UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad #3

Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas.

Título: *Desarrollo del componente de estandarización de instrumentos jurídicos de la Dirección de Asuntos Legales de la Aduana General de la República de Cuba.*

Autor(es): Reinier Malagón Toledo

Tutores: Ing. Alain Eduardo Rodríguez
Ing. Rosalina Ibarra González.
Ciudad de la Habana, junio 23, 2011.Año 53 de la Revolución.

DECLARACIÓN DE AUTORÍA

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo a 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 _____.

Reinier Malagón Toledo	Alain Eduardo Rodríguez Áreas	Rosalina Ibarra González
_____	_____	_____
Firma del Autor	Firma del Tutor	Firma del Tutor

Resumen

La implementación del componente de Estandarización de Instrumentos Jurídicos, constituirá un aporte significativo a la informatización de la Dirección de Asuntos Legales de la Aduana General de la República de Cuba. Logrando una mayor uniformidad en los instrumentos jurídicos que se generan diariamente en esta institución, así como en los que son emitidos por las entidades de base autorizadas.

La estandarización de dichos instrumentos tiene como objetivo, normar su elaboración en aras de garantizar la entrada de los mismos datos desde varias entidades, reduciendo la posibilidad de hallar redundancias, errores ortográficos, datos innecesarios, estructuras legalmente incorrectas, entre otros. Posibilitándole al cliente obtener documentos de calidad, así como la extracción desde el módulo Dirección de Asuntos Legales (DAL), reportes estadísticos importantes para la jefatura con los datos introducidos a través de los estándares.

La propuesta de componente de estandarización que se presenta en este trabajo, será anexada al módulo referido anteriormente, una vez que sea terminado. Debido a que dicho módulo es a su vez, una actualización del Sistema Único de Aduanas (SUA), el cual es la herramienta de gestión aduanera que emplea actualmente esta institución.

Para tal desarrollo, será empleado el Modelo de desarrollo de software orientado a componentes y la arquitectura se centrará en el uso del patrón Modelo Vista Controlador (MVC)

Palabras Claves

Estándares, Sistema Único de Aduanas, Dirección de Asuntos Legales, Instrumentos jurídicos, Resolución.

ÍNDICE

Contenido

ÍNDICE

ÍNDICE

ÍNDICE

Introducción

El papel de las aduanas en el siglo XXI afrontando los desafíos planteados por la globalización, las iniciativas de facilitación del comercio y las preocupaciones en cuanto a seguridad, necesita un renovado acercamiento profesional a la gerencia y a las operaciones de las administraciones aduaneras a nivel mundial. Es por ello que la aduana ejerce su papel como uno de los principales organismos ejecutores de las leyes y regulaciones de cualquier estado en función de mantener controladas la entrada y salida de productos y bienes de un país.

Ubicada generalmente en fronteras, esta constituye toda oficina recaudadora fiscal, establecida por el gobierno nacional en los puertos marítimos, fluviales, fronterizos y aeropuertos del país, para aplicar y hacer cumplir la ley de aduanas y sus concordantes, recauda los derechos que fija el Arancel y los demás que se hallen a su cargo. Corre con las operaciones de entrada y despacho, tránsito y trasbordo, depósito y entrega de mercancías de importación y exportación, reprime el contrabando y el fraude a la renta nacional de aduanas y controla el comercio marítimo internacional y de cabotaje en cumplimiento de las leyes referidas, así como la entrada y salida de viajeros internacionales, y hace efectivas las prohibiciones de orden sanitario que las mismas leyes establecen. (Ramírez, 2005)

De esta manera en su haber jurídico surgen leyes y disposiciones de carácter obligatorio que son manejadas por la Dirección de Asuntos Legales (DAL) de la Aduana General de la República de Cuba (AGR), en la cual se gestionan numerosas tramitaciones que son registradas a través de diferentes instrumentos jurídicos.

De estos últimos, el que le compete a este trabajo es la Resolución. La cual puede derivarse en varios tipos de resolución de acuerdo al trámite que la genere. De su uso, se pueden obtener varios datos de gran importancia para la generación de reportes estadísticos de interés para la jefatura de la aduana. Sin embargo, a pesar de que existe el Decreto Ley 162 de Aduanas, que establece cómo debe ser elaborada una resolución determinada, aún estas se crean con diferencias significativas en su estructura y contenido, de manera tal que la información en algunas ocasiones suele ser redundante, inexacta, errónea o insuficiente. Afectando las disposiciones y veracidad de las mismas, pues ningún documento legal con errores es válido.

En adición, esta situación limita a los abogados con respecto a la elaboración de reportes sobre infracciones, personas de interés para la aduana, transacciones y medidas impuestas, entre otros. Dificultando la representación estadística de su funcionamiento, la extracción de datos con un grado de exactitud mayor al actual. Además de que todas estas operaciones deben realizarse manualmente, implicando cálculos y recolección de datos del propio abogado, quien puede equivocarse y obtener resultados erróneos.

Las resoluciones además, aportan diferentes datos que pueden ser referenciados desde varias direcciones aduaneras o áreas de trabajo. Pero si estos no pueden ser accedidos can rapidez y seguridad, posibles amenazas a la seguridad del país, o de cualquier otra índole podrían pasar inadvertidas para los inspectores en las áreas de enfrentamiento, o de inspección al arribo de las aeronaves y buques.

Por lo que encontrar una manera de normar la elaboración de las resoluciones y de garantizar la extracción de datos de las mismas en tiempo real con gran exactitud, es una prioridad de la DAL. (soluciones aduaneras, 2010)

Para corregir estas deficiencias y sentar un precedente en cuanto a la utilización de estándares de instrumentos jurídicos tanto nacional como internacionalmente, la AGR, el Centro de Automatización y Dirección de la Información (CADI) y la Universidad de las Ciencias Informáticas (UCI) conformaron un equipo de trabajo con la misión de hallar una solución a esta problemática.

A partir de la situación descrita anteriormente se identifica como **problema a resolver:** ¿Cómo estandarizar la elaboración de los instrumentos jurídicos de la Dirección de Asuntos Legales de la Aduana General de la República? Una vez identificado el problema a resolver se define como **objeto de estudio:** los instrumentos jurídicos de la Dirección de Asuntos Legales de la Aduana General de la República de Cuba. Definiendo como **campo de acción**: la estandarización de instrumentos jurídicos generados en la Dirección de Asuntos Legales de la Aduana General de la República Cuba.

Objetivo General: Desarrollar el componente de estandarización de los instrumentos jurídicos de la Dirección de Asuntos Legales de la Aduana General de la República de Cuba. El cual se desglosa en los siguientes **objetivos específicos**:

1. Elaborar la fundamentación teórica de la investigación.

2. Desarrollar el componente de estandarización.
3. Realizar pruebas al componente obtenido.

Para el cumplimiento de los objetivos de esta investigación se plantean las **tareas de investigación** que a continuación se enumeran:

1. Elaborar marco teórico de la investigación mediante el estudio de los diferentes conceptos involucrados.
2. Estudiar las tecnologías propuestas por el proyecto para la realización del componente.
3. Planificar entrevistas de captación de información con el cliente.
4. Elaborar documentación con toda la información obtenida de las entrevistas con el cliente.
5. Definir los estándares para los instrumentos jurídicos.
6. Describir los estándares de los instrumentos jurídicos.
7. Identificar procesos de negocio y listarlos.
8. Describir procesos del negocio
9. Identificar las reglas de negocio.
10. Listar y describir las reglas de negocio.
11. Modelar procesos de negocio con BPMN.
12. Listar los involucrados.
13. Obtener lista de requisitos de software.
14. Describir los requisitos.
15. Diseñar los estándares para los instrumentos jurídicos.
16. Implementar los estándares para los instrumentos jurídicos.
17. Elaborar pruebas para validar la propuesta.

La investigación es desarrollada mediante el empleo de **métodos científicos** que sirven para estudiar la realidad, la sociedad y la naturaleza que forman en conjunto el entorno donde todos los procesos legales

de la aduana tienen lugar y a su vez donde se genera el problema a resolver facilitando el descubrimiento de su esencia y sus principales relaciones, implicando además una combinación de inducción y deducción que se retroalimentan:

Métodos Teóricos:

Histórico-Lógico: estudia los aspectos concretos en la trayectoria real de los fenómenos y acontecimientos en el transcurso de su historia; en este caso es empleado en el estudio del arte del tema a investigar, pues de esta manera se puede conocer acerca de la existencia y características de sistemas de este tipo que hayan sido creados anteriormente.

Analítico-Sintético para resumir, enunciar y describir los requisitos enunciados por los profesionales.

Análisis Histórico – lógico: se refleja durante la investigación relacionada con las herramientas a utilizar para la confección del producto final. Además se pone de manifiesto durante la identificación de las necesidades de información y el levantamiento de requisitos. Para investigar sobre los sistemas de gestión aduanera desplegados en Cuba y el resto del mundo.

Modelación: Se utiliza con el objetivo de lograr un mayor entendimiento de todos los procesos y para crear un modelo que explique cómo debería quedar el resultado final. Consiste en realizar una reproducción simplificada de la realidad. Permite descubrir nuevas relaciones y cualidades del objeto en estudio.

Dentro de los **Métodos Empíricos** se utiliza la **Entrevista** para conocer ciertas especificidades que los estándares deberían contener, en relación a la información constante en cada uno de los instrumentos jurídicos y a la que cambia en correspondencia con el caso legal en cuestión.

Síntesis de la Estructura de la investigación por Capítulos.

El presente trabajo está estructurado por **tres capítulos**, distribuidos de la siguiente manera

Capítulo 1: Fundamentación Teórica. En este capítulo se plantean todos los elementos que sustentan el desarrollo y solución del problema a resolver. Además se definen los principales conceptos asociados a

las herramientas a utilizar en el desarrollo de las aplicaciones y se exponen sus características. Se exponen además las tecnologías a utilizar, la metodología, el lenguaje de programación entre otras.

Capítulo 2: Análisis y diseño de la solución. En este capítulo se muestran todas las definiciones del negocio, a través de procesos modelados en BPMN, captura y especificación de requisitos, definición de reglas del negocio e involucrados. Además de los diagramas de clases del sistema, modelos de datos y modelo conceptual. Se aplicarán métricas al diseño y se reflejará el uso de patrones

Capítulo 3: Implementación y prueba. Se desarrolla la solución propuesta en el capítulo 2. Además se validarán los elementos que lo requieran y la funcionalidad del producto también a través de casos de prueba e informes de resultados.

Capítulo 1: Fundamentación teórica

1 Introducción

En el presente capítulo se ofrece una panorámica relacionada con los instrumentos jurídicos de la Aduana General de la República de Cuba, los procesos que los generan, así como una breve descripción de los sistemas aduaneros que han sido empleados con anterioridad. Además se relacionan los elementos más importantes inherentes a la investigación como Ingeniería de software, Requisitos, Técnicas de captura de requisitos, Tecnologías, Herramientas y otros. En adición, se muestran referencias de los Marcos de Trabajo, los Entornos de desarrollo Integrado (IDES) y lenguajes de programación que facilitarán el desarrollo de la solución que se propone.

1.1 Sistemas aduaneros de referencia

El avance de las Tecnologías de la Información y las Comunicaciones (TICs) se hace evidente en todas las esferas de la sociedad y la economía de un país, dígase su Educación, Medicina, Innovación tecnológica, Investigación científica, Gestión de entidades, entre otras. El desarrollo tecnológico, ha propiciado la proliferación de sistemas de gestión aplicables a la gran variedad de instituciones que componen los sectores productivos y de control de las naciones, de manera que los alcances de estos diversos sistemas se enfocan en el desarrollo de los procesos, su agilización, la protección de la información, la eliminación de duplicidad en los datos, la erradicación del manejo irregular de materiales gastables, entre otras razones que justifican las grandes ventajas palpables durante su explotación.

En Cuba, se han puesto en práctica varios de estos sistemas aportando un cúmulo significativo de beneficios que han permitido el desarrollo y crecimiento de instituciones estatales como la Aduana General de la República de Cuba. En torno a la cual se pudieran referenciar sistemas aduaneros como el SIDUNEA (Sistema Aduanero Automatizado), Sistema de Órganos Aduaneros (SOA), Sistema Automatizado de Personas de Interés Aduanal (SAPIA), Sistema Automatizado de Despacho de Operaciones No Comerciales (SADONCE), Sistema Automatizado de Despacho Mercantil (SADEM), el Sistema Automatizado de Control Mercantil (SACOM) y el Sistema Único de Aduanas (SUA) que se encuentra en desarrollo, para tratar de integrar y mejorar las facilidades que ofrecen los sistemas relacionados previamente. Además dentro de la competencia internacional se encuentran el Sistema Aduanero Nacional de El Salvador y el Sistema María (SIM) de producción Argentina, los cuales gozan de gran popularidad entre las aduanas del mundo.

De estos sistemas antes mencionados, se puede concluir que son muy útiles para realizar funciones específicas en cada una de las aéreas de las aduanas, tales como optimizar los tiempos

y recursos del proceso aduanero, aplicar la ley con justicia y exactitud, cobrar los impuestos y tasas correctamente, monitorear el pago de los impuestos para la importación y exportación de mercancías ya sea de forma definitiva o temporal, minimizar el contrabando, además de que se puede configurar la forma de operar y recuperar información. Sin embargo, entre las funcionalidades inexistentes hasta el momento en la mayoría de ellos para no absolutizar el término; está la gestión de los procesos legales, que en el caso de Cuba tienen lugar en la Dirección de Asuntos Legales de la Aduana General de la República de Cuba y para el resto de las aduanas internacionales es frecuentemente denominado Asuntos Jurídicos o Customs' Legal Issues.

Esta Dirección o Área aduanera, es la encargada de todos los trámites que involucran a las personas naturales o jurídicas que se acogen a los servicios de la aduana cubana. Esta funciona como órgano legislativo y de representación de dicha institución ante casos que impliquen judicialmente a la misma. Debido a su gran importancia e interrelación con todos los restantes subsistemas integrados al SUA, el equipo de desarrollo comenzó a elaborar una solución informática, para la centralización de los datos y el establecimiento de un flujo más directo de información facilitando la retroalimentación de todas las áreas y la generación de reportes estadísticos donde se identificó un problema que afecta la entrada de datos, principalmente, debido a que la mayoría de la información que se maneja, está contenida en instrumentos jurídicos que son gestionados de forma manual y que al digitalizarse sin un patrón o plantilla dificultan la administración de los datos por estar redactados en forma de documentos.

1.2 Instrumentos jurídicos

Los instrumentos jurídicos no son más que el conjunto de resoluciones administrativo/legales, circulares, instrucciones, decretos, dictámenes, cartas resolutivas o de respuesta a interesados y providencias, que constituyen la entrada y salida de todos los procesos que generan este tipo de documentación, conocidos como Procesos Legales; que desde el punto de vista jurídico pueden ser definidos como el conjunto de actos coordinados que se ejecutan por una persona ante los funcionarios competentes del órgano judicial del Estado para las aduanas; con el objetivo de obtener mediante la actuación de la ley en un caso concreto, la declaración, la defensa o la realización coactiva de los derechos que pretendan tener las personas privadas o públicas, en vista de su incertidumbre, de su desconocimiento o de insatisfacción (en lo civil, laboral o litigante administrativo), para la investigación, prevención y represión de los delitos y las contravenciones

(en materia penal), y para la tutela del orden jurídico y de la libertad individual y la dignidad de las personas, en todos los casos (civil, penal, etcétera).

Dichos instrumentos jurídicos contienen datos importantes que pueden ser relacionados en reportes u otro tipo de referencia cruzada en aras de establecer o calcular estadísticas de funcionamiento o eficiencia del trabajo en las aduanas de base. Pero como su elaboración sólo está condicionada por resoluciones o dictámenes que rigen qué datos y estructura tendrán y no el cómo se elaborará, ni qué información puede ser mantenida constante, se estableció la meta de la creación y generación de estándares que facilitarán el trabajo, asegurando la calidad de dichos instrumentos, la homogeneidad en la redacción, la eliminación de incongruencias y ambigüedades, además del ahorro considerable del material gastable de la entidad.

1.3 Estandarización

El proceso de estandarización definido para los instrumentos jurídicos de la DAL es la normalización de toda la información que se trabaja sobre platillas, la cuales prueban todos los datos que le conciernen a un determinado proceso jurídico o legal.

Los estándares son valores, documentos, instrumentos y normas que constituyen un mecanismo de calidad que permite regular la representación de datos, su proyección, su transmisión, su almacenamiento y retroalimentación o recuperación, protegiendo la información contra la duplicidad, incongruencia, incoherencia y los errores de otro tipo como ortográficos, legislativos o de ubicación de los casos, logrando además la homogeneidad de la información y la agilidad en los procesos per se.

Para la estandarización de los elementos que lo requieran, se definen 4 actividades fundamentales que se describen a continuación:

Establecimiento de estándares: Es la primera etapa del control, que establece los estándares o criterios de evaluación o comparación. Un estándar es una norma o un criterio que sirve de base para la evaluación o comparación siguiendo criterios definidos previamente acordados y aprobados. Existen cuatro tipos de estándares; los cuales se presentan como:

- Estándares de cantidad: Regulan proporciones, de manera que las cantidades propuestas o controladas sean equitativamente distribuidas de acuerdo al estándar definido sea por la

organización, un especialista o un procedimiento en específico. Por ejemplo se pueden aplicar estándares al volumen de producción, cantidad de existencias, cantidad de materias primas a emplear en la creación de un objeto, al número de horas para ejecutar una tarea o una respuesta, entre otros.

- Estándares de calidad: Definen una normativa de racionalización y control de la calidad en el proceso de creación o desarrollo de algún producto en específico, dígase un software, un automóvil, un producto químico, etc. De tal manera que establece la mejor forma o más cercana al término óptimo de modelar, describir, especificar, desarrollar, validar o probar un objeto.

- Estándares de tiempo: Establece la cuota de tiempo a emplear en actividades específicas que por sus características pueden ser controladas a través de un cronograma fijo, debido a que fueran analizadas y dispuestas de una manera determinada, o que se definiera el nivel óptimo para su ejecución identificándose un patrón de tiempo, dígase tiempo de respuesta de un buscador, el tiempo de cocción para pastelería, el tiempo de mezcla y homogenización de exámenes clínicos en instrumentos previamente programados, el tiempo de curación del tabaco o de añejo para los vinos, entre otros.

- Estándares de costos: Estos estándares definen costos previamente calculados para la realización de transacciones u otro tipo de acción contable o que pueda ser cuantificada y no necesariamente relacionada con el dinero, que pueda implicar cuotas únicas de capital ya sea monetario, humano, de ejercicio mental, de tiempo entre otros. Por ejemplo costos de administración, costos de importación, costo de esfuerzo/horas/hombre, entre otros.

Una vez que son aplicados los diferentes estándares de forma integrada o indistintamente, se debe dar un seguimiento al desarrollo de la propia aplicación, los resultados generados y los posibles errores a corregir. Para ello, se relacionan otras tres actividades que conjuntamente con la antes expuesta complementan el proceso de creación, especificación, desarrollo, validación y prueba a estándares.

Evaluación del desempeño: Es la segunda etapa del control, que tiene como fin evaluar lo que se está haciendo, para establecer los elementos que determinen si la aplicación del estándar está siendo realizada de forma efectiva (Estándares, 2000)

Comparación del desempeño con el estándar establecido: Es la tercera etapa del control, que compara el desempeño con lo que fue establecido como estándar para verificar si hay desvío o variación con respecto al objetivo trazado al comenzar la estandarización y si existe algún error o falla con relación a los resultados que se espera obtener (Estandares, 2000)

Acción correctiva: Es la cuarta y última etapa del control que busca corregir el desempeño para adecuarlo al estándar esperado. La acción correctiva es siempre una medida de corrección y adecuación ante algún desvío o variación con relación al estándar esperado (Estandares, 2000)

Para poder estandarizar con la calidad requerida, inicialmente se deben realizar actividades y aplicar técnicas que permitan conocer a cabalidad las funcionalidades y especificidades de dichos estándares, es por eso que la Ingeniería de Requisitos juega un papel primordial en esa labor (Estandares, 2000)

1.4 Ingeniería de requisitos (IR)

La Ingeniería de Requisitos, cumple un papel primordial en el proceso de construcción y producción de un software, es decir que, estará basada en función de las necesidades planteadas por los clientes en un nivel muy general donde se descubre, documenta, analiza y se definen los servicios o componentes de lo que se desea producir, además de las restricciones que tendrá el producto o software. Su principal tarea consiste en la definición del proceso a seguir en la construcción de un software y de facilitar la comprensión de lo que el cliente requiera. La obtención correcta de los requisitos puede llegar a describir con claridad, sin ambigüedades, en forma consistente y concisa, el comportamiento de un sistema.

Existen varios conceptos o significados acerca de la IR que proporcionan los autores según su nivel de experiencia, sentido común o simplemente por su forma de ver los requisitos respecto al desarrollo de un determinado proyecto. La “Ingeniería de Requisitos es la disciplina para desarrollar una especificación completa, consistente y no ambigua, la cual servirá como base para acuerdos comunes entre todas las partes involucradas y en dónde se describen las funciones que realizará el sistema”. (Boehm, 1979).

Por su parte, Pressman plantea que la Ingeniería de Requisitos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas

que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software. (Pressman, 2006)

En adición, Sommerville sostiene que la ingeniería de requisitos es el proceso de desarrollar una especificación de software. Las especificaciones pretenden comunicar las necesidades del sistema del cliente a los desarrolladores del sistema. (Sommerville, 2005)

Sin embargo, no puede hablarse de Ingeniería de requisitos sin referirse a los requisitos como tal, especificándose que:

“La Ingeniería de Requisitos es el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener con un costo reducido el análisis, documentación, evolución continua de las necesidades del usuario y la necesidades del usuario y la especificación del comportamiento externo que satisfaga las necesidades del usuario”(Pressman, 2005).

Los requisitos están clasificados en dos categorías, requisitos funcionales y requisitos no funcionales. (Pressman, 2005)

- Los requisitos funcionales: definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Es importante que se describa el ¿Qué? y no el ¿Cómo? se deben hacer esas transformaciones.

- Los requisitos no funcionales: tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, entre otras.

Para lograr el objetivo que se plantea la Ingeniería de Requisitos diversos autores han establecido varias actividades que facilitan la interpretación, descripción, modelación y validación de un negocio específico.

1.4.1 Actividades de la IR

Las actividades que se realizan en la IR varían según las prácticas de sus autores de acuerdo a las características del proyecto (tamaño, modelo del proceso). Estas no son criterios

esquemáticos sino que evolucionan según las experiencias de cada desarrollador en este campo de la Ingeniería de Software.

Entre las actividades definidas en varias bibliografías están la elicitación, análisis y validación. (Durán, 2000). Sin embargo Pressman define 5 pasos distintos: Identificación de Requisitos, Análisis y Negociación de Requisitos, Especificación de Requisitos, Modelado del Sistema, Validación de Requisitos y Gestión de Requisitos. (Pressman, 2005)

En el proceso de desarrollo de software del proyecto colaborativo con la AGR se identificaron tres actividades principales a realizar:

- Elicitación de requisitos: “La elicitación de requisitos es la parte de la Ingeniería de Requisitos en la que se tiene contacto con los clientes y usuarios y donde deben quedar claros el dominio del problema, las necesidades reales del cliente y usuarios finales, y la negociación con estos de los requisitos”. (Durán, 2000).

- Análisis de requisitos: El objetivo de esta metodología es la definición de las tareas a realizar, los productos a obtener y las técnicas a emplear durante la actividad de análisis de requisitos de la fase de ingeniería de requisitos del ciclo de vida de la ingeniería del software. (Durán, 2001)

- Validación de requisitos: evalúa la calidad de la especificación, examinando las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos, y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto. (Pressman, 2005)

1.4.2 Técnicas de captura de requisitos

El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años técnicas que permitan hacer este proceso de una forma más precisa y eficaz.

Las técnicas, de forma tradicional han sido utilizadas para esta actividad en el proceso de desarrollo de todo tipo de software. Entre ellas se citan las entrevistas, el Joint Application Development JAD (Desarrollo conjunto de aplicaciones), la revisión de documentos, modelo de negocio, tormenta de ideas, mapas conceptuales, esbozos o guiones gráficos, listas de chequeo, entre otras. (Durán, 2001)

Para definir los requisitos funcionales de los procesos del subsistema de la DAL se usaron algunas de estas técnicas de captura de requisitos favorables a las necesidades del equipo y que posibilitarán un dominio previo de los conceptos del negocio, permitiendo una familiarización con los mismos en un corto período de tiempo.

Entrevistas: Resultan, una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Básicamente, la estructura de la entrevista abarca cuatro pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados. (Durán, 2001)

Revisión de documentos: Esta técnica depende de la información almacenada por las entidades acerca de los procesos y términos que se manejan dentro de la misma. Las entidades guardan información referente a sus procesos, los modelos o informes necesarios para el desarrollo de la misma o para rendir cuenta a los organismos superiores. Este cúmulo de información es estudiado por los analistas en busca de captar bien todos los procesos para determinar los requisitos asociados a estos, y que luego deben ser verificados por otras técnicas. La revisión de documentos no es efectiva por sí sola, para la Captura de Requisitos, debe ser vinculada con otra(s) para lograr un resultado efectivo. (Durán, 2001)

Join Application Development (JAD): Es una práctica, en grupo, que se desarrolla varios días donde el cliente es partícipe junto al equipo de desarrollo de las actividades expresando los problemas así como las posibles soluciones. El objetivo es romper las barreras con el cliente que no se sienta excluido del proceso de desarrollo y de esta forma puede dar lugar a una declaración más exacta de los requisitos del sistema a una comprensión mejor de metas comunes, y a una comisión más fuerte al éxito del nuevo sistema. Cuando los usuarios participan en el proceso del desarrollo de los sistemas, son más probables a sentir un sentido de la propiedad en los

resultados, y la ayuda para el nuevo sistema. Comparado con métodos tradicionales, JAD es más costoso y puede ser incómodo si el grupo es demasiado grande concerniente al tamaño del proyecto. Muchas compañías encuentran, sin embargo, que JAD permite que los usuarios dominantes participen con eficacia en los requisitos que modelan proceso. (Durán, 2001)

Modelo de Negocio: describe el funcionamiento actual del negocio del cliente, es decir, los procesos del negocio. Es fundamental para entender el contexto en el que se usará el sistema a desarrollar y permite mejorar los procesos de negocio al tener una visión más general de los mismos. El nivel de detalle es menor que en los modelos del sistema a desarrollar. (Durán, 2001)

En la revisión de documentos utilizados en el subsistema DAL, permitió adquirir una visión inicial del negocio. Se empleó la técnica JAD con la participación de especialistas de la aduana donde se tomaron las ideas en diferentes sesiones de trabajo y se logró eliminar posibles brechas entre lo técnico de la informática y lo conceptual de los procesos en la práctica .Se modeló el negocio con la participación de los especialistas funcionales, lo que demostró que es una vía muy eficiente para que la comunicación sea en ambos sentidos y muy fluida. Creándose de esta manera diagramas para un mejor entendimiento y se determinaron y especificaron los procesos a desarrollar.

1.4.3 Análisis de requisitos

El objetivo de esta actividad es la definición de las tareas a realizar, los productos a obtener y las técnicas a emplear durante el análisis de requisitos de la fase de ingeniería de requisitos del ciclo de vida de la ingeniería del software. En esta actividad se distinguen dos tipos de productos: los productos entregables y los productos no entregables o internos. Los productos entregables son aquellos que se entregan oficialmente al cliente como parte del desarrollo en fechas previamente acordadas, mientras que los no entregables son productos internos al desarrollo que no se entregan al cliente. (Durán, 2001)

La especificación describe la función y características de un sistema de computación y las restricciones que gobiernan su desarrollo. Para la actividad de especificación de requisitos existe un gran número de técnicas propuestas. Las más relevantes son glosarios, lenguaje natural, lenguaje formal, plantillas, escenarios, etcétera. Para especificar los requisitos se utilizaron las siguientes técnicas.

Glosarios: la diversidad de personas que forman parte de un proyecto de software hace que sea necesario establecer un marco de terminología común. Por esta razón son muchas las propuestas que abogan por desarrollar un glosario de términos en el que se recogen y definen los conceptos más relevantes y críticos para el sistema. En esta línea se encuentra también el uso de ontologías, en las que no sólo aparecen los términos, sino también las relaciones entre ellos.

Plantillas: esta técnica, recomendada por varios autores, tiene por objetivo el describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea esta, menos ambigüedad ofrece. Sin embargo, si el nivel de detalle elegido es demasiado estructurado, el trabajo de rellenar las plantillas y mantenerlas, puede ser demasiado tedioso.

Mediante el análisis de las técnicas existentes para definir los requisitos, se decidió utilizar el glosario y las plantillas. El glosario, para conceptualizar las terminologías y lograr un entendimiento entre el equipo de desarrollo y el cliente. Las plantillas, para estructurar de una forma estándar la información, previendo que el nivel de detalles no fuera tan estructurado evitando así las ambigüedades.

1.4.4 Validación de requisitos

La validación es la etapa final de la IR. Los requisitos una vez definidos necesitan ser validados. La validación tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos.

Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la especificación de requisitos con el usuario para detectar errores o inconsistencias. Las más conocidas son las revisiones, prototipos, matrices de trazabilidad y auditorías. Para validar los requisitos en el proyecto se utilizaron las siguientes técnicas:

Revisión: esta técnica consiste en la lectura y corrección completa de la documentación o modelado de la especificación de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida. Más difícil es verificar la consistencia de la documentación o información faltante.

Prototipos: algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final.

Posterior a la obtención de varios requisitos y su modelado se realizaron revisiones para garantizarse que la interpretación por parte del analista fue correcta y que no faltó información. Para obtener una idea de la interfaz de usuario se desarrolló un prototipo con las funcionalidades críticas del sistema.

1.5 Modelo de desarrollo de software

El modelo de desarrollo de software propuesto describe la secuencia de actividades de alto nivel para la instrucción y desarrollo de soluciones. Se logra con la combinación entre los modelos: basado en componentes, el iterativo y el incremental. (GEIGE, 2009)

1.5.1 Desarrollo iterativo e incremental

Es un enfoque en el que el ciclo de vida está compuesto por iteraciones, estas son pequeños procesos compuestos de varias actividades cuyo objetivo es entregar una parte del sistema parcialmente completo, probado, integrado y estable. Todo el software es integrado en cada entrega de cada iteración hasta obtener el producto de software completo en la última iteración. En cada iteración se obtiene como resultado un incremento. (GEIGE, 2009)

1.5.2 Desarrollo basado en componentes

Nos lleva a alcanzar un mayor nivel de reutilización de software, aún en contextos distintos a aquellos para los que fue diseñado. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema. Dado que un componente

puede ser construido y luego mejorado continuamente, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo. (GEIGE, 2009)

1.6 Herramientas de desarrollo y tecnologías

A continuación se detallan las herramientas que ayudarán al cumplimiento del objetivo general de este trabajo. Estas, en su mayoría, fueron las que se utilizaron para el desarrollo de la primera versión. Las mismas ponen en práctica el aprovechamiento y reutilización de la tecnología, un punto que siempre debe tenerse en cuenta cuando se va a desarrollar una versión superior de un software, pues buscar otras herramientas conllevaría a un gasto innecesario de tiempo y de recursos. Esto atado al hecho de que las definiciones arquitectónicas del proyecto están sobre estos pilares de desarrollo y sería impropio violentarlas si con las mismas se puede resolver la problemática en cuestión.

1.6.1 Lenguajes de modelado

El lenguaje de modelado es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un software. En la mayoría de los casos son utilizados en combinación con una metodología de desarrollo de software para realizar la especificación del desarrollo de un software y de este modo hacerlo extensivo a todo el equipo de desarrollo. El uso de un lenguaje de modelado es más sencillo que la auténtica programación. Por su robustez y fiabilidad el equipo de desarrollo decidió que se utilizaría el lenguaje UML (Unified Modeling Language, lenguaje unificado de modelado) para especificar los artefactos que se deben generar.

1.6.1.1 BPMN

Business Process Modeling Notation (BPMN) por sus siglas en ingles traducidas, traduciadas el español como (notación del modelado de los procesos de negocio), es una notación gráfica estandarizada para modelar procesos de negocios en flujos de trabajo. Fue desarrollada por Business Process Management Initiative (BPMI), y está siendo mantenida por el Object Management Group (OMG) desde que ambas organizaciones se asociaron en el 2005.

La notación BPMN permite separar la información de negocio, de la información técnica (elementos técnicos del sistema de información) para maximizar su capacidad de ser transferida de una compañía a la otra. Se puede calificar como una notación UML aplicada a la administración de procesos de negocio. (Dumas, 2005).

El objetivo primario de BPMN es proporcionar una notación estándar que sea legible y entendible por todos los involucrados en el negocio. Estos involucrados incluyen desde los analistas del negocio, quienes crean y refinan los procesos, los desarrolladores técnicos responsables de la implementación de los procesos, hasta los administradores de procesos quienes monitorean y administran los procesos. BPMN pretende servir como lenguaje común para facilitar la comunicación que generalmente se establece entre el diseño y la implementación de los procesos de negocios. La adopción de la notación estándar BPMN ayuda con la expresión de conceptos básicos de procesos del negocio (procesos públicos y privados) así como con el modelado de conceptos avanzados (manipulación de excepciones, compensación de transacciones).

Los modelos BPMN son hechos por simples diagramas formados por un conjunto de elementos gráficos. Estos le facilitan tanto a los desarrolladores como a los usuarios del negocio comprender los procesos. Hay cuatro categorías básicas de elementos: (Dumas, 2005)

- Objetos de flujos
- Conectores
- Piscinas
- Artefactos

1.7 Herramientas Computer Aided Software Engineering (CASE)

Las **Herramientas CASE** son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación, detección de errores, entre otras.

Objetivos:

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la planificación de un proyecto.

- Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
- Automatizar, desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.
- Ayuda a la reutilización del software, portabilidad y estandarización de la documentación.
- Gestión global en todas las fases de desarrollo de software con una misma herramienta.
- Facilitar el uso de las distintas metodologías propias de la ingeniería del software.

1.7.1 Visual paradigm para UML

Visual Paradigm para UML es una herramienta CASE que soporta UML 2.1 como lenguaje de modelado y BPMN. Esta herramienta apoya el ciclo de vida completo de desarrollo del software, análisis, diseño, implementación y prueba. Este se encuentra enfocado al negocio y ofrece un software de mayor calidad, además presenta disponibilidad de múltiples versiones, para cada necesidad disponibilidad en múltiples plataformas.

Visual Paradigm resulta una herramienta amigable para el usuario ya que contiene facilidades para redactar especificaciones, utilizando plantillas que se encuentran definidas o que pueden ser creadas por los usuarios, permite la sincronización entre diagramas de entidad relación y diagramas de clases, la generación de código / ingeniería inversa.(International, 2006)

1.8 Diseño de software

Son los que definen la estructura de un sistema software, los cuales a su vez se componen de subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema. (Fabien Potencier, 2008)

1.8.1 Arquitectura de software

Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que

cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas. Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los Componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas. Este escenario contrasta con la aproximación monolítica típica de muchos programas Java. Todos tienen un Frame que contiene todos los elementos, un controlador de eventos, un montón de cálculos y la presentación del resultado. Ante esta perspectiva, hacer un cambio aquí no es nada trivial. (DCC.)

1.8.2 Patrones de software

Un patrón es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación.
Muchos diseñadores y arquitectos de software han definido el término de patrón de diseño de varias formas que corresponden al ámbito a la cual se aplican los patrones. Luego, se dividió los patrones en diferentes categorías de acuerdo a su uso. Los diseñadores de software extendieron la idea de patrones de diseño al proceso de desarrollo de software. Debido a las características que proporcionaron los lenguajes orientados a objetos (como herencia, abstracción y encapsulamiento) les permitieron relacionar entidades de los lenguajes de programación a entidades del mundo real fácilmente, los diseñadores empezaron a aplicar esas características para crear soluciones comunes y reutilizables para problemas frecuentes que exhibían patrones similares.

1.8.2.1 Patrones de arquitectura

En la figura siguiente, vemos la arquitectura MVC en su forma más general. Hay un Modelo, múltiples Controladores que manipulan ese Modelo, y hay varias Vistas de los datos del Modelo, que cambian cuando cambia el estado de ese Modelo.
La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

El *Modelo* es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

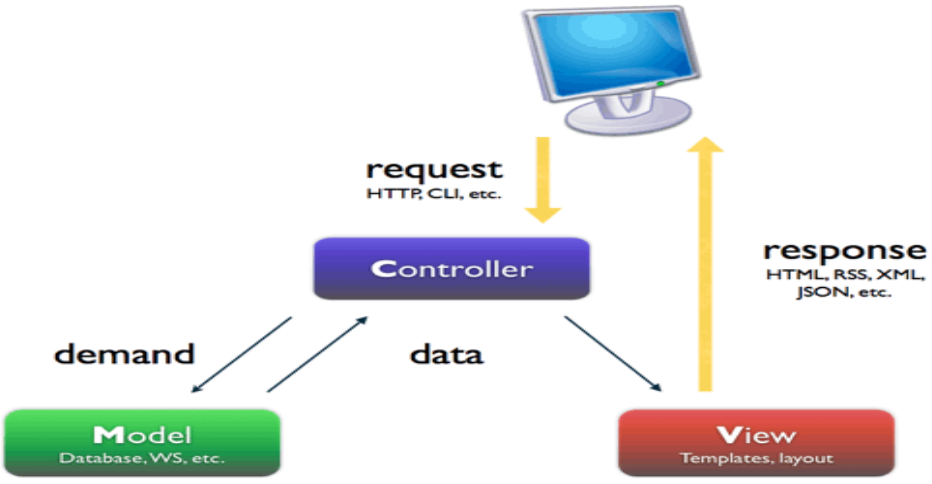


Figura 11: Relación Modelo-Vista-Controlador.

En el componente de Estandarización, la capa del **modelo** define la lógica de negocio (la base de datos pertenece a esta capa). Symfony guarda todas las clases y archivos relacionados con el modelo en el directorio lib/model/.

La **vista** es lo que utilizan los usuarios para interactuar con la aplicación (los gestores de plantillas pertenecen a esta capa). En Symfony la capa de la vista está formada principalmente por plantillas en PHP. Estas plantillas se guardan en varios directorios llamados templates/ repartidos por todo el proyecto.

El **controlador** es un bloque de código que realiza llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario. Todas las peticiones se canalizan a través de los controladores frontales (index.php y frontend_dev.php). Estos controladores frontales realmente delegan todo el trabajo en las acciones.

1.8.2.2 Patrones de diseño

El sistema emplea numerosos patrones de diseño, algunos de los más usados serán descritos a continuación:

1.8.2.2.1 Patrones GRASP

Para la Asignación general de Responsabilidad (o patrones GRASP por sus siglas en ingles), los mismos constituyen principios básicos a tener en cuenta cuando se quiere construir eficazmente un software orientado a objetos. Entre los más evidentes en el diseño propuesto se encuentran los patrones: Creador, Experto, Alta Cohesión, Bajo Acoplamiento y Controlador.

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

Creador: Este patrón como su nombre lo indica es el que crea, el guía la asignación de responsabilidades relacionadas con la creación de objetos, se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:

1. B contiene A
2. B es una agregación (o composición) de A
3. B almacena A
4. B tiene los datos de inicialización de A (datos que requiere su constructor)
5. B usa A.

A la hora de crear objetos se deben tener en cuenta las características de la clase.

Bajo Acoplamiento: El acoplamiento es una medida de fuerza con que un elemento está a, tiene conocimiento de, confía en, otros elementos. Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador

no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema. (Larman, 1999)

Alta Cohesión: La cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento, Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable, una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo. (Larman, 1999).

Controlador: Es un evento generado por actores externos. Se asocian con operaciones del sistema, operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos. Normalmente un controlador delega en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad. No realiza mucho trabajo por sí mismo. (Larman, 1999).

Decorador: Responde a la necesidad de añadir dinámicamente funcionalidad a un Objeto. Esto nos permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera. (Fowler).

Su aplicación se basa principalmente en los siguientes casos:

- Añadir objetos individuales de forma dinámica y transparente
- Responsabilidades de un objeto pueden ser retiradas
- Cuando la extensión mediante la herencia no es viable
- Hay una necesidad de extender la funcionalidad de una clase, pero no hay razones para extenderlo a través de la herencia.
- Hay la necesidad de extender dinámicamente la funcionalidad de un objeto y quizás quitar la funcionalidad extendida.

1.8.2.2.2 Patrones GOF

Para contribuir a una implementación eficiente se hizo necesario el estudio de los patrones del grupo de los cuatro (GOF, Gang Of Four), de ellos se seleccionaron cuatro: Singleton, Abstract Factory, Decorator, Composite, los cuales se explicarán a continuación.

- ✓ En la categoría Creacionales:

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En las acciones se usan los métodos ->getRequest (), ->getUser (), esto se debe a que, en la acción, el método getContext () guarda una referencia a todos los objetos del núcleo de Symfony, estos métodos pueden ser accedidos desde la vista y desde el controlador, sólo varía la forma de llamarlos. (Elers Pérez, 2009)

- ✓ En la categoría Estructurales:

Decorator (Envoltorio): Añade funcionalidad a una clase, dinámicamente. En cada archivo layout.php se define el código HTML común de cada vista, evitando que este sea repetido en cada página.

1.9 Marcos de trabajo

Un Marco de Trabajo, en el desarrollo de software es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. (CodeBox, 2001)

El Marcos de Trabajo simplifica el desarrollo de las aplicaciones mediante la automatización de muchas de las tareas comunes. Además, un Marco de Trabajo proporciona estructura al código fuente, forzando al programador a crear código más legible y más fácil de mantener. (ExtJs, 2004)

1.9.1 Marco de trabajo ExtJs 3.0

ExtJs es una librería JavaScript ligera y de alto rendimiento, compatible con la mayoría de navegadores que nos permite crear páginas e interfaces web dinámicas (ExtJs, 2006). Permite realizar aplicaciones Web enriquecidas basándose en tecnología AJAX, JSON, DHTML y DOM. Ext 2.0 está patentado bajo licencia LGPL lo que posibilita su uso para aplicaciones empresariales privativas de código cerrado.

Con ExtJs, se pueden desarrollar aplicaciones web multiplataforma con facilidad. El modelo de componente de ExtJs mantiene su código bien estructurado por lo que incluso las aplicaciones más grandes pueden ser de fácil mantenimiento.

1.9.2 Marco de trabajo Symfony

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

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares)
- Independiente del sistema gestor de bases de datos
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web
- Preparado para aplicaciones empresariales y adaptable, a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. (Fabien Potencier, 2008)

1.10 Ides de desarrollo

1.10.1 NetBeans

NetBeans IDE es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE,

además es un producto libre y gratuito sin restricciones de uso. (Oracle Corporation, 2010). Posee un amplio soporte para el lenguaje PHP así como para el Marco de Trabajo Symfony y el Marco de Trabajo ExtJs.

Para el desarrollo de la solución se ha elegido NetBeans IDE, en su versión 6.9, como Entorno de Desarrollo Integrado.

1.11 Lenguajes de programación

Un lenguaje de programación es un conjunto de sintaxis y reglas semánticas que definen los programas del computador. Es una técnica estándar de comunicación para entregarle instrucciones al computador. Un programa escrito en un lenguaje de programación necesita pasar por un proceso de compilación, interpretación o intermedio, es decir, ser traducido al lenguaje de máquina para que pueda ser ejecutado por el ordenador. (Tucumán, 2009)

1.11.1 Lenguajes utilizados en el lado del servidor

El sistema GINA se ha implementado sobre la base del PHP como lenguaje de programación del lado del servidor.

1.11.1.1 PHP 5.2

El **PHP** es un lenguaje de script incrustado dentro del HTML. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas.

Es un lenguaje interpretado de alto nivel, diseñado originalmente para la creación de páginas web dinámicas de manera rápida y fácil. Es un lenguaje de programación usado principalmente en interpretación del lado del servidor. Esta versión sigue mejorando el soporte para la Programación Orientada a Objeto así como los temas de seguridad, el rendimiento y manejo de excepciones (PHP.net, 2001)

- **Velocidad:** No sólo la velocidad de ejecución, la cual es importante, sino además no crear demoras en la máquina. Por esta razón no debe requerir demasiados recursos de sistema.

PHP se integra muy bien junto a otro software, especialmente bajo ambientes Unix, generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz.

- **Estabilidad:** Ninguna aplicación es 100% libre de errores, pero PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente. PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
- **Seguridad:** PHP provee diferentes niveles de seguridad, estos pueden ser configurados de archivos de configuración que se encuentran en el servidor.
- **Simplicidad:** Es un lenguaje de programación simple de implementar por lo que usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente.

1.11.2 Lenguajes utilizados en el lado del cliente

Para realizar la programación en el lado del cliente fueron elegidos los siguientes lenguajes:

1.11.2.1 Java Script

Java Script es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, Java Script es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con Java Script se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (Pérez, 2011).

1.11.2.2 HTML

HTML es la sigla de **HiperText Markup Language (Lenguaje de Marcación de Hipertexto)** es un lenguaje es se utiliza comúnmente para establecer la estructura y contenido de un sitio web, tanto de texto, objetos e imágenes. Los archivos desarrollados en HTML usan la extensión .htm o .html.

El lenguaje HTML es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado W3C (World Wide Web Consortium). Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma

página HTML se visualiza de la misma manera en cualquier navegador de cualquier sistema operativo.

El propio W3C define el lenguaje HTML como “un lenguaje reconocido universalmente y que permite publicar información de forma global”. Desde su creación, el lenguaje HTML ha pasado de ser un lenguaje utilizado exclusivamente para crear documentos electrónicos a ser un lenguaje que se utiliza en muchas aplicaciones electrónicas. (Pérez, 2007)

1.11.2.3 CSS

CSS es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas.

La separación de los contenidos y su presentación presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados “documentos semánticos”). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes. (Pérez, 2007)

1.11.2.4 XHML

El **XHTML** (eXtensible Hypertext Markup Language) o Lenguaje de Etiquetado Hipertextual Extensible. Es una reformulación del lenguaje HTML como aplicación XML que se recoge en la Recomendación del World Wide Web Consortium (W3C) XHTML 1.0 The Extensible HyperText Markup Language. La familia del xhtml es el siguiente estadio en la evolución de Internet. A través del cambio a XHTML, los creadores de contenidos pueden adentrarse en el mundo XML con todos los beneficios que esto conlleva, a la vez que aseguran la compatibilidad de sus contenidos pasados y futuros."(Pérez, 2008).

1.11.3 Otros lenguajes

XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos, sirve para estructurar, almacenar e intercambiar información.

Representa una manera distinta de hacer las cosas, más avanzada, cuya principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. Así pues, el XML juega un papel importantísimo en este mundo actual, que tiende a la globalización y la compatibilidad entre los sistemas, ya que es la tecnología que permitirá compartir la información de una manera segura, fiable, fácil. (Web, 2001)

1.12 Herramientas de diseño de pototipos WEB

1.12.1 Axure RP 5.5

Axure RP 5.5 es una herramienta para la rápida creación de prototipos y especificaciones para aplicaciones y sitios web, usado principalmente por analistas y profesionales de experiencia, permite además crear diagramas de flujo. Axure es fácil de aprender, admite la navegación de los prototipos basado en un navegador con una estructura web para un mejor entendimiento de los mismos para su validación.

1.13 Gestor de base de datos

Sistema Gestor de Bases de Datos (SGBD) o DBMA (DataBase Management System). Es un software que proporciona servicios para la creación, el almacenamiento, el procesamiento y la consulta de la información almacenada en base de datos de forma segura y eficiente. Un SGBD actúa como un intermediario entre las aplicaciones y los datos. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

Los SGBD pueden residir una máquina diferente a la que ejecuta las aplicaciones. De hecho, las aplicaciones modernas se programan de forma que se puede utilizar esta característica de distribución física, aunque a la hora de instalar la aplicación no se utilice y se ubique todo el software en la misma máquina. Esto ha dado lugar a diferentes configuraciones de la arquitectura de las aplicaciones, todas ellas conocidas como arquitecturas multi-capa. (CAVSI.com)

1.13.1 Oracle 11 G

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

Con Oracle 11g se reducen los costes informáticos y ofrecer una calidad de servicio superior:

1. Consolidando las aplicaciones de negocio en redes de bases de datos rápidas, fiables y ampliables.
2. Maximizando la disponibilidad y eliminando la redundancia del centro de datos inactivo.
3. Comprimiendo datos en particiones de almacenamiento de bajo coste para un rendimiento más rápido.
4. Protegiendo con seguridad la información y permitiendo el cumplimiento.
5. Duplicando la productividad de la DBA y reduciendo el riesgo de cambios. (Corporation, 2002)

1.13.1.1 Principales ventajas de Oracle

- Las entidades complejas del mundo real y la lógica se pueden modelar fácilmente, lo que permite reutilizar objetos para el desarrollo de base de datos de una forma más rápida y con mayor eficiencia.
- Los programadores de aplicaciones pueden acceder directamente a tipos de objetos Oracle, sin necesidad de ninguna capa adicional entre la base de datos y la capa cliente.
- Las aplicaciones que utilizan objetos de Oracle son fáciles de entender y mantener porque soportan las características del paradigma orientado a objetos.
- Tiene buen rendimiento y hace buen uso de los recursos.
- Posee un rico diccionario de datos.
- Brinda soporte a la mayoría de los lenguajes de programación.
- Es un sistema multiplataforma, disponible en Windows, Linux y Unix.

Permite tener copias de la base de datos productiva en lugares lejanos a la ubicación principal. Las copias de la Base de Datos productiva pueden estar en modo de lectura solamente. (Corporation, 2002)

Conclusiones del capítulo 1

Se demostró con el estudio de los sistemas aduaneros existentes, a través del presente capítulo; que estos no contribuyen a la solución de las deficiencias plateadas en la problemáticas de este trabajo. Por lo que se pone de manifiesto la necesidad de crear un nuevo sistema; mediante el uso de las herramientas estudiadas, que brindaran un gran apoyo para el inicio del mismo y que permitirán sentar las bases para el comienzo del desarrollo del componente.

Capítulo 2. Análisis y diseño de la solución

2 Introducción

En este capítulo se muestran todas las definiciones del negocio, a través de procesos modelados en BPMN, captura y especificación de requisitos, entre otros.

Se pretende además, alcanzar un enfoque general de las actividades que se llevan a cabo, estos procesos se analizan de forma detallada, identificando quienes participan, los artefactos y las reglas del negocio, la comunicación del módulo DAL con otros subsistemas, la composición que presenta la base de datos, además de los diagramas de clases del sistema, modelos de datos y modelo conceptual. Se aplicarán métricas al diseño y se reflejará el uso de patrones.

2.1 Procesos del negocio

Según Henry J. Johansson un proceso de negocio es un conjunto de actividades relacionadas que permiten crear un producto o servicio final a través de la transformación de uno o varios productos o servicios iniciales. El impulso del proceso es el que debe aportar valor a las entradas iniciales.

Para desarrollar un proceso es imprescindible conocer al detalle el funcionamiento del negocio en la entidad y a su vez cada pieza del mismo. En la compresión de dicho proceso es obligatorio que analistas y clientes se comuniquen entre ellos, utilizando un lenguaje natural y sencillo, creando una comunicación de excelencia, para cualquier intercambio de datos que sea necesario manejar entre ellos.

A partir de la información captada, se definieron los trabajadores, involucrados y artefactos asociados a los procesos del negocio, que se detallan a continuación.

2.1.1 Trabajadores del negocio

Los trabajadores puntualizan el perfil y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan unidos como un conjunto. Ellos realizan las actividades y son propietarios de elementos. (BOLIVIA, 2009)

A cada uno de estos trabajadores de la aduana se le asigna un grupo de funciones que se corresponden con los procesos pertenecientes a la DAL.

Trabajadores	Descripción
Abogado de la DAL	Son los encargados de realizar todo el estudio preliminar para la selección de los datos válidos a estandarizar, mediante la consulta hecha a las normas legislativas e instrumentos jurídicos. Permitiendo definir elementos estándares y editables.
Técnico Revisor	Es el encargado de llevar a cabo el proceso de estandarización de instrumentos jurídicos. Desarrollando la estructura inicial, identificada mediante consultas a los elementos definidos por los abogados. Una vez identificados los elementos válidos, este confecciona la propuesta final de estándar de documentos jurídicos.

Tabla1: Trabajadores del negocio

2.1.2 Reglas del negocio

Las reglas de negocio describen políticas que deben cumplirse o condiciones que deben satisfacerse, por lo que regulan algún aspecto del negocio.

- **Reglas Textuales:** Contienen "instrucciones", se expresan de forma libre (no estructurada) en lenguaje natural.

- **Reglas del Modelo de Datos:** Engloba todas aquellas reglas que se encargan de controlar que la información básica almacenada para cada atributo o propiedad de una entidad u objeto sea válida.

- **Reglas de Relación:** Incluye todas aquellas reglas que controlan las relaciones entre los datos. Estas reglas especifican, por ejemplo, que todo pedido debe ser realizado por un cliente, y que el mismo debe ser atendido. Además, una vez que un cliente haya hecho algún pedido, se deberá garantizar que no es posible eliminarlo, a menos que previamente se eliminen todos sus pedidos.

- **Reglas de Derivación:** Es frecuente que a partir de cierta información se pueda derivar otra, este conjunto de reglas especifican y controlan la obtención de información que se puede calcular a partir de la ya existente.

Relación de las reglas de negocio

No:	1
Tipo:	Regla de derivación
Nombre:	Correspondencia
Descripción:	Esta regla determina que cualquier documento de corte legal que sea elaborado o modificado en la aduana o por la aduana, se acoja a las especificidades que establece el Manual de Procedimientos de la Dirección de Asuntos Legales en su epígrafe III Elaboración de Instrumentos Jurídicos.

No:	2
Tipo:	Regla textual
Nombre:	Identificación
Descripción:	Esta regla determina que cualquier elemento identificado en los instrumentos jurídicos pertenecientes a la DAL propenso a ser estandarizado, no debe afectar el propósito original del Instrumento.

No:	3
Tipo:	Regla del Modelo de Datos
Nombre:	Funcionalidad
Descripción:	Esta regla establece que los datos seleccionados sean completamente válidos y funcionales, de manera que no limite el estándar sino que facilite el trabajo haciéndolo más flexible y configurable.

2.1.3 Modelo conceptual

En el presente modelo conceptual (**figura 1)**, se muestran todos los conceptos relacionados con la DAL.

Figura 1: Diagrama de Objetos: Modelo Conceptual de Resoluciones Aduanera

Los elementos principales a mostrar en el modelo conceptual son:

Conceptos. Elemento lógico o físico que ayuda a entender el problema, es parte del lenguaje utilizado por el cliente y generalmente se nombra como sustantivo. Se representan con el símbolo de una clase. (Orosco, 2009)

Atributos. Información que caracteriza al concepto en el mundo real. Se muestra en el segundo compartimiento de las clases. (Orosco, 2009)

Asociaciones. Relaciones lógicas o físicas que existen en el mundo real entre dos conceptos. Si se puede armar una frase con dos conceptos, significa que se puede representar mediante una relación de asociación entre esos dos conceptos. Puede colocársele el verbo que se usa para relacionar los conceptos en la frase, mostrándolo sobre la asociación con una punta de una flecha para indicar la dirección en que se debe leer la frase. (Orosco, 2009)

Rol. El rol también puede aclarar la relación entre dos conceptos, indica el rol que juega un concepto con respecto a otro en una relación de asociación. (Orosco, 2009)

Multiplicidad. El número de instancias de un concepto relacionados con el otro concepto. Ejemplo: Una póliza tiene una lista de uno a diez beneficiarios. (Orosco, 2009)

Generalización. En lugar de poner una asociación para armar la frase “es-un-tipo-de” se puede poner una generalización. Aunque esto podría crear confusión en los lectores no técnicos; por lo que hay que asegurarse que el lector del modelo entienda perfectamente el significado de la notación. Ejemplo: El Plan Oro es un tipo de plan de seguro de vida, al igual que el plan tradicional. (Orosco, 2009)

Agregación y composición. Indican una relación donde uno de los conceptos es el contenedor del otro. (Orosco, 2009)

Para mayor comprensión se incluyen las entidades descritas en el diccionario de datos, en la sección de anexos.

2.1.4 Artefactos a generar

Dentro del ciclo de desarrollo del software es necesario documentar utilizando los productos o artefactos que propone el modelo o metodología a seguir, en este caso dígase el Modelo de desarrollo de software orientado a componentes, así como las disposiciones hechas por el proyecto aduana.

Por su parte un artefacto es un segmento de información que es producido, modificado o usado en cada una de las fases. Valiosos para la conclusión de cada etapa, los artefactos son los resultados tangibles del proyecto, los entes que se van creando y usando hasta obtener el producto final. (Jacobson, 2000)

Para el avance del presente trabajo, se consideraron los artefactos propuestos por el proyecto aduana.

Artefactos	Descripción
Modelado de procesos de negocios.	Son los modelos realizados para plasmar en mayor medida todos los detalles de cada uno de los procesos de negocio, para facilitar la comunicación entre el usuario y el equipo de desarrollo.
Descripción de Proceso de Negocio.	Describir los procesos del negocio, detallando el comportamiento de cada proceso identificado en el Mapa de Procesos. (GEIGE, 2009)
Lista de requisitos.	Documento en el cual se listan todos los requisitos capturados con el cliente.
Especificación de requisitos del Software.	Describir los requisitos que deben ser cumplidos por el software. (GEIGE, 2009)
Prototipos de interfaz de usuario	Validar y/o explorar el diseño de la interfaz de usuario. El grado de formalidad y herramientas utilizadas para generarlo puede variar mucho de proyecto en proyecto, pudiendo ir desde sólo unas cuantas imágenes de pantallas hasta un esqueleto de interfaz de usuario ejecutable producido en un ambiente de Desarrollo rápido de aplicaciones (RAD: Rapid Aplicación. (GEIGE, 2009)
Glosario de términos.	Describir las definiciones concisas para favorecer la comunicación entre todos los involucrados. Los miembros del proyecto utilizarán el glosario inicialmente para comprender sus términos específicos. (GEIGE, 2009)
Modelo conceptual	Explicar los conceptos más significativos en un dominio del problema, identificando los atributos y las asociaciones, que representa cosas del mundo real. (GEIGE, 2009)
Modelo de diseño.	Contener los diagramas de clases y de secuencia.

Diagrama de clases	Describir la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.
Diagrama de secuencia Orientado a Actividades del Negocio.	Provee un mapa secuencial del paso de los mensajes entre los objetos a lo largo del tiempo.
Diagrama de componentes.	Mostrar los componentes y sus relaciones.
Modelo de Datos	Describir las estructuras de datos físicas del sistema, es decir los objetos de la base de datos y las relaciones entre ellos. Debe describir además las operaciones de manipulación y recuperación de la información. (GEIGE, 2009)
Diseño de casos de prueba	Definir paso a paso las instrucciones que indican cómo llevar a cabo una prueba. Pueden ser documentos con información textual que describan cómo realizar la prueba manualmente o archivos de instrucciones legibles por las máquinas que posibiliten la ejecución automatizada de la prueba. (GEIGE, 2009)

Tabla 2: Artefactos definidos por el proyecto Aduana.

2.2 Modelado del proceso de negocio y subprocesos de negocio

Se realizó el modelado de los procesos y subprocesos que ocupan esta investigación relacionados al negocio del subsistema de la DAL, para lograr un mayor entendimiento empleando la notación BPMN y la herramienta CASE Visual Paradigm.

2.2.1 Proceso de gestión de estándares para instrumentos jurídicos

El proceso de Gestión de Estándares para Instrumentos Jurídicos **(figura 2)**, se encarga de establecer todos los parámetros necesarios, para el manejo adecuado de la estandarización.

Iniciando por la entrevista con el cliente, el técnico revisor presenta la solicitud de selección de datos para comenzar la estandarización; dando lugar al origen del subproceso de Selección de Datos (**figura 3)**, que permite al abogado de la DAL, mediante las consultas a normas legislativas e instrumentos jurídicos, definir todos los elementos estándares o editables. Luego es aprobada por el abogado la lista de elementos que le permiten al técnico revisor continuar con el proceso de gestión de dichos estándares.

Inmediatamente se procede a identificar la estructura de la lista de elementos **(figura 2)** que permite fijar la correspondencia de campos estáticos o editables. El técnico revisor establece las regiones del Marco Legal y actualiza la estructura erigida, para definir el apartado dispositivo por el cual se elaborarán las regiones del cuerpo legal que generan al estándar. Finalmente el abogado se encarga de validar todos los estándares y liberarlos para su desarrollo.

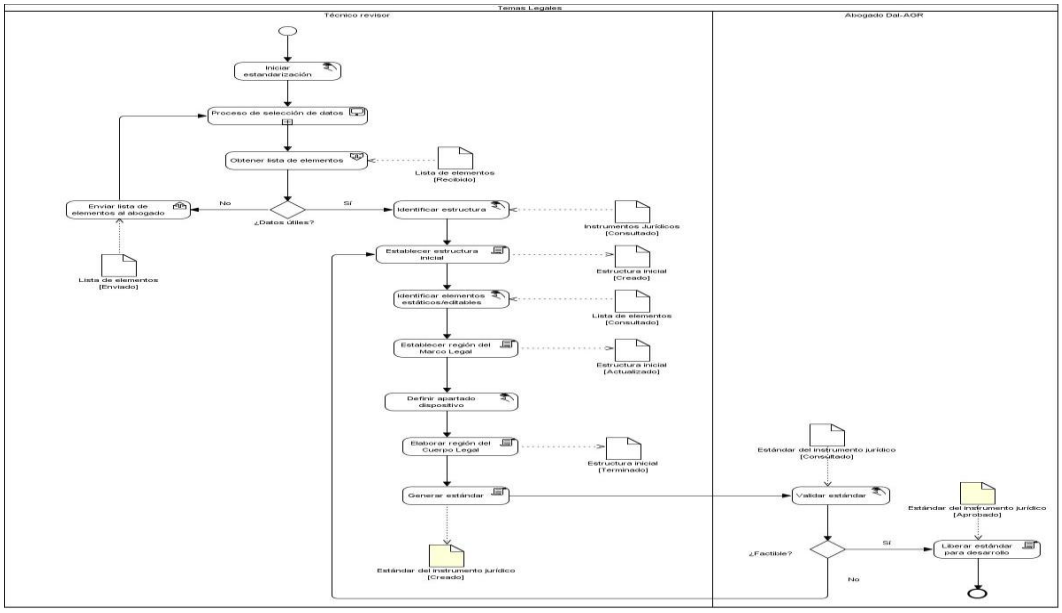


Figura 2. Modelado de Proceso de Gestión de Estándares para Instrumentos Jurídicos.

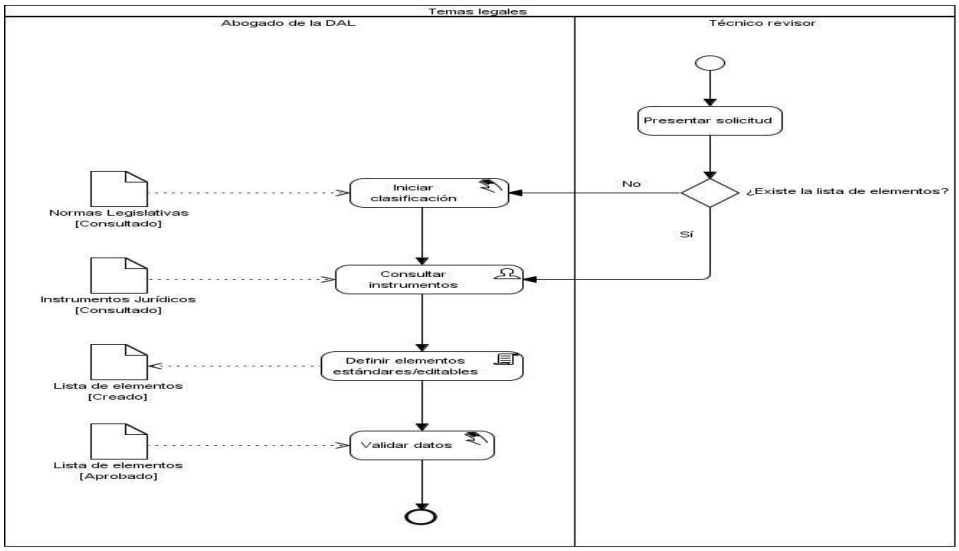


Figura 3. Modelado del Subproceso de Selección de Datos

2.2.2 Estándar

La **tabla 3** muestra el estándar liberado para el desarrollo del componente de estandarización, obtenido en los talleres con el cliente. Se muestran sus campos estáticos y editables así como todo el marco de su estructura definida previamente por los abogados de la Dirección de Asuntos Legales de la Aduana General de la República de Cuba.

<<Título General>>
<<Escudo>>
<<Unidad>>
<<Número Resolución>>
Por Cuanto: En fecha……..., en el área de……., se detectó a……..., con pasaporte no. …… con domicilio en……**.** Procedente de con destino a en la Línea Aérea…… Vuelo No…… que hubo de……

Por Cuanto: Los hechos que se describen, constituyen una violación de los dispuesto en la legislación aduanera vigente, …….. **Por Cuanto:** El artículo 206 del decreto Ley 162, de Aduanas de 3 de abril de 1996, estipula que el Jefe de la Aduana General de la República, designará a las autoridades facultadas para imponer las sanciones por las infracciones administrativas aduaneras. **Por Cuanto:** En virtud de lo dispuesto en la Resolución No. …….. Del Jefe de la Aduana General de la República, queda facultado el que resuelve para imponer la sanción que hace referencia esta Resolución. **Por Tanto:** En el ejercicio de las facultades que me están conferidas,…… << Norma Sanción>>
<<Resuelvo>> PRIMERO: …… SEGUNDO: …….. TERCERO……..... NOTIFÍQUESE:…… ARCHÍVESE: el original firmado en el consecutivo de Resoluciones del área legal de esta Aduana. DADA: en....., a los días……..de " Año…… del triunfo de la Revolución" Autoridad Facultada_____ Firma del Infractor Notificado_____ <<Norma Precepto>>

Tabla 3. Estándar Liberado

2.2.2 Definición de requisitos.

El flujo de trabajo de Levantamiento de Requisitos radica su mayor esfuerzo en el establecimiento de un acuerdo entre los clientes y los desarrolladores, sobre lo que el sistema debe hacer. Se precisa el ámbito del sistema y se pretende entender el comportamiento del software definiendo una interfaz enfocada a las

necesidades y metas del usuario. Este flujo de trabajo intenta proveer a los desarrolladores de un mejor entendimiento de los requisitos del sistema. (Jacobson, et al., 2004).

A continuación se listan los requisitos identificados:

Requisitos funcionales	Descripción
Gestionar estándar	Este requisito funcional permitirá de forma general, el manejo de todos los estándares como estructuras primarias para los instrumentos jurídicos.
Crear estándar	Este requisito permitirá la creación de un nuevo estándar, que no es más que la visualización de un estándar seleccionado, correspondiente con el nuevo documento a crear.
Crear Documento Resolución Multa ➢ Crear documento Resolución de Multa por vía comercial ➢ Crear documento Resolución de Multa por vía no comercial	Permitirá la creación de un nuevo documento de tipo Multa, que no es más que la visualización de un estándar específico al documento en cuestión, al cual se le insertan datos, correspondientes con su tipo.
Crear Documento Resolución Decomiso ➢ Crear documento Resolución de Decomiso por vía comercial ➢ Crear documento Resolución de Decomiso por vía no comercial	Permitirá la creación de un nuevo documento de tipo Decomiso, que no es más que la visualización de un estándar específico al documento en cuestión, al cual se le insertan datos, correspondientes con su tipo.
Crear Documento Resolución Apelación	Permitirá la creación de un nuevo documento de tipo Apelación, que no es más que la visualización de un estándar especifico al documento en cuestión, al cual se le insertan datos,

	correspondiente con el nuevo documento a crear.
Crear Documento Resolución Sanción Conjunta	Permitirá la creación de un nuevo documento de tipo Sanción Conjunta, que no es más que la visualización de un estándar especifico al documento en cuestión, al cual se le insertan datos, correspondiente con el nuevo documento a crear.
Eliminar estándar	Permitirá eliminar un estándar seleccionado, de acuerdo a un cambio en la norma que rige su estructura básica.
Exportar estándar	Permitirá exportar un estándar después de introducidos los datos y creado el instrumento jurídico, en formato pdf.
Validar datos	Permitirá validar los datos introducidos en los campos editables de acuerdo a los siguientes tipos: numérico, alfabético y alfanumérico.

2.2.3 Prototipos de interfaz de usuario

El diseño de interfaces de usuario es una tarea que ha adquirido relevancia en el desarrollo de un sistema. La calidad de la interfaz de usuario puede ser uno de los motivos que conduzca a un sistema al éxito o al fracaso. Los principios que se presentan son de utilidad para creación de interfaces funcionales y de fácil operación. (ExtJs, 2004)

La Interfaz de usuario, en lo adelante IU, de un programa es un conjunto de elementos hardware y software de un computador que presentan información al usuario y le permiten interactuar con la información y con el ordenador. Si la IU está bien diseñada, el usuario encontrará la respuesta que espera a su acción. Si no es así puede ser frustrante su operación, ya que el usuario habitualmente tiende a culparse a sí mismo por no saber usar el objeto.

Los siguientes prototipos de interfaz fueron diseñados para usuarios con distintos niveles de conocimientos, desde principiantes hasta expertos. En la **(figura 4)** se muestra la ventana principal de la aplicación con los eventos y campos necesarios para que el usuario interactúe de manera sencilla y amena con todos los estándares de los instrumentos jurídicos, asimismo la **(figura 5)** muestra un ejemplo de uno de los estándares. Se especifican los restantes prototipos de interfaz en los anexos de este trabajo de diploma.

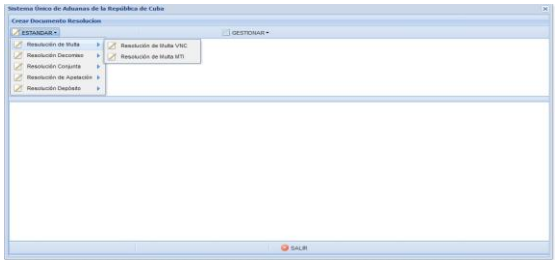


Figura 4. Pantalla Principal de la Interfaz de Usuario

Figura 5. Prototipo de pantalla de Interfaz de Usuario para Estándar

2.3 Modelo de diseño

El modelo de diseño es una abstracción del modelo de implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a la implementación. El modelo de diseño puede contener: los diagramas, las clases, paquetes, subsistemas, cápsulas, protocolos, relaciones, colaboraciones, atributos, las realizaciones de los casos de uso y los prototipos de interfaces de usuario, entre otros que se puedan considerar para el sistema en desarrollo. (Pedre, 2011)

El diseño de sistemas se define como el proceso de aplicar ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso o un Sistema, con suficientes detalles como para permitir su interpretación y realización física.

El diseño del sistema encierra cuatro etapas:

Trasforma el modelo de dominio de la información, creado durante el análisis, en las estructuras de datos necesarios para implementar el software.

1. El diseño de los datos**.**

Define la relación entre cada uno de los elementos estructurales del programa.

2. El diseño arquitectónico***.***

Describe como se comunica el Software consigo mismo, con los sistemas que operan junto con él y con los operadores y usuarios que lo emplean.

3. El diseño de la interfaz***.***

4. El diseño de procedimientos***.***

Transforma elementos estructurales de la arquitectura del programa. La importancia del diseño del Software se puede definir en una sola palabra ***Calidad***, dentro del diseño es donde se fomenta la calidad del proyecto. El diseño es la única manera de materializar con precisión los requerimientos del cliente. (Pedre, 2011)

2.3.1 Diagrama de paquetes

El diagrama de paquetes brinda información del módulo DAL y plasma el vínculo con diferentes paquetes dentro de los que sobresale el **LIB,** es el responsable de acceder a la capa de almacenamiento de datos.

Dentro de los subsistemas que se relacionan con el componente de estandarización se encuentra TC que es el encargado de administrar las clases que son nomencladoras y comunes para todos los subsistemas del GINA. También se halla MTI brindando servicios especializados sobre confirmación de información. Se utilizan los subsistemas eAduana para el trabajo con los archivos y DateTime para las soluciones de edición de fechas. Así como la interacción con el núcleo de Symfony y la librería. Dentro del mismo módulo se materializa el vínculo con diferentes paquetes dentro de los que sobresale el WEB encargado de la capa de presentación, el Controlado que presenta las clases y objetos para el control de las peticiones y el flujo de acciones a realizar y el LIB donde se encuentran las clases del negocio, los formularios y algunas de las encargadas del mapeo y abstracción de la base de datos.

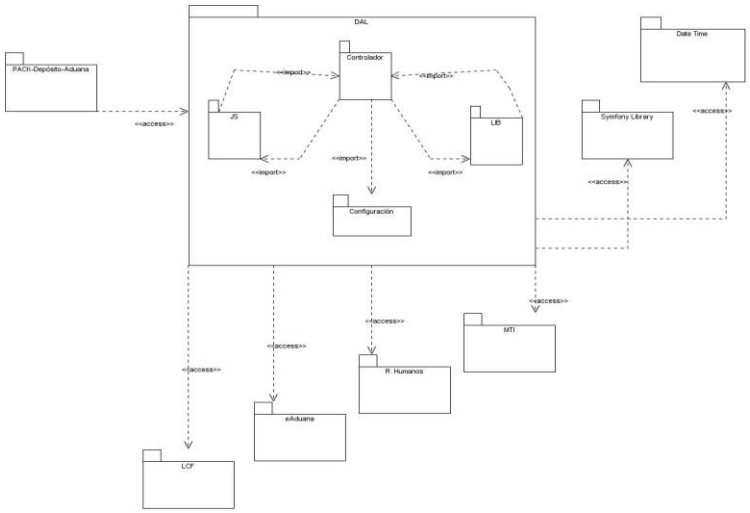


Figura 6. Diagrama de Paquetes del Componente de Estandarización de Instrumentos Jurídicos.

2.3.2 Diseño de la base de datos

El modelo entidad-relación **figura 7** generado para el sistema contiene las tablas (y las relaciones entre ellas) que guardarán la información de los IJ del sistema Dirección de Asuntos Legales.

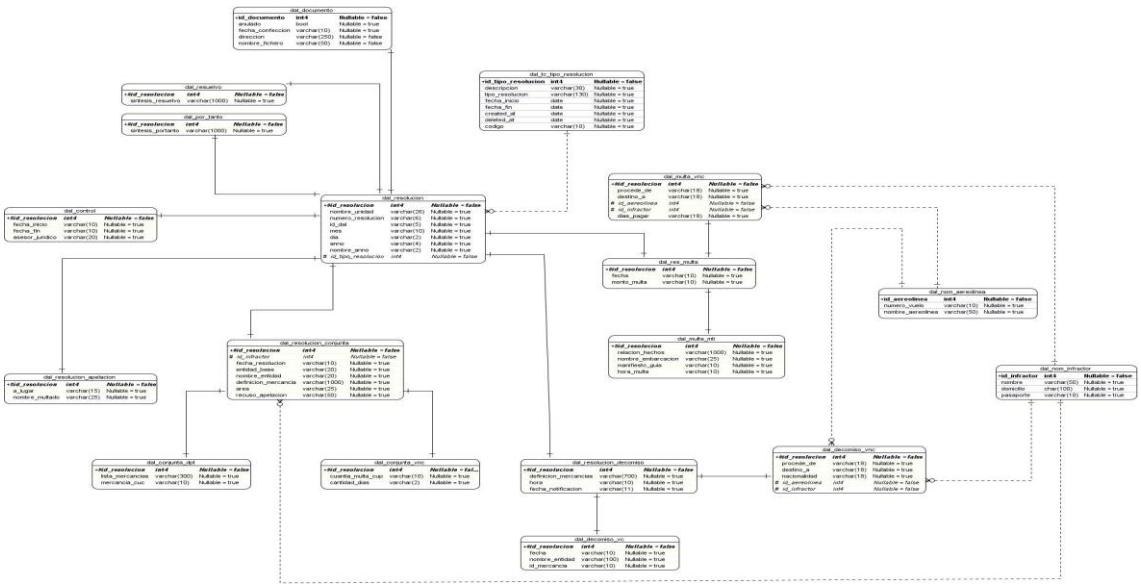


Figura 7. Modelo Entidad-Relación del módulo DAL

2.3.2.1 Modelos Entidad Relación

Las tablas de color verde son las más importantes arquitectónicamente debido a que en ellas se almacenan los datos principales de los Estándares de los instrumentos jurídicos.

Las tablas de color carmelita son del tipo hijas con una fuerte presencia de la herencia, donde se tiene una o varias clases padre o superclase, y una o varias clase hija o subclase.

Las tablas azules son nomencladoras, contribuyen al completamiento de la información de las tablas dichas anteriormente.

UML soporta tanto herencia simple como herencia múltiple. Todas en su conjunto conforman un modelo de entidad-relación, normalizado en Tercera Forma Normal (3FN), según la propuesta original de (Cood, 1992) que plantea que una relación está en tercera forma normal si todos los atributos de la relación dependen funcionalmente sólo de la clave, y no de ningún otro atributo.

La normalización de los datos puede considerarse como un proceso durante el cual los esquemas de relación que no cumplen las condiciones se descomponen repartiendo sus atributos entre esquemas de relación más pequeños que cumplen las condiciones establecidas. Un objetivo del proceso de normalización realizado para el modelo antes presentado, es garantizar que no ocurran anomalías de actualización. (Cood, 1992)

2.3.3 Diagrama de clases del sistema

Un diagrama de clase comparte las mismas propiedades comunes como lo hacen todos los otros diagramas un nombre y contenido gráfico. (BOLIVIA, 2009)

Los diagramas de clase contienen, clases, interfaces, colaboraciones, y relaciones de dependencia, generalización y asociación.
El siguiente diagrama (**Figura 8)** muestra un total de 18 clases. De las clases más significativas se encuentran, **Resolución** que es contenedora de todas las funcionalidades y atributos como clase padre a ser heredados, la clase **Documento**, que es la que conserva los datos de los Documentos Resolución generado por los estándares. Se anexa el diagrama al artefacto modelo de diseño.

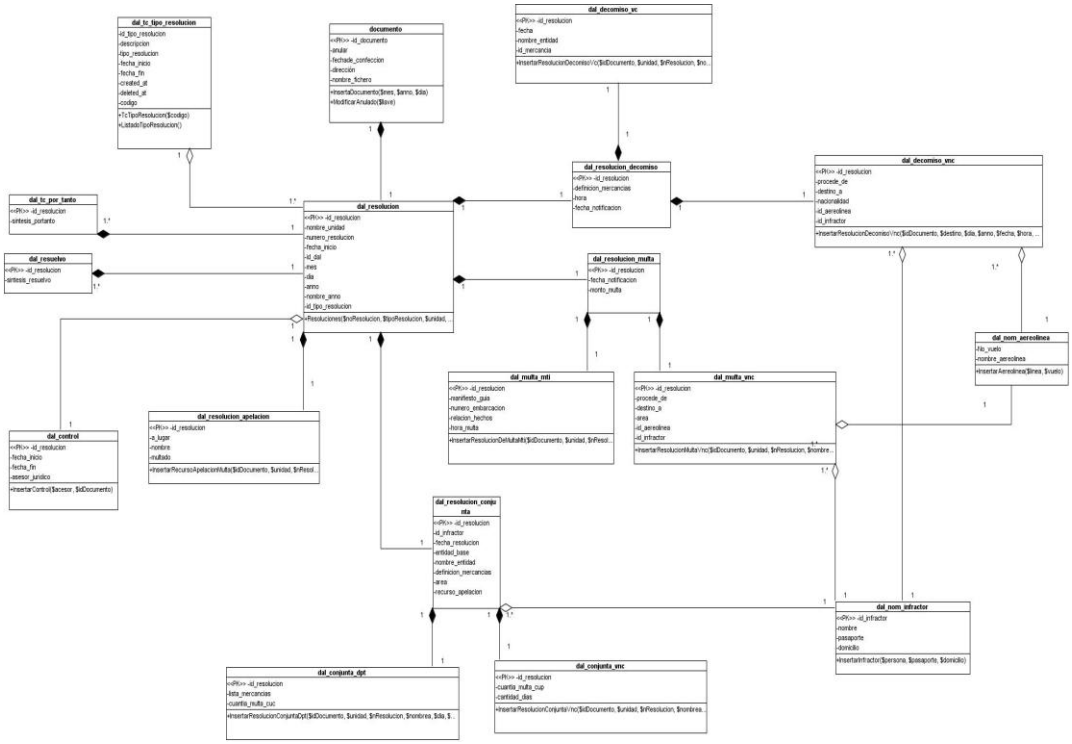


Figura 8. Diagrama de Clases del Sistema.

2.3.4 Modelado mediante estereotipos web

Para la realización de los Diagramas de Clases del Diseño será utilizada la extensión de UML para el modelado de aplicaciones Web (Laram, 1999).

Esta extensión presenta como elementos más significativos a tres clases UML: Server Page (Página Servidora), Client Page (Página Cliente) y Form (Formulario) permitiendo representar ficheros contenedores de sentencias script, empleadas para el código servidor, código cliente y formularios respectivamente. A continuación se brinda una explicación de cómo son usados estos estereotipos en el diseño de la propuesta del componente y qué representa cada cual:

<<Server Page>>: Representa la clase que tiene código que se ejecuta en el servidor, la cual se encarga de construir (build) o generar el resultado HTML y/o realizar peticiones a la capa inferior.

<<Client Page>>: Es una página Web con formato XHTML. Mezcla de datos, presentación y lógica. Son interpretadas por el navegador. Sus atributos son las variables declaradas dentro del script que son accesibles para cualquier función dentro de la página. Cada página cliente es construida por una sola página de servidor.

<<FormHTML>>: Es una colección de elementos de entrada que están contenidos en la página cliente. Sus atributos son los elementos de entrada del formulario (input boxes, radio buttons, check boxes, hidden fields, entre otros). No tienen operaciones, el método para el paso de los parámetros es $_POST y se comunican con las páginas servidores mediante submit. (Larman, 1999)

2.3.4.1. Relaciones entre los elementos del diseño

Entre las páginas servidoras pueden existir relaciones de inclusión (<<include>>).

Las páginas servidoras construyen el resultado XHTML que conforma el código cliente (<<build>>).

Los formularios forman parte del resultado XHTML (<<aggregation/ aggregation by).

Los formularios envían los datos al código servidor para su procesamiento (<<submit>>).

Entre las páginas clientes pueden existir vínculos (<<link>>) o redirecciones (<<redirect>>).

Las páginas clientes pueden incluir ficheros script (<<include>>).

DESDE/HASTA	Client Page	Form	Server Page
Client Page	<<link>> <<redirect>>	aggregation	<<link>>
Form	aggregation by		<<submit>>
Server Page	<<build>>		<<include>>

Figura 9. Relaciones entre los elementos del diseño.

2.3.5. Diagramas de clases del diseño

Los diagramas de clases del diseño son diagramas estáticos que describen la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de diseño de un sistema para crear el diseño conceptual de la información que se manejara en el mismo y los componentes que se encargaran su funcionamiento así como la relación entre ellos como se muestra en la **Figura 10**. (Laram, 1999).

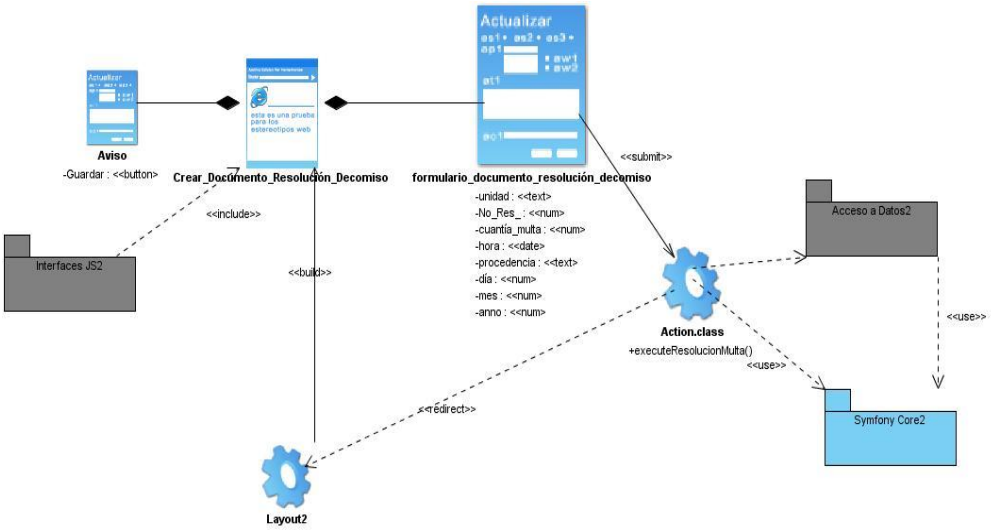


Figura 10. Diagrama de clases del diseño Insertar Resolución.

El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades, y su principal objetivo es la elaboración de los diagramas de clases de diseño, que muestra las clases participantes en la realización en un caso de uso con todos sus atributos.

2.3.6 Diagrama de interacción del diseño

El diagrama de interacción, representa la forma en cómo un Cliente (Actor) u Objetos (Clases) se comunican entre sí en petición a un evento. Esto implica recorrer toda la secuencia de llamadas, de donde se obtienen las responsabilidades claramente. (Chile, 2006)

Hay dos tipos de diagramas de interacción, ambos basados en la misma información, pero cada uno enfatizando un aspecto particular: Diagramas de Secuencia y Diagramas de Colaboración. (Pedre, 2011).

2.3.6.1 Diagrama de secuencia orientado a las actividades del negocio.

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el escenario. En aplicaciones grandes además de los objetos se muestran también los componentes y casos de uso. El mostrar los componentes tiene sentido ya que se trata de objetos reutilizables, en cuanto a los casos de uso hay que recordar que se implementan como objetos cuyo rol es encapsular lo definido en el caso de uso. (Montes, 2008)

Para mostrar la interacción con el usuario o con otro sistema se introducen en los diagramas de secuencia. En las primeras fases de diseño el propósito de introducir estas clases es capturar y documentar los requisitos de interfaz, pero no el mostrar, como se va a implementar dicha interfaz.
Los diagramas de secuencia, formalmente diagramas de traza de eventos o de interacción de objetos; documentan el diseño desde el punto de vista de los requisitos. Observando qué mensajes se envían a los objetos, viendo a groso modo cuánto tiempo consume el método invocado, los diagramas de secuencia nos ayudan a comprender los cuellos de botella potenciales, para así poder eliminarlos. A la hora de documentar un diagrama de secuencia resulta importante mantener los enlaces de los mensajes a los métodos apropiados del diagrama de clases. Los ejemplos de los diagramas de Secuencia Orientado a las Actividades del Negocio definidos por el proyecto Aduana, se encuentran anexos en el artefacto, Modelo de Diseño.

2.5 Evaluación del diseño propuesto

Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen medidas de la cohesión, acoplamiento y complejidad del subsistema. Estas tres medidas pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de los componentes (Pressman, 1998)

Entre los tipos de métricas que se utilizan para evaluar el diseño están:

- **Tamaño operacional de clase (TOC):** está dado por el número de métodos asignados a una clase.

- **Relaciones entre clases (RC):** está dado por el número de relaciones de uso de una clase con otras.

  Para evaluar el diseño se empleó la métrica Tamaño operacional de clase (TOC), la cual mide la calidad de acuerdo a los atributos Responsabilidad, Complejidad de implementación y Reutilización de las clases.

  Atributos de calidad que se abarcan en la métrica TOC:

- **Responsabilidad**. Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

- **Complejidad de implementación**. Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

- **Reutilización**. Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

2.6 Patrones Empleados

Decorador

Se utiliza en la composición recursiva para organizar un número abierto de objetos.
El decorador permite agregar responsabilidades a los objetos sin la utilización de subclases; evita la explosión de subclases que puede levantarse de intentar cubrir cada combinación de responsabilidades estáticamente. Por ejemplo, un textview y al cual le puedes agregar la funcionalidad de scrollbar o ponerle borde negro.

Bajo Acoplamiento

A cada clase Symfony se le asigna una responsabilidad, de forma tal que mantiene pocas dependencias entre las mismas.

Alta Cohesión

En cada clase Actions se definen las acciones para las plantillas, además estas colaboran con otras para realizar diferentes operaciones, se instancian objetos, se acceden a las properties, es decir en una Actions se utilizan diferentes funcionalidades estrechamente relacionadas entre sí, lo que proporciona un software flexible ante los cambios.

Los patrones mencionados anteriormente son los más visualizados y utilizados dentro del desarrollo de la solución. Todos están evidenciados principalmente en los Marco de Trabajo escogidos y en el diseño realizado para darle cumplimiento a los requerimientos.

2.7 Evaluación de las métricas

En este epígrafe se mostrarán las gráficas y las valoraciones que resultaron después de aplicar las métricas, para las cuales se tuvieron en cuenta el número de métodos de las clases Controladoras y Modelos reflejados en el diseño de acuerdo a la propuesta planteada por la arquitectura.

2.7.2 Métricas de software

Según Pressman en su libro Ingeniería de software, un enfoque práctico, plantea lo siguiente: “El proceso del software y las métricas del producto son una medida cuantitativa que permite a la gente del software tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un Marco de Trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software” (Pressman, 2005).

2.6.2.1 Métricas de usabilidad

Se pueden definir las mismas como aquellos criterios o variables que son medibles de forma objetiva. Mientras que la interpretación de una opinión es un análisis cualitativo o subjetivo por parte del experto, la interpretación de datos objetivos responde a un análisis cuantitativo. (Métricas, 2010).

De acuerdo con lo anterior el siguiente gráfico representa los resultados generales obtenidos de la aplicación de las métricas Tamaño operacional de clases, de acuerdo a la cantidad de métodos.

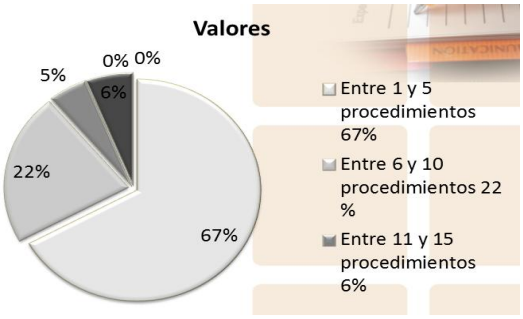


Figura 12: Gráfico de los resultados generales de acuerdo a la cantidad de procedimientos.

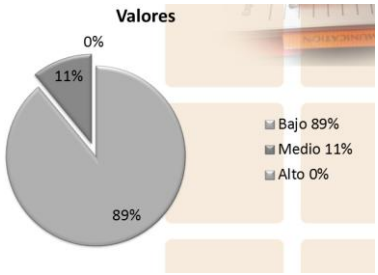


Figura 13: Gráfico de la Responsabilidad por clases.

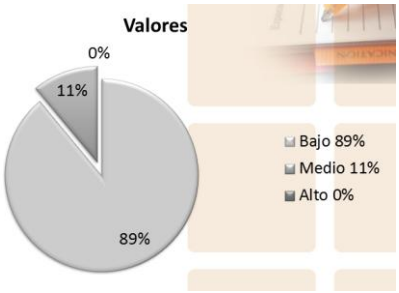


Figura 14: Gráfico de la Complejidad de implementación por clases.

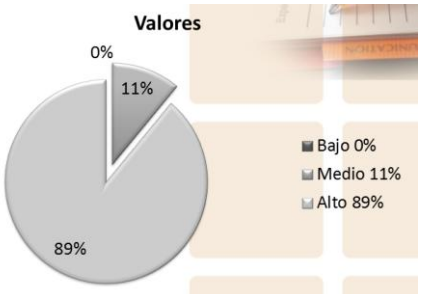


Figura 15: Gráfico del nivel de Reutilización de las clases.

Valoraciones del resultado de las métricas aplicadas

Analizando los resultados obtenidos en la evaluación de la métrica TOC se puede concluir que el diseño del componente de estandarización tiene una calidad aceptable teniendo en cuenta que el 67 % de las clases incluidas posee menos cantidad de operaciones que la mitad del valor máximo registrado en las mediciones. Además el 89% de las clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad **Figura 13**, Complejidad de implementación **Figura 14** y Reutilización **Figura 15**).

Conclusiones del capítulo

En este capítulo se le da inicio al análisis de la solución iniciándose con el modelo de negocio y la descripción de sus procesos y subprocesos. Se muestran los modelos conceptuales, los requisitos identificados así como las descripciones de aquellos que resultan significativos para la presente investigación. Se contemplaron las técnicas utilizadas para la captura de los requerimientos y los métodos de validación de los mismos. Se realizó el modelado del componente de estandarización, con los artefactos definidos por el proyecto Aduana, los cuales muestran los diagramas correspondientes al análisis y diseño. Por último se agregaron los prototipos de interfaces que demuestran que todos los requerimientos captados y descritos contienen todas las necesidades de los clientes, además se realizó una descripción de los patrones propuestos para realizar el diseño de la herramienta, quedando de esta manera el sistema listo para pasar a la fase de implementación.

Capítulo 3. Implementación y prueba

Introducción

En este capítulo se muestran todas las definiciones de la fase de implementación y prueba, a través de procesos de modelado de la implementación y la especificación de los estándares de codificación utilizados por el proyecto, el tratamiento de errores y la comunicación entre capas. Además se explica el diagrama de componentes para visualizar con más facilidad la estructura general del sistema y el acoplamiento del servicio.

Se realizarán casos de pruebas para validar los elementos e informar los resultados, permitiendo inferir la validez funcional del producto.

3.1 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros. (CIENFUEGOS, 2000).

3.1.1 Estándar de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. (Barcelona, 2001)

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La «mantenibilidad» del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento.

Aunque la legibilidad y la mantenibilidad son el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente es en la técnica de codificación. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas.

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. (Barcelona, 2001)

El objetivo de crear un estándar de codificación es que los programadores se rijan por el mismo a la hora de implementar. Este estándar debe servir para el personal del proyecto para identificar de forma sencilla cuál es el objetivo y las funcionalidades que brinda cada una de las clases, funciones y demás componentes de Software dada su nomenclatura, es necesario que esto se pueda identificar a simple vista. Además debe servir de guía para los implementadores que tienen que continuar el desarrollo de las aplicaciones luego. (CEIGE, 2011)

La implementación llevada a cabo para el desarrollo del módulo está regida por el documento “Propuesta de un estándar de codificación”. Comprende todas las aplicaciones desarrolladas bajo el Departamento de Soluciones para la Aduana del Centro de Informatización y Gestión de Entidades y todo el código generado bajo la tecnología y el lenguaje PHP y que utilicen la arquitectura regida por la utilización del Marco de Trabajo Arquitectónico Symfony. (CEIGE, 2011)

Como la implementación de las aplicaciones está basada en la utilización de un Marco de Trabajo que en su totalidad está codificado en inglés, es imposible desligar la implementación completamente de este idioma. Por lo tanto la implementación se realizará teniendo en cuenta los idiomas de inglés y español, incluso la combinación de ambos al mismo tiempo, sin olvidar la utilización de los sufijos y prefijos “Set”, “Get”, “Is” y “Max”. (Barcelona, 2001)

Teniendo en cuenta las políticas de las llaves se utilizará la tradicional de Unix de 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. (CEIGE, 2011)

Cuando se declara un arreglo inicializado con valores que contienen diversos índices y se extiende más allá del largo de una línea de código se abre el paréntesis y se continúa en la próxima línea utilizando una sangría. (CEIGE, 2011)

Para comentar una línea se utilizará el o los caracteres que dicta el lenguaje de programación utilizado, al igual que para los comentarios de bloque, es decir, que requiera más de un línea. (CEIGE, 2011).

3.1.2 Tratamiento de errores

El control de los errores que pueden producirse en una aplicación es una tarea importante del desarrollo, que muchas veces se deja aparcada o a la que se le presta escasa atención. El controlar estas situaciones de forma correcta nos permitirá ofrecer a nuestros usuarios siempre una respuesta, incluso aunque haya sucedido algo que impida realizar su petición. Estas respuestas irán desde un simple mensaje descriptivo del error, o puede incluir un manejo más complejo de la situación que intente resolver el problema por otros medios. (Echarte, 2005)

Las validaciones juegan el primer papel importante y son controladas en las vistas por medio de expresiones regulares y en el negocio por medio de los formularios. Todas tienen como función primordial, evitar la inserción de datos incorrectos en los diferentes campos, logrando así evitar ataques contra el sistema por esta vía.

El tratamiento de errores es una medida importante para la aplicación. Muchos son los errores cometidos por los usuarios, unos por accidentes y otros no tan accidentales. En los Documentos Resolución, creados en el subsistema Asuntos Legales, mediante el Componente de Estandarización, se controlan información importante que es perenne para la DAL de la ARG, en la toma de decisiones de su quehacer diario, es por eso que se realizan varios tipos de validaciones de errores.

3.1.3 Comunicación entre caspa

El Marco de Trabajo que se utiliza es Symfony y este utiliza el patrón arquitectónico MVC que separa el sistema en tres capas diferentes, por lo que se hace necesario realizar la comunicación entre las mismas.

La capa de la Vista es la encargada de recoger la información deseada y enviarla a través de sus formularios hacia la capa del controlador, a través de objetos JSON en algunos casos por medio de

peticiones AJAX para lograr mayor rapidez .En otras ocasiones por XML logrando el intercambio de información y objetos completos entre las capas de presentación y la del controlador.

En Symfony, la capa del controlador, que contiene el código que liga la lógica de negocio con la presentación, está dividida en varios componentes que se utilizan para diversos propósitos:

- El controlador frontal es el único punto de entrada a la aplicación. Carga la configuración y determina la acción a ejecutarse. (Potencier)
- Las acciones contienen la lógica de la aplicación. Verifican la integridad de las peticiones y preparan los datos requeridos por la capa de presentación. (Potencier)
- Los objetos *request, response y session* dan acceso a los parámetros de la petición, las cabeceras de las respuestas y a los datos persistentes del usuario. Se utilizan muy a menudo en la capa del controlador. (Potencier)
- Los filtros son trozos de código ejecutados para cada petición, antes o después de una acción. Por ejemplo, los filtros de seguridad y validación son comúnmente utilizados en aplicaciones web. Puedes extender el Marco de Trabajo creando tus propios filtros. (Potencier)

Symfony utiliza PROPEL que es un ORM para PHP, transformando las tablas de la base de datos en objetos y realizando la transformación de la información mediante ellos, con los que se puede, insertar, modificar y anular datos, proveyendo facilitación en las conexiones y abdicar la presión de todo el código SQL a los programadores.

3.1.3.1 Ejemplo de comunicación entre capas

En este ejemplo se plasmará la solución del Requisito Funcional “Crear Documento Resolución Multa”.

El usuario inicia el proceso dentro de la ventana principal de Componente de Estandarización, con el propósito de crear un Documento Resolución. Dentro presenta un menú que contiene todas las funcionalidades que se llevan a cabo para la Gestión de Estándares. Selecciona la opción Resolución Multa por medio de un menú que lo dirige a la ventana de dicho estándar para crear el fichero con los datos del Documento.

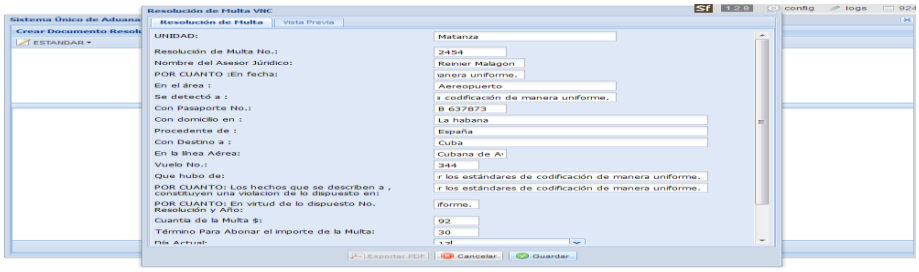


Figura 16. Página Principal del Componente de Estandarización.

Al seleccionar la opción de Guardar entonces se ejecuta el siguiente código.

```
buttons: [{
    text: 'Guardar y Exportar PDF',
    iconCls: 'iconPDF',
    id: 'idButtonPDFSiete',
    disabled: true,
    handler: function(){
        Ext.getCmp('idFormidSiete').getForm().submit({
            url: 'asuntosLegales/ResolucionSancionConjuntaVNC',
            waitMsg:'Guardando Datos...',
            success: function(form, action){
                // Ext.Msg.alert('Success', action.result.msg);
                Ext.Msg.show({
                    title: 'INFORMACIÓN',
                    msg: action.result.msg,
                    icon: Ext.Msg.INFO,
                    buttons: Ext.Msg.OK,
                    iconCls: 'iconOK',
                    fn: function(){
                        var numeroR = Ext.getCmp('idNumeroSiete').getValue();
                        Ext.getCmp('miFormSiete').form.submit({
                            waitMsg: 'Enviando PDF...',
                            waitTitle: 'Espere por favor',
                            params: {
                                numero: numeroR
                            },
                            success: function(){
```

Figura 17. Código en el ExtJs para guardar el fichero.

El XML es transformado en un arreglo para el trabajo de sus datos.

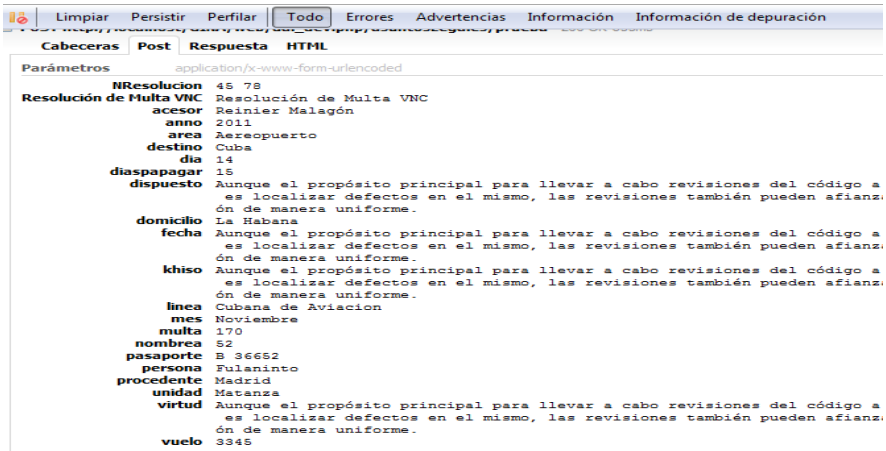


Figura 18. XML transformado en arreglo

Funcionalidad que brinda el servicio. Recibe todos los datos en un arreglo y los guarda en la Base de Datos.

```
class DalConjuntaVncPeer extends BaseDalConjuntaVncPeer {

    public static function InsertarConjuntaVnc($unidad, $nResolucion, $asesor, $nombrea, $dia, $mes, $anno, $area, $dinero,
        $resolucionForm = new DalResolucionForm();
        $paramResolucion = array();
        $paramResolucion['nombreUnidad'] = $unidad;
        $paramResolucion['numeroResolucion'] = $nResolucion;
        $paramResolucion['nombre'] = $asesor;
        $paramResolucion['nombreAnno'] = $nombrea;
        $paramResolucion['dia'] = $dia;
        $paramResolucion['mes'] = $mes;
        $paramResolucion['anno'] = $anno;
        $resolucionForm->bind($paramResolucion);
        if ($resolucionForm->isValid()) {
            $resolucionForm->updateObject($paramResolucion);
            // creando el formulario DalResolucionConjuntaForm
            $resolucionConjuntaForm = new DalResolucionConjuntaForm();
            $paramResolucionConjunta = array();
            $paramResolucionConjunta['idInfractor'] = $tipoRes;
            $paramResolucionConjunta['fechaRes'] = $fecha;
            $paramResolucionConjunta['area'] = $area;
            $paramResolucionConjunta['definicionMercancia'] = $mercancia;
            $paramResolucionConjunta['recursoApelacion'] = $interponer;
            $resolucionConjuntaForm->bind($paramResolucionConjunta);
            if ($resolucionConjuntaForm->isValid()) {
                $resolucionConjuntaForm->updateObject($paramResolucionConjunta);
                $resolucionConjuntaForm->updateObject()->setDalResolucion($resolucionForm->getObject());
                // creando el formulario DalConjuntaVncForm
                $resolucionConjuntaVNCForm = new DalConjuntaVncForm();
                $paramResolucionConjuntaVNC = array();
```

Figura 19. Código PHP con parte de la funcionalidad que brinda el servicio para guardar en la BD.

3.1.4 Diagrama de componentes

Un componente es el empaquetamiento físico de los elementos de un modelo.

En los diagramas de componentes se muestran los elementos de diseño de un sistema de software. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. (Microsoft, 2011)

Puede usar un diagrama de componentes para describir un diseño que se implemente en cualquier lenguaje o estilo. Sólo es necesario identificar los elementos del diseño que interactúan con otros elementos del diseño a través de un conjunto restringido de entradas y salidas. Los componentes pueden tener cualquier escala y pueden estar interconectados de cualquier manera. (Microsoft, 2011)

En el diagrama que muestra la **figura 20,** se representa la relación de componentes que se utilizan en el subsistema DAL. El acceso al subsistema, quien es el encargado de comunicarse con las clases actions, es a través de un controlador frontal, según la petición realizada; esta clase interactúa con las interfaces

que utilizan las librerías del Marco de Trabajo ExtJs y que son las encargadas de vincularse directamente con el usuario; además con las clases creadas por Propel que obtiene información de los diferentes Plugins y subsistemas, por medios de servicios, al igual de la interacción de los componentes de configuración que están presentes en todas las acciones realizadas en el sistema.

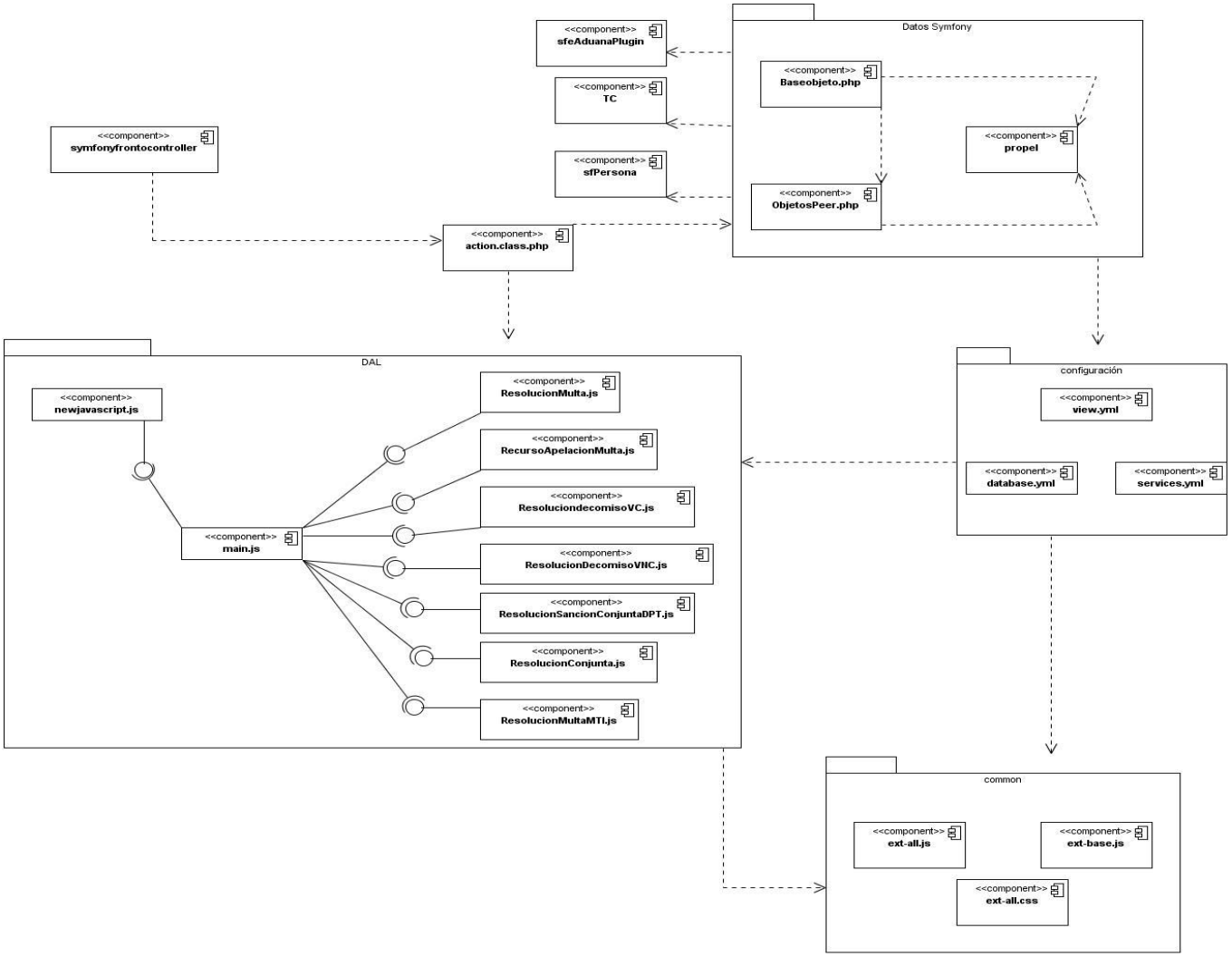


Figura 20. Diagrama de Componentes, del Componente de Estandarización del módulo DAL

3.2 Validación de la solución implementada

El interés por la calidad crece hoy de forma continua, a medida que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no dan respuesta a sus necesidades. El aseguramiento de la calidad toma en cuenta todas aquellas acciones planificadas y sistemáticas necesarias para proporcionar la confianza de que un producto o servicio satisfaga los requisitos de calidad establecidos (Mendoza, 2003).

3.2.1 Casos de prueba

La ejecución de las pruebas conlleva a la realización de una serie de actividades, descritas por (Vegas, 2005) como:

1. Diseño de las pruebas: Comprende la identificación de la técnica o técnicas de pruebas que se utilizarán para probar el software. Distintas técnicas de prueba ejercitan diferentes criterios como guía para realizar las pruebas.

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

3. Definición de los procedimientos de la prueba: Conlleva a una especificación de cómo se va a llevar a cabo el proceso, quién lo va a realizar y cuándo.

4. Ejecución de la prueba: Es el momento de aplicar los casos de prueba generados previamente e identificar los posibles fallos producidos al comparar los resultados esperados con los resultados obtenidos.

3.2.2 Pruebas de caja negra

Aquellas que trabajan con la interfaz del sistema. Verifican los dominios de entrada y salida del programa o programas para descubrir errores de funcionalidad, comportamiento y rendimiento. . El objetivo es

demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene. Fundamentado en el estudio de la especificación de las funciones, la entrada y la salida para derivar los casos. La prueba consiste en probar todas las posibles entradas y salidas del programa sin considerar el código en lo absoluto.

Figura 21: Estrategia de Prueba Aplicada.

Las pruebas de caja negra se aplican al sistema con el objetivo de encontrar posibles errores en la optimización del Acceso a Datos. Se trata de hallar una " tasas de errores" para dar una medida del número de requisitos que se han probado. Los resultados han determinado márgenes específicos de errores, demostrando la valides de las pruebas aplicada, permitiendo corregir los errores lanzados.

El conjunto de datos posibles suele ser muy amplio (por ejemplo, la Gestión de Contextos).

En la **figura 21** se muestra un fragmento de ejecución a un Requisito Funcional, desarrollado en el documento Diseño de casos de Pruebas anexos a este trabajo de diploma.

Nombre del requisito	scripción general	Escenarios de pruebas	Flujo del escenario
1: Registrar Documento Resolución Multa VNC	Permite crear una resolución mediante el estándar correspondiente a la misma, establecido en el Componente de Estandarización.	EP 1.1: Registrar Documento Resolución Multa.	-Seleccionar opción Estándar y solicita Crear Resolución Multa. -Muestra una ventana para captar los datos de la solicitud.
		EP 1.2: Ver Datos en Formato PDF.	Carga los datos básicos de la Resolución en los campos de información del mismo
		EP 1.3: Validar Datos • Unidad • Numero Resolución • Asesor • Área • Fecha(día-mes-año) • Persona (nombre y apellidos) • Pasaporte	Comprueba que los datos insertados sean correctos.

Figura 22. Pruebas a Requisito funcional: Registrar Resolución de Multa.

Id del escenario	Escenario	Unidad	Numero de Resolución	Pasaporte	Fecha	Hora	Domicilio	Procede, Destino,	Persona	Plazo(días)	Respuesta del sistema	Resultado la de prueba
EP 1.1	Solicita Registrar Documento Resolución.	V	NA	NA	V	NA	V	V	V	V	No se pueden dejar Campos Vacíos	Se pide al usuario establecer correctamente todos los datos
EP 1.2	: Ver Datos en Formato PDF.	I	V	V	NA	V	V	NA	V		Marca los formularios que tengan caracteres incorrectos.	La aplicación no guarda os datos de la Resolución, notificando el error encontrado.
EP 1.4	Validar Dato	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	I	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	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		I	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el error.

[Las celdas de la tabla contienen **V, I, o NA.** V indica válido**, I** indica inválido**,** y **NA** se debe proporcionar un valor del dato en este caso].

Figura 23: Juegos de datos a probar (Registrar Documento Resolución de Multa)

3.2.3 Pruebas de caja blanca

Se denomina cajas blancas a un tipo de pruebas de software que se realiza sobre las funciones internas de un módulo. Las pruebas de caja blanca están dirigidas a las funciones internas. Entre las técnicas usadas se encuentran; la cobertura de caminos (pruebas que hagan que se recorran todos los posibles caminos de ejecución), pruebas sobre las expresiones lógico-aritméticas, pruebas de camino de datos (definición-uso de variables), comprobación de bucles (se verifican los bucles para 0,1 y n iteraciones, y luego para las iteraciones máximas, máximas menos uno y más uno. (E.T.S.I., 2000)

3.2.3.1 Herramientas para la realización de pruebas de caja blanca.

Cuando se emplea php como lenguaje de programación existe diversidad de Marcos de Trabajo para crear pruebas unitarias y funcionales, siendo los más usados PhpUnit y SimpleTest. Symfony incluye su propio Marco de Trabajo para pruebas unitarias llamado Lime. El mismo utiliza la librería Test::More de Perl y es compatible con TAP. Lo que significa que los resultados de las pruebas se muestran con el formato definido en el "Test Anything Protocol", creado para facilitar la lectura de los resultados de las pruebas. El Lime presenta además una serie de ventajas dentro de las cuales resaltan por su importancia las siguientes: (Angel, 2009)

- Ejecuta los archivos de prueba en un entorno independiente para evitar interferencias entre las diferentes pruebas.
- Las pruebas son fáciles de leer y sus resultados también lo son. En los sistemas operativos que lo soportan, los resultados utilizan diferentes colores para mostrar de forma clara la información más importante.

- Está escrito con PHP, es muy rápido y está bien diseñado internamente. Consta únicamente de un archivo llamado lime.php y no tiene ninguna dependencia.

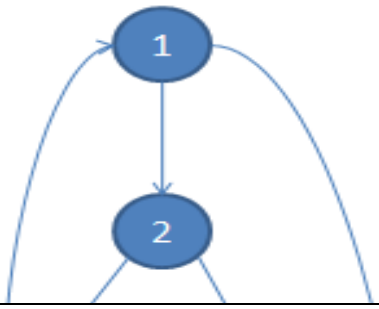
A continuación se muestran algunos ejemplos de las pruebas de caja blanca realizadas al sistema.

- "Obtener Contexto": Permite obtener un contexto dado su id.

```
public function obtenerResolucionbyID($id_resolucion, $resolucion) {
    $$resolucion = null;                                          //1
    foreach ($lista as $index => $obj) {                          //1
        if ($obj->getId() == $id_resolucion)                      //2
            $resolucion = objet;                                  //3
    }                                                             //4
    return $resolucion;                                           //5
}
```

Figura 24 Método obtenerResolucionByID($id_resolucion, $resolucion).

La figura 24 muestra el código correspondiente al método *obtenerResolucionByID($id_resolucion, $resolucion).* En el cual, pasándole por parámetro el id de una resolución y el listado de resoluciones hace una búsqueda dentro de la lista y devuelve la resolución con dicho id.

Complejidad ciclomática V (G) = A – N + 2 V (G) = 6 – 5 + 2 V (G) = 3	

Posibles caminos - 1-2-3-4-5 - 1-2-4-5 - 1-5	

Juegos de datos definidos para la prueba.

```
    $arr = array (array("uploads/Resolucion/Modelo de datos",
        "Modelo de Datos.jpg","este es el modelo de datos","1"),array("uploads/Resolucion/001","2"));
    $resolucion = null;
    foreach($arr as $key =>$val){
        $c= new Resolucion();
        $c->setURL($val[0]);
        $c->setFileName($val[1]);
        $c->setEnunciado($val[3]);
        $c->setId($val[4]);
        $resolucion []= $c;
    }
    $Res =new resolucion_Modelo();
    $var1 = $Res->obtenerResolucionbyID(1,$resolucion);
    $t->is($var1, "Modelo de datos", 'Correcto');

    $var2 = $Cont->obtenerResolucionbyID(4, $resolucion);
    $t->is($var,false ,'Incorrecto, No se encontro contecto con dicho id');
}
```

Figura 25: Juegos de datos para la prueba del método obtenerResolucionByID().

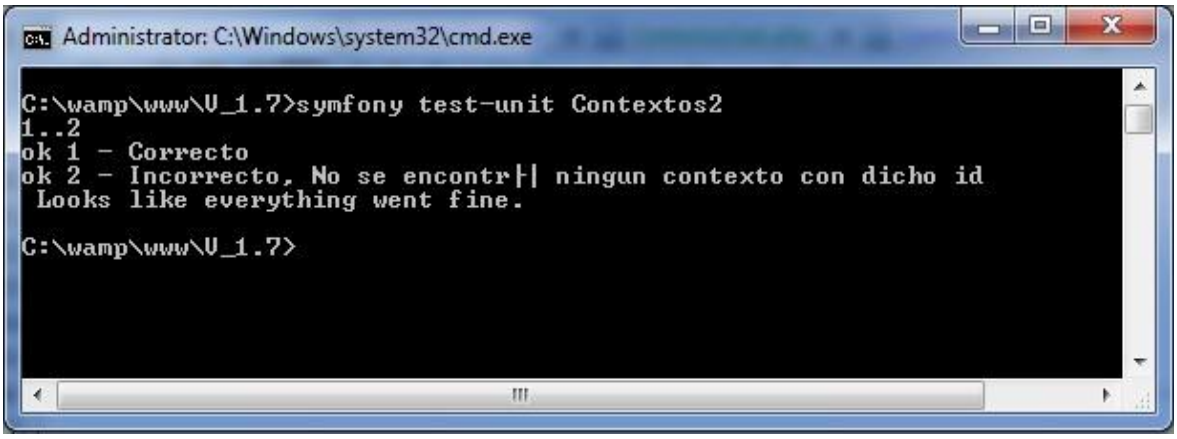


Figura 26: Resultados de la prueba del método obtenerResolucionByID() con la herramienta Lime.

Con la prueba del método obtenerResolucionByID ($id_resolucion, $resolucion) **figura 24**, y dados los juegos de datos introducidos como se muestra en la **figura 25**, se valida el correcto funcionamiento del metodo obtenerContextoByID(), mostrado en la **figura 26**.

3.2.2 Informe de resultados a las pruebas aplicadas

Prueba de Sistema Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados. En un ciclo iterativo estas pruebas ocurren más temprano, tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados.

3.2.2.1 Prueba de funcionalidad

Objetivo

Verificar la función del sistema al fijar la tensión en la validación de las funciones, métodos, servicios y casos de usos.

Metas

- ✓ El Sistema cumple con los requisitos funcionales especificados en el diseño de la solución.
- ✓ El Sistema tiene resultados satisfactorios en las restricciones de entrada y salida de la información especificada en el diccionario de Datos, de cada caso de uso.

3.2.2.2 Prueba de usabilidad

Objetivo

Determinar si la organización de los contenidos y las funcionalidades que se ofrecen desde el Sitio Web son entendidas y utilizadas por los usuarios de manera simple y directa.

Metas

Validado en la aplicación:

- ✓ Para poder ver las páginas adecuadamente necesita utilizar un navegador compatible con estándares Web.
- ✓ Proporcionar al usuario información relacionada con el estado actual del sistema.

3.3 Informe de resultado

Para la certificación del éxito o no de una prueba se tiene en cuenta el análisis de los resultados esperados contenidos en la descripción de los casos de prueba con los resultados reales. Se considera:

- Prueba Satisfactoria: Si el resultado esperado es idéntico al resultado obtenido.
- Prueba Insatisfactoria: Si el resultado esperado es diferente al resultado obtenido.

El equipo encargado de las pruebas determinó las no conformidades en los fallos en la validación del sistema y las erradicó, determinado continuar con los restantes casos de prueba y dar por finalizada la validación del sistema.

Conclusiones del Capítulo

Se logró implementar el componente de estandarización con éxito. Se le realizaron Casos de Prueba al sistema detectándose no conformidades en los 9 casos de prueba aplicados. Las no conformidades encontradas fueron registradas y solucionadas para un mejor funcionamiento del software.

Conclusiones Generales

Tras la realización de esta investigación se puede arribar a la conclusión fundamental de que la puesta en práctica del componente de estandarización desarrollado, permite un mejor control y la creación uniforme de todos los Documentos Resolución, manejado por el personal jurídico de las aduanas del país.

El componente implementado posee una interfaz agradable e interactiva que permite a cualquier asesor jurídico aduanero hacer uso del mismo sin complicaciones idiomáticas o de simbología. En cuanto a su estructura arquitectónica se puede concluir que cuenta con un diseño que permite realizar adecuadamente el manejo de Instrumentos Jurídicos, cumpliendo los objetivos de este trabajo de diploma.

Recomendaciones

- Desarrollar un grupo de funcionalidades que no están implementadas por escapar al alcance definido para esta investigación y que se considera aumentarían el valor agregado de la misma con la implementación del Módulo Dirección de Asuntos Legales al cual se integra este componente de estandarización.
- Implementar el resto de las funcionalidades del componente de estandarización que corresponden a los restantes módulos de la Dirección de Asuntos Legales.
- Incluir el componente de estandarización al Sistema Único de Aduanas.

- Incluir además servicios de autenticación de usuarios restringidos al sistema, con el objetivo de controlar el manejo de Documentos Resolución, sólo por el personal autorizado de la DAL en el SUA.

Bibliografía

. Pérez, Javier Eguíluz. 2008. *Introducción a XHTML.* 2008.

Angel. 2009. angelfire.com. [En línea] angelfire.com, 2009. [Citado el: 15 de 6 de 2011.] http://www.angelfire.com/empire2/ivansanes/bywbox.htm.

Barcelona, Universidad de. 2001. *Revisiones de Codigos y Estandares de Codificación.* 2001.

BOLIVIA, UNIVERSIDAD SALESIANA DE. 2009. *Análisis y Diseño.* 2009.

Booch, Grady. rational. *rational.* [En línea] [Citado el: 20 de 1 de 2011.] http://www.rational.com/uml/.

—. 1996. rational. *rational.* [En línea] 1996. [Citado el: 20 de 1 de 2011.] http://www.rational.com/uml/.

CAVSI.com. ¿Qué es un Sistema Gestor de Bases de Datos o SGBD? *¿Qué es un Sistema Gestor de Bases de Datos o SGBD?* [En línea] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

CEIGE, Dpto Aduana. 2011. *Propuesta de un Estándar de Codiicación.* 2011.

Chile, Departamento de las Ciencias de la Computación de. 2006. [En línea] 2006. [Citado el: 4 de 4 de 2011.] http://www.dcc.uchile.cl/~psalinas/uml/interaccion.html.

CIENFUEGOS, UNIVERSIDAD DE. 2000. ELECCIÓN DE METODOLOGÍAS DE DESARROLLO PARA APLICACIONES WEB EN LA FACULTAD DE INFORMÁTICA DE LA UNIVERSIDAD DE CIENFUEGOS. [En línea] 2000. [Citado el: 2011 de 4 de 20.] http://www.eumed.net/libros/2009c/584/RUP%20Diseno%20e%20implementacion%20del%20sistema.htm.

CodeBox. 2001. CodeBox. [En línea] CodeBox, 2001. [Citado el: 12 de 4 de 2011.] http://www.codebox.es/glosario.

Comunidad de Desarrollo de ExtJs . *ExtJs en Español.* [En línea] [Citado el: 4 de Febrero de 2011.] http://extjs.es/.

Cood. 1992. *UML Data Base.* 1992.

Corporation, Oracle. 2002. oracle.com. *oracle.com.* [En línea] 2002. [Citado el: 14 de 6 de 2011.] http://www.oracle.com/us/index.html.

Datos, Gestor de Base. cnx.org. [En línea] [Citado el: 26 de 2 de 2011.] http:00cnx.org/contet/m17543/latest/html.

DCC., .Departamento de Ciencias de Computación Universidad de Chile. [En línea] [Citado el: 10 de 3 de 2011.] http://www.dcc.uchile.cl/~psalinas/uml/interaccion.html..

Dumas. 2005. *Business Process Modeling Notation.* 2005.

Durán, A. 2001. *Metodología para el Análisis.* 2001.

E.T.S.I. 2000. s.l. : Departamento de Lenguajes y sistema Informáticos. [En línea] Universidad de Granada., 2000.

Echarte, Patxi. 2005. www.eslomas.com. *www.eslomas.com.* [En línea] 05 de 02 de 2005. [Citado el: 4 de 19 de 2011.] http://www.eslomas.com/index.php/archives/2005/02/25/manejo-de-errores-en-php/.

Echevarria. 2008. 2008.

Elers Pérez, Alberto y Sariol Pérez. 2009. Implementación del Módulo Gestión de Actividades del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas. 2009.

Estandares. 2000. wcoomd.org. *wcoomd.org.* [En línea] 2000. [Citado el: 16 de 12 de 2010.] http://www.wcoomd.org.pdf.

ExtJs, Comunidad de Desarrollo de. 2004. ExtJs en Español. [En línea] Requirements Engineering So Things Don't Get Ugly. 2004., 2004. http://extjs.es/. JACOBSON.

—. 2006. ExtJs en Español. [En línea] 2006. [Citado el: 27 de 2 de 2011.] http://extjs.es/.

Fabien Potencier, François Zaninotto. 2008. *Symfony la Guía Definitiva.* 2008.

Fabien Potencier, Fransiçois Zaninotto. 2009. *Symfony la guia definitiva.* 2009.

Fowler, Martin. *Patterns of Enterprise Application Architecture.*

GEIGE. 2009. *Proceso de Desarrollo y Gestión de Proyectos de Software.* 2009.

GEIGE, Dpto Aduana. 2010. *Estándar de Codificación.* Ciudad Habana : s.n., 2010.

Ibarra, Rosalina. 2009. Biblioteca.uci.cu. *Biblioteca.uci.cu.* [En línea] mayo de 2009. [Citado el: 7 de 05 de 2011.] http://biblioteca.uci.cu/sbd/biuci/index.html.

Jacaboson, I., Booch, G., Rumbaugh J. 2000. scrib. *scrib.* [En línea] El Proceso Unificado de Desarrollo de Software, 2000. [Citado el: 05 de 02 de 2011.] http://es.scribd.com/doc/7844685/CONCEPTOS-DE-RUP.

JACOBSON. 2004. Requirements Engineering So Things Don't Get Ugly. 2004. *Requirements Engineering So Things Don't Get Ugly. 2004.* [En línea] 2004. [Citado el:

4 de 2 de 2011.] http://extjs.es/. JACOBSON, y. o. (2004). Requirements Engineering So Things Don't Get Ugly. 2004.

Lago, Ramiro. 2007. Proactiva. *Proactiva.* [En línea] 2007. [Citado el: 8 de 11 de 2011.] http://www.proactiva-calidad.com/java/patrones/.

Laram, Craig. 1999. *UML y patrones. Mexico : Prentice-Hall Hispanoamericana.* 1999.

Larman, Craig. 1999. *UML.* 1999.

Mendoza. 2003. *Casos de Prueba.* 2003.

Metricas, Aplicables. 2010. 2010.

2011. Microsoft. [En línea] 2011. [Citado el: 15 de mayo de 2011.] http://msdn.microsoft.com/es-es/library/dd409390.aspx.

Microsoft. 2011. Diagrama de componetes. [En línea] 2011. [Citado el: 15 de mayo de 2011.] http://msdn.microsoft.com/es-es/library/dd409390.aspx.

mitecnologico. 2000. EspecificacionesDeRequerimientos. *EspecificacionesDeRequerimientos.* [En línea] 2000. [Citado el: 21 de 3 de 2011.] http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos.

Montes. 2008. disenomontes. [En línea] 2008. [Citado el: 6 de 4 de 2011.] http://disenomontes.over-blog.es/article-diagramas-de-secuencia-37559775.html.

Oracle. *Oracle.* [En línea] [Citado el: 02 de Febrero de 2011.] http://www.oracle.com/us/index.html.

Oracle Corporation. 2010. *Portal del IDE Java de Código Abierto.* [En línea] 2010. [Citado el: 17 de 3 de 2011.] http://netbeans.org/index_es.html.

Orosco, Sergio. 2009. Dominando_el_problema_el_modelo_conceptual. *Dominando_el_problema_el_modelo_conceptual.* [En línea] 2009. [Citado el: 4 de 4 de 2011.] http://www.milestone.com.mx/articulos/dominando_el_problema_el_modelo_conceptual.htm.

Pedre, Pozo. 2011. Clikear. *Clikear.* [En línea] 2011. [Citado el: 30 de 1 de 2011.] http://www.clikear.com/manuales/uml/diagramasinteraccion.aspx.

Pérez, Javier Eguíluz. 2007. *Introducción a XHTML.* 2007.

—. 2007. *Introducción al CSS.* 2007.**PHP.net. 2001.** PHP General Infotmation. [En línea] 2001. [Citado el: 1 de 3 de 20111.] http//ww.php.net/en/faq/general.php.

Potencier, François Zaninotto y Fabien. *Symfony La Guía Definitiva.*

Pressman. 1998. 1998.

—. 2006. 2006.

Ramírez, Arturo. 2005. La Aduana. *aduanas.com.* [En línea] 2005. [Citado el: 20 de 10 de 2010.] http://www.aduanas.com.ve/boletines/boletin_12/aduana.htm.

Reyero, Eusebio. 2005. 2005.

Riordan, Rebecca. 2005. office.microsoft.com. *office.microsoft.com.* [En línea] 2005. [Citado el: 18 de 2 de 2011.] http://office.microsoft.com/es-mx/access-help/conceptos-basicos-del-diseno-de-una-base-de-datos-HA001224247.aspx.

Rodriguez, Gorgel, Arce, San Marcos. 2008. *La Importancia de las Aduanas en el Comercio Exterior.* 2008.

SIDUNEA, UNCTAD. asycuda.org/. *asycuda.org.* [En línea] [Citado el: 3 de 3 de 2011.] http://www.asycuda.org/spanish.

soluciones aduaneras, Departamento de. 2010. *Alcance de Asuntos Legales. UCI-AGR.* La Habana : s.n., 2010.

T.Ambrosio. 2001. *Un Proceso de Ingeniería de Requerimientos basado en Reutilización.* 2001.

Tucumán, Universidad Tecnológica Nacional - Facultad Regional. 2000. [En línea] Lenguajes de Programación, 2000. [Citado el: 3 de 3 de 2011.] http://frt.utn.edu.ar/sistemas/paradigmas/lenguajes.htm.

VALDÉS. 2000. ¿Qué son las bases de datos? *¿Qué son las bases de datos?* [En línea] 2000. http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

Web, Desarrollo. 2001. DesarrolloWeb.com. [En línea] 2001. [Citado el: 10 de 11 de 2010.] http://www.desarrolloweb.com/articulos/449.php.