Universidad de las Ciencias Informáticas

Facultad 3

Título: Componente para Gestionar Documentos Complementarios en los Procesos aduaneros del sistema GINA

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

Autores: Yakelin Milagro León López

Tutores: Ing. Julio César Bravo Rodríguez

La Habana, 27 de Junio de 2010
"Año 53 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ _____

Yakelin Milagro León López Ing. Julio César Bravo Rodríguez

AGRADECIMIENTOS

Primeramente agradecer a mi Padre, mi Rey y Señor, a mi Dios, por haber sido mi sustento en todo este tiempo y quien siempre me ayudo a enfrentar cada amanecer, por ser el autor de mi vida, por darme la inspiración y las fuerzas y por alentar mi alma en las etapas más desoladas, por hacerme fuerte en mi debilidad y enriquecerme aún en mi pobreza, a Él, mil gracias.

A mi tutor por su ayuda en todo tiempo, por sus consejos e insistencia, muchas gracias por ayudarme a formar como ingeniera.

A cada uno de mis compañeros de aula y más que compañeros mis amigos, aún para aquellos que abandonaron la barca, pero que igual en el tiempo que estuvieron dejaron su huella y no pasaron desapercibidos, gracias a ustedes amigos míos por haberme enseñado la belleza de tener un grupo del cual me siento orgullosa y dichosa, por las horas de estudio y la compañía, por los apuros de los que me sacaron y los aprietos en los que me metieron, por todo, muchísimas gracias.

A mis hermanitas Lixanis, Tamara, Misleydis, Daylin e incluso a la sin tierra de Ingris, a la madre de todas la excelentísima Negra (Rowe), a ustedes gracias por las rizas y los buenos y malos momentos que pasaron conmigo y en especial a La Negra gracias por estar desde el principio y llegar hasta el final, me enseñaste que con metas bien definidas y mucho trabajo, siempre se puede llegar, aunque incluso la gente y uno mismo llegue a dudar.

A los varones especiales de mi grupo, a Dayan por tanto estorbo y cada una de las cosas agradables que siempre compartíamos en nuestra forma de tratarnos, a Yordani por siempre ser una ayuda en cualquier necesidad, a Boris por sus locuras y carisma, a Yasel por sus ocurrencias repentinas, a Malagón por su hiperactividad, a Cervela por enseñarme la belleza de las flores existentes en esta escuela y a Eyeris por ser mi hermano y amigo siempre.

Muchísimas gracias a mis hermanos en la fe, de todos aprendí algo, todos aportaron a mi vida grandes cosas, de ustedes me llevo las mejores experiencias jamás compartidas. Por toda su ayuda y sustento, por las conversaciones y regaños, por aguantarme a pesar de mi carácter chocante, por estar siempre para mí, gracias.

A mis amigas de Bayamo Anita y Roxemary, por sus llamadas y consejos, por sus insistencias y cada una de sus oraciones, gracias chicas por todo.

A mis regalos más preciados, mis amigos. A esas personitas que dejaron siempre algo y que dieron más de lo que se llevaron, a Mayi la sobreprotectora, Yanet la super F, Lisandra y Yuniel, Adri la chica genio y a Katy la más pequeñita de la pandilla.

A mi amiga del alma, Marita, gracias por estar siempre, por quedarte donde muchos abandonaron y por ser el David que todo Jonathan desea.

A mi Mana por amarme tanto, por ser la hermanita pequeña que se me fue de las manos pero que aún así me siguió admirando y alentando, por cuidar a mamá en mi ausencia, por siempre ser mi niña bella, te amo mi cielo y me siento orgullosa de ti y de todo lo que has logrado, gracias por ser parte activa de mi vida.

A Vito, mi mami, y más que mi madre, mi ejemplo, gracias por traerme a este mundo, por ayudarme en cada momento, por ser como eres, te admiro, te amo y cada uno de mis logros se los debo a tus regaños y consejos, a tu insistencia y paciencia, a tu amor incondicional, gracias doy a Dios por ti cada día porque fuiste siempre la fuente de mis energías, te amo mucho mama, gracias por la vida.

A mi tesoro, mi novio y futuro esposo, a Ronny, por su amor y más que nada paciencia, por aprender junto conmigo, por todas las cosas que hemos pasado juntos, porque con él la vida ha sido más llevadera, gracias por hacer de cada día en esta universidad un destello de detalles y amor. Te amo.

DEDICATORIA

Dedico este trabajo primeramente a mi Dios por ser el creador de mi vida.

A mi madre por ayudarme a llegar hasta aquí, tú has sido el principal motivo por el cual hoy puedo llegar a ser profesional y mejor persona, aunque sé que nunca podré pagarte todo cuanto me has dado. Te amo.

RESUMEN

La Aduana General de la República (AGR) tiene especial interés en la implantación del sistema GINA para poder agilizar cada uno de los procesos que se realizan en esta institución que actualmente son ejecutados manualmente. Durante la realización de estos procesos aduaneros son generados una serie de documentos complementarios que actualmente no son gestionados dentro del sistema.

Dado que el principal interés de la Aduana con la informatización de sus procesos es lograr controles aduaneros más eficaces con el objetivo de evitar fraudes y el contrabando, se procedió a realizar la captura de los requerimientos. Los resultados obtenidos fueron validados por el cliente manifestando su total satisfacción debido a lo modelado por parte del analista implicado en dicho proceso.

Partiendo de lo anteriormente planteado, en el presente trabajo se realiza el diseño y la implementación de un componente que sea capaz de automatizar la recepción y control de los documentos complementarios generados durante la realización de los diferentes procesos aduaneros en el sistema GINA, para garantizar la eficiencia y eficacia respondiendo en su totalidad a las demandas de sus clientes.

PALABRAS CLAVE

Aduana General de la República, GINA, documentos complementarios, componente

TABLA DE CONTENIDOS

INTRODUCCIÓN

En la actualidad el mundo está minado de desarrollo y competitividad, cada día son mayores los retos a los que se tienen que enfrentar las empresas para lograr subsistir en medio de los estándares y requisitos que impone la sociedad. La crisis mundial que se enfrenta va labrando un camino escabroso y difícil de superar por lo cual la gestión empresarial es una tarea primordial para poder sobrevivir en el territorio hostil que se impone.

Lograr un buen control y dominio sobre toda la documentación generada en una empresa, principalmente si es clasificada como confidencial, es uno de los grandes desafíos que se enfrenta en la actualidad pues de esta actividad dependerá, en parte, la confidencialidad e integridad de la empresa a nivel mundial.

La utilización de sistemas de gestión documental en cualquier institución acarrea grandes beneficios pues reduce los costos de los procesos empresariales, mediante el rediseño de procesos, permite la sustitución del trabajo administrativo no productivo y la reducción del espacio físico de almacenamiento. También logran la reducción de los ciclos de trabajo, aumentando la concurrencia de las distintas actividades necesarias y la unificación de los procesos empresariales en los distintos ámbitos departamentales y geográficos potenciando los canales formales y los procedimientos de trabajo lo que facilitará el cumplimiento de los requerimientos que los sistemas de calidad imponen. Además logra un aumento en las capacidades de comunicación de toda la organización, mejorando la integridad y seguridad de la información.

El sistema GINA (Gestión Integral de Aduanas) tiene como principal objetivo lograr informatizar todos los procesos aduaneros que se desarrollan en la Aduana General de la República (AGR) y por lo tanto facilitar el control por parte de la aduana de cada uno de los procesos desarrollados.

En la actualidad dentro del sistema GINA no existe un componente que se encargue del proceso de gestión de los documentos complementarios que son generados en los procesos aduaneros en la AGR situación que puede acarrear pérdida de información, además de no tener archivada de forma segura todos los documentos que son generados y de no garantizar un control del acceso y modificación de estos documentos, situación que provoca que se incurra en la seguridad de los datos generados dentro de estos documentos.

Introducción

Ante todas las ventajas que nos ofrecen los Sistema de Gestión Documental y atendiendo a la necesidad que presenta el sistema GINA de lograr tener un componente que permita gestionar cada uno de los documentos que son generados en los diferentes procesos aduaneros que se realizan dentro de este sistema y atendiendo a la necesidad de crear un sistema lo más completo posible y que responda a las necesidades e intereses de nuestro país, se ha decidido realizar un componente que permita gestionar los documentos complementarios generados en cada uno de los procesos aduaneros.

A causa de esto se identifica la siguiente **situación problemática**: En el sistema GINA perteneciente a la Aduana general de la República (AGR) en la realización de los diferentes procesos aduaneros se requiere una gran cantidad de documentos complementarios. Estos documentos complementarios actualmente no son gestionados lo que incurre en un gran problema de seguridad de la información y datos que estos generan. Estos documentos complementarios son manejados en la aduana en texto duro situación que permite una posible pérdida de los documentos ya sea por robo, pérdida o algún desastre que afecte el local donde son almacenados y además hace muy engorroso el proceso de consulta de los mismos para hacerlos coincidir con los documentos principales que lo requieren.

Al analizar cada una de las dificultades existentes, se plantea el **problema a resolver**: ¿Cómo favorecer la gestión de los documentos complementarios en la Aduana?

Teniendo en cuanta el problema planteado el **objeto de estudio** se enfocará en realizar un análisis de losSistemas informáticos de Gestión Aduanera.

El **Campo de acción** estará enmarcado al**:**Desarrollo de un componente para Gestionar Documentos complementarios dentro del sistema GINA.

Como **objetivo general** de este trabajo se tiene el siguiente: Realizar el diseño e implementación del Componente para Gestionar Documentos Complementarios en los procesos aduaneros del sistema GINA.

Descrito todo lo anterior se plantea la siguiente **Idea a defender:**Mediante el diseño y la implementación del componente para gestionar documentos complementarios generados en los procesos aduaneros dentro del sistema GINA se logrará gestionar toda la información contenida en ellos.

Teniendo como **tareas a realizar**:

- Realización de un estudio detallado acerca de cada uno de los principales conceptos y estado del arte referentes al tema de sistema de gestión documental.
- Estudio de cada una de las herramientas a utilizar para darle cumplimiento a los objetivos propuestos.
- Realización del modelo de datos del componente para Gestionar Documentos Complementarios
- Realización del diagrama de clases del negocio del componente para Gestionar Documentos Complementarios
- Realización de los diagramas de diseño de la solución del componente para Gestionar Documentos Complementarios.
- Diseñar la implementación de las vistas del Diseño del componente para gestionar documentos complementarios.
- Implementación del modelo escogido para el Diseño del componente para gestionar documentos complementarios.
- Realización del diagrama de componentes para el componente para Gestionar Documentos Complementarios.
- Realización de pruebas funcionales al sistema.
- Documentación de las pruebas realizadas.

Para la realización de cada uno de los objetivos específicos expuestos anteriormente se han utilizado los siguientes **métodos científicos**:

Métodos Teóricos:

Histórico – lógico: Para el estudio del estado del arte de los sistemas de gestión documental, del proceso de desarrollo de software y las técnicas de diseño de software más usados.

Analítico – sintético: Este método es utilizado para analizar y comprender los procesos aduaneros llevados a cabo en cada uno de los subsistemas que forman parte del sistema GINA. Además se realiza un estudio del sistema GINA en general, lo que posibilita una mayor comprensión del problema en concreto

Modelación: Se usa este método para la creación de modelos y diagramas que reflejen la lógica de la creación del componente para la gestión de documentos complementarios en su diseño e implementación.

Implementación: Este método se utiliza con el objetivo de medir si el componente implementado es una solución que da respuesta a las necesidades planteadas y por la cual se generó el problema a resolver y además si este componente funciona completamente bien para ser de utilidad a la hora de aplicarse.

Estructura de la Tesis

El trabajo se encuentra distribuido en tres capítulos los cuales están conformados como se describe a continuación:

Capítulo 1. Fundamentación Teórica

En este capítulo se realiza una explicación de todos los aspectos teóricos y técnicos concernientes específicamente al diseño e implementación de los sistemas gestores de documentos. Se realiza una síntesis del estado actual de estos sistemas en el mundo y específicamente en las aduanas, así como la experiencia que se tiene sobre el tema en Cuba. Se analizan conceptos fundamentales que ayudaran a desarrollar una visión objetiva y precisa de cada unos de los temas relacionados con la creación y desarrollo de un sistema gestor de documentos. También se explican de las herramientas, lenguajes y frameworks propuestos a utilizar para el desarrollo del componente.

Capítulo 2. Diseño e implementación

El capítulo expone cada uno de los artefactos que son generados en la etapa del diseño y la implementación para desarrollar el componente para gestionar documentos complementarios, estos artefactos serán: el modelo de datos, las clases del negocio, los diagramas de diseño, el diagrama de componente, el diagrama de secuencia, el código fuente, diagrama de paquetes y el diagrama de despliegue.

Capítulo 3. Análisis de la solución

En este capítulo se analizarán los resultados obtenidos luego de la implementación y diseño del componente por medio de pruebas funcionales realizadas al sistema y cada uno de los resultados obtenidos serán debidamente documentados y referenciados.

CAPÍTULO 1. **FUNDAMENTACIÓN TEÓRICA**

1.1 Introducción

En el presente capítulo se realizará un análisis sobre el estado del arte a nivel mundial de los sistemas automatizados para gestionar los diferentes procesos aduaneros. También se verán diferentes conceptos relacionados con el tema de investigación para poder llegar a una comprensión más profunda del mismo y se analizarán las diferentes herramientas utilizadas para desarrollar la aplicación.

1.2 Conceptos

Documento: "toda representación material destinada a reproducir una manifestación del pensamiento, dentro de la cual no sólo caben las representaciones escritas denominadas instrumentos que no son más que una especie de documentos, sino que también otros documentos de carácter no instrumental como son las fotografías, películas, cintas magnetofónicas, discos, radiografías, electrocardiogramas, planos, cuadros, dibujos, etc."(2)

Documento electrónico: "son aquellos generados por y a través de un medio automatizado y pueden además estar memorizados en dispositivos susceptibles de ser leídos por los mismos." (3)

Gestión (del latín ***gestio***)*:* acciónde administrar. Gestión, dirección. Actividad profesional tendiente a establecer los objetivos y medios de su realización, a precisar la organización de sistemas, a elaborar la estrategia del desarrollo y a ejecutar la gestión del personal. (4)

Gestión documental: consiste en el uso de tecnología y procedimientos que permiten la gestión y el acceso unificado a información generada en la organización. (5)

Sistema de gestión documental: es un sistema computarizado, un conjunto de programas, utilizado para rastrear y almacenar documentos electrónicos y/o imágenes de documentos soportados en papel. (5)

1.3 Estado del arte.

En la actualidad en el mundo existen diferentes sistemas automatizados en las aduanas que se encargan de la gestión de los documentos complementarios que se generan en cada uno de los procesos aduaneros que se llevan a cabo en sus instalaciones, aquí se listan algunos de estos sistemas y se

explican sus principales características para poder llegar a una valoración crítica de la tendencia internacional respecto al desarrollo de este tipo de software en las aduanas del mundo.

Sistema Aduanero Automatizado (SIDUENA)

Es la herramienta informática para el control y administración de la gestión aduanera, desarrollada por La Conferencia de las Naciones Unidas sobre el Comercio y el Desarrollo (UNCTAD), y que actualmente es usada con éxito en más de 80 países.

SIDUNEA permite realizar un seguimiento automatizado de las operaciones aduaneras y controlar efectivamente la recaudación de los impuestos aduaneros, porque este sistema verifica automáticamente los registros, calcula los impuestos y contabiliza todo lo relativo a cada declaración, con la mínima intervención del factor humano subjetivo.

Al ser un sistema multidisciplinario, está especializado en cada área del trabajo aduanero para ser la herramienta de trabajo de todos los clientes de la aduana, sean usuarios internos o externos, privados o públicos. De este modo se convierte en un único lenguaje, seguro y comprensible para todos los actores del proceso. SIDUNEA se puede configurar de acuerdo a las características nacionales de cada régimen aduanero, al arancel nacional y a la legislación de cada país, además de implementar los estándares internacionales para procesar los datos de comercio exterior ya acordados por la Organización Mundial de Aduanas (WCO[1]) y por la Organización Internacional para la Estandarización (ISO[2]) Entre las ventajas que se pueden obtener con la aplicación del Sistema Aduanero Automatizado se encuentran:

- Optimizar los tiempos y recursos del proceso aduanero.
- Aplicar la ley con toda justicia.
- Cobrar correctamente los impuestos y tasas.

[1]WorldCustomsOrganization, por sus siglas en inglés.

[2] International OrganizationforStandardization, por sus siglas en inglés.

- Detectar los errores en los valores de la declaración.
- Monitorear el pago de los impuestos.
- Evitar la evasión de impuestos.
- Minimizar el contrabando.
- Crear incentivos para el declarante.
- Administrar efectivamente el proceso de despacho.
- Poner en práctica un esquema de garantía con la modalidad de pago anticipado, para facilitar el comercio y asegurar el cobro de los derechos aduaneros.
- Controlar la ruta de comercio por medio de las oficinas de despacho de mercancía de cada aduana.

Sistema Integrado de Comercio Exterior (SICE)

Es un sistema que permite hacer toda la documentación que se genera en el proceso de una importación y exportación, tomando el control desde el inicio de un trámite, con esta información se podrá generar una variedad de consultas y reportes respecto a seguimientos, costos y tiempos. Esta aplicación está totalmente integrada con una Base de Datos de información sobre Aranceles, la misma que permite consultar al momento que lo requiera, facilitando una precisa clasificación de mercancías y obteniendo información de interés con respecto a productos a importarse al Ecuador, sus impuestos y tasas vigentes, sus preferencias y sus licencias.

La aplicación es permanentemente actualizada cada vez que existen cambios en el Comercio Exterior y su entorno.

Es una solución integral para pequeñas, medianas y grandes empresas que desarrollan la actividad de Agentes de Aduana; que permite realizar la documentación necesaria en la importación o exportación para

su posterior envío a la Aduana. Proporciona la máxima facilidad de uso y el mayor rendimiento aún en las necesidades de negocio más exigentes.

Ventanilla Única de Comercio Exterior (VUCE)

La Ventanilla Única de Comercio Exterior se define como un mecanismo de facilitación que permite a las partes involucradas, en el comercio y el transporte, alojar información estandarizada y documentos en un solo punto de entrada para cumplir con todos los trámites de importación, exportación y tránsito. La información al ser electrónica, debe ser remitida una sola vez.

La VUCE se conceptualiza como "un sistema integrado que permite a las partes involucradas en el comercio exterior y transporte internacional gestionar, a través de medios electrónicos, los trámites requeridos por las entidades competentes de acuerdo con la normatividad vigente, o solicitados por dichas partes, para el tránsito, ingreso o salida del territorio nacional de mercancías".

Las ventajas que brinda el uso de esta herramienta son las siguientes:

- Mayor efectividad y eficiencia en la utilización de recursos.
- Mejor gestión de riesgo aduanero.
- Incremento en la satisfacción de los usuarios de comercio exterior.
- Mayor integridad y transparencia.
- Incremento en la seguridad.

Ventanilla Única para las Exportaciones (VUPE)

La Ventanilla Única para las Exportaciones se encarga de centralizar y coordinar las instituciones involucradas en trámites y procedimientos de exportación para facilitar la comercialización externa de los productos guatemaltecos, impulsando el desarrollo de proyectos de Comercio Exterior para contribuir a mejorar la competitividad del país.

Un grupo de exportadores conscientes de la necesidad de mejorar estos trámites a fin de hacer más viable el comercio Internacional, en el año 1986 solicitó al Gobierno de la República de Guatemala a través del Ministerio de Economía, su apoyo para unir físicamente todas las instituciones públicas y privadas participantes en el proceso exportador, y es así como en el mes de Septiembre de l986 es aprobado y publicado el acuerdo 790 –86 que crea la Ventanilla Única para las exportaciones bajo la responsabilidad directa del Ministerio de Economía. Posteriormente se delega a AGEXPRONT[3] la función de VUPE.

Los servicios que ofrece son los siguientes:

- Estadísticas.
- Exportación de documentos.
- Seminarios y cursos de capacitación.
- Continuo desarrollo de sistemas electrónicos para facilitar sus procesos.
- Asistencia personalizada y telefónica al exportador.
- Edición de material impreso que contiene los procedimientos aduaneros.
- Información en línea disponible a través de una página web.
- Pago en línea para los procesos de exportación.
- Control sobre los acuerdos de exportación entre Guatemala y otros países.

Documentos:

- Código de exportador.

[3] Asociación Guatemalteca de Exportadores.

- Servicio Electrónico de Autorización de Exportaciones (SEADEX).
- Certificados de Origen.
- Servicios de confirmaciones de DUA-GT[4].
- Servicios de acompañamiento de contenedores.
- Certificados fitosanitarios y zoosanitarios locales.
- Certificados fitosanitarios electrónicos.

Sistema de Certificación de Origen Digital de Cuba (SCOD-CUBA)

Es un programa capaz de gestionar las solicitudes de Certificación de Origen Digital (COD) y que permite automatizar procesos para la confección del mismo.

La creación de este programa surge por una solicitud y acuerdos de la Asamblea de Asociados de la Cámara de Comercio, en el año 2008 se creó un grupo de trabajo conformado por la Dirección informática y la Dirección jurídica y supervisado por SG-CCRC[5] que da seguimiento al proyecto.

El SCOD acelera y enriquece el proceso de solicitud y confección de los CO[6], son este sistema la solicitud del CO se realiza a través del correo electrónico, se realiza el trámite de certificación en cuestiones de minutos y se validan los datos registrados a partir de los datos generados automáticamente en la factura digital.

Los beneficios que ofrece este sistema a la AGR son:

[4] Se refiere al documento Declaración de Mercancía en Guatemala.

[5] Secretaría General de la Cámara de Comercio de la República de Cuba.

[6] Certificado de Origen.

- Mayor seguridad en cuanto a la integridad de los CO recibidos, disminuyendo considerablemente la posibilidad de fraudes.
- Reduce costos de verificación de autenticidad y validez de las firmas que constan en dichos CO.
- Permite la integración de los CO a los sistemas informáticos de las aduanas.
- Facilita la determinación de los controles basados en la evaluación de riesgos.

Valoración de los sistemas estudiados

La creación y desarrollo de sistemas automatizados para el control de los procesos aduaneros en todos los países es una actividad que ha obtenido grandes avances por la necesidad creciente de lograr calidad y rapidez en estos servicios, cada uno de los sistemas que han sido analizados anteriormente muestran cómo según las necesidades específicas de cada uno de los países estos sistemas han sido implementados y adaptados a sus requerimientos, sin embargo a pesar de todas las ventajas que presentan solo pueden servir como base material de estudio ya que estos sistemas son privativos y la adquisición de la licencia de cada uno de ellos es bastante costoso y otros solo están adaptados a las necesidades específicas del país que los desarrolló.

El sistema de creación de CO creado en nuestro país no da solución a la necesidad planteada por parte de la aduana pues solo permite la gestión de uno de los documentos complementarios y no está diseñado para poder gestionar los otros documentos complementarios que son generados en los diferentes procesos aduaneros dentro del sistema GINA.

Dada esta situación se da la necesidad de implementar un componente que permita gestionar los documentos complementarios que son generados en los procesos aduaneros, tomando en cuenta cada uno de los requisitos descritos según la necesidad especifica de la Aduana General de la República (AGR).

1.4 Herramientas usadas para el desarrollo de la aplicación

Patrones de Diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, basadas en la experiencia y que se ha demostrado que funcionan. Se dividen en tres grandes grupos:

Patrones de creación: muestran la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Estas decisiones normalmente serán resueltas dinámicamente decidiendo que clases instanciar o sobre que objetos un objeto delegará responsabilidades.

- **Singleton(Instancia Única):** garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.

Patrones estructurales: describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.

- **Adapter(Adaptador):** convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permiten que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.

- **Decorator(Decorador):** añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.

- **Facade(Fachada):** proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.

Patrones funcionales o de comportamiento: se utilizan para organizar, manejar y combinar comportamientos.

- **TemplateMethod(Método Plantilla):** define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.

Patrones GRASP[7]: son patrones generales de software para asignación de responsabilidades, son una serie de buenas prácticas de aplicación recomendable en el diseño de software.

- **Bajo Acoplamiento:** aumentan la reutilización y eliminan las dependencias entre las clases para propiciar un fácil mantenimiento y entendimiento.
- **Alta Cohesión:** cada elemento dentro del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificables.
- **Controlador:** facilita la centralización de actividades, delega las actividades en otras clases con las que mantiene un modelo de alta cohesión.
- **Controlador Frontal:** centraliza todas las peticiones de los usuarios y los provee de un punto único de entrada a la aplicación.

ER Studio

Es una herramienta de modelado para el análisis, visualización y comunicación de base de datos, aplicaciones de diseños de datos y arquitectura de la información. Sus principales características son la combinación de procesos, datos, modelado UML[8], y presentación de reportes en un potente entorno de diseño en multi-niveles.

Los principales beneficios que brinda son:

- Promoción de la reutilización de datos y la colaboración en tiempo real a través de una gestión eficaz del modelo de la empresa y la capacidad de publicación de metadatos.
- Mejora de la visibilidad y la calidad de información con la ingeniería inversa y el soporte al ciclo de vida completo de base de datos.

[7] General ResponsabilityAssignament, por sus siglas en inglés.

[8] UnifiedModelingLenguage, por sus siglas en inglés.

- Comunicación efectiva de los modelos de toda la empresa con informes de metadatos en tiempo real y herramientas de publicación.

Visual Paradigm

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Características:

- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio.
- Modelado colaborativo con CVS[9] y Subversion[10].
- Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse) a través de XMI[11].
- Ingeniería de ida y vuelta. Ingeniería inversa - Código a modelo, código a diagrama. Ingeniería inversa Java, C++, Esquemas XML, XML, .NET exe/dll, CORBA IDL.
- Generación de código - Modelo a código, diagrama a código.

[9] ConcurrentVersioningSystem, por sus siglas en inglés.

[10] Software de Sistema de Control de Versiones, también puede ser representado por las siglas SVN.

[11] Estándar para el intercambio de modelos usando XML.

- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Diagramas de flujo de datos.
- Soporte ORM[12]- Generación de objetos Java desde la base de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS[13]) existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.

Sistema Gestor de BD Oracle

Oracle es un Sistema Gestor de Bases de Datos con características objeto-relacionales. Sus características principales son las siguientes:

- Entorno cliente/servidor.
- Gestión de grandes bases de datos.
- Usuarios concurrentes.

[12]ObjectRelationalMaping, por sus siglas en inglés.

[13]Database Management System, por sus siglas en inglés.

- 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.
- Compatibilidad.
- Conectividad.
- Replicación de entornos.

Framework Symfony

Symfony es un framework de PHP5 que facilita el desarrollo de las aplicaciones web, se encarga de todos los aspectos comunes y aburridos de las aplicaciones web, dejando que el programador se dedique a aportar valor desarrollando las características únicas de cada proyecto. Es además un framework que cuenta con miles de páginas de documentación distribuidas en varios libros gratuitos y decenas de tutoriales.

Sus principales características son:

- Fácil de instalar y configurar en sistemas Windows, Mac y Linux.
- Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server).

- Compatible solamente con PHP 5 desde hace años, para asegurar el mayor rendimiento y acceso a las características más avanzadas de PHP[14].
- Basado en la premisa de convenir en vez de configurar, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Preparado para aplicaciones empresariales, ya que se puede adaptar con facilidad a las políticas y arquitecturas propias de cada empresa u organización.
- Extensible mediante un completo mecanismo de plugins.
- Publicado bajo licencia MIT de software libre y apoyado por una empresa comprometida con su desarrollo.
- Traducido a más de 40 idiomas y fácilmente traducible a cualquier otro idioma.

PHP

Es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML[15]. PHP es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales.

Principales características:

- Lenguaje multiplataforma
- Soporta la mayoría de los servidores web que existen hoy.
- A partir de PHP5 incluye la programación procedimental o programación P.O.O.

[14] Hypertext Preprocessor

[15] Hyper Text Markup Lenguage

- Incluye habilidades como creación de imágenes y archivos PDF, también puede presentar otros resultados como XHTML[16] y cualquier otro tipo de ficheros XML.
- Abstracción de la BD lo que permite usar de forma transparente cualquier BD.
- Soporte para comunicarse con otros servicios usando protocolos como LDAP, POP·, INAM, SNMP, HTTP y muchos otros.
- Varias características muy útiles para el procesamiento de texto.

Ext JS

Es un framework de JavaScript que permite realizar aplicaciones web enriquecidas basándose en tecnología AJAX, JSON, DHTML y DOM. Se encuentra patentada bajo la licencia LGPL[17] lo que posibilita su uso para aplicaciones empresariales privativas de código cerrado.

Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts, razón por la cual provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno.

Los principales beneficios son:

- Balance entre Cliente – Servidor: la carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- Comunicación asíncrona: en este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.

[16] Extensible Hyper Text MarkupLenguage

[17] Lesser General PublicLicense, por sus siglas en inglés.

- Eficiencia de la red: el tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

UML

El Lenguaje Unificado de Modelado (UML, por siglas en inglés UnifiedModelingLanguaje) es un lenguaje muy popular de modelado de sistemas de software. UML usa técnicas de notación gráfica para crear modelos visuales de sistemas de desarrollo de software.

Este lenguaje se encarga de documentar, visualizar y especificar las funciones y procesos de los sistemas de software orientados al objeto. Representa un modelo estándar para visualizar un blueprint (dibujo técnico) de sistema, que incluye elementos como el actor (que especifica el papel que juega un usuario que interactúa con el sujeto), el proceso de negocio (tareas relacionadas lógicamente para lograr un negocio definido), el componente (encapsula el contenido del sistema), la actividad (tarea que toma lugar para cumplir un contrato de operación), los estatutos del lenguaje de programación, los esquemas de la base de datos y los componentes reusables del software.

Los principales beneficios que ofrece son:

- Mejores tiempos totales de desarrollo (de 50 % o más).
- Modelar sistemas (y no sólo de software) utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.

1.5 Conclusiones parciales

En este capítulo se analizaron los principales sistemas automatizados para Aduanas que existen en el mundo valorando cada una de sus ventajas y desventajas y las razones por las cuales no son sistemas óptimos para utilizar en nuestro país. También se abordaron cada una de las herramientas a utilizar en el proceso de diseño e implementación de la aplicación, teniendo como principales recursos las tecnologías libres dado el proceso de migración que se desea llevar a cabo en nuestro país. Además se analizaron conceptos básicos relacionados con el tema de investigación.

CAPÍTULO 2. **DISEÑO E IMPLEMENTACIÓN**

2.1 Introducción

En el presente capítulo se desarrollaran los flujos de trabajo de diseño e implementación del Componente para la Gestión de Documentos Complementarios. Se describirán cada uno de los artefactos obtenidos en el modelo de diseño y en el diseño de la base datos para una mejor comprensión del trabajo realizado. También se explicarán otros aspectos de interés que fueron tomados en cuenta a la hora de implementar el componente y se explicarán los artefactos generados durante este flujo de trabajo.

2.2 Diseño del Sistema

En el diseño es donde se modela el sistema de tal forma que se cumpla los requisitos funcionales y no funcionales que fueron identificados en el análisis. Se dedica principalmente a crear una jerarquía adecuada de módulos de programas y de interfaces entre ellos, además descompone el trabajo de implementación en partes más manejables. También en esta actividad se transforma el modelo de datos entidad-relación en un diseño de base de datos.

“En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos incluyendo los requisitos no funcionales y otras restricciones que se le suponen”. (7)

2.2.1 Modelo del Diseño

El modelo de diseño es un modelo físico, no genérico, específico para una implementación, es dinámico ya que está centrado en las secuencias, el mismo debe ser mantenido durante todo el ciclo de vida del software y logra dar forma al sistema mientras que intenta preservar la estructura definida por el modelo de análisis lo más posible. Este es el artefacto más importante que se obtiene durante el flujo de diseño.

El Modelo de Diseño es una abstracción de la implementación del sistema, para lograr que tenga calidad debe cumplir las siguientes características:

- Ser resistente a cambios en el entorno de implementación.
- Ser fácil de mantener en relación con otros posibles modelos de objetos y para la implementación del sistema.

2.2.1.1 Diagramas de clases del negocio

En un diagrama de clases del negocio se representan las clases que intervienen en la creación de la aplicación y cada uno de sus atributos y funcionalidades y las relaciones que se establecen entre estas.

A continuación en la f**igura 2.1** se muestra el diagrama de Clases del Negocio perteneciente al Componente Gestionar Documentos complementarios, se puede ver detalladamente cada uno de los atributos y funcionalidades que componen a cada una de las clases y las relaciones que existen entre cada una de ellas. Todas las clases que no tienen ni atributos ni funcionalidades y solo se especifican sus relaciones son aquellas clases pertenecientes a otros esquemas que son utilizadas por el componente.

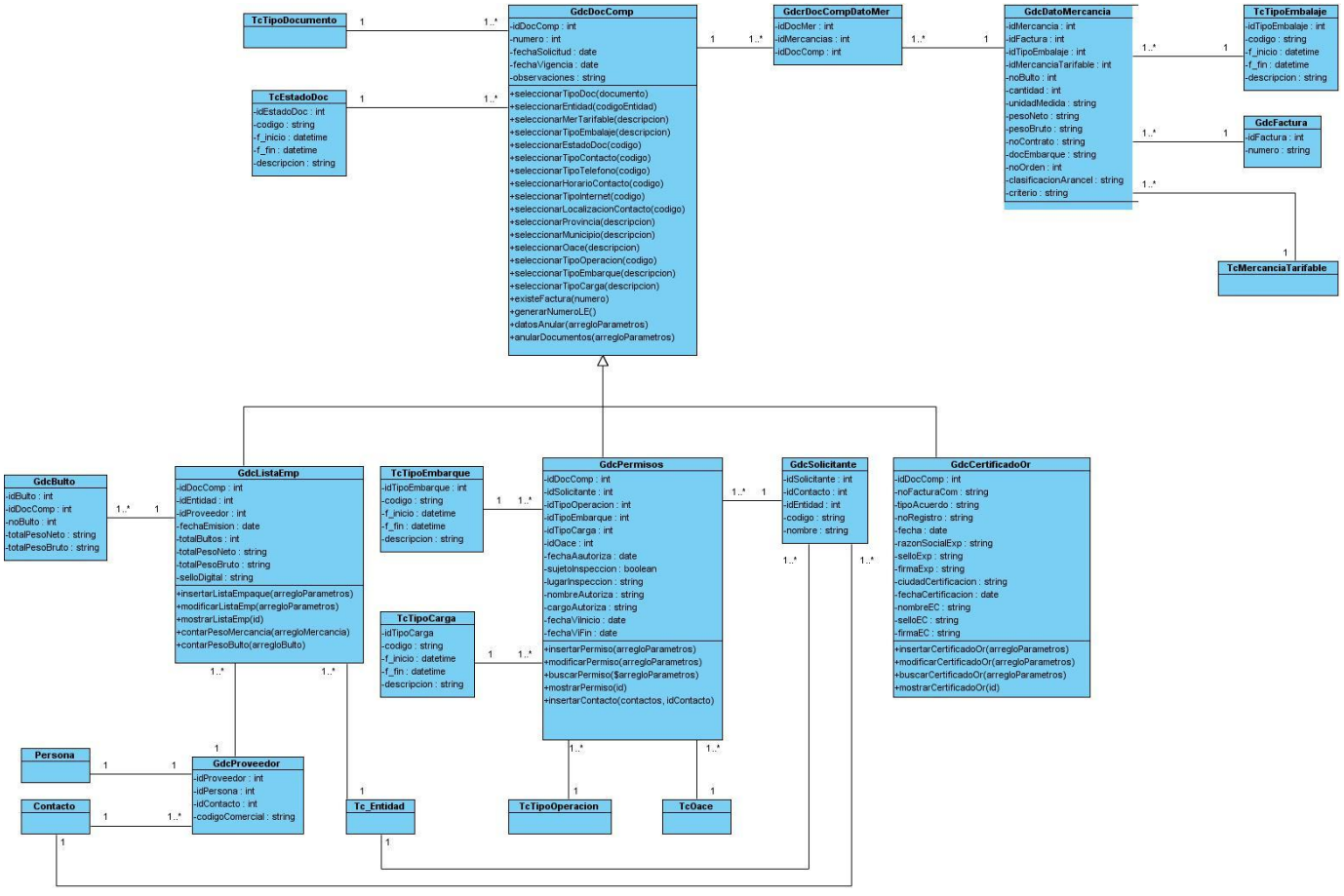


Figura 2.1 Diagrama de Clases del Negocio

2.2.1.2Diagramas de Secuencia de Actividades de los RF

Los Diagramas de Secuencia de Actividades de los RF representados en las f**iguras 2.2, 2.3 y 2.4**, son diagramas de interacción que muestran la interacción entre los objetos para cumplir con los requisitos funcionales identificados para el sistema.

Este diagrama de secuencia tiene la particularidad de representar las clases como calles, las mismas se comunican para la transmisión de los datos, pero cada una de las actividades que necesiten un cambio de estado es graficada con todos los pasos que necesita para culminar su tarea.

Con la realización de este tipo de diagramas se trata de evitar el trabajo engorroso que realiza el diseñador en el momento de diagramar una solución que resulte rápida, explicativa y eficaz para el programador.

Cada una de estas adecuaciones dentro del proyecto Aduana son explicadas en el trabajo de diploma titulado Procedimiento para la Ingeniería de Requisitosen el Departamento de Desarrollo de Solucionespara la Aduana del CEIGE.

Estos diagramas se van a dividir en dos grandes grupos los diagramas de Actividades del Negocio y los de Actividades de Componentes Visuales.

2.2.1.2.1 Actividades del negocio

Los diagramas de Actividades del Negocio son una respuesta del diseño a los requisitos funcionales capturados en fases anteriores y expresan paso a paso la comunicación entre las tres capas de arquitectura y cada uno de los pasos que debes seguir el programador para lograr concluir la aplicación final satisfactoriamente.

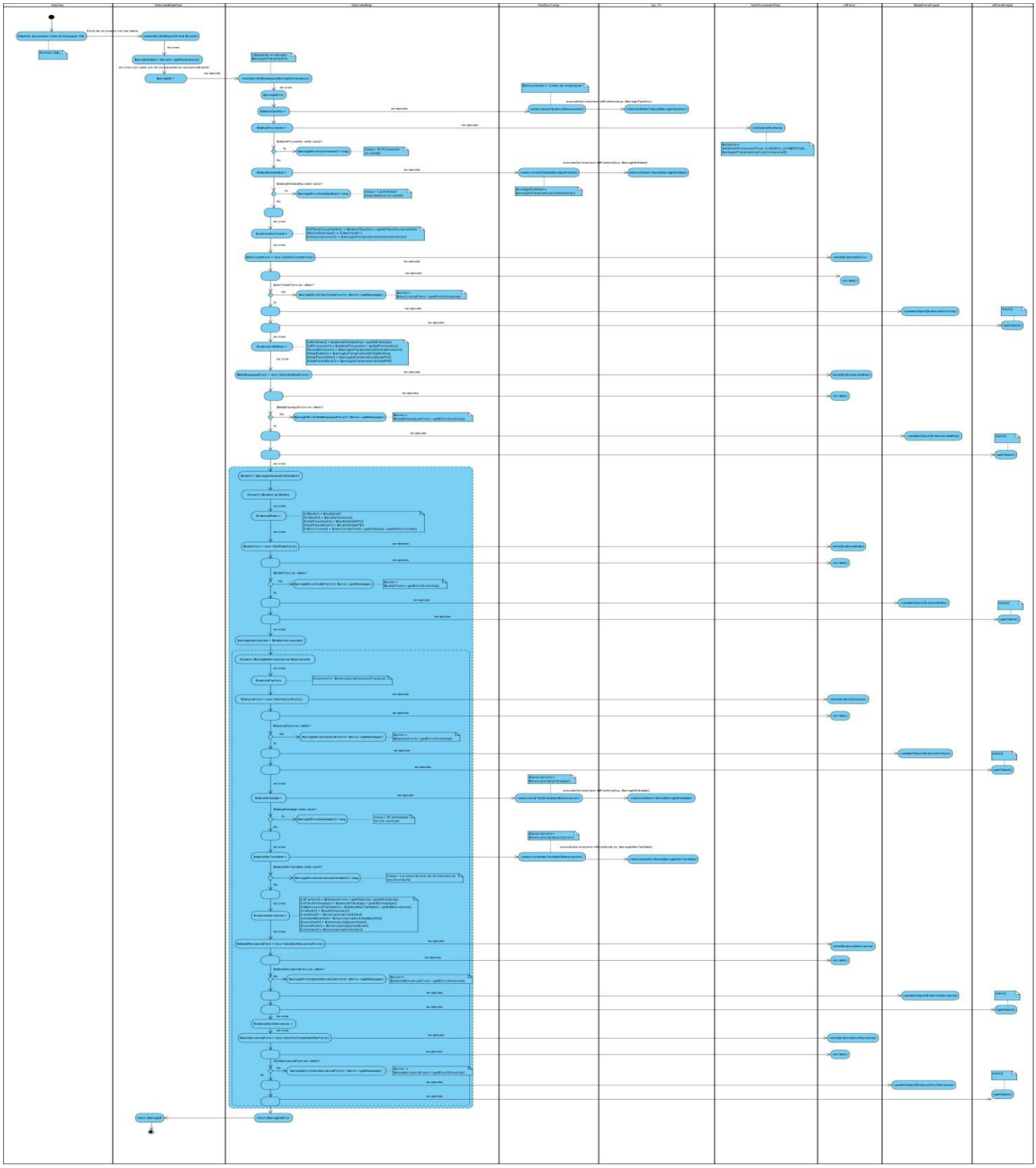


Figura 2.2 Diagrama de Secuencia Registra Lista de Empaque

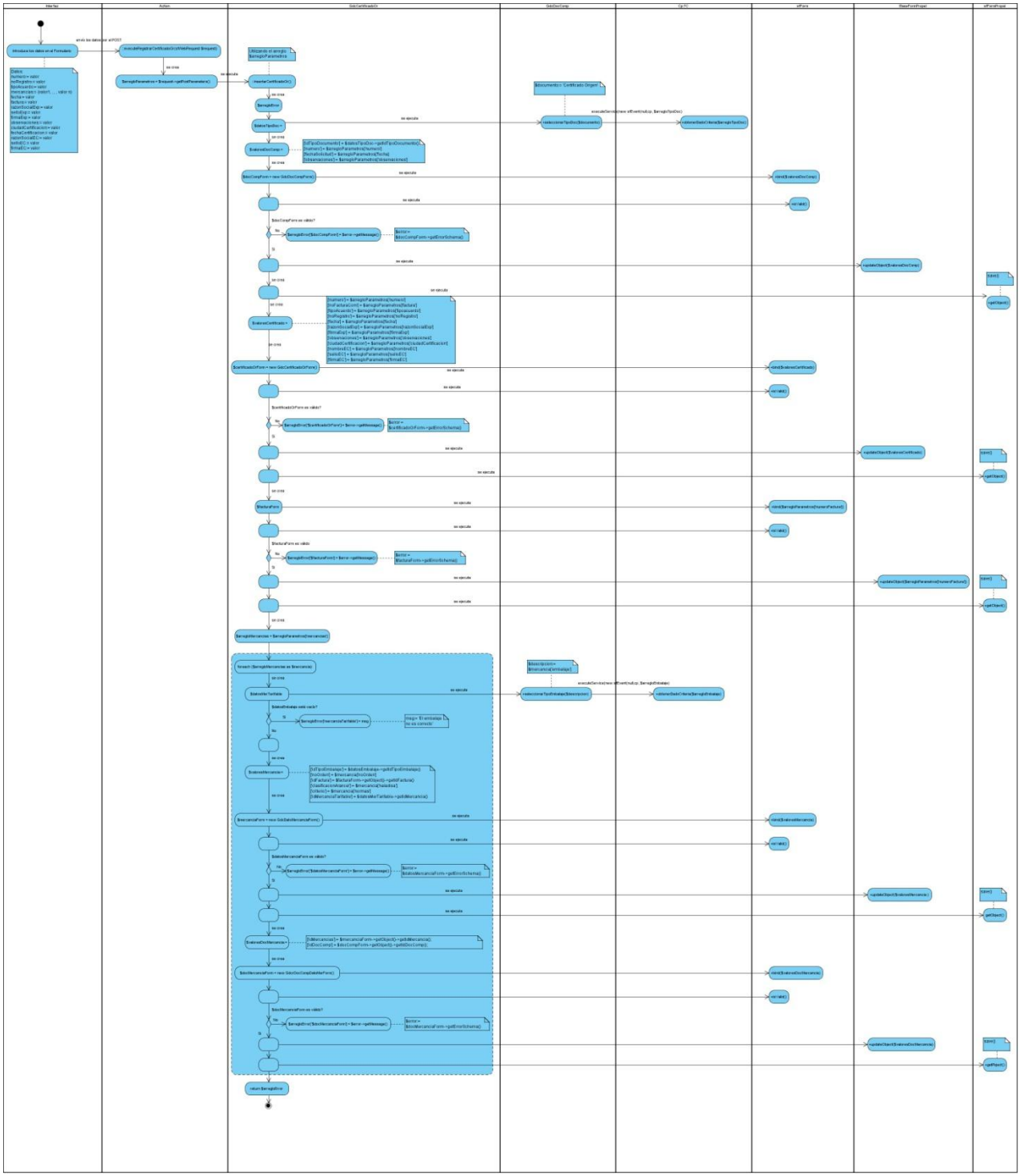


Figura 2.3 Diagrama de secuencia Registrar Certificado de Origen

En el **Anexo I**sonmostrados otros diagramas de secuencia de los principales escenarios de la aplicación.

2.2.1.2.1 Actividades de componentes visuales

Los diagramas de actividades de componentes visuales van a representar las acciones en la interfaz, mostraran cada una de las funcionalidades que se necesitan para incorporar a la vista elementos que permitan la selección de los datos para la ejecución de los diagramas de actividades del negocio y no requieren de la petición de datos a través del controlador frontal para que funcionen.

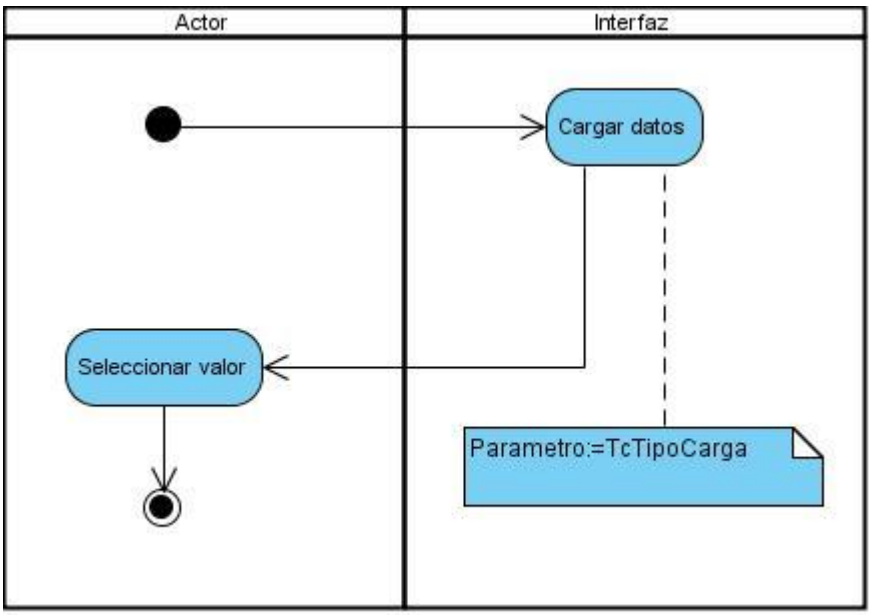


Figura 2.4 Diagrama de secuencia Tipo de Carga

Todos los diagramas de actividades de componentes visuales que faltan son representados en el **Anexo II**.

2.2.1.3 Diagrama de Paquetes

Un diagrama de paquetes permite dividir el sistema orientado a objetos organizándolo en subsistemas y detallando sus relaciones.

Contiene dos tipos de elementos:

- Paquetes: son una agrupación de elementos que pueden ser casos de usos, clases o componentes.
- Dependencias: indica que un elemento de un paquete requiere a otro de un paquete distinto.

En la **figura 2.5** es representado el diagrama de paquetes correspondiente al componente.

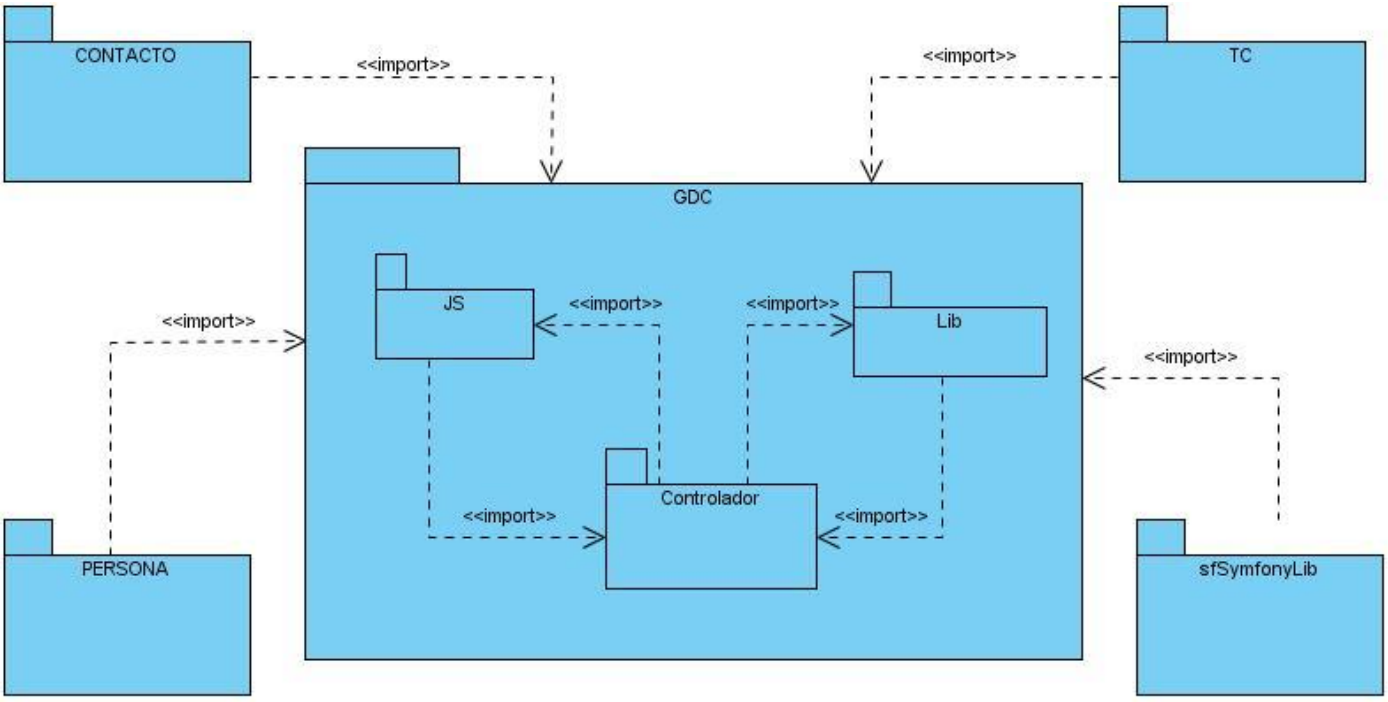


Figura 2.5 Diagrama de paquetes

2.2.2 Diseño de la BD

Una base de datos correctamente diseñada permite obtener acceso a información exacta y actualizada. Un diseño correcto es esencial para lograr los objetivos fijados para la base de datos. El objetivo del diseño de una base de datos relacional es generar un conjunto de esquemas de relaciones que permitan almacenar la información con un mínimo de redundancia, pero que a la vez faciliten la recuperación de la información.

2.2.2.1 Modelo lógico de datos

El modelo lógico de datos se usa para describir datos en los niveles conceptuales y de visión, es decir se representan los datos de la misma forma en que se captan en el mundo real.

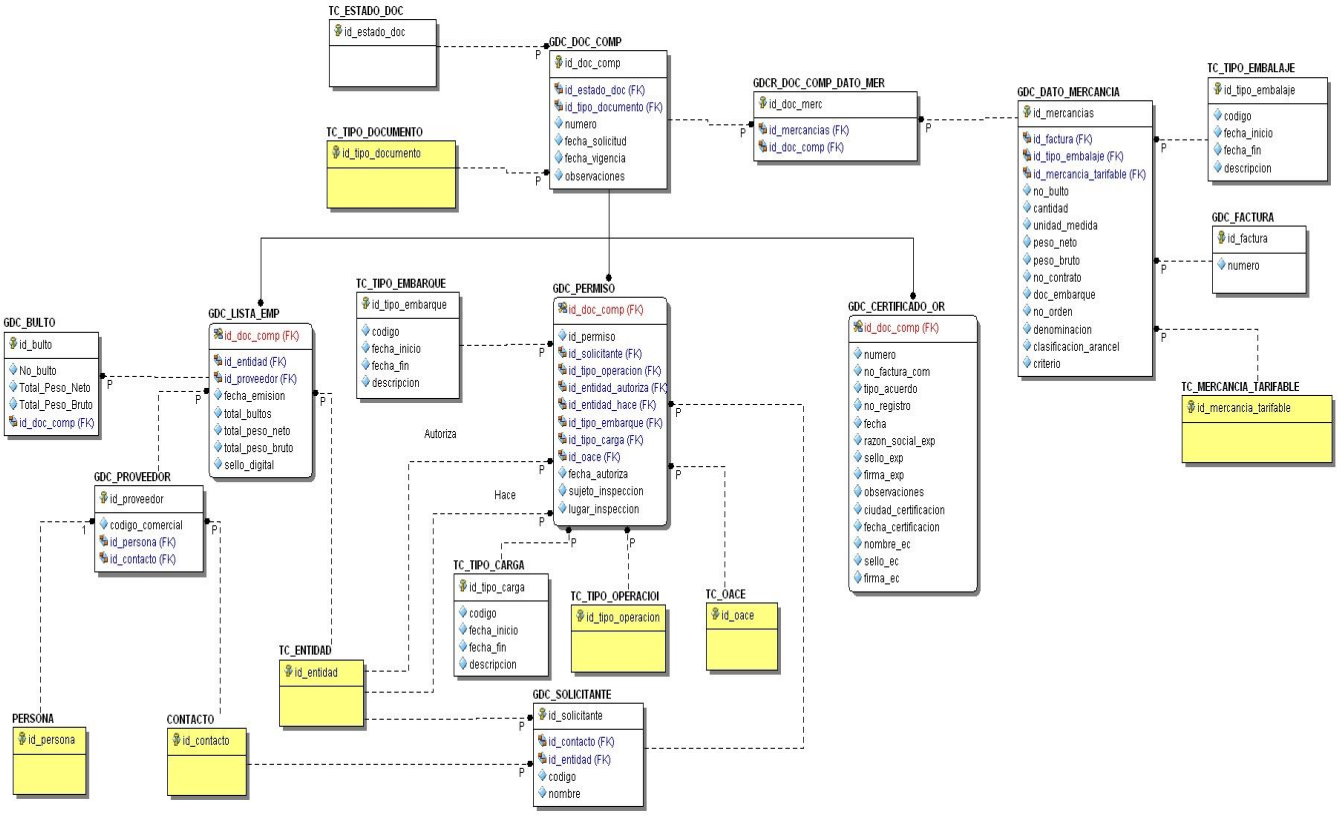


Figura 2.6 Diagrama de clases persistentes

2.2.2.2 Modelo físico de datos

El modelo físico de datos es usado para describir los datos a niveles más bajos,básicamente se logra capturar aspectos de la implementación de los sistemas de base de datos.

El Modelo de Datos obtenido luego de realizar el diseño de la Base de Dato del componente se puede ver en la **figura 2.7**, este modelo se encuentra formado por un total de 21 tablas distribuidas de la siguiente forma: 14 tablas pertenecen al esquema del componente y las 7 restantes forman parte de otros esquemas existentes.

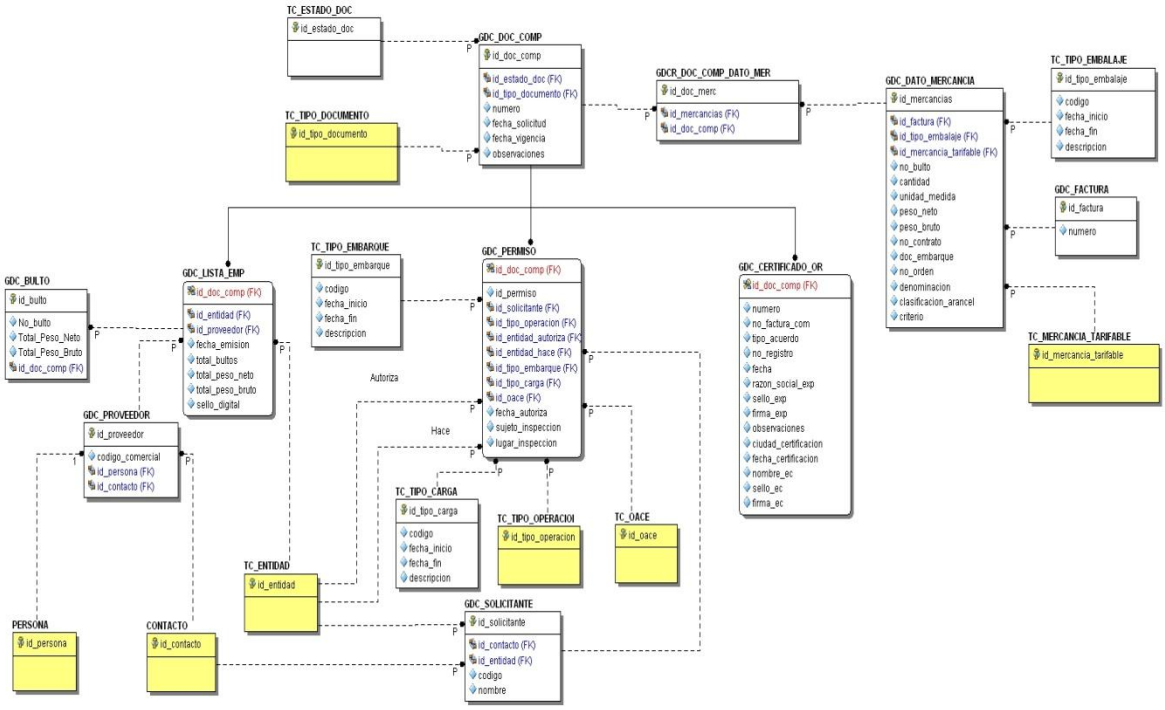


Figura 2.7 Modelo de datos

2.3 Implementación del Sistema

En la implementación se describe cómo los elementos del Modelo de Diseño se implementan en términos de componentes. Los principales objetivos que se deben lograr en la implementación son:

- Definir la organización del código, en términos de los subsistemas de implementación organizados en capas.
- Implementar los elementos de diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables y otros).
- Probar y desarrollar componentes como unidades.
- Integrar los resultados producidos por los implementadores individuales (o equipos) en un sistema ejecutable.

2.3.1 Estándares de codificación

Los estándares de codificación son pautas de codificación que no se encuentran enfocadas específicamente a la lógica del programa sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código por parte de todo el equipo de trabajo.

Al definir el estándar de codificación a seguir en la implementación del componente se utilizó la notación propuesta en el Estándar de Codificación del Departamento de Soluciones para la Aduana del Centro de Informatización y Gestión de Entidades, en este estándar se comprende todo el código generado bajo la tecnología y el lenguaje PHP y que utilicen la arquitectura regida por la utilización del Framework Arquitectónico Symfony.

Para nombrar el plugin se debe iniciar con el prefijo **sf** y seguido del nombre que debe contener la menor cantidad posible de palabras y se le coloca el sufijo **plugin**.

Las clases se le escribirán delante el identificador del Componente y seguido de nombre con letra inicial mayúscula por cada palabra, este nombre debe ser sugerente y ajustarse al alcance y responsabilidad de la misma. Las clases del modelo generadas por Propel mantendrán la nomenclatura que propone el Framework y siempre tendrán el sufijo **Peer**, para los formularios se usará el sufijo **Form** para poder identificarlos.

Los nombres de todas las funciones siempre empieza la primera palabra con letra inicial minúscula y las siguientes con letra inicial mayúscula, deben dejar reflejado claramente la acción a realizar por el mismo, pudiéndose apoyar en la utilización de sufijos que ayuden a identificar el resultado final de la ejecución de un método y en prefijos que ayuden a expresar la acción que realiza sobre un elemento determinado.

En el caso de las acciones se debe seguir la nomenclatura establecida por el Framework que indica que las clases deben comenzar con la palabra **execute** y seguido del nombre de la acción en nomenclatura CamelCase.

Para las variables sus nombres deben expresar claramente el contenido de la misma pueden estar referidas en singular o plural y son definidas al principio de las estructuras donde son utilizadas, sino se le asigna un valor inicial se debe inicializar con un valor que indique el tipo de dato al que debe pertenecer, los tipo de datos cadena se definen con comillas doble (“) y los tipos de datos caracteres se definen con comillas simples (´), en caso de que vaya a almacenar diferentes tipos de datos no se inicializa.

Para los bloques de códigos se pueden utilizar cualquiera de las estrategias de codificación siguiente:

1. Colocar las llaves en líneas separadas del código al mismo nivel del inicio del bloque.

2. Colocar la primera llave en la misma línea del comienzo del bloque separada por un espacio y la llave de cierre del bloque en la última línea (política tradicional del UNIX).

Los paréntesis no se utilizarán seguido de palabras claves del lenguaje, se sitúan seguido del nombre de las funciones y no se utilizan en las sentencia **return** de no ser necesario

Para comentar el código se utilizará en el caso de una línea los caracteres // y en caso de un bloque /* */. En la documentación de las variables y las funciones se utilizará un comentario de bloque donde se especifique un comentario de la misma, el o los tipos de parámetros y el tipo de retorno de ser necesario para la función.

Los nombres de las tablas siempre se escriben en mayúscula y singular. Las tablas deben pertenecer a un esquema que consta de un identificador de no más de 4 caracteres, los nombres de las tablas se inicializaran con el esquema al que pertenecen, seguido del guión bajo "_" y el nombre de la tabla. En el caso de las tablas de relación (relaciones de N a M) deben nombrarse utilizando los nombres de las tablas intervinientes, separadas con un guión bajo y al nombre del esquema se le asignará una R al final.

Los campos claves se ubican al principio de la tabla, se escriben en minúscula y siempre están antecedidos del prefijo id y luego el nombre de la tabla.

Las tablas de control (TC) deben tener invariantemente los siguientes campos:

Pos	Nombre	Tipo Dato	Null
01	id_nombre_tabla	autocremental	PK
02	codigo	varchar(50)	No
03	f_inicio	DATETIME	No
04	f_fin	DATETIME	Sí
05	descripcion	Varchar(250)	Sí

Tabla 2.1 Representación de los campos de las Tablas de Control

Además de estos pueden tener los demás campos que sean necesarios. Pueden tener más de un campo de código de negocio, estos se manejarán con la nomenclatura código_1, código_2, …, código_n.

2.3.2 Tratamiento de errores

El tratamiento de errores es una acción fundamental para lograr garantizar el correcto funcionamiento de cualquier sistema ya que permite identificar y controlar todos errores que pueden aparecer durante la interacción por parte del usuario con el sistema.

Para garantizar un correcto tratamiento de errores en el presente sistema se tratan todos los errores que puedan aparecer durante la interacción con la base de datos, persiguiendo este objetivo se toman las siguientes acciones, validación para comprobar la corrección de los datos a tratar, se trata que en el formulario el usuario introduzca la menor cantidad de valores posibles para evitar incoherencias e incorrecciones de los mismos, para los datos que introduce el usuario se implementan funciones que validen cada uno de estos datos para en caso de errores mostrar mensajes con la información de todos los errores cometidos durante la inserción o modificación de datos.

2.3.3 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos de un sistema que representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas.

El uso más importante de un Diagrama de Componentes es mostrar la estructura de nivel elevado del Modelo de Implementación, específicamente:

- Subsistemas de implementación y sus dependencias de importación.
- Los subsistemas de implementación organizados en capas.

En la f**igura 2.8** se muestra el Diagrama de Componentes correspondiente al Componente Gestionar Documentos Complementarios.

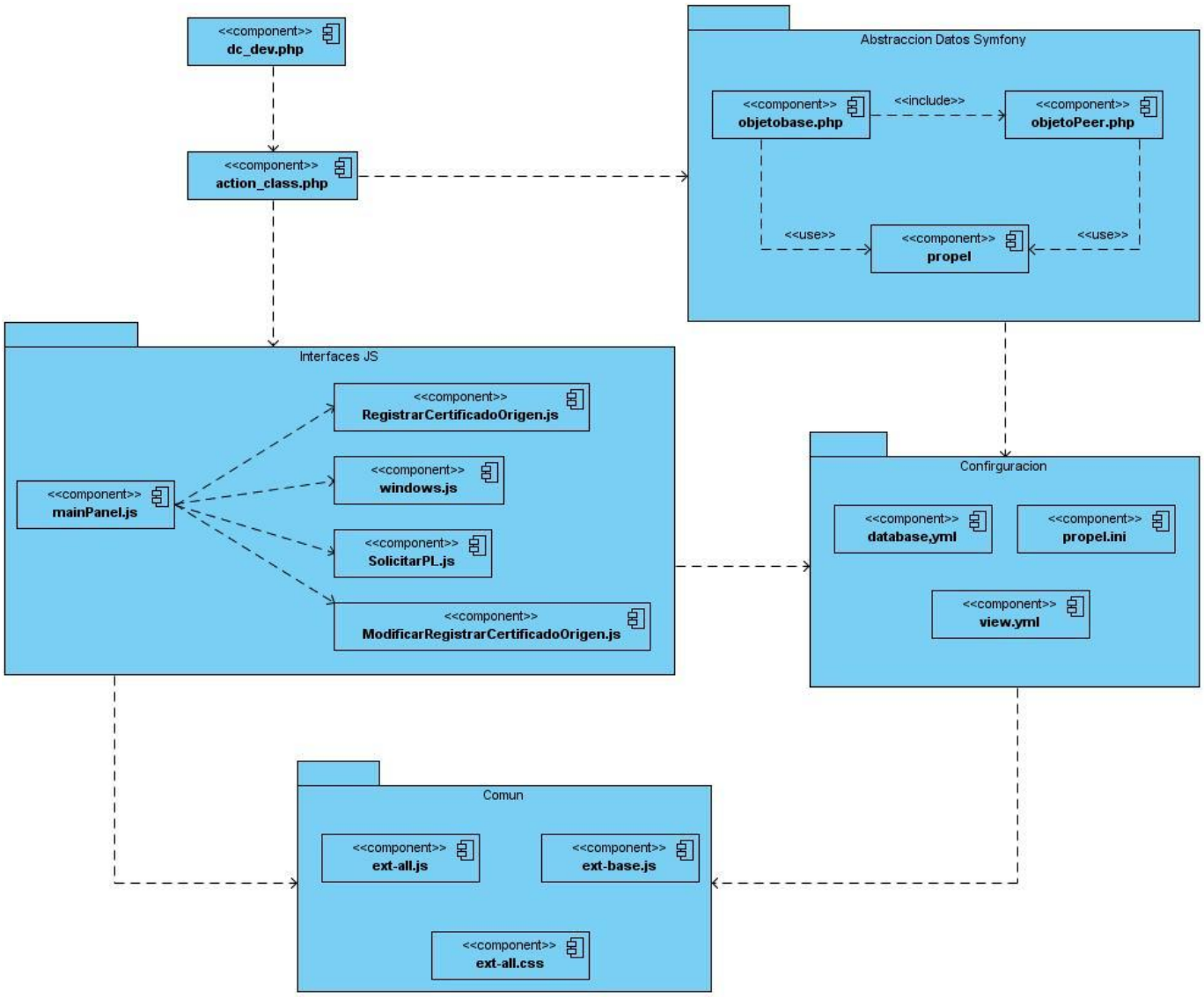


Figura 2.8 Diagrama de Componentes

2.4 Comunicación entre capas

En la comunicación entre capas en la que está dividida la arquitectura se encuentra orientada a mantener los paradigmas de la Programación Orientada a Objeto y posibilitar realizar sistemas mucho más portables.

Entre el cliente y el controlador frontal y viceversa se utiliza tecnología JSON, por medio de esta los datos pasados por el usuario son serializados en objetos PHP, esto permite interactuar con el servidor de la aplicación con cualquier tipo de interfaz y además da seguridad en la comunicación pues los implementadores de interfaz de usuario no van a tener acceso al comportamiento de los objetos que son enviados desde la capa controladora.

También se combina esta tecnología con AJAX para aumentar la velocidad y el dinamismo de la interacción con la aplicación siempre que sea posible. Se utiliza AJAX con comunicación basada en XML para cuando no es posible enviar los datos por JSON.

La comunicación entre las capas del Controlador y el Modelo se realiza por el envío de objetos con la información necesaria para el manejo de las peticiones del usuario y de las acciones a realizar, por medio de la tecnología ORM.

En la comunicación con la Base de Datos se utiliza PDO, librería que trae el Propel 1.3 para la abstracción de la BD.

2.4.1 Ejemplo de comunicación entre capas

A continuación es explicado cómo se desarrolla la comunicación entre capas dentro de la aplicación a través del caso de uso Registrar Certificado Origen para que sea más ejemplificado y entendible este proceso.

El usuario interactúa con la página principal del sistema (ver **figura 2.9**) donde está anexado el complemento, se puede ver la barra de menú con las funcionalidades que brinda el sistema con el nombre específico de Documentos, el usuario selecciona cualquiera de las acciones desplegada en el menú, en este caso específicamente la que responde a Registrar Certificado Origen (ver **figura 2.11**), el sistema muestra la pantalla correspondiente a la acción seleccionada (ver **figura 2.10**).

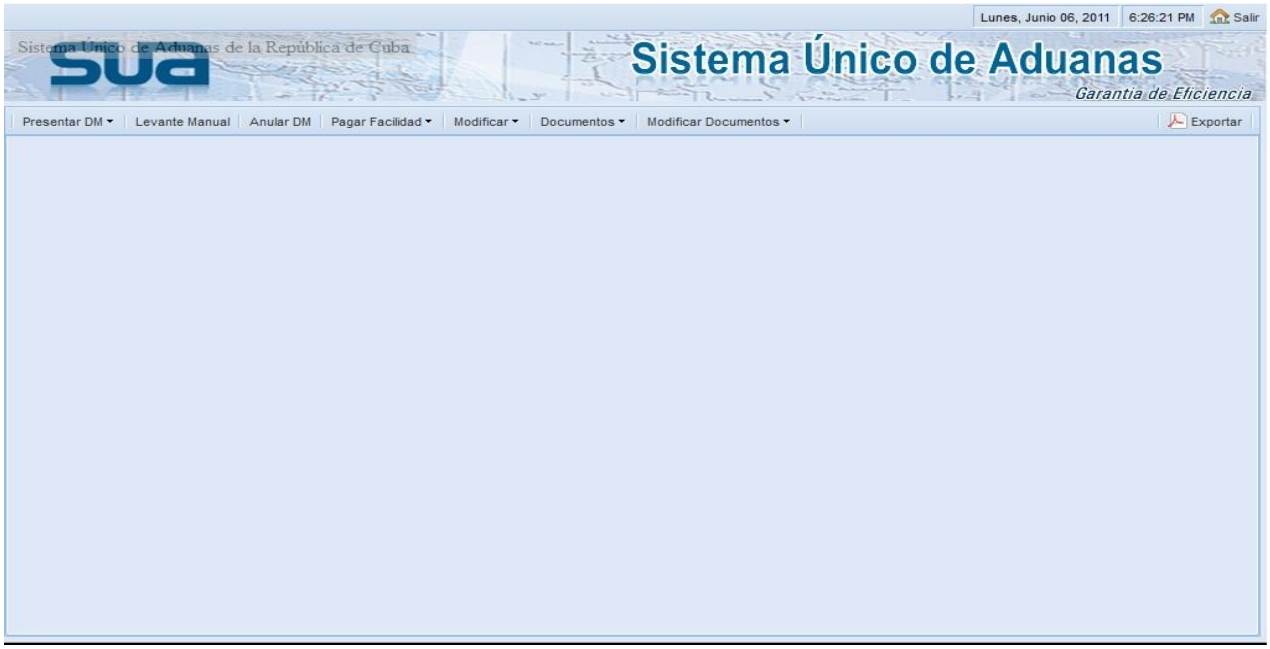


Figura 2.9 Pantalla Principal

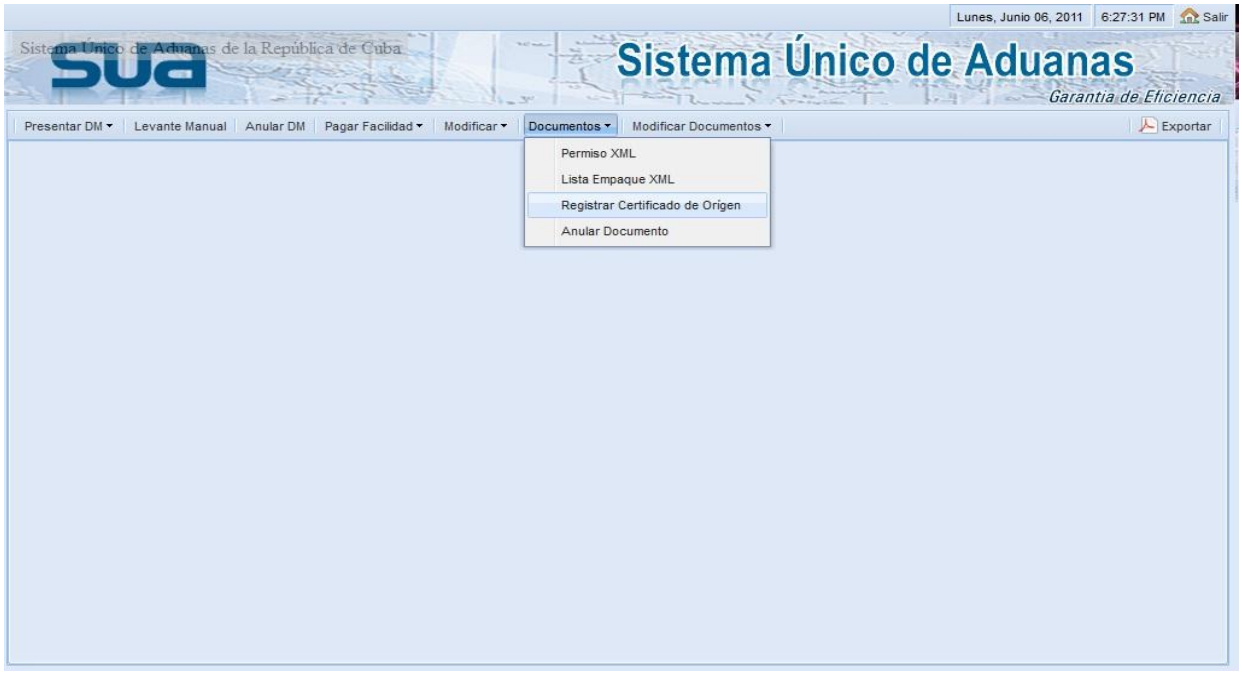


Figura 2.10 Menú del Componente

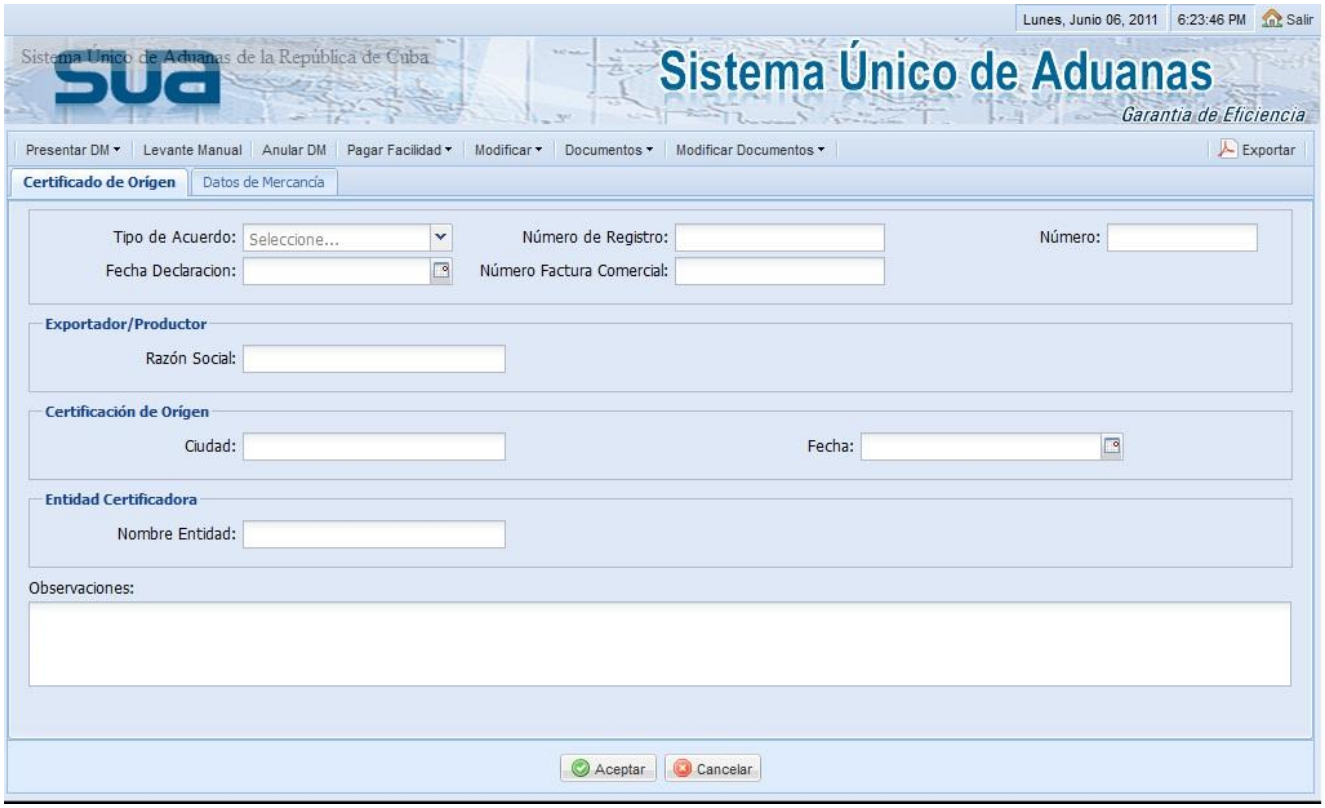


Figura 2.11 Pantalla Registrar Certificado Origen.

Para poder registrar el Certificado Origen el usuario debe insertar los datos necesarios, en caso de que estén incorrectos el sistema muestra un mensaje de error que es lanzado desde el javascript como se puede ver en la **figura 2.12 y 2.13.**

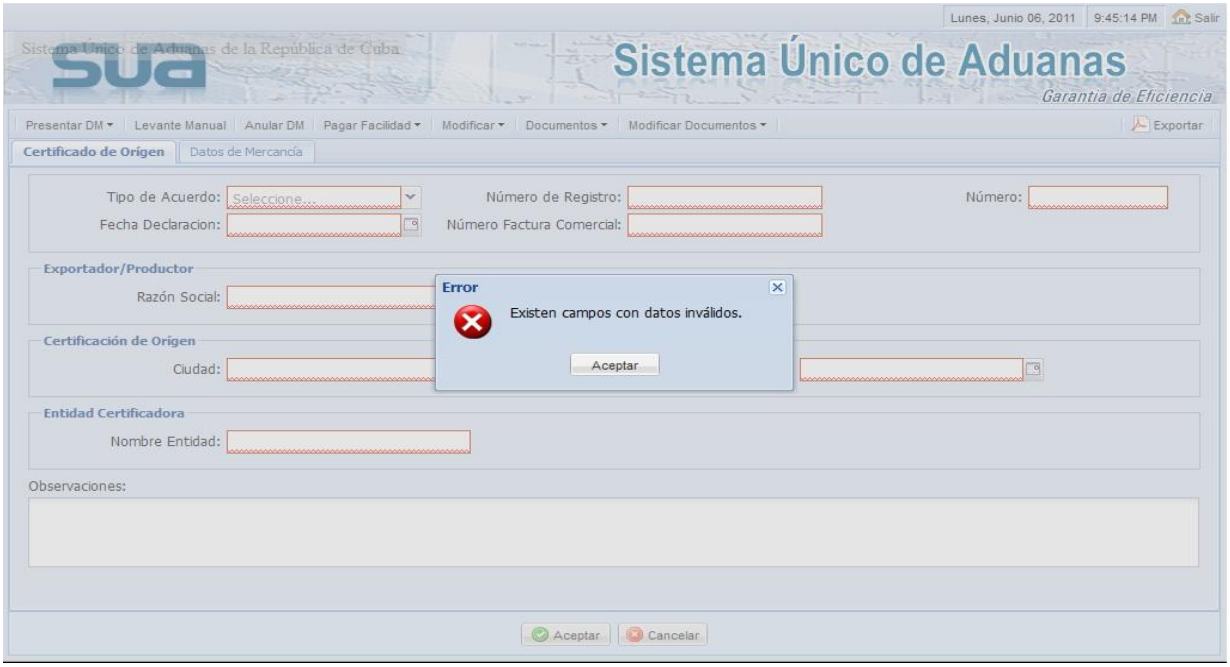


Figura 2.12 Pantalla de error cuando existen campos inválidos

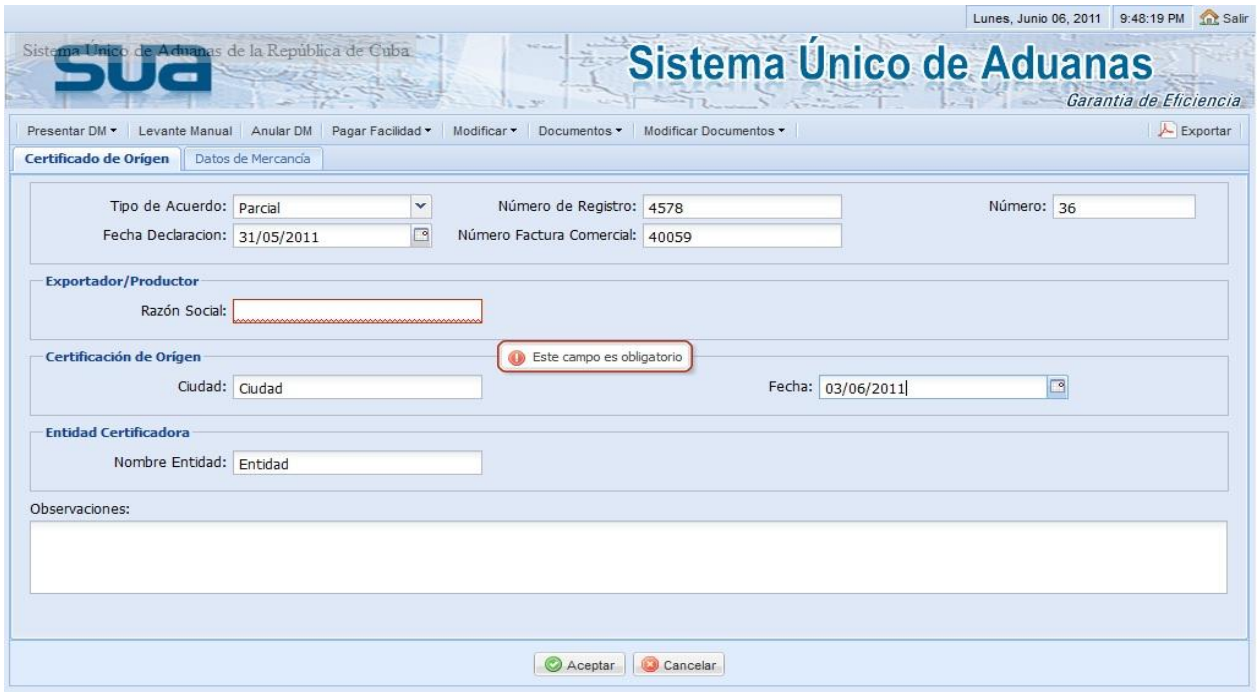


Figura 2.13 a) Pantalla de notificar errores en campos específicos

Una vez que el usuario inserta correctamente los datos en la pantalla tal como se muestra en la **figura 2.14** y acepta, estos son enviados a la controladora `sfDocumentosActions` en el evento del submit del formulario Registrar Certificado Origen por medio del método POST como se observa en la **figura 2.15** donde se puede observar en el url la dirección del `action` que recibirá los datos y el evento submit del formulario.

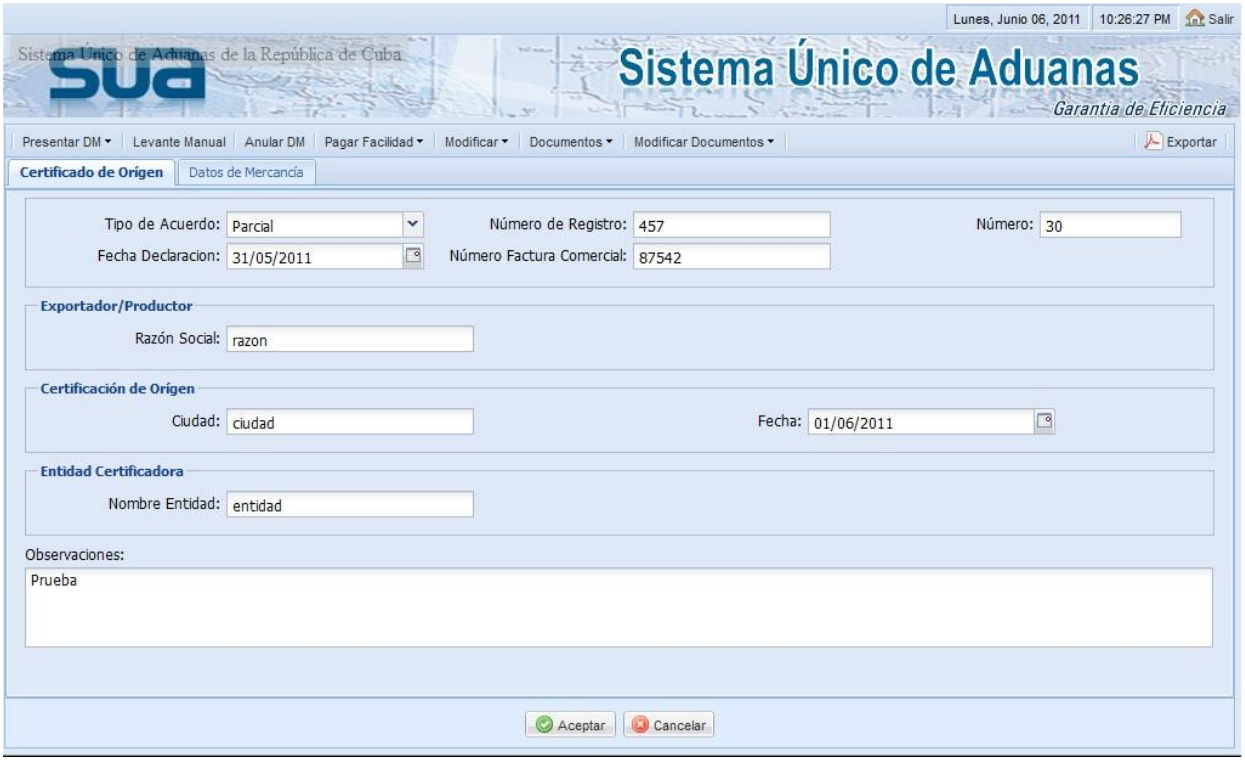


Figura 2.14 a) Pantalla con los datos correctos

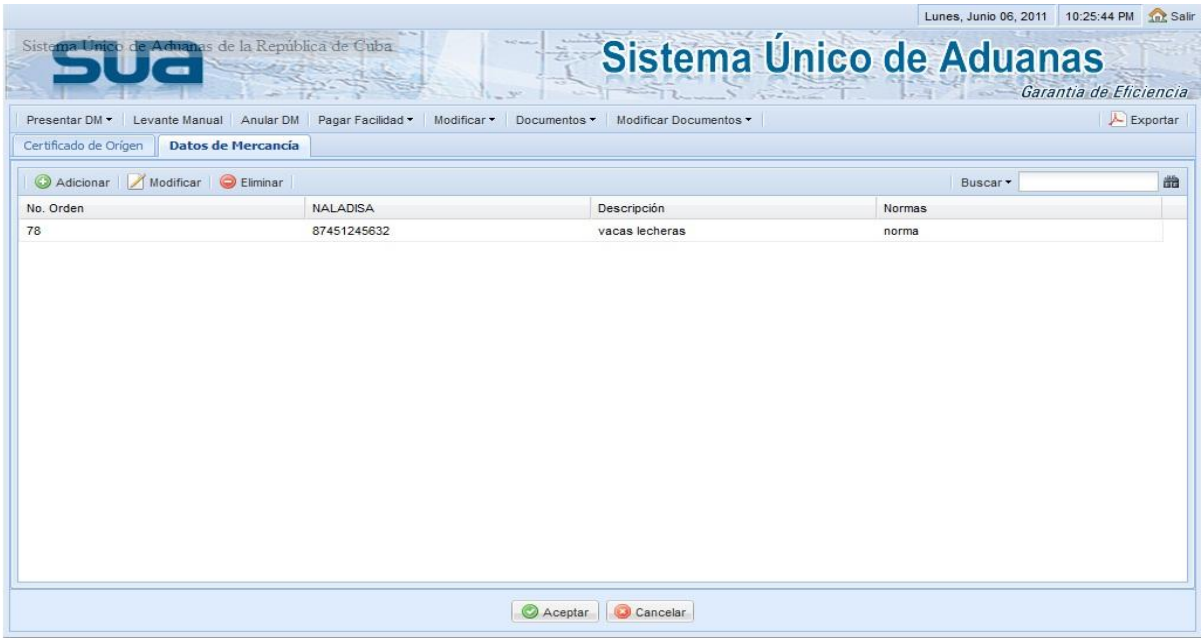


Figura 2.14 b) Pantalla con los datos correctos

```
buttons: [{
    iconCls: 'iconOk',
    text: 'Aceptar',
    handler: function(){
        if (obj.form.isValid() && Ext.getCmp('idFormGridPanel').getGrid().getStore().getCount() != 0) {
            obj.form.submit({
                url: 'sfDocumentos/registrarCertificadoOr',
                waitMsg: 'Enviando...',
                waitTitle: 'Espere por favor',
                success: function(form, action){
                    var msg = Ext.decode(action.response.responseText);
                    msg = "El Certificado Origen se ha guardado satisfactoriamente con el n&uacute;mero " + msg.numero;
                    Ext.Msg.show({
                        title: obj.title,
                        msg: msg,
                        icon: Ext.Msg.INFO,
                        buttons: Ext.Msg.OK,
                        width: 350
                    });
                   Ext.getCmp('mainPanelExt').getLayout().setActiveItem(0);
                },
                failure: function(form, action){
                    new ErroresTemporalidad({
                            data: Ext.decode(action.response.responseText).errores ||
                            []
                    }).show();
                }
            });
        }
        else {
            Ext.Msg.show({
                title: 'Error',
                msg: 'Existen campos con datos inv&aacute;lidos.',
                icon: Ext.Msg.ERROR,
                buttons: Ext.Msg.OK,
                width: 300
            });
        }
    }
```

Figura 2.15 Código de los botones Aceptar y Cancelar del formulario

Luego de ser enviados los datos al `action` de acuerdo a lo que se ve en la **figura 2.16** especificado en la clase controladora de acuerdo a lo que es representado en la **figura 2.17** por medio del método `getPostParameters`() se obtienen todos los atributos que son enviados desde la interfaz por parte del cliente. Con estos parámetros se conforma un arreglo de datos que es pasado como parámetro al método `insertarCertificadoOr()`.

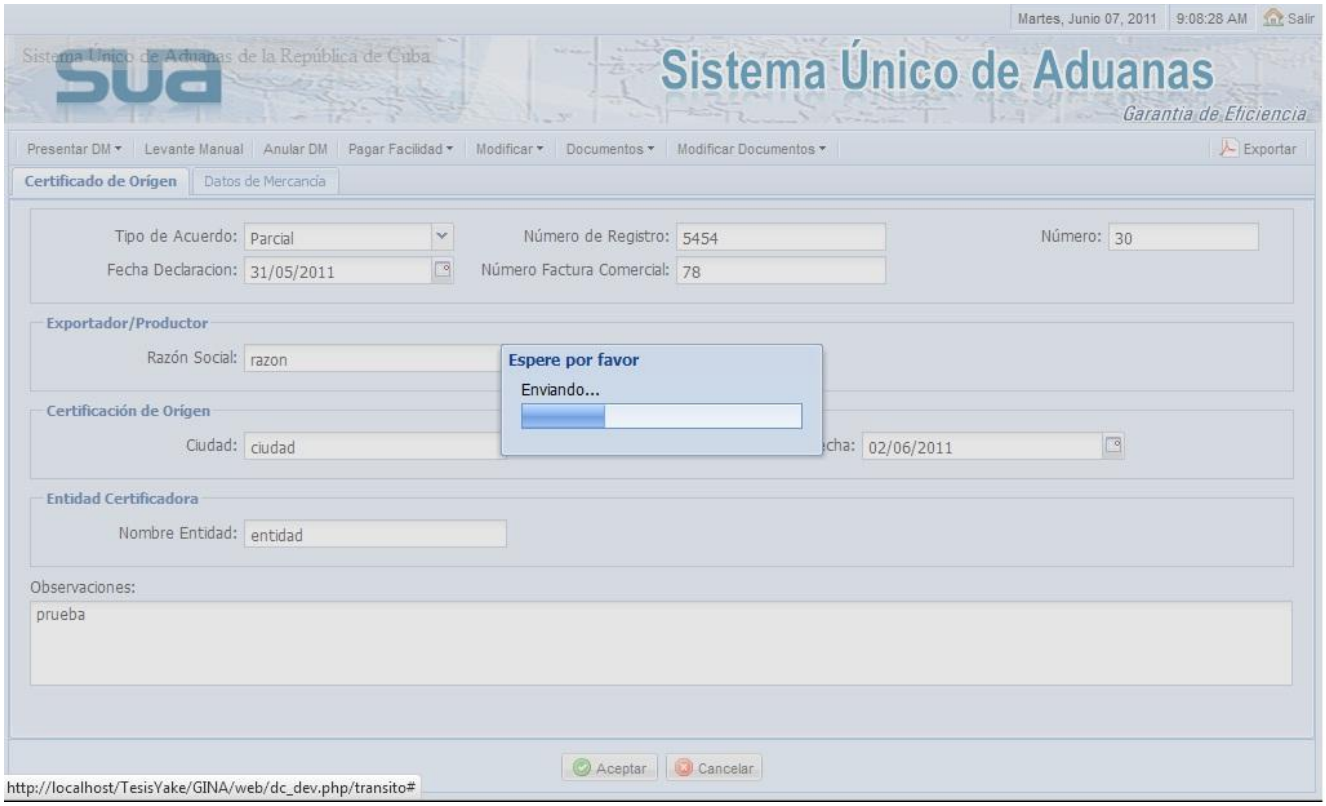


Figura 2.16 Pantalla de espera mientras se envían los datos

```php
    /**
     * Accion RegistrarCertificadoOr
     * RF:
     * Esta accion toma los datos del formulario y los inserta
     * @param sfWebRequest $request
     * @return <JSON>
     */
    public function executeRegistrarCertificadoOr(sfWebRequest $request) {
        try {
            $arregloParametrosRequest = $request->getPostParameters();
            $arregloParametros = array();
            $arregloE = array();

            //Formando el arreglo de mercancias
            $mercanciastmp = json_decode($arregloParametrosRequest['mercancia']);
            $mercancias = array();
            $cont = 0;
            foreach ($mercanciastmp as $mercanciatmp) {

                $mercancias[$cont]['noOrden'] = $mercanciatmp->noOrden;
                $mercancias[$cont]['naladisa'] = $mercanciatmp->naladisa;
                $mercancias[$cont]['descripcion'] = $mercanciatmp->descripcion;
                $mercancias[$cont]['normas'] = $mercanciatmp->normas;
                $cont+=1;
            }

            //Formando el arreglo de parametros para insertar
            $arregloParametros['numero'] = $arregloParametrosRequest['numero'];
            $arregloParametros['observaciones'] = $arregloParametrosRequest['observaciones'];
            $arregloParametros['factura'] = $arregloParametrosRequest['numeroFactura'];
            $arregloParametros['tipoacuerdo'] = $arregloParametrosRequest['id'];
            $arregloParametros['noRegistro'] = $arregloParametrosRequest['numeroRegistro'];
            $arregloParametros['fechaDeclaracion'] = $arregloParametrosRequest['fechaDeclaracion'];
            $arregloParametros['razonSocialExp'] = $arregloParametrosRequest['razon'];
            $arregloParametros['ciudadCertificacion'] = $arregloParametrosRequest['ciudad'];
            $arregloParametros['fechaCertificacion'] = $arregloParametrosRequest['fechaCertificacion'];
            $arregloParametros['nombreEC'] = $arregloParametrosRequest['nombreEC'];
            $arregloParametros['mercancias'] = $mercancias;

            $con = Propel::getConnection('gdc');
            $con->beginTransaction();
            $respuesta = GdcCertificadoOr::insertarCertificadoOr($arregloParametros);
            if ($respuesta['success']==true) {
                $con->commit();
            }
            return $this->renderText(json_encode($respuesta));
        } catch (Exception $e) {
            $con->rollBack();
            throw new Exception($e->getMessage());
        }
    }
```

Figura 2.17 Código del actionRegistrarCertificadoOr

En el método insertarCertificadoOr() perteneciente a la clase GdcCertificadoOr.php es donde son validados cada uno de los datos insertados (ver **figura 2.18**) y en caso de error se muestra el error identificado como se puede ver en la **figura 2.19**.

```
/**
 * Metodo para insertar el Certificado de Origen
 * @param <array> $arregloParametros
 * @return <array> $arregloError
 */
public static function insertarCertificadoOr($arregloParametros) {

    //Arreglo para guardar todos los errores que aparecen en la validacion de los datos

    $arregloError = array();
    $contError = 0;

    //Verificar si ya existe un Certificado Origen insertado con ese numero
    $existeDoc = GdcDocComp::seleccionarDocumentos($arregloParametros['numero'], 'Certificado Origen');

    if (empty($existeDoc)) {

        //Buscar el id del Tipo de Documento
        $datosTipoDoc = GdcDocComp::seleccionarTipoDoc('Certificado Origen');

        //Buscar el id del Estado del Documento
        $datosEstado = GdcDocComp::seleccionarEstadoDoc('001');

        //Tomando los valores de DocComp
        $valoresDocComp = array();
        $valoresDocComp['idTipoDocumento'] = $datosTipoDoc[0]->getIdTipoDocumento();
        $valoresDocComp['numero'] = $arregloParametros['numero'];
        $valoresDocComp['fechaSolicitud'] = Date('m/d/Y');
        $valoresDocComp['fechaVigencia'] = $arregloParametros['fechaCertificacion'];
        $valoresDocComp['observaciones'] = $arregloParametros['observaciones'];
        $valoresDocComp['idEstadoDoc'] = $datosEstado[0]->getIdEstadoDoc();
        //Guardando en el Formulario de DocComp
        $docCompForm = new GdcDocCompForm();
        $docCompForm->bind($valoresDocComp);
        if (!$docCompForm->isValid()) {
            $error = $docCompForm->getErrorSchema();
            $arregloError[$contError]['msg'] = $error->getMessage();
            $arregloError[$contError]['grupo'] = 'Dato Documento Complementario';
            $contError++;
        } else {
            $docCompForm->updateObject($valoresDocComp);
            $docCompForm->getObject()->save();
            //Tomando los valores de Certificado Origen
            $valoresCertificado = array();
            $valoresCertificado['idDocComp'] = $docCompForm->getObject()->getIdDocComp();
            $valoresCertificado['numero'] = $arregloParametros['numero'];
            $valoresCertificado['noFacturaCom'] = $arregloParametros['factura'];
            $valoresCertificado['tipoAcuerdo'] = $arregloParametros['tipoacuerdo'];
            $valoresCertificado['noRegistro'] = $arregloParametros['noRegistro'];
            $valoresCertificado['fecha'] = $arregloParametros['fechaDeclaracion'];
            $valoresCertificado['razonSocialExp'] = $arregloParametros['razonSocialExp'];
            $valoresCertificado['selloExp'] = 'selloExp';
            $valoresCertificado['firmaExp'] = 'firmaExp';
            $valoresCertificado['observaciones'] = $arregloParametros['observaciones'];
            $valoresCertificado['ciudadCertificacion'] = $arregloParametros['ciudadCertificacion'];
            $valoresCertificado['fechaCertificacion'] = 'fechaVig';
            $valoresCertificado['nombreEc'] = $arregloParametros['nombreEC'];
            $valoresCertificado['selloEC'] = 'selloEC';
            $valoresCertificado['firmaEC'] = 'firmaEC';

            //Guardando en el formulario de Certificado Origen

            $certificadoOrForm = new GdcCertificadoOrForm();
            $certificadoOrForm->bind($valoresCertificado);

            if (!$certificadoOrForm->isValid()) {
                $error = $certificadoOrForm->getErrorSchema();
                $arregloError[$contError]['msg'] = $error->getMessage();
                $arregloError[$contError]['grupo'] = 'Dato Certificado Origen';
```

```
                    $contError++;
                } else {
                    $certificadoOrForm->updateObject($valoresCertificado);
                    $certificadoOrForm->getObject()->save();
                    $existeFactura = GdcDocComp::existeFactura($arregloParametros['factura']);
                    if (!empty($existeFactura)) {
                        $idFactura = $existeFactura[0]->getIdFactura();
                    } else {
                        //Guardando en el formulario de factura
                        $valoresFactura = array();
                        $valoresFactura['numero'] = $arregloParametros['factura'];
                        $facturaForm = new GdcFacturaForm();
                        $facturaForm->bind($valoresFactura);
                        if (!$facturaForm->isValid()) {
                            $error = $facturaForm->getErrorSchema();
                            $arregloError[$contError]['msg'] = $error->getMessage();
                            $arregloError[$contError]['grupo'] = 'Dato Factura';
                            $contError++;
                        } else {

                            $facturaForm->updateObject($valoresFactura);
                            $facturaForm->getObject()->save();
                            $idFactura = $facturaForm->getObject()->getIdFactura();
                        }
                    }
                    //Tomando el arreglo de mercancias
                    $arregloMercancias = $arregloParametros['mercancias'];
                    //Buscando el id del Tipo Embalaje
                    $datosEmbalaje = GdcDocComp::seleccionarTipoEmbalaje('Ninguno');
                    $contMercancia = 1;
                    foreach ($arregloMercancias as $mercancia) {
                        //Buscando el id de Mercancia Tarifable
                        $datosMerTarifable = GdcDocComp::seleccionarMerTarifable($mercancia['descripcion']);
                        if (empty($datosMerTarifable)) {
                            $arregloError[$contError]['msg'] = 'La descripcion '.$datoMercancia['descripcion'].'
de la Mercancia '.$contMercancia.' es incorrecta';
                            $arregloError[$contError]['grupo'] = 'Mercancia '.$contMercancia;
                            $contMercancia++;
                            $contError++;
                        }else{
                            //Tomando los valores de mercancia
                        $valoresMercancia = array();
                        $valoresMercancia['idTipoEmbalaje'] = $datosEmbalaje[0]->getIdTipoEmbalaje();
                        $valoresMercancia['noOrden'] = $mercancia['noOrden'];
                        $valoresMercancia['idFactura'] = $idFactura;
                        $valoresMercancia['clasificacionArancel'] = $mercancia['naladisa'];
                        $valoresMercancia['criterio'] = $mercancia['normas'];
                        $valoresMercancia['idMercanciaTarifable'] = $datosMerTarifable[0]->getIdMercancia();
                        //Guardando en el formulario de Mercancia
                        $mercanciaForm = new GdcDatoMercanciaForm();
                        $mercanciaForm->bind($valoresMercancia);
                        if (!$mercanciaForm->isValid()) {
                            $error = $mercanciaForm->getErrorSchema();
                            $arregloError[$contError]['msg'] = $error->getMessage();
                            $arregloError[$contError]['grupo'] = 'Mercancia '.$contMercancia;
                            $contError++;
                        } else {
                            $mercanciaForm->updateObject($valoresMercancia);
                            $mercanciaForm->getObject()->save();
                            $contMercancia++;
                        }
                        //Tomo los id de Doc Complementario y de Mercancia
                        $valoresDocMarcancia = array();
                        $valoresDocMarcancia['idMercancias'] = $mercanciaForm->getObject()->getIdMercancia();
                        $valoresDocMarcancia['idDocComp'] = $docCompForm->getObject()->getIdDocComp();
                        //Guardando en el formulario de Doc Complementario y Mercancia
                        $docMercanciaForm = new GdcrDocCompDatoMerForm();
                        $docMercanciaForm->bind($valoresDocMarcancia);
                        if (!$docMercanciaForm->isValid()) {
```

```
                        $error = $docMercanciaForm->getErrorSchema();
                        $arregloError[$contError]['msg'] = $error->getMessage();
                        $arregloError[$contError]['grupo'] = 'Documento y Mercancia ' + $contMercancia;
                        $contError++;
                    } else {
                        $docMercanciaForm->updateObject($valoresDocMarcancia);
                        $docMercanciaForm->getObject()->save();
                    }
                    $contMercancia++;
                    }
                }
            }
        }
    } else {
        $arregloError[$contError]['msg'] = 'Ya existe un Certificado Origen con ese numero';
        $arregloError[$contError]['grupo'] = 'Certificado Origen';
        $contError++;
    }

    if (empty($arregloError)) {
        $respuesta['success'] = true;
        $respuesta['numero'] = $arregloParametros['numero'];
    } else {
        $respuesta['errores'] = $arregloError;
        $respuesta['success'] = false;
    }
    return $respuesta;
}
```

Figura 2.18 Código del método `insertarCertificadoOr()`

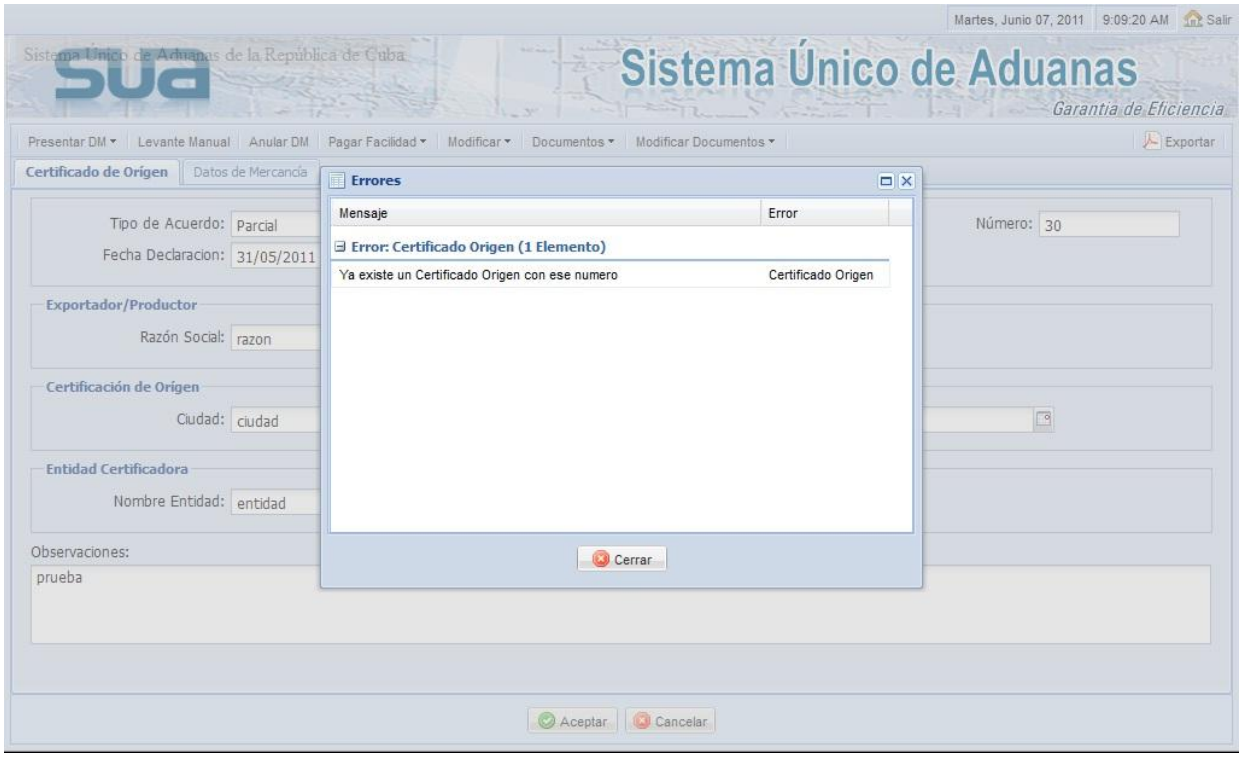


Figura 2.19 Pantalla para mostrar los errores

De lo contrario se guardan los valores en cada una de las tablas relacionadas a través de los formularios generados por symfony de acuerdo a lo que se ve en la **figura 2.18** y finalmente se muestra un mensaje al usuario comunicando el número con el que fue guardado el documento en la BD como se muestra en la **figura 2.20.**

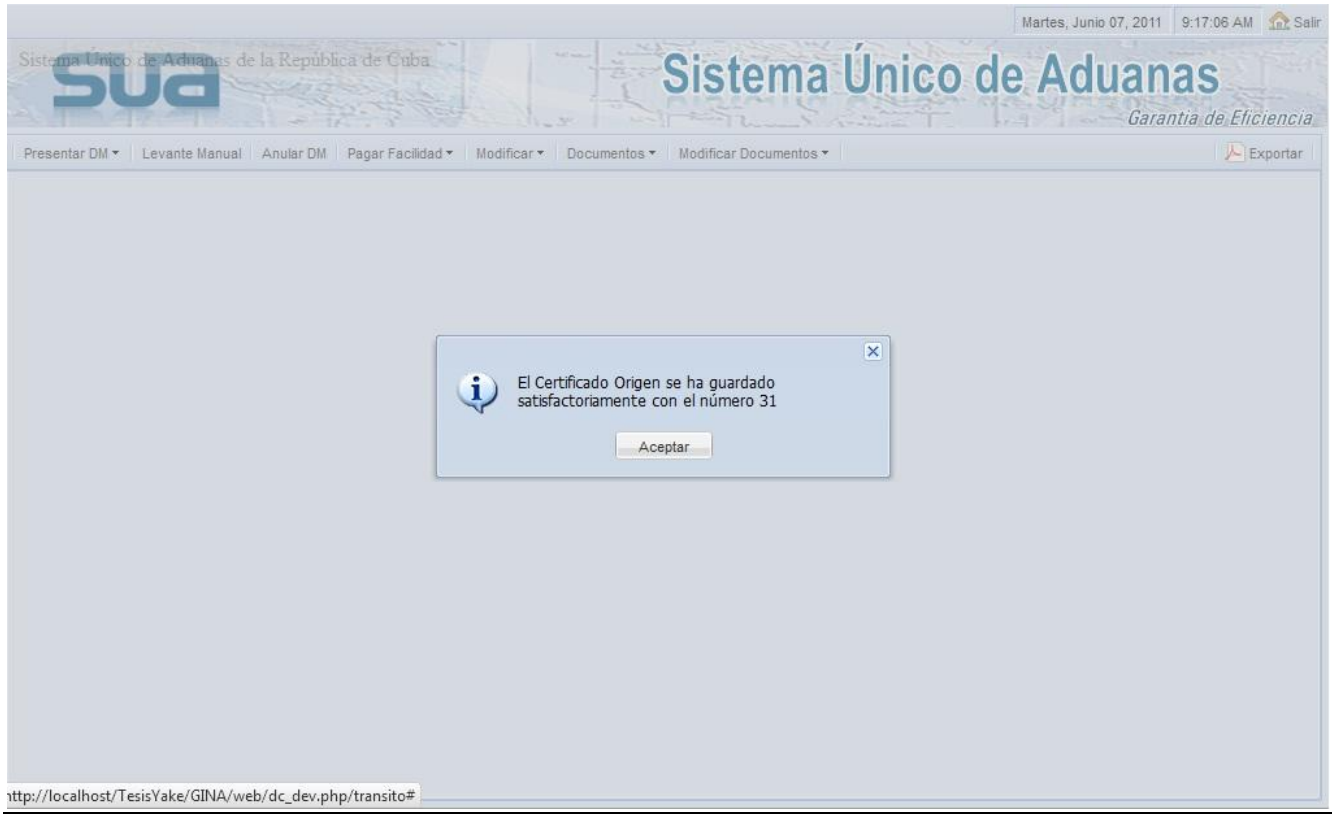


Figura 2.20 Pantalla mostrando el número con el que se guardó el Certificado Origen

2.5 Resumen de los resultados de la implementación

A continuación se realiza una descripción de los aspectos más importantes del componente Gestionar Documentos Complementarios en materia de implementación.

En total fueron implementados 41 funcionalidades, de estos 7 pertenecen a la clase controladora actions, estos son los que acceden a las funcionalidades implementadas en cada una de las clases del negocio generadas por symfonydonde están distribuidas las funcionalidades restantes. Para poder guardar y modificar la información en la base de datos se generaron formularios de cada una de las clases.

El envío y recepción de información desde la interfaz a la clase controladora fue realizada a través de tecnología JSON. Todos los errores identificados, durante cualquier acción realizada por el usuario,son guardados en un arreglo y retornados en conjunto, en vez de ser retornados uno a uno para que fuese más factible realizar todos los cambios necesarios y ganar en tiempo.

Se utilizó el servicio obtenerDatoCriteria perteneciente al subsistema TC mediante el cual se obtienen los datos almacenados en la Base de Datos de cualquier TC que se referencie. También se usó el servicio insertarContacto del pluginsfContactoPlugin para insertar los contactos correspondientes al solicitante del Permiso y Liberación. El pluginsfeAduanaPlugin fue utilizado para manejar y validar los datos que se recibían a través de los documentos XML, haciendo uso de los XSD generados para cada uno de los documentos recibidos por esta vía.

En el componente se generaron dos servicios recibirListaEmpaque implementado en la clase GdcListaEmpPeer y recibirPermiso implementado en la clase GdcPermisoPeer, estos servicios son los encargados de recibir los datos contenidos en los documentos XML que son entregados, luego de haber sido validados por parte del plugineAduana y acto seguido insertarlos en la BD.

2.6 Conclusiones parciales

En el presente capítulo fueron expuestos los diagramas más importantes para una mejor compresión del diseño y la implementación del presente sistema, son presentados los diagramas de clases de negocio, de interacción específicamente los de secuencia y el diagrama de paquetes, se muestra además el diseño de la BD y el diagrama de componentes. También son explicados algunos elementos de la implementación como el tratamiento de errores, los estándares de codificación y la comunicación entre capas que es explicada con un ejemplo práctico del sistema.

CAPÍTULO 3. **ANÁLISIS DE LA SOLUCIÓN**

3.1 Introducción

En el presente capítulo sonabordadas las métricas que son usadas para corroborar que el diseño propuesto cumple con toda la serie de requisitos de calidad necesarios para que pueda ser considerado adecuado y factible para el proceso de implementación.

Además es validada la solución propuesta mediante la realización de pruebas funcionales al sistema desarrollado con el propósito de lograr determinar si el mismo responde,funcionalmente, con todos los requisitos funcionales identificados durante el flujo de análisis y se analizan los resultados obtenidos para valorar si responden todas las exigencias solicitadas por el cliente.

3.2 Métricas para el software

En la ingeniería de software se ha vuelto extremadamente esencial comprobar si cada uno de los artefactos que son generados en la búsqueda de una solución son factibles, a causa de esto es importante aplicar técnicas que permitan obtener resultados cuantificados que evidencie si se ha tomado el camino correcto.

Las métricas que existen para el software se aplican con el objetivo de centralizar las funcionalidades operativas de un programa. Como la calidad es el principal factor para poder determinar objetivamente si se ha obtenido un producto que cumpla con las características que debe tener toda solución funcional, estas métricas para el software se especializan en cada una de las fases por las que atraviesa la aplicación informática, basadas en los principios generales que referencian la calidad con la que se ha estructurado los diferentes artefactos.

3.2.1 Métricas propuestas por Lorenz y Kidd

Este grupo de métricas muy conocidas fueron propuestas por los ingenieros Mark Lorenz y Jeff Kidd en el año 1994 y están divididas en categorías y enfocadas en el código. También están basadas en las clases como base para obtener métricas especializadas, las definiciones a las que están dirigidas en su concepción son:

- Tamaño
- Herencia

- Valores internos
- Valores Externos

Del conjunto de técnicas propuestas por Lorenz y Kidd fueron seleccionadas para validar el diseño:

- Tamaño operacional de las clases (TOC)
- Relaciones entre clases (RC)

3.2.1.1 Métrica Tamaño operacional de las clases

Esta es una métrica muy usada por los diseñadores de software que consta con una amplia documentación y literatura, es fácil de calcular y bastante efectiva. Se basa en el número de métodos asignados a una clase y evalúa los atributos de calidad que son mostrados en la **tabla 3.1**.

Atributos de Calidad	Modo de afectación
Responsabilidad	El aumento del TOC implica el aumento de la responsabilidad de la clase.
Complejidad de implementación	El aumento del TOC implica el aumento de la complejidad de implementación de la clase.
Reutilización	El aumento del TOC implica la disminución de la reutilización de la clase.

Tabla 3.1 Descripción de los atributos de calidad de la métrica TOC

Con un total de 15 clases pertenecientes al diseño se obtienen tras aplicar la métrica valores promedio de 2,73 operaciones por clases (ver **tabla 3.2**).

Total de clases	Promedio de operaciones
15	2,73

Tabla 3.2 Promedio de operaciones por clases

Para poder realizar una evaluación de las métricas es necesario conocer los valores de los umbrales para evaluar los atributos de calidad. Los umbrales aplicados en el diseño fueron los propuestos por algunos especialistas para esta métrica que es basarse en el promedio de operaciones obtenido por clases, cada uno de los umbrales son descritos a continuación.

Atributo de Calidad	Categoría	Criterio
Responsabilidad	Baja	<=Promedio (<= 2,73)
	Media	>Promedio y <= 2*Promedio (> 2,73 y <= 5,46)
	Alta	>2*Promedio (> 5,46)

Tabla 3.4 Valores de los umbrales para la responsabilidad

Atributo de Calidad	Categoría	Criterio
Complejidad de implementación	Baja	<=Promedio (<= 2,73)
	Media	>Promedio y <= 2*Promedio (> 2,73 y <= 5,46)
	Alta	>2*Promedio (> 5,46)

Tabla 3.5 Valores de los umbrales para la complejidad de implementación

Atributo de Calidad	Categoría	Criterio
Reutilización	Baja	>2*Promedio (> 5,46)
	Media	>Promedio y <= 2*Promedio (> 2,73 y <= 5,46)
	Alta	<=Promedio (<= 2,73)

Tabla 3.6 Valores de los umbrales para la reutilización

Luego de representar las clases en correspondencia con el umbral especificado se obtuvo como resultado que 13 de ellas son de tamaño pequeño, 1 mediana y 1 grande, esto puede ser visualizado en la **figura 3.1**:

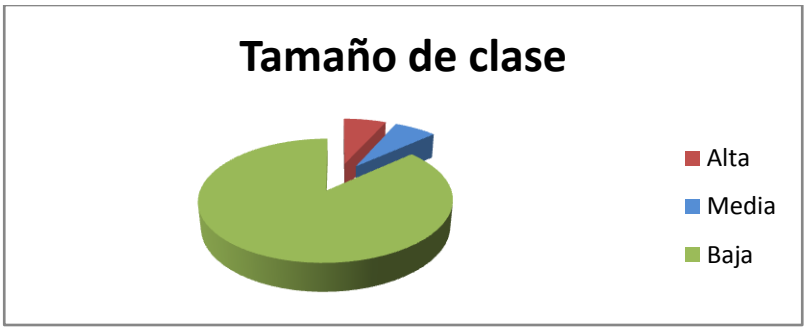


Figura 3.1 Tamaño de clase

A continuación se muestran los resultados obtenidos al aplicar los umbrales definidos para cada uno de los atributos de calidad.

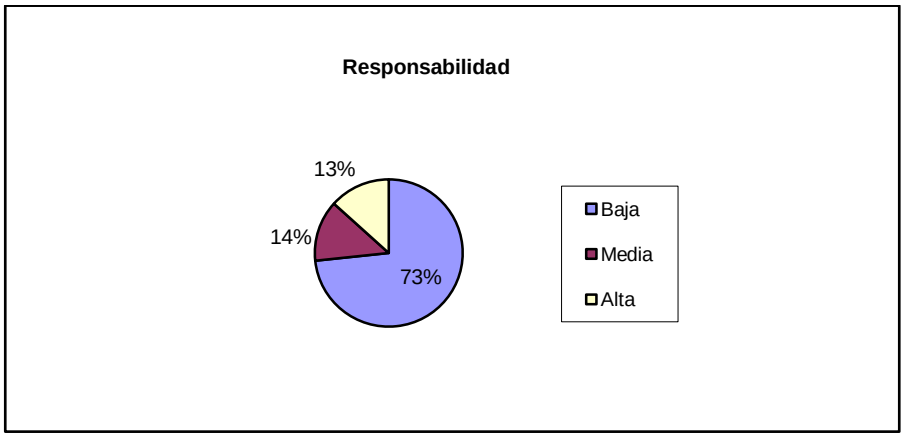


Figura 3.2 Resultado de la responsabilidad en clases

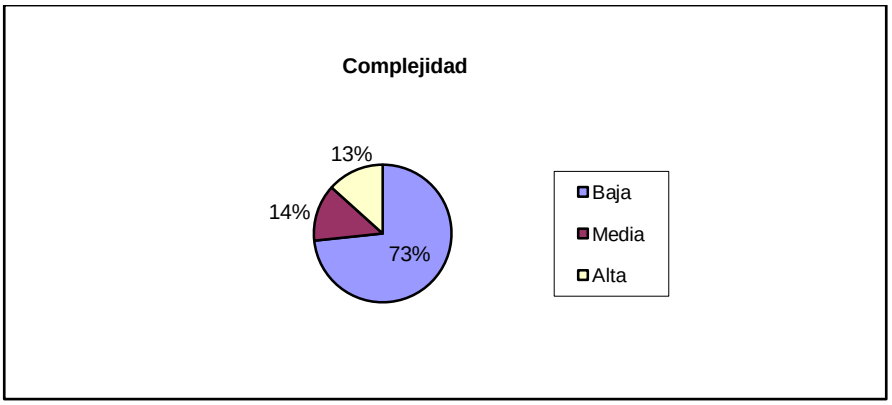


Figura 3.3Resultado de la complejidad de implementación en clases

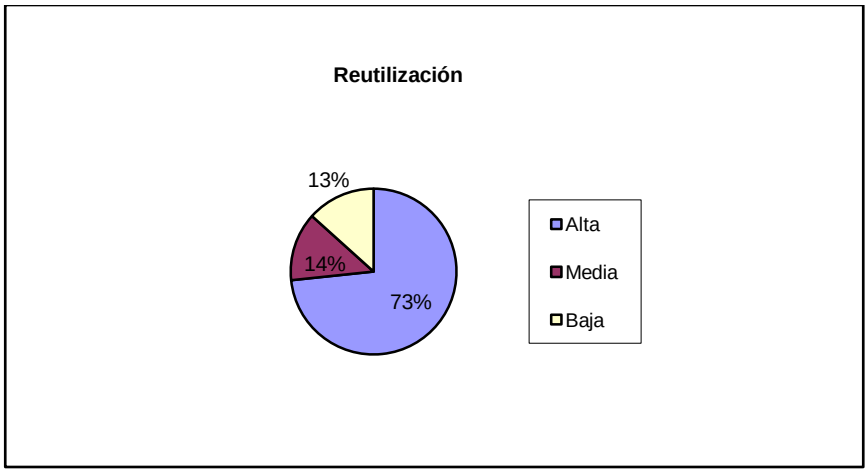


Figura 3.4Resultado de la reutilización en clases

Analizando los resultados obtenidos bajo la medición de la métrica TOC se puede observar que un 73% de las clases presentan valores bajos de responsabilidad, lo que hace que carezcan de mucha complejidad a la hora de la implementación y por lo tanto se tornan más reutilizables. Sólo un 13% de las clases muestran valores altos de responsabilidad y complejidad lo que incurre en un bajo grado de reutilización de las mismas.

A partir de los valores obtenidos se puede concluir que el diseño realizado es efectivo ya que va a permitir, en el mayor porciento de sus clases,una baja responsabilidad evitando darle a la mayoría de las clases demasiada responsabilidad lo que aumenta la reutilización de las mismas, logrando evitar a la vez un alto porciento de complejidad en la implementación, factor que puede impedir el correcto funcionamiento de un diseño.

3.2.1.2 Métrica Relaciones entre clases

Esta métrica trata de encontrar todas las relaciones que existan entre la clase A con una o varias clases B. Estas relaciones se definen por los atributos o métodos que utilice la clase A de la clase B al menos una vez para darle respuesta a alguna petición.

En el proceso de aplicación de esta métrica fue necesario definir de todas las clases persistentes del diseño, las que se apoyan en las operaciones o datos que no se encuentran dentro de sus responsabilidades.

Por medio de esta métrica se evalúan los atributos de calidad que se describen a continuación:

Atributos de Calidad	Modo de afectación
Acoplamiento	El aumento del RC implica el aumento del acoplamiento de la clase.
Complejidad de mantenimiento	El aumento del RC implica el aumento de la complejidad de mantenimiento de la clase.
Reutilización	El aumento del RC implica la disminución de la reutilización de la clase.

Tabla 3.7Descripción de los atributos de calidad de la métrica RC

Para un total de 15 clases pertenecientes al diseño se obtienen tras aplicar la métrica valores promedio de 2,8 operaciones por clases (**ver tabla 3.8**).

Total de clases	Promedio de asociaciones de uso
15	2,8

Tabla 3.8 Promedio de asociaciones de uso por clases

Los valores de los umbrales para aplicar esta métrica se basan en el promedio de asociaciones de uso por clases y se describen a continuación.

Atributo de Calidad	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2

Tabla 3.9 Valores de los umbrales para el acoplamiento

Atributo de Calidad	Categoría	Criterio
Complejidad de mantenimiento	Baja	<=Promedio (<= 2,8)
	Media	>Promedio y <= 2*Promedio (> 2,8 y <= 5,6)
	Alta	>2*Promedio (> 5,6)

Tabla 3.10 Valores de los umbrales para la complejidad de mantenimiento

Atributo de Calidad	Categoría	Criterio
Reutilización	Baja	>2*Promedio (> 5,6)
	Media	>Promedio y <= 2*Promedio (> 2,8 y <= 5,6)
	Alta	<=Promedio (<= 2,8)

Tabla 3.11 Valores de los umbrales para la reutilización

Luego de ser evaluadas cada una de las clases en correspondencia con los valores de los umbrales definidos para cada uno de los atributos de calidad se obtienen los siguientes resultados:

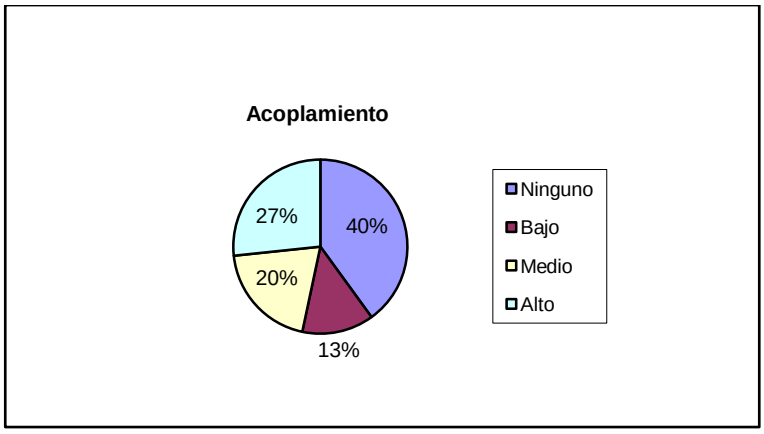


Figura 3.5Resultado del acoplamiento en clases

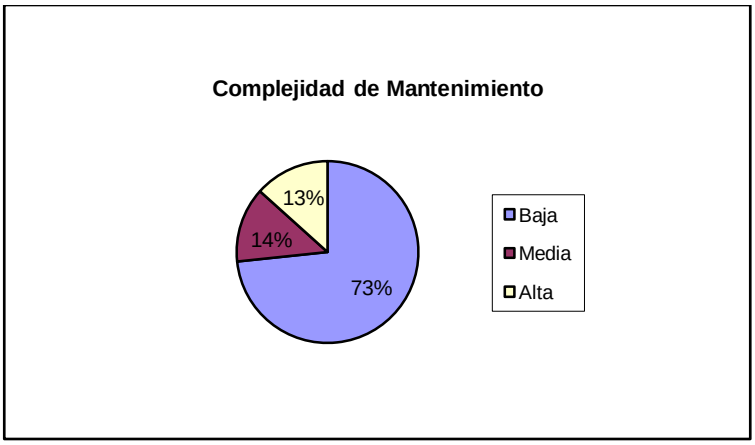


Figura 3.6Resultado de la complejidad de mantenimiento en clases

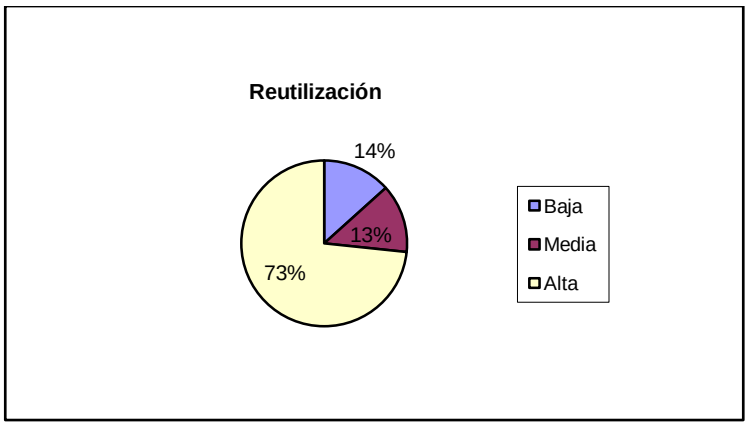


Figura 3.7Resultado de la reutilización en clases

La evaluación de los resultados obtenidos durante la aplicación de la métrica RC, demuestra que el diseño propuesto no presenta problemas para poder realizar un adecuado funcionamiento. Al lograrse un valor de acoplamiento de un 73% que oscila entre bajo y medio, lo que representa un valor aceptable desde el punto de vista de implementación del software, además la complejidad de mantenimiento es baja a un 73%, dado estos valores es posible la reutilización de las operaciones con un valor alto de 73%, lo que representa un aspecto clave para mejorar la obtención de respuestas.

3.3 Pruebas de Software

Las pruebas de software son actividades en las cuales un sistema o componente es ejecutado bajo condiciones y requerimientos específicos donde los resultados son observados y registrados y se realiza una evaluación a algún aspecto del sistema o componente. Se puede decir que las pruebas de software son un elemento crítico para garantizar la calidad del software y representan una revisión final de las especificaciones del diseño y de la codificación.

La eminente importancia que ha adquirido el software como componente de varios sistemas y todos los posibles costos que acarrea un fallo del mismo, motivan la creación de pruebas más minuciosas y bien planificadas.

Es bueno recalcar que el proceso de pruebas no va orientado a demostrar la ausencia de fallos en el software, sino más bien a encontrar cuantos fallos existan, por escondidos que se encuentren o difícilmente reproducibles que sean.

3.3.1 Pruebas de Caja Negra

Las pruebas de software son aplicadas en diferentes niveles, dentro de estos niveles entra la prueba de unidad que son las pruebas enfocadas principalmente a los elementos verificables del software, dentro de estas pruebas están las llamadas pruebas de caja negra.

Las pruebas de caja negra son aquellas pruebas que se llevan a cabo sobre la interfaz del software. Con este tipo de pruebas se pretende demostrar que:

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

Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener en cuenta la estructura interna del software, se centra principalmente en los requisitos funcionales del sistema.

Los problemas que nos permiten detectar este tipo de pruebas son los siguientes:

- Funciones incorrectas o ausentes.

- Errores de interfaz.
- Errores en la estructura de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Las pruebas realizadas a la solución propuesta fueron Pruebas de Funcionalidad que se encuentran contenidas dentro de las pruebas de caja negra. El objetivo principal de este tipo de pruebas es lograr localizar fallas funcionales dentro de sistema e identificar situaciones en las cuales la respuesta del sistema no se apega a las especificaciones establecidas.

Estas pruebas se realizan a través de casos de usos de pruebas insertando valores válidos e inválidos para verificar que:

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

3.3.1.1 Casos de Pruebas

Los casos de pruebas son un conjunto de condiciones o variables con las que se determina si el requisito de una aplicación es parcial o completamente satisfactorio.

Existen muchos casos de prueba para determinar que un requisito es satisfactorio, para poder comprobar que todos los requisitos de una aplicación fueron revisados debe existir un caso de prueba para cada requisito y si el requisito tiene requisitos secundarios se debe hacer un caso de prueba para cada uno de los requisitos secundarios.

Durante la realización de las pruebas al sistema se realizaron 6 Casos de Pruebas (uno para cada requisito funcional) con cada uno de sus Escenarios de Prueba (EP), a continuación se muestran cada uno de estos casos de prueba:

1. Caso de Prueba Registra Lista de Empaque por XML.

- EP 1.1: Registrar Lista de Empaque por XML.
- EP 1.2: Validar datos XML.

2. Caso de Prueba Registrar Permisos y Liberaciones por XML.
    - EP 2.1: Registrar Permisos y Liberaciones por XML.
    - EP 2.2: Validar datos XML.
3. Caso de Prueba Registrar Certificado Origen.
    - EP 3.1: Registrar Certificado Origen.
    - EP 3.2: Registrar datos del Certificado Origen.
4. Modificar Permisos y Liberaciones.
    - EP 4.1: Modificar Permisos y Liberaciones.
    - EP 4.2: Registrar datos para buscar el Permisos y la Liberación.
    - EP 4.3: Buscar Permiso y Liberación.
    - EP 4.4: Modificar datos de la Mercancía, 1 no conformidad.
5. Modificar Certificado Origen.
    - EP 5.1: Modificar Certificado Origen.
    - EP 5.2: Registrar datos para buscar el Certificado Origen.
    - EP 5.3: Buscar Certificado Origen.
    - EP 5.4: Modificar datos de la Mercancía, 2 no conformidades.
6. Anular Documentos Complementarios.
    - EP 6.1: Anular Documentos Complementarios.

- EP 6.2: Registrar datos.

Se realizó una primera iteración donde se pusieron en práctica cada uno de los casos de pruebas con sus respectivos escenarios, en esta iteración fueron identificadas un total de 3 no conformidades que se enfocan en la validación de los datos que son introducidos por parte de usuario, estas no conformidades fueron solucionadas de inmediato. Luego se procedió a realizar una segunda iteración para identificar cualquier otra no conformidad que pudiese existir. En la **figura 3.8** se pueden observar los resultados obtenidos en ambas iteraciones.

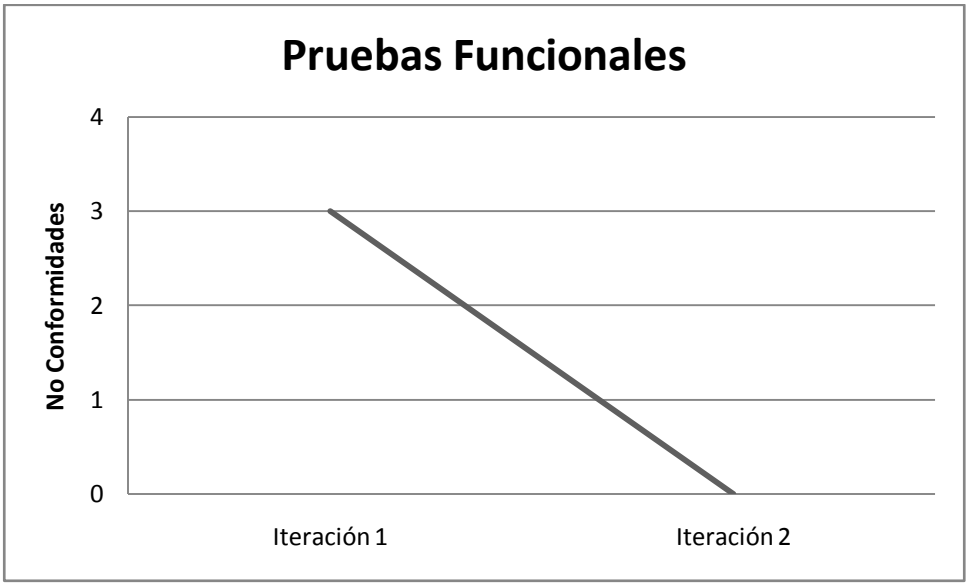


Figura 3.8 Iteraciones de las pruebas funcionales realizadas

Dado los resultados obtenidos en la segunda iteración el sistema quedó en total funcionamiento y acorde a las necesidades funcionales requeridas por el cliente.

3.3.1.2 Ejemplo de un Caso de Prueba

A continuación en la **tabla 3.12** se muestra el caso de prueba que fue aplicado al RF Registrar Certificado Origen.

Id del escenario	Escenario	Número Registro	Número Factura Comercial	Número	Razón Social	Ciudad	Nombre entidad	Observación	No. Orden	NALADISA	Descripción	Normas	Respuesta del sistema	Resultado de la prueba
EP 3.1	Registrar Certificado Origen	V	V	V	V	V	V	V	V	V	V	V	Registrar el documento complementario y notificar el número con el que fue guardado.	Se guarda la información y se notifica el número del documento complementario.
EP 3.2 Registrar datos del Certificado Origen		I	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el error.
		NA	I	NA	NA	NA	NA	NA	NA	NA	NA	NA	Se validan los datos según el escenario.	El sistema no reconoce que la longitud de la

													cadena es mayor que 35.
	NA	NA	I	NA	NA	NA	NA	NA	NA	NA	NA	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el error.
	NA	NA	NA	I	NA	NA	NA	NA	NA	NA	NA	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el error.
	NA	NA	NA	NA	I	NA	NA	NA	NA	NA	NA	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el error.
	NA	NA	NA	NA	NA	I	NA	NA	NA	NA	NA	Se validan los datos según el	El sistema notifica que

												escenario.	existen campos con datos inválidos especificando el error.
	NA	NA	NA	NA	NA	NA	I	NA	NA	NA	NA	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el error.
	NA	NA	NA	NA	NA	NA	NA	I	NA	NA	NA	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el error.
	NA	NA	NA	NA	NA	NA	NA	NA	I	NA	NA	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el

													error.
	NA	NA	NA	NA	NA	NA	NA	NA	NA	I	NA	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el error.
	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	I	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el error.

Tabla 3.12 Caso de Prueba Registrar Certificado Origen

Como se pudo ver en el caso de prueba anterior fueron probados cada uno de los valores a insertaren correspondencia con los escenarios de prueba definidos, de esta misma forma se procedió con los 6 casos de pruebas definidos en las 2 iteraciones realizadas.

3.3 Conclusiones Parciales

En este capítulo se realizó un análisis de los resultados obtenidos para la validación del diseño propuesto para la aplicación, donde a través de la aplicación de métricas se pudo ver que este diseño cumple con todos los requisitos necesarios para ser considerado un diseño funcional. También fueron explicadas y analizadas la realización de las pruebas de caja negra,estas fueron aplicadas a través de diferentes casos de pruebas que cubrían cada una de las funcionalidades del sistema, acción que permitió identificar errores que obstaculizaban un correcto funcionamiento de la aplicación e inmediatamente estos fueron corregidos permitiendo que el sistema se encuentre en un 100% de su total funcionamiento.

CONCLUSIONES

Al finalizar el trabajo presente se puede afirmar que las tareas de investigación propuestas para el mismo fueron cumplidas exitosamente, contribuyendo a la resolución del problema planteado a través del desarrollo de una la solución informática acorde a las necesidades y exigencias actuales de la AGR. Se realizó un análisis teórico de los principales sistemas automatizados para Aduanas existentes en el mundo estableciendo comparaciones entre ellos. Fueron expuestas las herramientas y metodologías como base para el desarrollo del sistema.

Para la modelación de la solución informática se generaron los artefactos necesarios, así como su grado de detalle en cada una de las fases y flujos de trabajo establecidos, que contribuyeron a la clarificación y comprensión de los modelos elaborados en el trabajo.

Se definió un diseño que cumple con todos los requisitos funcionales definidos para el sistema.

Cada una de las funcionalidades del sistema fueron validadas mediante las pruebas de caja negra, identificando errores que dificultaban el correcto funcionamiento de la aplicación, dichos errores fueron corregidos permitiendo la eficiencia requerida y confiabilidad deseada.

RECOMENDACIONES

Luego de cumplir con cada una de las tareas de la investigación y partiendo de la experiencia acumulada en el desarrollo de la aplicación se proponen las siguientes recomendaciones:

- La implementación del resto de los documentos complementarios una vez que sean estandarizados cada uno de sus datos y su representación.
- La utilización del componente en todos los módulos dentro del sistema GINA que lo precisen.
- Continuar con la investigación de nuevas tecnologías informáticas que garanticen mejoras en futuras versiones del Componente para Gestionar Documentos Complementarios en los procesos aduaneros del sistema GINA.

BIBLIOGRAFÍA

1. **Centro Nacional de Tecnologías de Información.** www.gobiernoenlinea.ve. *www.gobiernoenlinea.ve.* [En línea] [Citado el: 9 de febrero de 2011.] http://www.gobiernoenlinea.ve/cartelera/Sidunea.html.

2. **Miquel, Cristian Maturana.** Los Medios de Prueba. s.l. : Departamento de Derecho Procedual, Facultad de Derecho, Universidad de Chile, 2003.

3. **Abarca, Lorena Donoso.** www.derechoinformatico.uchile.cl. [En línea] Mayo de 2004. [Citado el: 9 de febrero de 2011.] http://www.derechoinformatico.uchile.cl/CDA/der_informatico_simple/0,1493,SCID%253D15836%2526ISID%253D567%2526PRT%253D15830,00.html. 4.

4. **Piñones, Ariel rementeria.***Concepto y alcance de Gestión.* Ciudad Habana : s.n., 2002.

5. **Cassal, Velmour Muñoz.***Sistema de Gestión Documental.* Ciudad Habana : s.n., 2007.

6. **Rodríguez, Jose Antonio Cobo.** Línea base arquitectónica para el polo Sistemas Tributarios de Aduana. Trabajo de Curso. Ciudad Habana : Universidad de las Ciencias Informáticas, 2008.

7. **JACOBSON, I. y RUMBAUCH, J.** El Proceso Unificado de Desarrollo de Software. Addison Wesley. 2000. 84-7829-036-2..

8. **Optimsoft Cía. Ltda. .** www.todocomercioexterior.com.ec. [En línea] diciembre 2010. http://www.todocomercioexterior.com.ec/pv_principal1.asp?pid=11.

9. **Ministerio de Comercio, Industria y Turismo.** www.vuce.gov.co. [En línea] diciembre 2010.http://www.vuce.gov.co/.

10. **AGEXPORT.** www.export.com.gt. [En línea] diciembre 2010.http://www.export.com.gt/Portal/Home.aspx?sub=VUPE.

11. **Tec. Prof.Dennys López Dinza, Lic. Susana Enríquez Domínguez.** SISTEMA DE CERTIFICACIÓN DE ORIGEN DIGITAL DE CUBA (SCOD-CUBA). La Habana : Cámara de Comercio de la República de Cuba, Septiembre, 2010.

12. **Eguiluz, Javier.** www.symfony.es. [En línea] enero 2011.http://www.symfony.es/.

13. **Visual Paradigm.** www.visual-paradigm.com. [En línea] enero 2011. http://www.visual-paradigm.com/.

14. **phpBB Group .** www.extjses.com. [En línea] enero 2011. http://www.extjses.com/.

15. **The PHP Group.** www.php.net. [En línea]enero 2011. http://www.php.net/.

16. **ORACLE.** www.oracle.com. [En línea] enero 2011. http://www.oracle.com/es/index.html.

17. **Conferencia de las Naciones Unidas sobre Comercio y Desarrollo.** www.unctad.org. [En línea] enero 2011. http://www.unctad.org/Templates/StartPage.asp?intItemID=2068.

18. **World Customs Organization.** www.wcoomd.org. [En línea]enero 2011. http://www.wcoomd.org/home.htm.

19. **Ministerio de Comercio, Industria y Turismo.** Ventanilla Unica de Comercio Exterior --VUCE--. *Ventanilla Unica de Comercio Exterior --VUCE--.* Republica de Colombia : s.n., 2007.

20. **Torres, Rafael.** Colombia. Bogotá, D.C. : s.n.enero 2011

21. **Almeida, Ing. Enrique.** Web Services en acción! Uruguay : s.n.enero 2011

22. **Mr. Joaquín Estuardo Arriaga Padilla.** Guatemala. Guatemala C.A. : s.n.enero 2011

23. **Information Management, Office Customs and Tariff Bureau, Ministry of Finance.** Tokyo, Japan : s.n. 1008940.enero 2011

24. **Korea Customs Service.** Republic of Korea Single Window Case. Daejeon, Republic of Korea : s.n., July 2010.

25. **Bermudez, Mairel H.** MODERNIZACIÓN DE LA ADUANA DE PANAMÁ. *"LA DIRECCIÓN GENERAL DE ADUANAS SU MODERNIZACIÓN Y DESARROLLO EN EL MARCO DEL COMERCIO INTERNACIONAL".* 2007.

26. **Gracia, Joaquin.***Diseño de Software Orientado a Objetos.* 27 de Mayo de 2005.

27. **Alvarado, Sergio.***Patrones de diseño.* 2003.

28. **Torossi, Profesor: A.U.S. Gustavo Marcelo.***Diseño de Sistemas.*marzo 2011

29. **Guerrero, Luis A.** Taller de UML. *Arquitectura física: Diagramas de Componentes.* marzo 2011

30. **Universidad de las Ciencias Informaticas.** Conferencia # 5. *Culminación del flujo de trabajo Análisis y Diseño. Diseño de la Base de Datos.* 2009-2010.

31. **Universidad de las Ciencias Informáticas.** Entorno Virtual de Aprendizaje. [Online] marzo 2011. http://eva.uci.cu/mod/resource/view.php?id=14070.

32. —. Entorno Virtual de Apredizaje. [Online] marzo 2011. http://eva.uci.cu/mod/resource/view.php?id=14094.

33. *http://geeks.ms*. (s.f.). Recuperado el Junio de 2011, de http://geeks.ms/blogs/mllopis/archive/2008/02/09/191-en-qu-233-consiste-el-proceso-de-pruebas-de-software.aspx

34. Ing. Violena Hernández Aguilar, I. M. (Junio de 2011). Proceso de pruebas de caja negra basado en la descripción de los casos de uso.

35. Isabel Blank, L. H. (Mayo de 2005). Pruebas de Funcionalidad.

36. Natalia Juristo, A. M. (2006). *Técnicas de Evaluación del Software.*

37. Prado, E. R. (2007). Casi todas las pruebas del software.

38. Pandero, C. F. ; Roman, J.V. (Febrero de 2006). Pruebas de Programas.

39. Universidad de las Ciencias Informáticas. (s.f.). *http://eva.uc.cu*. Obtenido de http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Complementarios/Material_de_caja_b_y_caja_n.pdf

40. Universidad de las Ciencias Informáticas. (Junio de 2011). *http://eva.uci.cu.* Obtenido de http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Basicos/Documentacion_sobre_Pruebas.pdf

41. Universidad de las Ciencias Informáticas. (Junio de 2011). *http://eva.uci.cu.* Obtenido de http://eva.uci.cu/file.php/259/Curso_2010-2011/Semana_9/Conferencia_7/Materiales_Basicos/Sobre_la_disciplina_de_Prueba.pdf

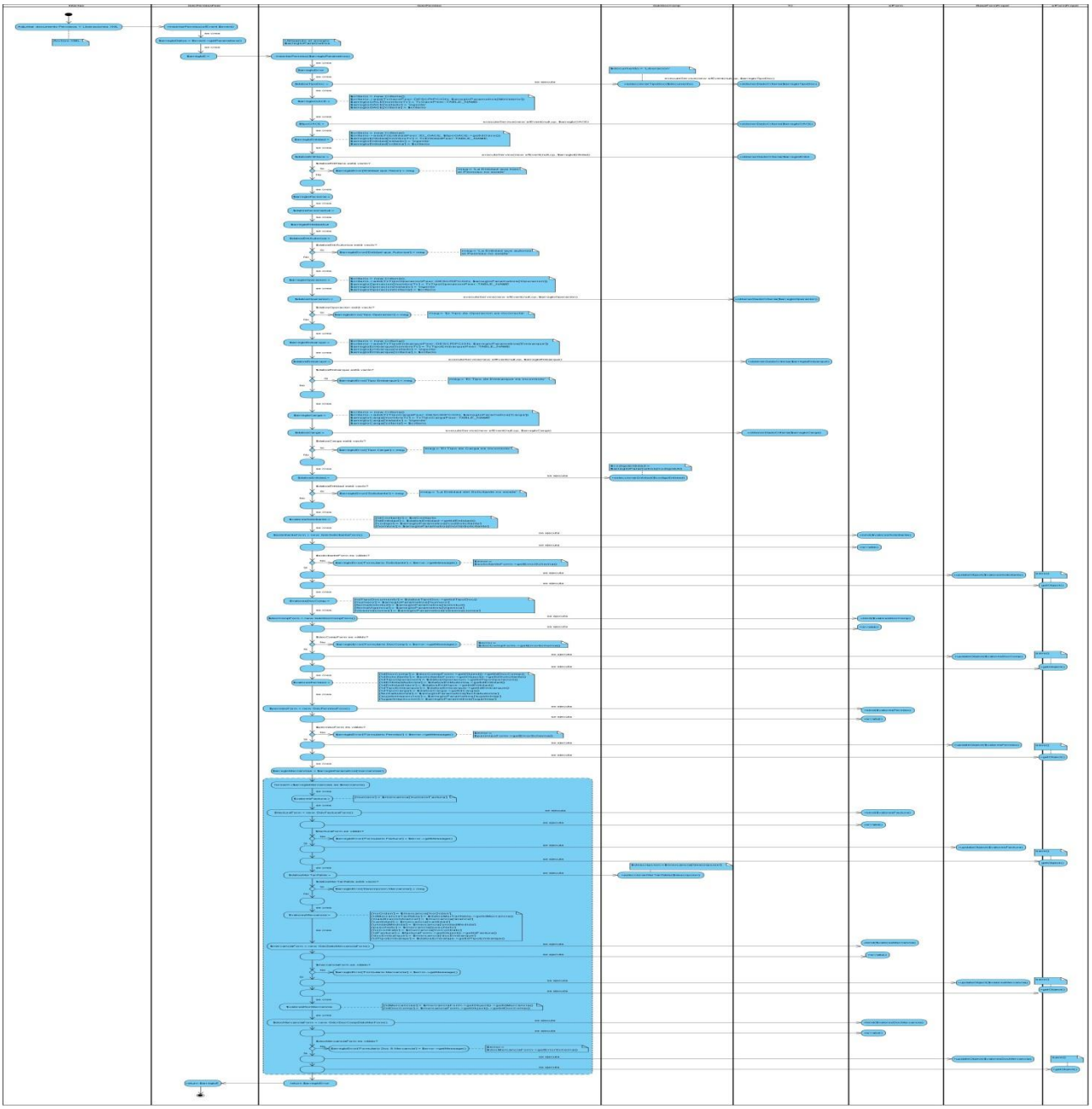
42. Arregui, J. J. (2005, Septiembre). Revisión Sistemática de Métricas de Diseño Orientado a Objetos.junio 2011

43. Harrison, D. R. (2007, Mayo). MEDICIÓN EN LA ORIENTACIÓN A OBJETOS. junio 2011

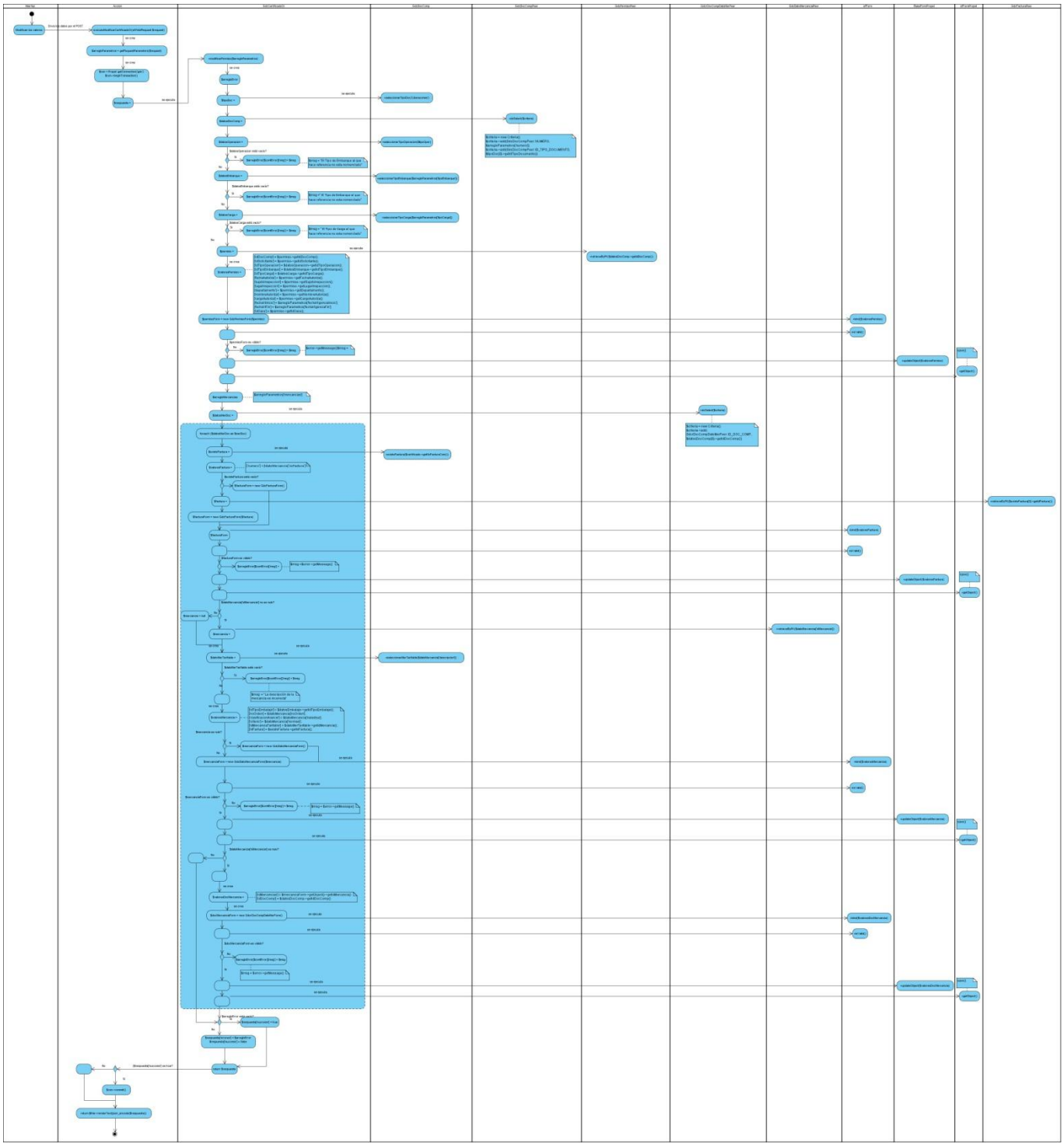
44. Reynoso, D. I. ; Batista, H.E. ; Martínez, I. (2007, Enero). MÉTRICAS EN INGENIERÍA DE SOFTWARE.junio 2011

ANEXOS

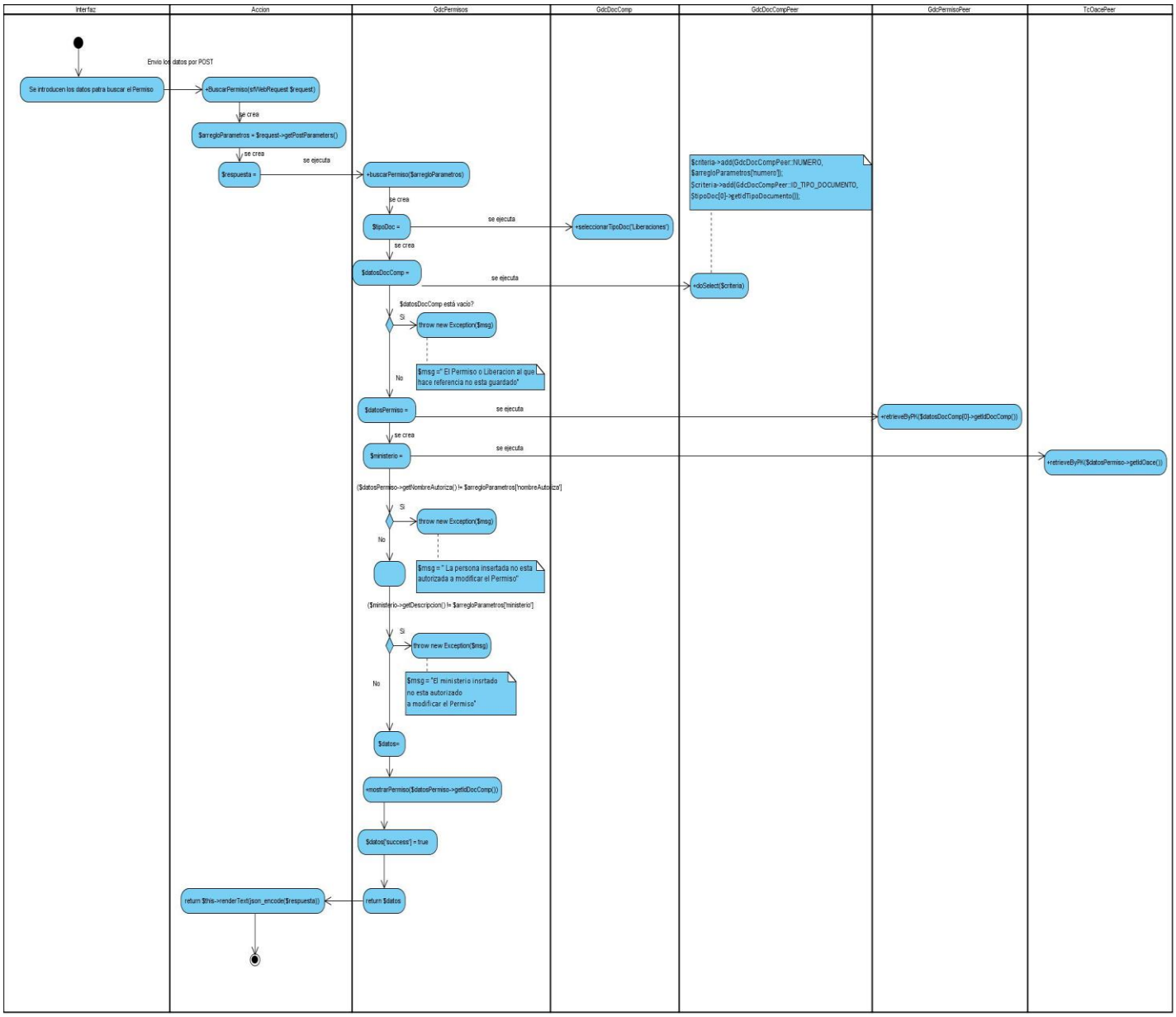
Anexo I Diagramas de Secuencia de Actividades del Negocio



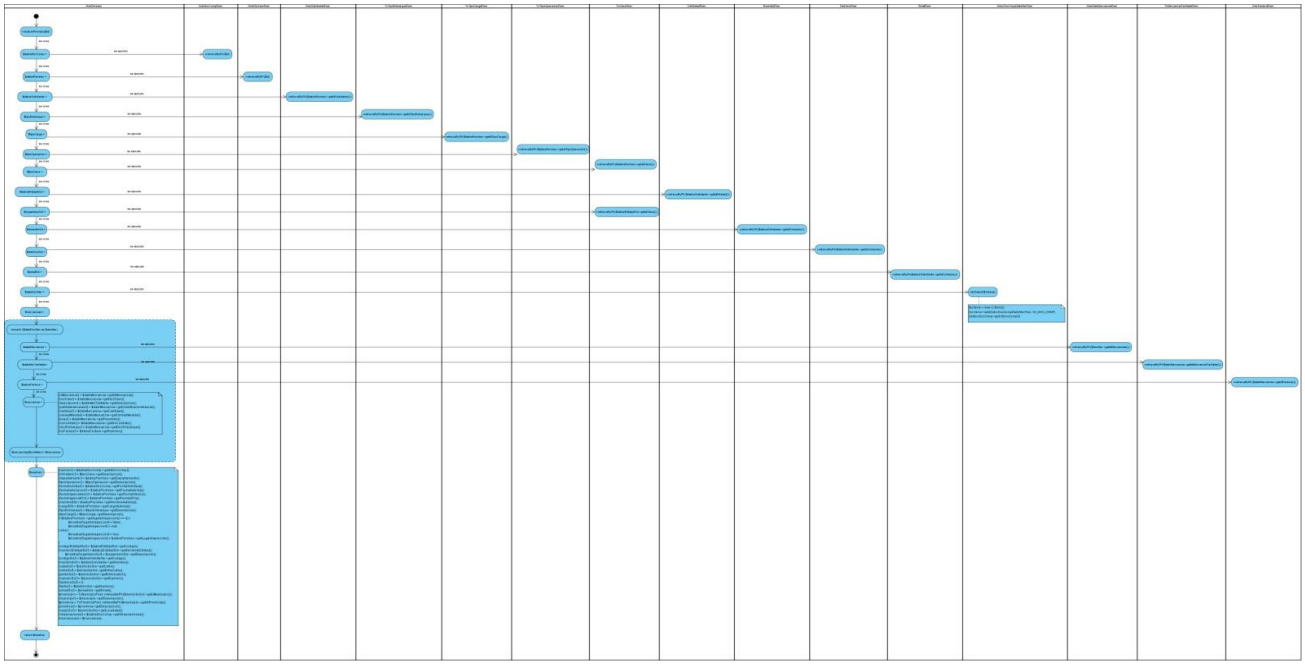
Registrar Permisos Y Liberaciones



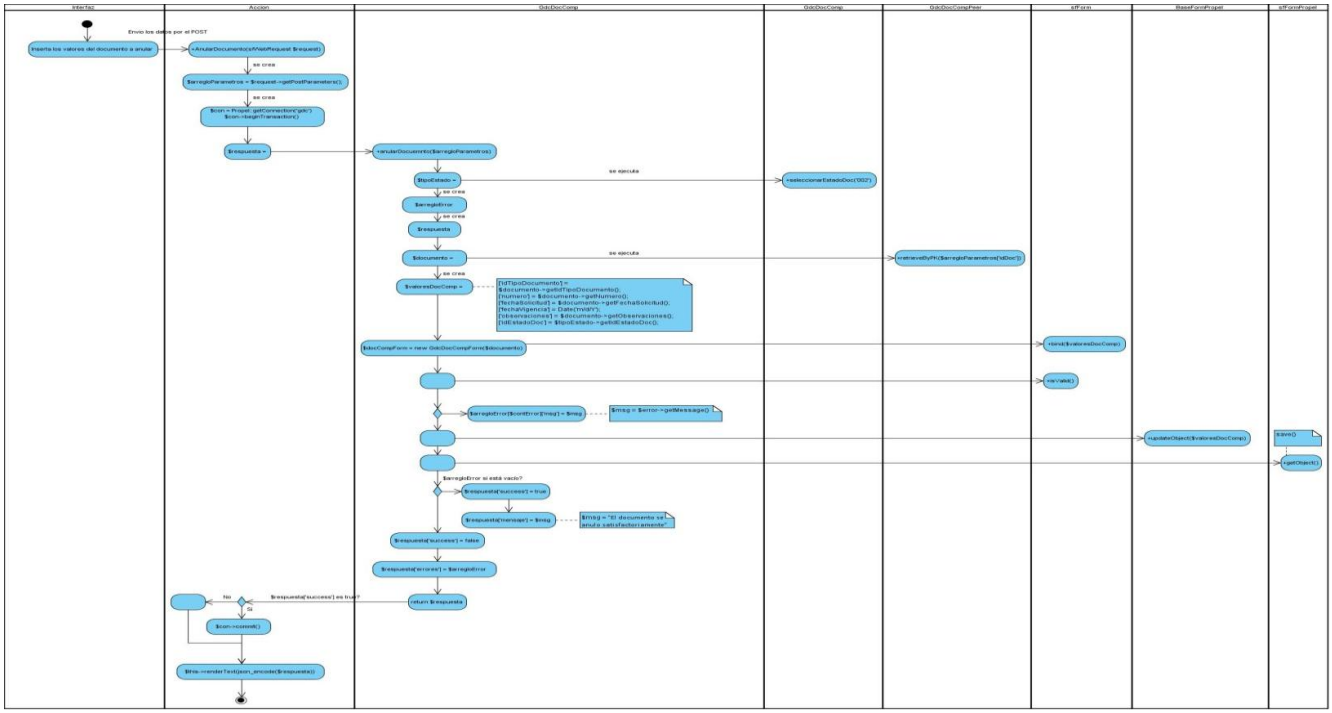
Modificar Permisos y Liberaciones



Buscar Permisos y Liberaciones

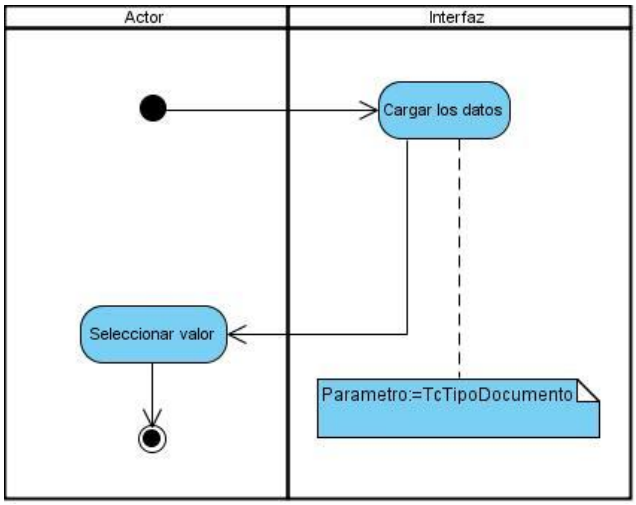


Mostrar Permisos y Liberaciones

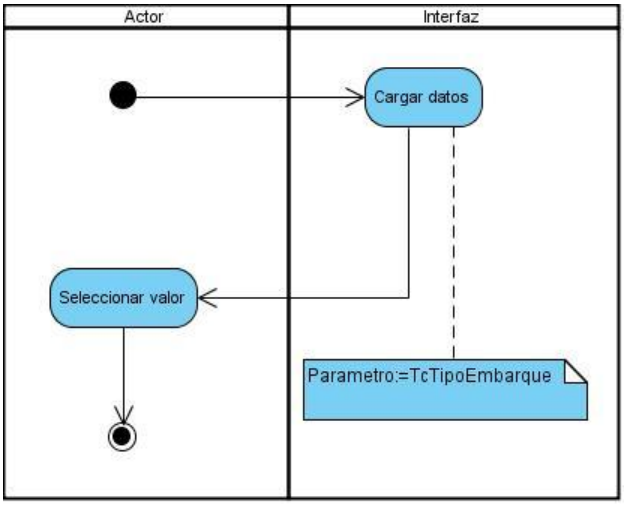


Anular Documentos Complementarios

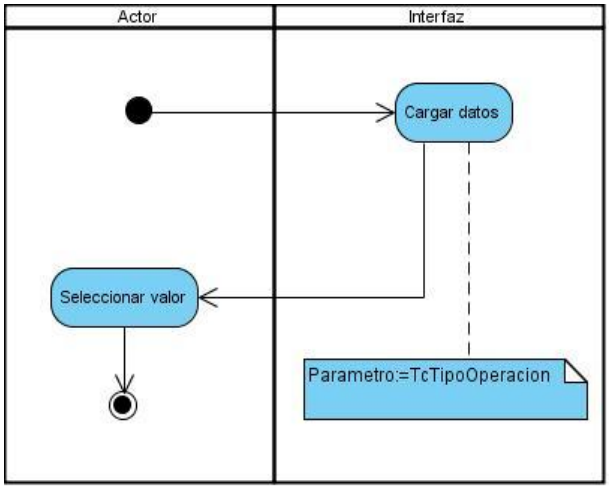
Anexo II Diagramas de Secuencia de Actividades de Componentes Visuales



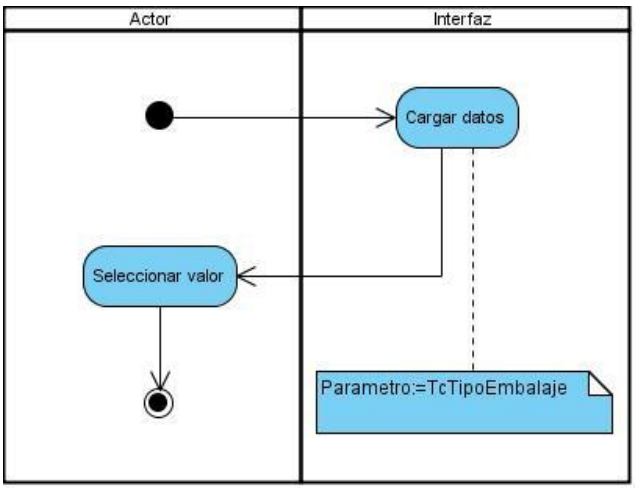
Tipo de Documento



Tipo de Embarque

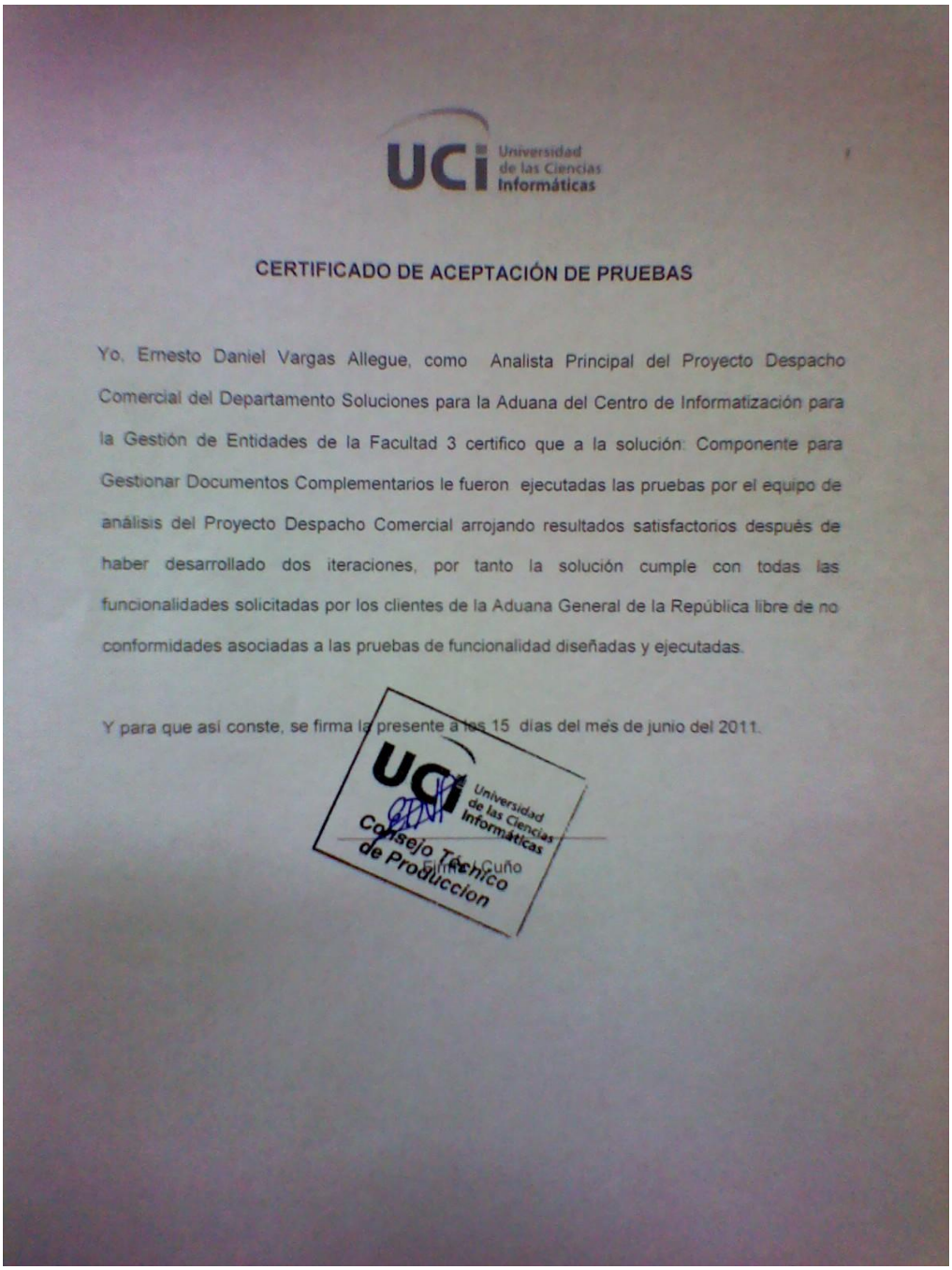


Tipo de Operación



Tipo Embalaje

Anexo III Certificado de Aceptación



Universidad de las Ciencias Informáticas

CERTIFICADO DE ACEPTACIÓN DE PRUEBAS

Yo, Ernesto Daniel Vargas Allegue, como Analista Principal del Proyecto Despacho Comercial del Departamento Soluciones para la Aduana del Centro de Informatización para la Gestión de Entidades de la Facultad 3 certifico que a la solución: Componente para Gestionar Documentos Complementarios le fueron ejecutadas las pruebas por el equipo de análisis del Proyecto Despacho Comercial arrojando resultados satisfactorios después de haber desarrollado dos iteraciones, por tanto la solución cumple con todas las funcionalidades solicitadas por los clientes de la Aduana General de la República libre de no conformidades asociadas a las pruebas de funcionalidad diseñadas y ejecutadas.

Y para que así conste, se firma la presente a los 15 días del mes de junio del 2011.

Universidad de las Ciencias Informáticas
Consejo Técnico de Producción

Certificado de Aceptación de Pruebas

GLOSARIO

1. UNTCAD: Es el órgano principal de la Asamblea General en la esfera del comercio y el desarrollo. Fue establecido en 1964, con el mandato de acelerar el desarrollo comercial y económico, haciendo especial énfasis en los países en desarrollo.
2. WCO: es un organismo intergubernamental independiente cuya misión es incrementar la eficiencia de las administraciones de aduanas, contribuyendo al bienestar económico y a la protección social de sus Miembros, favoreciendo de esta forma un entorno aduanero honesto, transparente y previsible. Esto permite el desarrollo del comercio internacional lícito y lucha eficaz contra las actividades ilegales.
3. Agentes de Aduana: profesional auxiliar de la función pública aduanera, cuya licencia lo habilita ante la Aduana para prestar servicios a terceros como gestor en el despacho de mercancías.
4. DUA-GT: es un documento uniformizar la presentación de la declaración de mercancías.
5. Certificado de Origen: Documento oficialmente válido que acredita que las mercancías amparadas en él son originarias de un determinado país.
6. Ingeniería inversa: el proceso de construir especificaciones de un mayor nivel de abstracción partiendo del código fuente de un sistema software o cualquier otro producto.
7. Java: es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems
8. C++: es un lenguaje imperativo orientado a objetos derivado del C.
9. ORM: es un componente de software que permite trabajar con los datos persistidos como si ellos fueran parte de una base de datos orientada a objetos básicamente son operaciones que permite transformar un registro en objeto y viceversa.
10. XML: es un sistema estándar de codificación de información, los programas que utilizan el formato XML pueden intercambiar fácilmente sus datos, ya que responden a una misma lógica interna.
11. .NET: es un conjunto de tecnologías de software, compuesto de varios lenguajes de programación que se ejecutan bajo el .NET Framework.
12. DBMS: Software que controla la organización, almacenamiento, recuperación, seguridad e integridad de los datos en una base de datos. Acepta solicitudes de la aplicación y ordena al sistema operativo transferir los datos apropiados.
13. UML: es una técnica para la especificación de sistemas en todas sus fases. Nació en 1994 cubriendo los aspectos principales de todos los métodos de diseño antecesores.

14. MySQL: es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB.
15. PostgreSQL: es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES, de la universidad de Berkeley.
16. MSSQL Server: es una plataforma global de base de datos que ofrece administración de datos empresariales con herramientas integradas de inteligencia empresarial.
17. SQlite: es un sistema de gestión de bases de datos relacional compatible con ACID, y que está contenida en una relativamente pequeña biblioteca en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp.
18. Licencia MIT (*Massachusetts Institute of Technology*): Es una licencia sin *copyright*, lo que nos permite modificarla y adaptarla a nuestras necesidades.
19. Licencia LGPL (LesserGnuPublicLicence): es una licencia que permite utilizarse junto a software no libre.
20. HTML: es el lenguaje que se utiliza para crear las páginas web le indica a los navegadores cómo deben mostrar el contenido de una página web.
21. POO: es lo que se conoce como un **paradigma o modelo de programación**. Esto significa que no es un lenguaje específico, o una tecnología, sino una **forma de programar.**
22. XHTML: es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web. XHTML es la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML.
23. LDAP (LightweightDirectory Access Protocol, Protocolo Ligero de Acceso a Directorios) : es un protocolo de tipo cliente-servidor para acceder a un servicio de directorio.
24. POP (Post Office Protocol, Protocolo de oficina de correo): es un protocolo estándar para recuperar correo electrónico, controla la conexión entre un cliente de correo electrónico POP y un servidor donde se almacena el correo electrónico.
25. INAM (intelligentnetworkapplicationpartprotocol): es unprotocolode capa de aplicaciónutilizada enredesinteligentes, se ejecutaenla parte superior deTCAPyse implementamedianteSS7.
26. SNMP (Protocolo simple de administración de red): Es un protocolo que les permite a los administradores de red administrar dispositivos de red y diagnosticar problemas en la red.

27. AJAX (Asynchronous JavaScript And XML): es un término que describe un nuevo acercamiento a usar un conjunto de tecnologías existentes juntas, incluyendo las siguientes: HTML o XHTML, hojas de estilo (Cascading Style Sheets o css), Javascript, el DOM (DocumentObjectModel), XML, XSLT, y el objeto XMLHttpRequest.
28. JSON (JavaScript ObjectNotation, Notación de Objetos de JavaScript): es un formato ligero de intercambio de datos, es subconjunto del estándar ECMA 262 publicado en diciembre de 1999. El formato de JSON es ampliamente reconocido por una gran variedad de lenguajes como Java, PHP, JavaScript, C++, C# entre otros,
29. DHTML: se trata de una tecnología (o mejor, un conjunto de tecnologías) que permiten hacer páginas web dinámicas.
30. DOM (DocumentObjetModel, Modelo de Objeto de Documento): es una estructura jerárquica donde existen varios objetos y unos dependen de otros.
31. Layouts: Es la ordenación y colocación de todos los elementos que componen una página web, es decir textos, imágenes, tablas y gráficos.