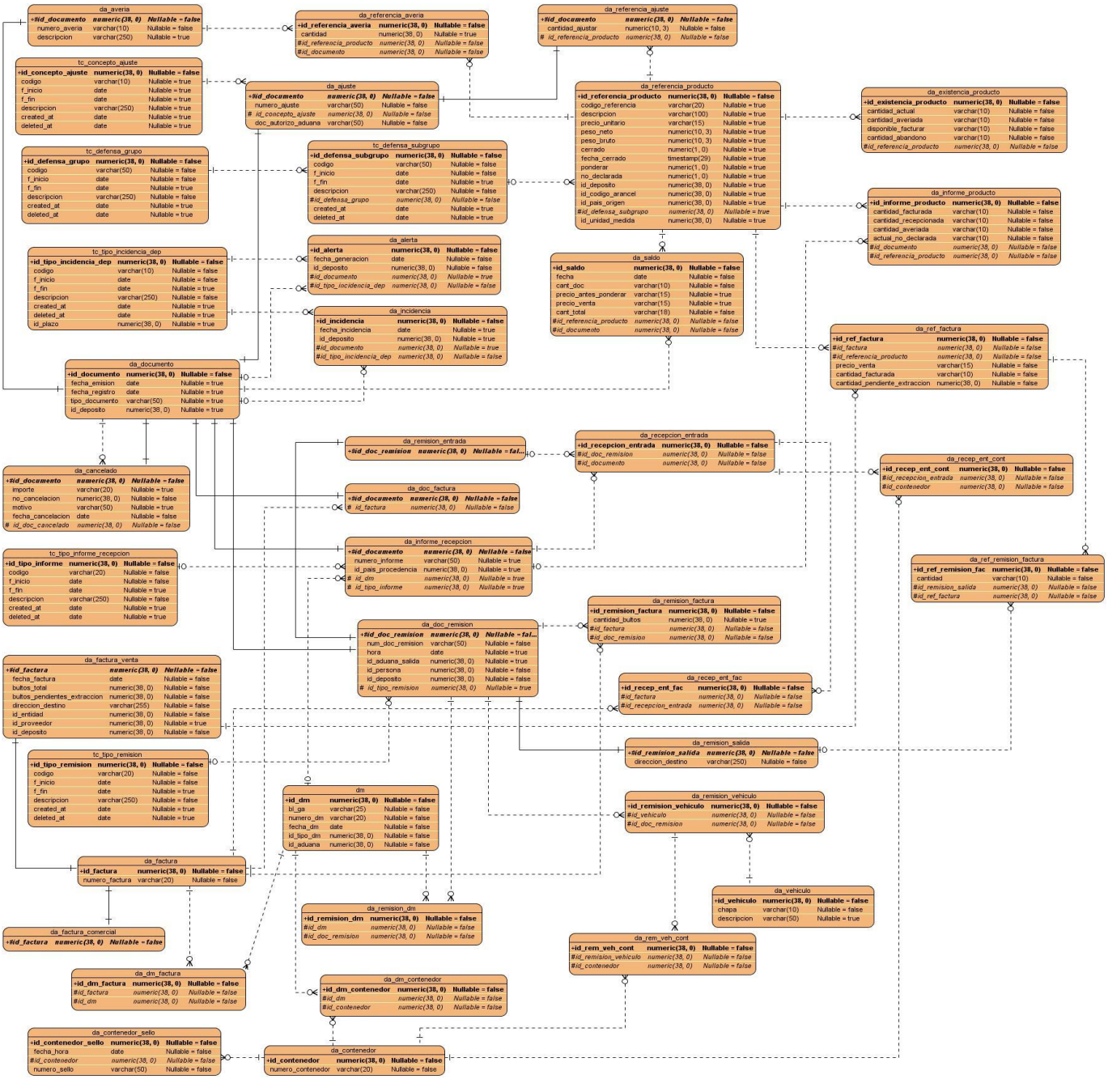

Fig. 9 Modelo de Datos

2.11 Conclusiones

En el capítulo se describieron los aspectos desarrollados durante el análisis y el diseño de la solución planteada, como son el modelado de los procesos de negocio asociados a los módulos Alertas y Prórrogas correspondientes al subsistema Depósito de Aduana, el modelo conceptual, una breve descripción de los requisitos funcionales así como los diagramas de clases del diseño con estereotipos web, de interacción y el modelo de datos correspondiente.

Implementación y Prueba

3.1 Introducción

En este capítulo se describen algunos elementos utilizados en la implementación de la aplicación, además se muestra la interacción de los componentes mediante el diagrama de componentes además del diagrama de despliegue del proyecto GINA al cual está integrado el subsistema Depósito de Aduana. Unido a lo anterior se realiza la validación de la solución mediante la aplicación de las pruebas de Caja Negra de las que se presenta un diseño de casos de prueba, las pruebas de Caja Blanca y pruebas unitarias así como los resultados obtenidos en las mismas.

3.2 Implementación

El flujo de trabajo de implementación es de gran importancia porque en el mismo se obtiene como resultado final un sistema ejecutable, siendo uno de los principales objetivos en el desarrollo de software, el mismo describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. En este flujo de trabajo son implementadas las clases y objetos en ficheros fuente, binarios, ejecutables y demás.

3.3 Estándares de codificación utilizados

Un estándar de codificación comprende todos los aspectos en la generación de código. La mejor forma para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código.

Los estándares de codificación son definidos por el equipo de desarrollo con el objetivo de lograr generalidad en la programación del sistema. Estos facilitan la lectura, comprensión, mantenimiento del código y la reutilización a lo largo del proceso de desarrollo del software.

La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta el final del proyecto de software. En el caso del Departamento de Soluciones para la Aduana, el estándar de codificación fue definido por la dirección del proyecto en sus inicios y se encuentra registrado en el documento “Propuesta de un estándar de codificación”.

3.4 Elementos utilizados para el desarrollo de la solución

3.4.1 Tareas de Symfony

Symfony permite generar tareas propias las cuales pueden aprovechar el procesamiento de argumentos y opciones de la línea de comandos, además de mostrar mensajes de ayuda sobre su uso y pueden ampliar las Tareas existentes.

Una tarea propia es una clase que hereda de sfBaseTask y cuyo código se encuentra en el directorio lib/task/ de la raíz del proyecto o en el directorio lib/task de un plugin. Estas tienen como restricción que su nombre debe terminar en Task.class.php.

Para el desarrollo de la aplicación se crearon varias tareas que son las responsables de generar alertas correspondientes a las infracciones cometidas en los Depósitos de Aduana, las cuales se describirán a continuación:

DepositoSinOperarTask.class: esta tarea es encargada de generar una alerta sobre los depósitos que hasta el día en curso su última operación supere el tiempo límite de un depósito sin operar.

InformeRecepcionSinDmPresentadaTask.class: esta tarea es encargada de generar una alerta sobre cada Informe de Recepción que no está cancelado y no esté asociado a la DM de régimen 7100.

MercanciaProximaAlAbandonoTask.class: esta tarea es encargada de generar una alerta sobre las Facturas de Venta que no están canceladas, posean bultos que no hayan sido extraídos, estén asociadas a la Declaración de Mercancías y se haya vencido el plazo establecido para que las mercancías estén próximas al abandono.

MercanciaEnAbandonoTask.class: esta tarea es encargada de generar una alerta sobre las Facturas de Venta que estaban próximas al abandono y haya vencido el plazo establecido para que las mercancías estén en abandono.

RemisionEntradaSinInformeRecepcionTask.class: esta tarea es encargada de generar una alerta sobre las Remisiones de Entrada que no tengan asociadas el Informe de Recepción y hayan superado la sumatoria de los tiempos límites para la realización y entrega del Informe de recepción.

3.4.2 Variables de Symfony

Las variables de Symfony son utilizadas para almacenar datos que se utilizan en la lógica de negocio de la aplicación y no están almacenados en una base de datos, evitando la unión de código fuente con datos referentes al negocio. Estas son acumuladas en archivo app.yml que contiene la configuración específica de la aplicación.

3.5 Diagrama de Componente

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones, además muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Estos representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas y pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente y otros.

Elementos del Diagrama de Componentes:

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

El siguiente diagrama muestra los componentes de los módulos Alertas y Prórrogas, y las relaciones entre estos, evidenciando la clase actions.php, cada una de las interfaces involucradas en estos módulos, las clases del negocio y su interacción con otros subsistemas, así como los elementos de configuración.

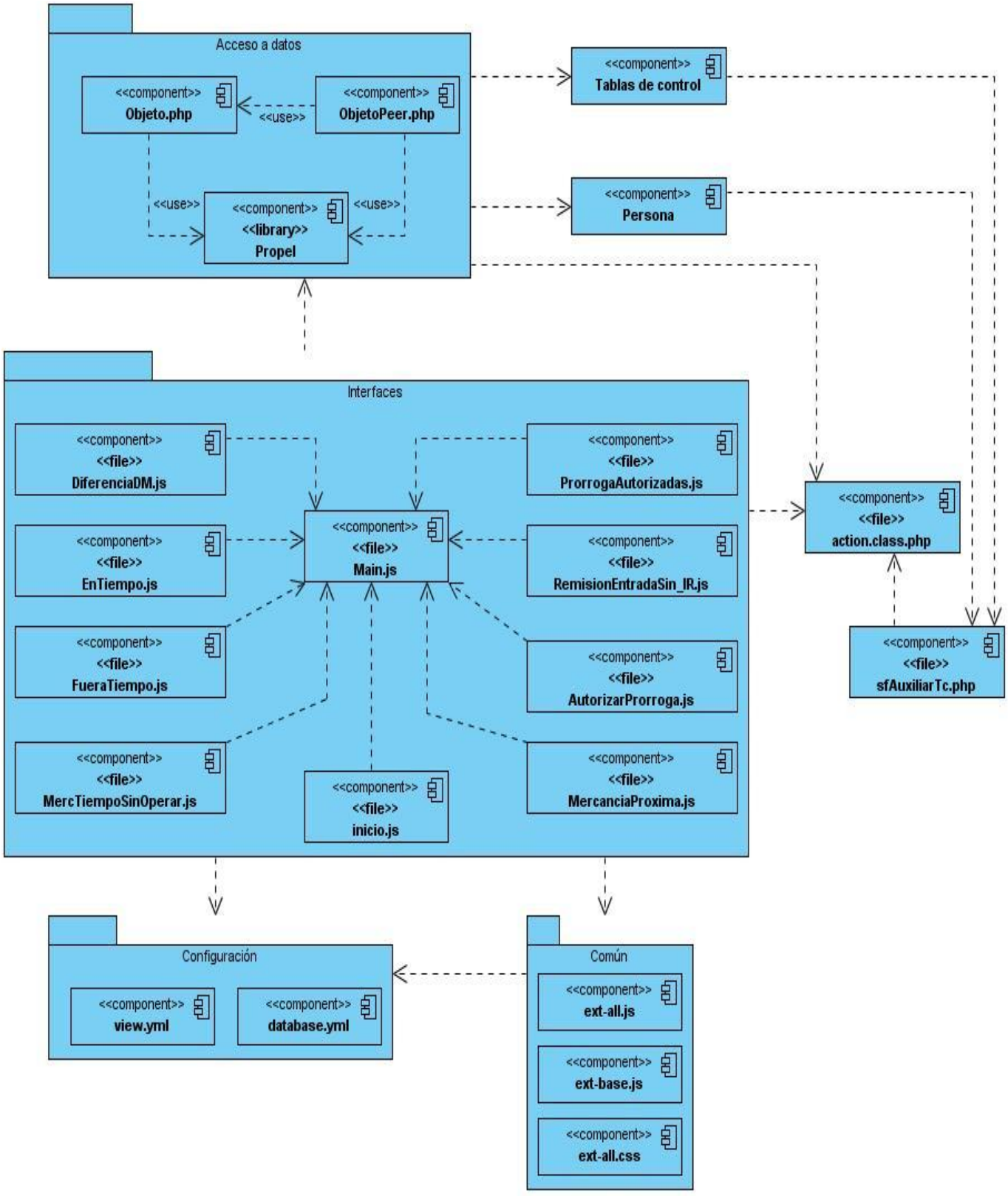

Fig. 10 Diagrama de componentes de los módulos Alertas y Prórrogas.

3.6 Diagrama de despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. Los estereotipos permiten precisar la naturaleza del equipo: dispositivos, procesadores y memoria.(18) A continuación se presenta el Diagrama de despliegue con los nodos definidos en el documento de Arquitectura para el despliegue del proyecto GINA.

Fig. 11 Diagrama de despliegue del proyecto Gina.

3.7 Validación de la solución

Las pruebas de software, son los procesos que permiten verificar y revelar la calidad de un producto software. Estas son utilizadas para identificar posibles imperfecciones e irregularidades en la implementación de un sistema. Las pruebas se integran dentro de las diferentes fases del ciclo de vida de un proyecto para detectar errores que en algunos casos pueden ser irreversibles y costosos.

3.7.1 Tipo de prueba a realizar

A la solución se le realizó diversas pruebas para comprobar su factibilidad, chequeando el software en busca de errores en el manejo de las restricciones de integridad de las clases del sistema. El objetivo

fundamental de las mismas es verificar las discrepancias entre los requerimientos y los resultados de la ejecución del software. Estas pruebas también determinan el nivel de calidad y comprueban el grado de cumplimiento respecto de las especificaciones del sistema inicialmente. Para realizar las pruebas al sistema se eligieron diferentes tipos de prueba debido a la variedad de pruebas existentes. Como son las pruebas de Caja Negra y Caja Blanca.

Las pruebas de Caja Negra se centran en lo que se espera de un módulo, ya que estas intentan encontrar casos en que el software no se corresponda a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Las pruebas de Caja Blanca se realizan sobre las funciones internas de un módulo. Entre las técnicas usadas se encuentran la cobertura de caminos, las cuales hacen que se recorran todos los posibles caminos de ejecución; pruebas sobre las expresiones lógico-aritméticas; pruebas de camino de datos (definición-uso de variables); comprobación de bucles, en las que se verifican los bucles para 0,1 y n iteraciones, y luego para las iteraciones máximas, máximas menos uno y más uno.

3.7.2 Aplicación de la Prueba de Caja Blanca o Estructural

Para realizar las pruebas de caja blanca se aplicó la técnica de cobertura de caminos debido a que esta permite obtener una medida de la complejidad lógica del diseño y usar la misma como guía para la definición de un conjunto de caminos básicos, garantizando que durante la prueba se ejecute al menos una vez cada sentencia del programa.

Para ello es necesario conocer el número de caminos independientes de un determinado algoritmo mediante el cálculo de la complejidad ciclomática. Se debe comenzar por un análisis del código, posteriormente son enumeradas cada una de las instrucciones, se construye el grafo de flujo asociado y según las fórmulas pertinentes se calcula dicha complejidad.

A continuación se analizan y enumeran las sentencias de código de uno de los métodos contenidos en la clase Actions.class.php, específicamente: executeInsertarprorroga, la que se manda a invocar desde la vista permitiendo insertar las prórrogas que son aceptadas por parte de la Aduana a los depositarios.

```
public function executeInsertarProrroga(sfWebRequest $request) {
    $fecha = $this->getRequestParameter('fecha'); //1
    $fecha = str_replace("/", "-", $fecha); //1
    $tiempoPro = $this->getRequestParameter('tiempoPro'); //1
    $tipoPro = $this->getRequestParameter('tipoProrroga'); //1
    $texto = $this->getRequestParameter('id'); //1
    $noRemision = $this->getRequestParameter('remisE'); //1
    $idTipoProrroga = sfAuxiliarTC::getIdTipoProrroga($tipoPro); //1
    $Remision = DaDocRemisionPeer::BuscarRemisionDadoNumero($noRemision); //1
    if (!empty($Remision)) {//2
        $idRemision = $Remision->getIdDocRemision(); //3
        $idDeposito = $Remision->getIdDeposito(); //3
        if ($idDeposito == $texto) {//3
            $mensaje = DaProrrogaPeer::AutorizarProrroga($tiempoPro, $fecha, $idTipoProrroga, $idRemision); //4
            return $this->renderText($mensaje); //4
        } else {
            return $this->renderText("{'success':false, 'msg':
                'La Remisión de Entrada no está asociada al depósito especificado'}"); //5
        }
    } else {
        return $this->renderText("{'success':false, 'msg':
            'La Remisión de Entrada  especificada no existe'}"); //6
    }
}
```

Fig. 12 Funcionalidad executeInsertarprorroga

Luego de la enumeración de las sentencias anteriores es necesario presentar el grafo de flujo asociado:

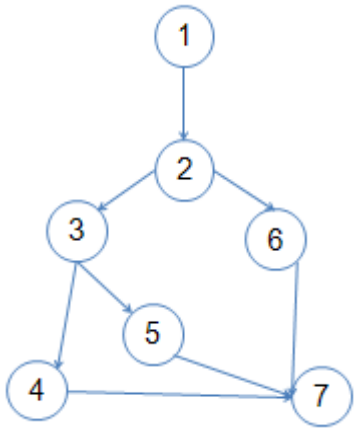

Fig. 13 Grafo de flujo asociado a la funcionalidad executeInsertarProrroga

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, este cálculo debe efectuarse de tres formas diferentes de forma tal que quede bien justificado el resultado, siendo el mismo en cada caso:

1. V (G) = (A – N) + 2

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

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

V (G) = 3.

2. V (G) = P + 1

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 2 + 1

V (G) = 3.

3. V (G) = R

Siendo “R” la cantidad total de regiones, se incluye el área exterior del grafo, contando como una región más.

V (G) = 3.

El cálculo efectuado muestra la complejidad ciclomática de valor 3, de manera que existen tres posibles caminos a través de los cuales el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento antes tratado.

Posteriormente es necesario especificar los caminos básicos por los que puede circular el algoritmo durante su ejecución.

Camino básico #1: 1 – 2 – 3 – 4 –7

Camino básico #2: 1 – 2 – 3 –5 – 7

Camino básico #3: 1 – 2 – 6 –7

Seguidamente se ejecutan los casos de pruebas[10] para cada uno de los caminos básicos determinados en el grafo de flujo. Para cada uno de los casos de prueba se debe tener en cuenta:

Descripción: Se describe el caso de prueba y se tratan algunos aspectos fundamentales de los datos de entrada.

Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada y de esa forma observar el funcionamiento del procedimiento.

Entrada: Se muestran los parámetros de entrada al procedimiento.

Resultados Esperados: Se expone el resultado que debe devolver el procedimiento después de ser realizado el caso de prueba.

Caso de prueba para el camino básico # 1.

Descripción: se debe insertar la prórroga correspondiente al Depósito y la Remisión de Entrada especificada.

Condición de ejecución: para el algoritmo es necesario especificar la fecha de la prórroga, el tiempo, el tipo de prórroga, el depósito y el número de remisión.

Entrada:

Fecha: 24/05/2011

Tiempo: 12

Tipo: Prórroga de realización.

Depósito: BCD

Remisión: 0018

Resultados esperados: Debe mostrar un mensaje indicando que la operación finalizó con éxito.

[10] *Casos de prueba: especifican una forma de probar el componente, incluye: la entrada, las condiciones bajo las cuales ha de probarse y los resultados esperados.*

El resultado obtenido fue correcto.

Caso de prueba para el camino básico # 2.

Entrada:

Fecha: 24/05/2011

Tiempo: 12

Tipo: Prórroga de realización.

Depósito: Cimex

Remisión: 0018

Resultados esperados: Debe mostrar un mensaje indicando que la remisión no pertenece al depósito especificado.

El resultado obtenido fue correcto.

Caso de prueba para el camino básico # 3.

Entrada:

Fecha: 24/05/2011

Tiempo: 12

Tipo: Prórroga de realización.

Depósito: BCD

Remisión: 001

Resultados esperados: Debe mostrar un mensaje indicando que la remisión especificada no existe.

El resultado obtenido fue correcto.

3.7.3 Aplicación de la Prueba de Caja Negra o Funcional.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
Autorizar prórroga.	EC 1.1: Autorizar con los campos vacíos.	El sistema notifica que es obligatorio llenar los campos con datos válidos.	- Se presiona el botón Aceptar. - El sistema muestra un mensaje de información.
	EC 1.2: Autorizar especificando el Tiempo de la prórroga.	El sistema notifica que es obligatorio llenar los campos con datos válidos.	- Introducir el tiempo de prórroga. - Se presiona el botón Aceptar. - El sistema muestra un mensaje de información.
	EC 1.3: Autorizar especificando el Tipo de prórroga.	El sistema notifica que es obligatorio llenar los campos con datos válidos.	- Introducir el tipo de prórroga - Se presiona el botón Aceptar. - El sistema muestra un mensaje de información.
	EC 1.4: Autorizar especificando el Depósito.	El sistema notifica que es obligatorio llenar los campos con datos válidos.	- Introducir el depósito de prórroga - Se presiona el botón Aceptar. - El sistema muestra un mensaje de información.
	EC 1.5: Autorizar especificando el Número de la Remisión.	El sistema notifica que es obligatorio llenar los campos con datos válidos.	- Introducir el número de la remisión. - Se presiona el botón Aceptar. - El sistema muestra un mensaje de información.
	EC1.6: Autorizar especificando la Fecha.	El sistema notifica que es obligatorio llenar los campos con datos válidos.	- Introducir la fecha. - Se presiona el botón Aceptar. - El sistema muestra un mensaje de

			información.
	EC 1.7: Autorizar especificando todos los campos.	El sistema permite autorizar la prórroga según los criterios.	- Introducir criterios. - Se presiona el botón Aceptar. - El sistema autoriza la prórroga.

Tabla 3.1 Caso de prueba Autorizar Prórroga.

ID del esce-nario	Escenario	Depósi-to	Fecha	Tipo	Tiempo	Número de Remisión	Respuesta del sistema	Resultado de la prueba
EC 1.1	Autorizar con los campos vacíos.	V (vacio)	V (vacio)	V (vacio)	V (vacio)	V (vacio)	El sistema notificar que los campos deben ser llenados con datos válidos.	Se muestra un mensaje de información.
EC 1.2	Autorizar especificando el Tiempo.	V (vacio)	V (vacio)	V (vacio)	V (2 días)	V (vacio)	El sistema debe notificar que este campo debe ser llenado obligatoriamente.	Se muestra un mensaje de información.
EC 1.3	Autorizar especificando el Tipo de prórroga.	V (vacio)	V (vacio)	V (Prórroga entrega)	V (vacio)	V (vacio)	El sistema debe notificar que este campo debe ser llenado obligatoriamente.	Se muestra un mensaje de información.
EC 1.4	Autorizar especificando el Depósito.	V (BDC)	V (vacio)	V (vacio)	V (vacio)	V (vacio)	El sistema debe notificar que este campo debe ser llenado obligatoriamente.	Se muestra un mensaje de información.
EC 1.5	Autori-zar especificando el Número de la Remisión.	V (vacio)	V (vacio)	V (vacio)	V (vacio)	V (0018)	El sistema debe notificar que este campo debe ser llenado obligatoriamente.	Se muestra un mensaje de información.

EC 1.6	Autorizar especificando la Fecha.	V (vacio)	V (20/05/2011)	V (vacio)	V (vacio)	V (vacio)	El sistema debe notificar que los campos deben ser llenados con datos válidos.	Se muestra un mensaje de información.
EC 1.7:	Autorizar especificando todos los campos.	V (BDC)	V (20/05/2011)	V (Prórroga entrega)	V (2 días)	V (0018)	El sistema debe insertar o no la prórroga solicitada.	Se inserta o no la prórroga.

Tabla 3.2 Datos a probar.

3.7.3.1 Resultados de la pruebas

Elemento	No.	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Resp. equipo de desarrollo
Datos insertados	**1**	*Datos insertados*	*Validar que si la fecha debe ser especificada.*	*Al insertar la prórroga.*	X			PD 20/05/2011	
Datos insertados	*2*	*Datos insertados*	*Validar que el número de días sea positivo.*	*Al insertar la prórroga.*	X			RA 25/05/2011 PD 06/05/2011	*Corrección de la no conformidad registrada el* 20/05/2011
Datos insertados	*3*	*Datos insertados*						RA 66/05/2011	*Corrección de la no conformidad registrada el* 06/05/2011

Tabla 3.3 Registro de defectos y dificultades detectadas

3.8 Conclusiones

En el capítulo que recién concluye se presentaron los elementos de la implementación y la calidad de la solución propuesta. Para medir la calidad de la solución se realizaron pruebas de software internas mediante casos de pruebas, para los cuales se tuvieron en cuenta las entradas, las salidas y los resultados esperados así como el funcionamiento interno del sistema.

Con la culminación de las pruebas se determinó que los módulos Alertas y Prórrogas del subsistema Depósito de Aduana que desde el punto de vista funcional cumplen con los requisitos definidos a partir de las necesidades del cliente.

Conclusiones

Una vez terminado el presente trabajo de diploma se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, para ello se analizaron ventajas y deficiencias de sistemas informáticos internacionales vinculados a la gestión de procesos en los Depósitos de Aduana, donde se detectaron varias dificultades para su utilización como son: el alto costo de sus licencias y soporte, sus arquitecturas no pueden ser integradas al proyecto GINA que se está desarrollando y muchas de ellas no cumplen con la legislación aduanera vigente. Evidenciándose de esta forma la necesidad de desarrollar una solución informática capaz de gestionar los procesos aduaneros asociados a las prórrogas y las infracciones cometidas a la normativa aduanera, que cumpla con los requisitos establecidos por la Aduana General de la República de Cuba.

También se realizó la modelación e implementación de los módulos Alertas y Prórrogas obteniendo el modelado de los procesos asociados al negocio, la especificación de los requisitos funcionales, los diagramas de clases del diseño con estereotipos web, interacción, componente y despliegue, así como el código fuente de los módulos; utilizando buenas prácticas como los patrones de diseño, con el objetivo de incorporarlos al sistema GINA. Unido a lo antes planteado se evaluó la viabilidad de dichos módulos a través de pruebas de software efectuadas, las cuales arrojaron resultados favorables posibilitando dar cumplimiento a los objetivos propuestos.

Recomendaciones

Se recomienda dar seguimiento al estudio de los procesos asociados a las prórrogas y las infracciones en los depósitos de aduana y al desarrollo del subsistema con el propósito de detectar nuevas necesidades e incorporarle funcionalidades a la solución y la optimización de las ya existentes con el objetivo de lograr un sistema más estable.

Glosario de Términos

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

AGR: Aduana General de la República.

GINA: Gestión Integral Aduanera.

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

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

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

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

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

Referencias bibliográficas

1. **Uzcátegui, Marco Antonio Osorio.** El Régimen de Almacenes In Bond y la sanción establecida en el artículo 115 de nuestra Ley Orgánica de Aduanas. [En línea] [Citado el: 11 de diciembre de 2010.] http://www.aduanas.com.ve/boletines/boletin_5/inbond.htm.

2. *Comercio Exterior.* [En línea] [Citado el: 15 de enero de 2011.] HTTP://WWW.EMBACUBAQATAR.COM/Q000031S.HTML.

3. Usinad. [En línea] [Citado el: 25 de enero de 2011.] http://www.usinad.es.

4. S4 tr@nsERP. *Depósitos Aduaneros.* [En línea] [Citado el: 13 de diciembre de 2010.] http://www.s-4.es/ms_Depositos.aspx.

5. **Arellano, Ana Alejandra Febres.** Conociendo al SIDUNEA - Sistema Aduanero Automatizado. *COMERCIO INTERNACIONAL.* [En línea] [Citado el: 13 de diciembre de 2010.] http://www.gestiopolis.com/canales2/economia/sidunea.htm.

6. Cotecna. [En línea] [Citado el: 16 de enero de 2011.] http://www.cotecna.com/COM/ES/bonded_warehouse_management.aspx.

7. ¿Qué es Ingeniería de Requisitos (IR)? [En línea] 27 de marzo de 2008. [Citado el: 29 de enero de 2011.] http://danielvn7.wordpress.com/2008/03/27/%C2%BFque-es-ingenieria-de-requisitos-ir/.

8. **Koch., N. y Escalona, M.J.** *Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo.* España : s.n.

9. *Estilos y Patrones en la Estrategia de.* **Reynoso, Carlos y Kiccillof, Nicolás.** BUENOS AIRES : s.n., 2004.

10. **Craig Larman.** *Introducción al análisis y diseño orientado a objeto.* México : UML y Patrones, 1999.

11. Scribd. [En línea] [Citado el: 12 de diciembre de 2010.] http://www.scribd.com/doc/31440864/Metodologia-RUP.

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

13. **Barriento, Manuel Sánchez.** *Modelado de procesos de negocio.* [En línea] 2008. [Citado el: 16 de enero de 2011.] http://www.aprendergratis.com/introduccion-a-bpmn.html ..

14. **José Enrique González Cornejo.** *El lenguaje de modelado unificado.* [En línea] [Citado el: 15 de enero de 2011.] http://www.docirs.cl/uml.htm.

15. Symfony.es. [En línea] [Citado el: 16 de enero de 2011.] http://www.symfony.es/.

16. **Roberto Hernando Velasco.** El SGBDR Oracle . [En línea] [Citado el: 20 de enero de 2011.] http://www.rhernando.net/modules/tutorials/doc/bd/oracle.html.

17. **Potencier, Fabien.** *Symfony 1.2, la guía definitiva.* 2008.

18. SCRIBD. *SCRIBD.* [En línea] [Citado el: 10 de 05 de 2011.] http://www.scribd.com/doc/19808824/diagramas-de-despliegue-2222.

Bibliografía

1. **Uzcátegui, Marco Antonio Osorio.** El Régimen de Almacenes In Bond y la sanción establecida en el artículo 115 de nuestra Ley Orgánica de Aduanas. [En línea] [Citado el: 11 de diciembre de 2010.] http://www.aduanas.com.ve/boletines/boletin_5/inbond.htm.

2. *Comercio Exterior.* [En línea] [Citado el: 15 de enero de 2011.] HTTP://WWW.EMBACUBAQATAR.COM/Q000031S.HTML.

3. Usinad. [En línea] [Citado el: 25 de enero de 2011.] http://www.usinad.es.

4. S4 tr@nsERP. *Depósitos Aduaneros.* [En línea] [Citado el: 13 de diciembre de 2010.] http://www.s-4.es/ms_Depositos.aspx.

5. **Arellano, Ana Alejandra Febres.** Conociendo al SIDUNEA - Sistema Aduanero Automatizado. *COMERCIO INTERNACIONAL.* [En línea] [Citado el: 13 de diciembre de 2010.] http://www.gestiopolis.com/canales2/economia/sidunea.htm.

6. **Bolivia, Aduana Nacional de.** *Manual de Usuario SIDUNEA (Sistema Aduanero Automatizado).* Bolivia : s.n.

7. Cotecna. [En línea] [Citado el: 16 de enero de 2011.] http://www.cotecna.com/COM/ES/bonded_warehouse_management.aspx.

8. ¿Qué es Ingeniería de Requisitos (IR)? [En línea] 27 de marzo de 2008. [Citado el: 29 de enero de 2011.] http://danielvn7.wordpress.com/2008/03/27/%C2%BFque-es-ingenieria-de-requisitos-ir/.

9. **Koch., N. y Escalona, M.J.** *Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo.* España : s.n.

10. *Estilos y Patrones en la Estrategia de.* **Reynoso, Carlos y Kiccillof, Nicolás.** BUENOS AIRES : s.n., 2004.

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

12. Scribd. [En línea] [Citado el: 12 de diciembre de 2010.] http://www.scribd.com/doc/31440864/Metodologia-RUP.

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

13. **Barriento, Manuel Sánchez.** *Modelado de procesos de negocio.* [En línea] 2008. [Citado el: 16 de enero de 2011.] http://www.aprendergratis.com/introduccion-a-bpmn.html ..

14. **José Enrique González Cornejo.** *El lenguaje de modelado unificado.* [En línea] [Citado el: 15 de enero de 2011.] http://www.docirs.cl/uml.htm.

15. Visual Paradigm for UML (ME). *Free Download Manager.* [En línea] [Citado el: 14 de enero de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/..

16. LinuxCentro.net. [En línea] [Citado el: 18 de enero de 2011.] http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.

17. Symfony.es. [En línea] [Citado el: 16 de enero de 2011.] http://www.symfony.es/.

18. Ext JS y frameworks JavaScript. [En línea] [Citado el: 21 de enero de 2011.] ttp://www.desarrolloweb.com/wiki/ext-js.html.

19. **Roberto Hernando Velasco.** El SGBDR Oracle . [En línea] [Citado el: 20 de enero de 2011.] http://www.rhernando.net/modules/tutorials/doc/bd/oracle.html.

20. **Orallo, Enrique Hernández.** El Lenguaje Unificado de Modelado (UML). [En línea] [Citado el: 16 de enero de 2011.] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

21. *Estilos y Patrones en la Estrategia de.* **Reynoso, Carlos y Kiccillof, Nicolás.** BUENOS AIRES : s.n., 2004.

22. INEI. *INEI.* [En línea] [Citado el: 09 de 05 de 2011.] http://www.inei.gob.pe/biblioineipub/bancopub/inf/lib5011/cap2-3.htm.

23. **Potencier, Fabien.** *Symfony 1.2, la guía definitiva.* 2008.

24. Definicion.de. *Definicion.de.* [En línea] [Citado el: 09 de 05 de 2011.] http://definicion.de/modelo-de-datos/.

25. ZANINOTTO, F. y POTENCIER, F. 2009. librosweb.es. Symfony 1.1, la guía definitiva. [En línea] 2009. http://www.librosweb.es/symfony_1_1.

26. SCRIBD. *SCRIBD.* [En línea] [Citado el: 10 de 05 de 2011.] http://www.scribd.com/doc/19808824/diagramas-de-despliegue-2222.

27. **Marca Huallpara Hugo Michael, Quisbert Limachi Nancy Susana.** [En línea] [Citado el: 10 de 05 de 2011.] virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc.

28. **A. Mañas, José.** Prueba de Programas. [Online] 03 16, 1994. http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s21.

29. Ecuared. *Ecuared.* [En línea] [Citado el: 10 de 05 de 2011.] http://www.ecured.cu/index.php/Flujo_de_Trabajo_de_Implementaci%C3%B3n.

30. Entorno virtual de aprendizaje. *Entorno virtual de aprendizaje.* [En línea] [Citado el: 10 de 05 de 2011.] http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Basicos.