Universidad de las Ciencias Informáticas

Facultad 3

Título: Diseño e Implementación del módulo Despacho de Cargas del subsistema Despacho No Comercial del Sistema de Gestión Integral de Aduana.

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

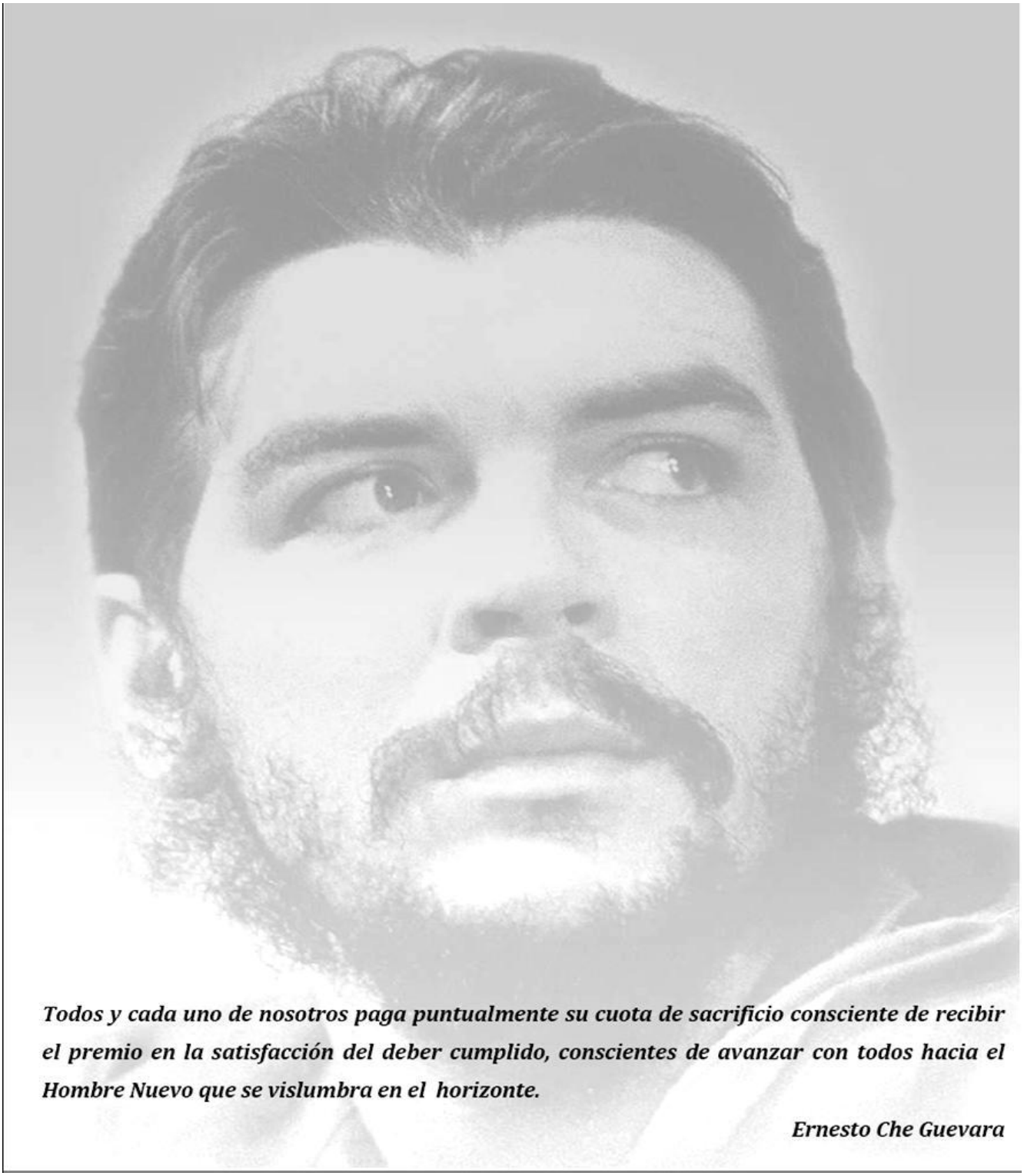
Autor: Odelkis Rivera Ortiz

Tutores:

Ing. Iliana de la Rosa Zayas Frias

Ing. Lázaro Manuel Gil Martínez

La Habana, junio del 2012



Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.

Ernesto Che Guevara

Declaración de autoría

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

Para que así conste firmo la presente.

A los ____ días del mes de _____ del año _____.

Odelkis Rivera Ortiz

_____

Firma del Autor

Ing. Lázaro Manuel Gil Martínez

_____

Firma del Tutor

Ing. Iliana de la Rosa Zayas Frias

_____

Firma del Tutor

Agradecimientos

A mi mamá Odalis por estar siempre pendiente de mí, por darme todo el amor y la dedicación del mundo, a mis hermanas bellas que las adoro, son las mejores del mundo, a mi padrastro que ha sido un padre para mí, a mi novio por darme el amor y el cariño, a mi familia por ser tan bella y siempre estar tan unida y dispuesta a ayudarme, a mis dos tutores Iliana y Lázaro que han sido incondicional, en todo momento me han ayudado desinteresadamente, a mis compañeros del departamento, los cuáles de una forma u otra me han ayudado, ellos son Adrián, Boris, Fernando, Widen, Ricardo, Eddy, en fin todos.

A: Susel y Daniel por dedicarme parte de su tiempo.

A: mis compañeros de aula de 5 años, que nunca los voy a olvidar y en especial a Maydel, Mariem, Lili, Carlos. Muchas gracias a todos!

Dedicatoria

Dedico este trabajo especialmente a mi mamá, quien ha confiado siempre en mí, a mis hermanas que son mi vida y que sigan mi ejemplo, a mi padre y abuelo quienes estarían muy orgullosos en estos momentos, a mi novio por ser maravilloso, a mi abuela, tíos, tías, primos, en fin a todas las personas que siempre han confiado en mí.

Resumen

El objetivo fundamental de la Aduana General de la República de Cuba es el control de personas, artículos y mercancías en las fronteras cubanas. Para ello, se basa en un conjunto de leyes y regulaciones que definen algunas reglas para la entrada y salida de los mismos. Actualmente existe un sistema que gestiona las mercancías en la Aduana General de la República de Cuba, el Sistema Único de Aduana, pero han surgido nuevos requerimientos del cliente como cancelar una declaración, modificar una declaración y registrarle una prórroga, que no están presentes en este. En este trabajo se hace referencia al Despacho No Comercial, para el cual se desarrolla el módulo Despacho de Cargas, con el objetivo de materializar los nuevos requerimientos y mejorar los existentes siguiendo las normas y regulaciones aduaneras.

Para darle solución a este, se desea realizar el diseño e implementación del módulo que cumpla con los requerimientos del cliente. Para ello se utilizaron las siguientes herramientas como medio de realización: Visual Paradigm para el modelado del diseño y Eclipse para la implementación, así como el framework de desarrollo Symfony y de JavaScript el ExtJS, siguiendo el Modelo de Desarrollo creado en el Departamento de Soluciones para la Aduana.

Tabla de contenidos

Introducción

En Cuba, la aduana constituye un órgano de control en la frontera y de fiscalización de la actividad vinculada al comercio exterior. Entre sus misiones está garantizar la seguridad y protección de la sociedad socialista y de la economía nacional, a través del cumplimiento de las políticas estatales de competencia aduanera para el tráfico internacional de viajeros, mercancías y medios de transporte.(AGR, 2000)

El comercio internacional implica la circulación de mercancías, personas y bultos postales entre fronteras. La capacidad de mover artículos a través de fronteras internacionales en forma rápida, segura y a un costo razonable y predecible, puede otorgar a un país ventajas notables con respecto a las competencias e impulsar el desarrollo socioeconómico del mismo. Junto a estos beneficios, el tráfico internacional conlleva riesgos de diversos tipos que pueden comprometer el desarrollo de la economía nacional, la salud y la seguridad del país. Por estas razones las aduanas juegan un rol crítico, no solo para lograr las metas gubernamentales, sino también en el cumplimiento de la legislación nacional y en garantizar la protección y seguridad de la sociedad. (PINEDA *et al.*, 2010)

Dentro del control que se lleva a cabo, existen las operaciones no comerciales. Estas se refieren a cualquier mercancía o producto importado, que ha sido suministrado a título gratuito por el proveedor al importador con fines promocionales, publicitarios, de exhibición u otra actividad análoga, con el objetivo de demostrar sus características o propiedades, es decir, se trata de muestras sin valor comercial y que no serán utilizadas para fines comerciales.

Estas operaciones son monitoreadas a través del Despacho No Comercial (DNC), donde se gestionan las importaciones y exportaciones realizadas por un conjunto de personas que pueden ser tratadas por la aduana como naturales o jurídicas.

En la aduana de Cuba se usa actualmente el Sistema Único de Aduana (SUA), el cual fue eficiente en el momento de su creación, pero han surgido modificaciones en las normativas por las cuáles se rige la aduana, lo que ha provocado grandes cambios en los procesos y reglas del negocio, este sistema es poco flexible, y adaptable a los cambios. Cada módulo de este tiene su Base de Datos (BD) separada, por lo que es difícil obtener información de otro módulo, ya que hay que realizar las consultas manualmente para obtener la información. El SUA contiene algunas funcionalidades como registrar una declaración, presentarla y liquidarla, pero producto a que han cambiado las normativas y regulaciones aduaneras es

necesario adaptar estos cambios, además de que han surgido nuevas funcionalidades como otorgamiento de prórrogas, cancelación por oficio (cancelar una declaración) y modificación de la declaración que no están presentes en este. Por estos motivos se propuso el desarrollo de un nuevo sistema que materializase todos los requisitos identificados y que permita a la Aduana General de la República de Cuba (AGR) contar con toda la información sobre los controles de las mercancías sin carácter comercial.

A partir de la situación planteada se define como **problema a resolver**:

¿Cómo materializar los requisitos identificados para el Despacho de Cargas del Sistema de Gestión Integral de Aduana?

En una profunda búsqueda de la solución al problema se determinó como **objeto de estudio**:

Los sistemas informáticos que gestionan las operaciones no comerciales para las aduanas.

Se define como **campo de acción** del presente trabajo:

El módulo Despacho de Cargas del Sistema de Gestión Integral de Aduana.

Como **idea a defender**:

Si se desarrolla el módulo Despacho de Cargas para el Sistema de Gestión Integral de Aduana se materializaran los requisitos identificados.

Objetivo General:

Desarrollar el módulo Despacho de Cargas para el Sistema de Gestión Integral de Aduanas teniendo en cuenta los requisitos identificados.

Objetivos Específicos:

- ✓ Elaborar el marco teórico relativo a las soluciones de software que gestionan las importaciones y exportaciones de mercancías con carácter no comercial.
- ✓ Realizar el diseño del módulo Despacho de Cargas.
- ✓ Realizar la implementación del módulo Despacho de Cargas.
- ✓ Validar el módulo Despacho de Cargas.

Tareas a realizar:

- ✓ Realización del estudio comparativo de las soluciones de software que gestionan las mercancías en las Aduanas.
- ✓ Diseño de la BD.
- ✓ Obtención del modelo de clases del diseño.
- ✓ Obtención del modelo de datos.
- ✓ Implementación de las interfaces visuales.
- ✓ Implementación de los requisitos funcionales.
- ✓ Realización de pruebas de validación.

Métodos de Investigación

Los métodos teóricos permiten estudiar las características del objeto de investigación que no son observables directamente, facilitando la construcción de modelos creando condiciones para ir más allá de las características.

Analítico-Sintético

El análisis permite la división mental del todo en sus múltiples relaciones y componentes, la síntesis es la operación inversa que establece mentalmente la unión entre las partes previamente analizadas. Este método se utiliza en el análisis de los elementos de la situación problemática, estos se relacionan entre sí y se vinculan llegando al problema como un todo. A su vez, la síntesis se produce sobre la base de los resultados dados previamente por el análisis. Este método se utiliza además para analizar la bibliografía y realizar el estudio del estado del arte. (ALVARADO *et al.*, 2007)

Modelación

Este método fue creado por los científicos para reproducir el fenómeno que se está estudiando, es una reproducción simplificada de la realidad, que cumple una función heurística, ya que permite descubrir y

estudiar nuevas relaciones y cualidades del objeto de estudio. A través de este método se crean modelos con vista a investigar la realidad, facilitando la construcción de los modelos del diseño, así como la construcción del software.

La estructura del trabajo queda constituida en tres capítulos:

Capítulo 1:

En el capítulo uno se analizan y seleccionan las herramientas, lenguajes de modelado y técnicas para el diseño e implementación de los requisitos funcionales.

Capítulo 2:

En el capítulo dos se realiza el diseño e implementación del módulo Despacho de Cargas, donde se obtienen diferentes artefactos, como el script de la BD, el Modelo de Diseño, incluyendo diferentes diagramas como: Diagrama de Paquetes, Diagrama Entidad-Relación(DER), Diagramas de secuencias orientado a actividades, así como la descripción de diferentes clases, con el objetivo de realizar la implementación.

Capítulo 3:

En el capítulo tres se realizan las validaciones del módulo Despacho de Cargas, donde se verifica el cumplimiento de los requisitos a través de pruebas y se evalúa la calidad del producto desarrollado.

Capítulo 1: Fundamentación teórica.

A continuación se describe el estado del arte, las herramientas, lenguaje de programación y metodologías a utilizar para el desarrollo del mismo, así como el framework[1] de desarrollo que será usado para la implementación.

1.1. *Estado del arte.*

Ámbito internacional

La Organización Mundial de Aduanas, es el único organismo intergubernamental enfocado a cuestiones aduaneras y que agrupa a 173 Administraciones de aduanas en todo el mundo y que dentro de sus actividades está desarrollar estándares globales, simplificar y armonizar los procedimientos aduaneros, facilitar el comercio internacional, entre otras. (ZAYAS, 2010)

- **Sistema Informático María (SIM)**

El SIM lo utiliza la aduana Argentina. El objetivo de este es crear un mecanismo de consulta a destinaciones de importaciones y totalmente abierto al público en general, a fin de, en base a distintos campos seleccionados, obtener el detalle de la información requerida, en pantalla, o bien armar un archivo para cargar posteriormente en un disquete, este sistema cubre aproximadamente el 90 % del total de las importaciones y exportaciones. (AFIP, 1996)

Este sistema tiene una arquitectura informática de la década del ochenta que combina BD, registros, visualizaciones y declaraciones electrónicas que utiliza la aduana para realizar el control y estadística de las declaraciones de importaciones y exportaciones. (NICODEMOS, 1010)

Producto a su arquitectura informática, está limitado ante los actuales progresos de transmisión de información aplicados por sistemas aduaneros de tecnología actual. Es fundamental para el comercio exterior puesto que da la posibilidad de consultar la historia de cada destinación, paso por paso, y usuario

[1] Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

por usuario, esto es un elemento clave, tanto para el control de cada operación, como para cualquier tipo de auditoría, pero este sistema no gestiona el DNC, todas las operaciones la realiza por la vía comercial, por lo que no es factible usarlo en la AGR.

Ámbito nacional

Desde hace varios años la aduana de Cuba trabaja en un proceso de perfeccionamiento y mejora de la gestión aduanera, buscando que las administraciones sean más eficaces y eficientes. Se impone el desarrollo de la informática, como una necesidad en el país, ya que cada día se requiere que la información sea más confiable, segura y rápida. (AGR, 2000)

- **Sistema Aduanero Automatizado (SIDUNEA)**

En 1994 se implantó el SIDUNEA. Este fue el primer sistema que se utilizó en la AGR, por lo que creó una cultura informática, dio a conocer las ventajas que reportaba el uso de la computación sustituyendo procesos completamente manuales y por ende en la agilización del proceso y en la consolidación de las estadísticas de recaudación y comercio exterior, ayudó a organizar el proceso de despacho y a emplear controles selectivos e impulsar el estudio de las particularidades y requerimientos de los procesos. Este es usado con éxito en más de 80 países.(AGR, 2006)

- **Sistema Automatizado de Despacho Mercantil (SADEM)**

En enero del 2001 surge el SADEM, paralelamente a este se desarrollaron otros sistemas aunque en otras plataformas, los cuales resolvían otros procesos o partes de ellos y al igual que el SADEM obtenían buenos resultados.

El objetivo del SADEM era perfeccionar el proceso de despacho, controlar de manera automatizada, los plazos de vencimiento de las facilidades y regímenes suspensivos, nomenclatura y acuerdos; permitiendo enfrentar el fraude de la política de comercio exterior y otras manifestaciones que afecten la economía nacional, resolver el problema de la doble moneda, teniendo bien delimitados y validados los ingresos por cada tipo de moneda, incrementar la calidad de servicio al cliente, facilitándole los trámites aduaneros al permitirles presentar la declaración de mercancías tanto en ventanilla a través del inspector, en oficinas conectadas a la Red de Área Local o desde sus oficinas.(AGR, 2006)

- **SUA**

Luego se le dio paso a la creación del SUA, sistema web vigente en la AGR, multiplataforma, desarrollado bajo las premisas del uso de software libre.

Incluye 5 procesos esenciales:

- ✓ Proceso de Control de los medios de transporte internacional.
- ✓ Proceso de Control de las importaciones y exportaciones de los viajeros.
- ✓ Proceso de Control de las importaciones y exportaciones comerciales.
- ✓ Proceso de Control de las importaciones y exportaciones no comerciales vía postal.
- ✓ Proceso de Control de las importaciones y exportaciones no comerciales por la vía marítima y aérea.

Entre sus objetivos se encuentra automatizar íntegramente los 5 procesos que se llevan a cabo en la aduana, perfeccionándolos para lograr que sean más sencillos y eficientes, asegurar la información para usuarios internos y externos, incluso el cruce de información entre las diferentes áreas y que esto ocurra de manera transparente, así como que la información se encuentre en una BD única y brindar facilidades para los trámites aduaneros, por ejemplo: envío electrónico de información adelantada del manifiesto, declaración de mercancías y liberaciones. (AGR, 2000)

Se llega a la conclusión que el SIDUNEA tiene una gran importancia puesto que creó una cultura informática y dio a conocer las ventajas que reportaba el uso de la computación, pero debido a la especificidad del comercio de Cuba, surgió la necesidad de realizar una serie de adecuaciones prácticas que en muchos casos provocaban la ejecución de operaciones innecesarias, la confección de documentos manuales que duplican información, retrasando y distorsionando el proceso de despacho; se comprobó que se recibían, revisaban, elaboraban y actualizaban alrededor de 129 documentos. Este sistema no diferencia las operaciones comerciales de las no comerciales, todas las operaciones la realiza de forma comercial, y además el costo de compra y de mantenimiento es muy alto. A continuación surge el SADEM, la principal limitante estaba dada por la interoperabilidad, ya que alimentarse este, de los otros sistemas que usaba la AGR para gestionar otras operaciones era imposible.

Teniendo en cuenta las necesidades de la AGR y la diversidad de sistemas automatizados existentes, en

diferentes plataformas, se decidió crear un Sistema Único de Aduana, el SUA. Este sistema es el que está presente en las aduanas de Cuba, aunque resuelve actualmente los problemas, tiene inconvenientes, puesto que han surgido nuevas normativas y funcionalidades por las cuáles se rige la aduana, y este sistema es poco flexible, por lo que es muy costoso adaptar los cambios, además que existe redundancia de información, ya que cada módulo tiene su BD separada.

1.2. *Tecnología utilizada.*

Modelo de desarrollo para el Departamento de Soluciones para la Aduana

En la actualidad, la utilización de metodologías para el desarrollo de aplicaciones es de gran importancia, debido a la gran necesidad de control de variables que conlleva el mismo desarrollo, y para la ordenada elaboración de las aplicaciones. (CHACON, 2006)

El Departamento de Soluciones para la Aduana tiene como principal responsabilidad desarrollar sistemas informáticos, para gestionar los procesos de negocio de las entidades aduaneras en su relación con el comercio exterior del país.

En el departamento se hace uso de un Modelo de desarrollo, el cual cumple con las especificaciones y necesidades del departamento, en la figura 1.1 se muestra el ciclo de vida de los proyectos del departamento.

Figura 1.1 Ciclo de vida del proyecto.

Este ciclo comienza con el Estudio Preliminar, dando como resultado el Estado del Arte, y a continuación se generan diferentes artefactos que serán utilizados como entrada en el diseño. Uno de los principales aspectos a resaltar del modelo dentro del diseño, lo constituye la particular utilización de un diagrama de secuencia orientado a actividades el cual se nutre de las principales bondades que brinda el diagrama de actividades y el diagrama de secuencias. Este modelo se realizó con el objetivo de facilitar el trabajo de los programadores que no tienen un alto nivel de involucración en el desarrollo del proyecto, pudiéndose así comprender con facilidad el diseño de cada una de las actividades orientadas por los diseñadores a cualquiera de los programadores del departamento. (FALGUERAS *et al.*, 2003)

Framework: Symfony 1.2.8

Symfony es un framework que simplifica el desarrollo de una aplicación, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y fácil de mantener. Además, el desarrollo web se hace más intuitivo y las aplicaciones resultantes son más robustas y más fáciles de mantener. Este 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.

Está desarrollado completamente con PHP 5, este hace un uso continuo de los mecanismos Orientados a Objetos. Además es compatible con la mayoría de gestores de BD como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows y es fácil de instalar y configurar en la mayoría de plataformas, pero lo suficientemente flexible como para adaptarse a los casos más complejos, sigue las 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. El código es fácil de leer, que permite un mantenimiento muy sencillo, y fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.(POTENCIER y ZANINOTTO, 2008)

Dentro de un proyecto, las operaciones se agrupan de forma lógica en aplicaciones.

Normalmente, una aplicación se ejecuta de forma independiente respecto a otras aplicaciones del mismo proyecto. Cada aplicación está formada por uno o más módulos. Un módulo normalmente representa a una página web o a un grupo de páginas con un propósito relacionado. Los módulos almacenan las acciones, que representan cada una de las operaciones que se puede realizar en un módulo.

Las acciones son el corazón de la aplicación, puesto que contienen toda la lógica de la aplicación. Las acciones utilizan el modelo y definen variables para la vista. Cuando se realiza una petición web en una aplicación Symfony, el Localizador Uniforme de Recursos (URL) define una acción y los parámetros de la petición. (POTENCIER y ZANINOTTO, 2008)

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

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

Symfony toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones

sea rápido y sencillo.

Herramienta de Entorno de Desarrollo Integrado: Eclipse 3.3

Eclipse es un software de desarrollo, de código abierto y multiplataforma que proporciona robustez, calidad comercial y una plataforma industrial para el desarrollo de aplicaciones de alta calidad. En el Eclipse se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los plugins adecuados.

La arquitectura de plugins de Eclipse permite, además de integrar diversos lenguajes sobre un mismo Entorno de Desarrollo Integrado (IDE), introducir otras aplicaciones que pueden resultar útiles durante el proceso de desarrollo como: herramientas de Lenguaje Unificado de Modelado (UML), editores visuales de interfaces, ayuda en línea para librerías, entre otros.

Las herramientas que proporciona Eclipse permiten al desarrollador la libertad de elegir el entorno de su proyecto y además proporciona un framework que hace más fácil crear, integrar y usar las herramientas de software. (MOTA *et al.*, 2012)

Herramienta de Ingeniería de Software Asistida por Computadoras (CASE) para el modelado: Visual Paradigm for UML 8.0

Visual Paradigm es una herramienta de UML profesional que soporta el ciclo de vida completo del desarrollo de software, análisis y diseño, construcción, pruebas y despliegue. Permite realizar diagramas de procesos de negocio, ingeniería inversa (de código a modelo), generación de código (de modelo a código), generación de BD (transformación de DER a tablas de BD), ingeniería inversa de BD (a partir del modelo físico de la BD obtener su DER), generación de informes de documentación, entre otros. (PACHECO *et al.*, 2010)

Lenguaje de programación

Los lenguajes de programación son herramientas que permiten crear programas y software. La Programación Orientada a Objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real. Con la POO tenemos que aprender a pensar las cosas de una manera distinta, para escribir los programas en términos de objetos, propiedades y métodos.(ALVARADO *et al.*, 2007)

Los lenguajes de POO tratan a los programas como conjuntos de objetos que se ayudan entre ellos para realizar acciones. Entendiendo como objeto a las entidades que contienen datos. Permitiendo que los programas sean más fáciles de escribir, mantener y reutilizar. (PÉREZ, 2006)

La POO permite concebir los programas de una manera bastante intuitiva y cercana a la realidad. La tendencia es que un mayor número de lenguajes de programación adopten la POO como paradigma para modelar los sistemas. Prueba de ello es la nueva versión de PHP 5 que implanta la programación de objetos como metodología de desarrollo.

PHP 5.3

PHP (acrónimo de PHP: Hypertext Preprocessor) es un lenguaje de script interpretado en el lado del servidor, utilizado para la generación de páginas Web dinámicas, especialmente adecuado para desarrollo web y que puede ser incrustado en Lenguaje de Marcado de Hipertexto (HTML).

La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas, este se ejecuta en el servidor por eso permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una BD. El programa PHP es ejecutado en el servidor y el resultado es enviado al navegador. El resultado es normalmente una página HTML.

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, sin embargo, para que sus páginas PHP funcionen el servidor donde están alojadas debe soportar PHP.

Dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas web dinámicas:

- Soporte para una gran cantidad de BD: MysSQL, PostgreSQL, Oracle, MS SQL Server, entre otras.
- Soportado por una gran comunidad de desarrolladores, como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente.
- El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.

- Con PHP se puede procesar información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas.

El modo de operación del PHP es el siguiente:

- El Navegador realiza una petición al servidor (se escribe la URL).
- Después el servidor ejecuta el código PHP solicitado y retorna el código HTML generado al Navegador.
- Por último el Navegador muestra la respuesta del servidor.

Este tipo de iteración permite algunas operaciones complejas como conexiones a BD o ejecución de complejos programas.

Las principales características de PHP son: su rapidez, su facilidad de aprendizaje, su soporte multiplataforma tanto de diversos Sistemas Operativos, como servidores de Protocolo de Transferencia de Hipertexto (HTTP) y de BD, y el hecho de que se distribuye de forma gratuita bajo una licencia abierta. (ACHOUR *et al.*, 1997)

ExtJS 3.0

ExtJS es una librería de JavaScript (JS) para el desarrollo de aplicaciones web interactivas que dispone de un conjunto de componentes para incluir dentro de una aplicación web, como son cuadros y áreas de textos, campos para fechas, campos numéricos, combos, radiobuttons, checkboxs, entre otros componentes.

Varios de estos componentes están capacitados para comunicarse con el servidor usando AJAX (JS asíncrono y XML) y Lenguaje de Marcado de Hipertexto Dinámico (DHTML).

También contiene numerosas funcionalidades que permiten añadir interactividad a las páginas HTML, como: cuadros de diálogo y quicktips para mostrar mensajes de validación e información sobre campos individuales.

Tiene una licencia de tipo software libre y otra licencia de tipo comercial si se desea obtener soporte técnico. (EGUÍLUZ, 2007)

Hojas de Estilo en Cascada (CSS) 2.0

CSS es un lenguaje de hojas de estilos, creado para controlar el aspecto o presentación de documentos electrónicos definidos con HTML y Lenguaje Extensible de Marcado de Hipertexto (XHTML). CSS es imprescindible para crear páginas web complejas y es la mejor forma de separar los contenidos y su presentación. A través de este lenguaje se define el aspecto de cada elemento como color, tamaño, posición, separaciones horizontales y verticales entre elementos dentro del texto, entre otros. (EGUÍLUZ, 2008)

Servidor de Aplicación Web: Apache 2.2

Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Las diferentes plataformas y entornos, hacen que a menudo sean necesarias diferentes características o funcionalidades. Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular. Este diseño permite a los administradores de sitios web elegir qué características van a ser incluidas en el servidor seleccionando qué módulos se van a cargar, ya sea al compilar o al ejecutar el mismo. Corre en multitud de sistemas operativos, lo que lo hace prácticamente universal. (PACHECO *et al.*, 2010)

Sistema Gestor de Base de Datos: Oracle 11g

Los Sistemas Gestores de Base de Datos (SGBD), son un tipo de software muy específico que ejecuta todas las solicitudes de acceso a la BD formuladas por los usuarios, dedicado a servir de interfaz entre la BD, el usuario y las aplicaciones. Es un sistema de mantenimiento de registros basado en ordenadores, es decir, un sistema cuyo propósito general es registrar y mantener información. Tal información puede estar relacionada con cualquier cosa que sea significativa para la organización donde el sistema opera, es decir, cualquier dato necesario para los procesos de toma de decisión inherentes a la administración de esa organización. (BERTINO y MARTINO, 1995)

El SGBD Oracle fue fabricado por la Corporación de Oracle, utiliza la arquitectura cliente/servidor. Ha incorporado en su sistema el modelo objeto-relacional, pero al mismo tiempo garantiza la compatibilidad con el tradicional modelo relacional de datos. Así ofrece un servidor de BD híbrido. Es uno de los más conocidos y ha alcanzado un buen nivel de madurez y de profesionalidad. Se destaca por su soporte de transacciones, estabilidad y escalabilidad.

El soporte que da Oracle a este tipo de sistemas no es otro que la arquitectura cliente/servidor. Donde nos encontramos una aplicación que se ejecuta tal y como su nombre indica parte en un servidor y parte en el cliente que accede a él.

Para ello utiliza un complejo sistema de buffers que utiliza para el traspaso y visualización de la información. Por ejemplo: Cuando un usuario ejecuta una instrucción, el SGBD interpreta la instrucción, si es correcta la ejecutará, sino devolverá un error. La función principal del SGBD es por lo tanto, la protección y control de la BD. (BERTINO y MARTINO, 1995)

Otras funciones del SGBD son:

- Definición de los datos: Ha de permitir crear todo el modelo de la BD y a la vez instrucciones para manipular los datos.
- Optimización del acceso: Debe buscar el camino más óptimo para realizar los accesos.
- Seguridad de los datos: Atendiendo a la seguridad que ha determinado el administrador de la BD debe encargarse de llevar a cabo la política de seguridad.

- Permitir acceso concurrente: Debe permitir a varios usuarios acceder a un mismo dato simultáneamente.

Gestión del diccionario de datos o repositorio: El repositorio incluye toda la BD propia de la configuración del SGBD, también incluye los distintos menús, programas, etc. Que realiza el usuario.

1.3. *Patrones empleados por Symfony.*

1.3.1 Patrón arquitectónico.

MVC es un patrón de arquitectura de software que separa la lógica de negocio de la interfaz de usuario, facilita la evolución por separado de ambos aspectos e incrementa la reutilización y la flexibilidad del sistema.

El modelo son las operaciones que se efectúan sobre los datos que se reciben, o las consultas a BD que se hacen, su objetivo es preparar los datos para que la vista solo se tenga que preocupar de pintarlo, así si hay que realizar cambios en la estructura de la BD o cambiar las operaciones que se hacen con los datos lo tendríamos que hacer sólo en el modelo, evitando que un cambio se tenga que realizar en todas las vistas que utilicen unos determinados datos.

Las vistas son el código HTML, lo que se muestra a ojos del usuario. Estas pintan, donde corresponda los datos que vienen por el request, en las variables globales y en las de sesión sin saber realmente lo que está pintando, puesto que el modelo es el que ha introducido los datos en ellas y pueden variar dependiendo de las circunstancias. Lo pintaría metiendo código PHP entre el HTML, simplemente consulta variables, algún bucle para recorrer arreglos, alguna condición que se deba cumplir para que pinte una parte o no (p ej.: comprobar si un arreglo viene vacío para pintarlo).(POTENCIER y ZANINOTTO, 2008)

El controlador es el que escucha los cambios en la vista y se los envía al modelo, el cual le regresa los datos a la vista, es un ciclo donde cada acción del usuario causa que se inicie de nuevo, un nuevo ciclo, es el que decide que vista se debe de imprimir y que información es la que se envía.

1.3.2 Patrones de diseño.

Los patrones son soluciones simples y elegantes a problemas específicos y comunes del diseño

Orientado a Objetos basadas en la experiencia, estos permiten reutilizar la experiencia de los desarrolladores, clasificar y describir formas de solucionar problemas que ocurren de forma frecuente en el desarrollo y está basado en la recopilación del conocimiento de los expertos en desarrollo de software.

Patrones (GoF)2

Patrones GoF (Gang of Four) o Cosa de Cuatro: Su traducción al español significa “Pandilla de Cuatro”, y describen las clases y objetos que se comunican entre sí, y se adaptan para resolver un problema general de diseño en un contexto particular. Estos vienen agrupados por los patrones creacionales para abstraer el proceso de instanciación y creación de objetos, los patrones estructurales, que describen como pueden ser combinados las clases y los objetos para formar grandes estructuras y por último los patrones de comportamiento para definir la comunicación entre los objetos del sistema. (BUSCHMANN y HENNEY, 1996)

Singlenton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia, Symfony lo utiliza para crear un contexto único, donde guarda las configuraciones del proyecto (sfcontext\frontend-dev.php) y el controlador frontal (sfWebFrontController) se encarga de enrutar todas las peticiones que se hagan a la aplicación.

Command (Comando): Encapsula una cierta cantidad de funcionalidades en una sola estructura haciendo este funcionamiento oculto para el usuario. Este patrón se observa en la clase sfWebFrontController, en el método dispatch. Esta clase esta por defecto y es la encargada de establecer el módulo y la acción que se va a usar según la petición del usuario. (BUSCHMANN y HENNEY, 1996)

Decorator (Decorador): Este método pertenece a la clase abstracta sfView, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente, el archivo nombrado layout.php es el que contiene el layout de la página, este conocido también como plantilla global, guarda el código HTML, que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout.

Iterator (Iterador): Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos, se encuentra dentro de PHP.

[2] GoF: (Gang of Four) llamados así por los cuatro autores del libro: “Patrones de Diseño”.

Visitor (Visitante): Permite definir nuevas operaciones sobre una jerarquía de clases sin modificar las clases sobre las que opera, Symfony lo emplea en el plugin sfDoctrine que es el que proporciona la interfaz para el Mapeo de objeto relacional (ORM).(BUSCHMANN y HENNEY, 1996)

Patrones GRASP (General Responsibility Asignment Software Patterns)

Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Constituyen el fundamento de cómo se diseñará el sistema.

Creador: Ayuda a identificar quien debe ser el responsable de la creación de nuevos objetos o clases, donde la nueva clase deberá ser creada por la clase que tiene toda la información necesaria para realizar la acción, que usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de clase y contiene o agrega la clase. En la clase Actions se encuentran las acciones definidas para el sistema y se ejecutan en cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que la clase Actions es “creador” de dichas entidades. (BUSCHMANN y HENNEY, 1996)

Bajo acoplamiento: La clase Actions hereda únicamente de sfActions para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista, lo que proporciona que la dependencia en este caso sea baja.

Alta cohesión: Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase Actions, la cual está formada por varias funcionalidades que están estrechamente relacionadas, siendo la misma la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, además se usa ya que cada elemento del diseño debe realizar una labor única dentro del sistema, con el objetivo de que las clases con responsabilidades estrechamente relacionadas no realicen un trabajo enorme.

Experto: Es uno de los patrones que más se utiliza cuando se trabaja con Symfony, con la inclusión de la librería Propel para mapear la BD. Symfony utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades

comunes de las entidades, las clases de abstracción de datos (Peer del Modelo) poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan.

Controlador: Todas las peticiones Web son manipuladas por un solo controlador frontal (sfActions), que es el punto de entrada único de toda la aplicación en un entorno determinado. Este patrón se evidencia en las clases sfFrontController, sfWebFrontController, sfContex, los “actions” y el index.php del ambiente. (BUSCHMANN y HENNEY, 1996)

Los patrones de diseño ayudan a una correcta elaboración de los proyectos y con los patrones GRAPS se asegura una correcta elaboración de las diferentes asignaciones de responsabilidades a objetos.

Conclusiones parciales

En el presente capítulo se demuestra la importancia que presenta la AGR para la seguridad y protección de la sociedad cubana, así como la importancia de tener un sistema informático que controle el Despacho de Cargas. Se describen diferentes sistemas informáticos tanto nacionales como internacionales que cumplen un objetivo similar, pero ninguno cumple con los requisitos funcionales requeridos, además la economía cubana no permite el mantenimiento de algunos de estos por su elevado costo.

Se propone como solución, el diseño e implementación del módulo, teniendo en cuenta la metodología, lenguajes y herramientas así como el modelo de la BD y los diferentes diagramas descritos en el capítulo.

Capítulo 2: Diseño e implementación del Despacho de Cargas.

En el presente capítulo se propone el diseño e implementación de un módulo, describiendo los artefactos del diseño que se generan durante el flujo de trabajo logrando entenderlos claramente para así describir las clases con sus atributos, métodos y algoritmos más importantes en la implementación del módulo, así como las estructuras de datos utilizadas para manejar los datos en el sistema, siendo compatible con la tecnología y arquitectura del Sistema de Gestión Integral de Aduana (GINA) y brindando una interface agradable, fácil de manipular y que se ajuste a los requerimientos del cliente.

2.1. *Modelo de Diseño.*

El modelo de diseño es un proceso y un modelado a la vez, es un conjunto de pasos repetitivos que permiten al diseñador describir todos los aspectos del sistema a construir, este debe implementar todos los requisitos explícitos contenidos en el modelo de análisis, debe ser una guía para los que construyan el código, los que prueban y mantienen el software, el cual debe ser claro y entendible, ya que debe proporcionar una completa idea de lo que es el software. El propósito del diseño es especificar una solución que trabaje y pueda ser fácilmente convertida en código fuente y construir una arquitectura simple y fácilmente extensible.

El diseño es la única manera de convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado.(PRESSMAN, 2005)

2.1.1 Requisitos

Una gran cantidad de aplicaciones informáticas son muy grandes y tienen muchos requerimientos por lo que es necesario desglosarlos para que el desarrollo del mismo sea más fácil, el módulo Despacho de Cargas del subsistema DNC cuenta con 8 requisitos funcionales, que constituyen un artefacto de entrada para realizar el diseño, los cuáles se listan a continuación:

Requisito	Descripción	Complejidad
Registrar	Registrar todos los datos de la declaración.	Alta

declaración		
Presentar declaración	Seleccionar todos los documentos complementarios al despacho de una operación no comercial.	Baja
Liquidar declaración	Poner al pago una declaración, lo cual implica calcular el total a pagar de acuerdo a las normativas aduaneras.	Alta
Cancelar declaración	Cancelar la declaración, donde se cancelan las operaciones que esta tenga registrada.	Media
Buscar Declaración	Buscar una declaración a partir de su número, año y aduana.	Media
Modificar declaración	Modificar los datos de la declaración.	Media
Registrar prórroga	Registrar los datos de una prórroga que se le otorgue a una declaración.	Baja
Registrar Operación sin DNC	Registrar los datos de una operación que no requiere de una declaración para su despacho.	Alta

2.1.2 Diagrama de paquetes

Un diagrama de paquetes muestra los paquetes de clases y las dependencias entre ellos, es decir, como el sistema está dividido en agrupaciones lógicas, se usa para reflejar la organización de los paquetes y sus elementos. Estos proporcionan una descomposición de la jerarquía lógica de un sistema, cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido.(ARQUITECT, 2007)

A continuación en la figura 2.1 se muestra el diagrama de paquetes donde se expone como interactúa el subsistema DNC con diferentes subsistemas del GINA, así como las librerías y paquetes del mismo.

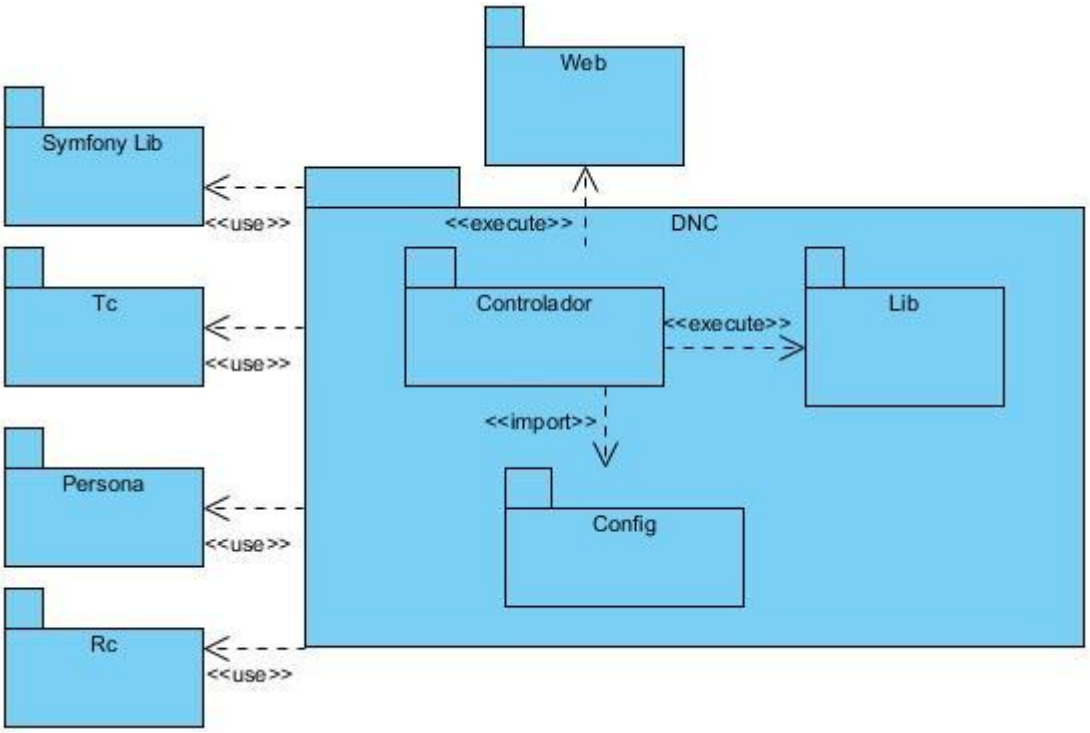


Figura 2.1. Diagrama de paquetes.

Dentro de los subsistemas que se relacionan con DNC se encuentra el subsistema Persona, este brinda toda la información referente a las personas, se encuentran también las Tablas de Control (TC), que se encargan de administrar los nomencladores de todos los subsistemas dentro del GINA, así como la interacción con el núcleo y la librería de Symfony. Dentro del mismo módulo se materializa el vínculo con diferentes paquetes como el Controlador que presenta las clases y objetos para el control de las peticiones y el flujo de acciones a realizar, el Lib donde se encuentran los formularios y algunas clases encargadas del mapeo de la BD, y el paquete Config, donde se encuentran las configuraciones del módulo, así como la relación con el paquete Web, encargado de la capa de presentación.

2.1.3 Diagrama de secuencia orientado a actividades

Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página con líneas discontinuas verticales, y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con qué otros

objetos y qué mensajes disparan esas comunicaciones. (SPARX, 2000)

En los diagramas de secuencias orientados a actividades se especifica claramente lo que el desarrollador tiene que implementar, ya que definen las relaciones entre las clases, así como el flujo de cada una de las actividades a realizar, por lo que es muy fácil para el implementador cuando no tiene conocimientos del negocio. Estos diagramas se dividen en dos partes, los orientados a la vista y los orientados al negocio.

En la figura 2.3 se representa el requisito Cancelar DNC orientado al negocio, donde se muestran los pasos lógicos a seguir para implementar dicho requisito, este flujo comienza en el método cancelarDnc en el actions recibiendo como parámetros el motivo por el cual desea cancelar la declaración y el identificador, con ese identificador se obtiene la declaración en forma de arreglo, en motivo y fecha de cancelación se le escribe los valores y se le cambia el estado a la declaración por cancelada, luego se crea el formulario para verificar que los datos introducidos sean correctos, si son correctos se guarda el objeto con los valores modificados y se muestra un mensaje satisfactoriamente al usuario.

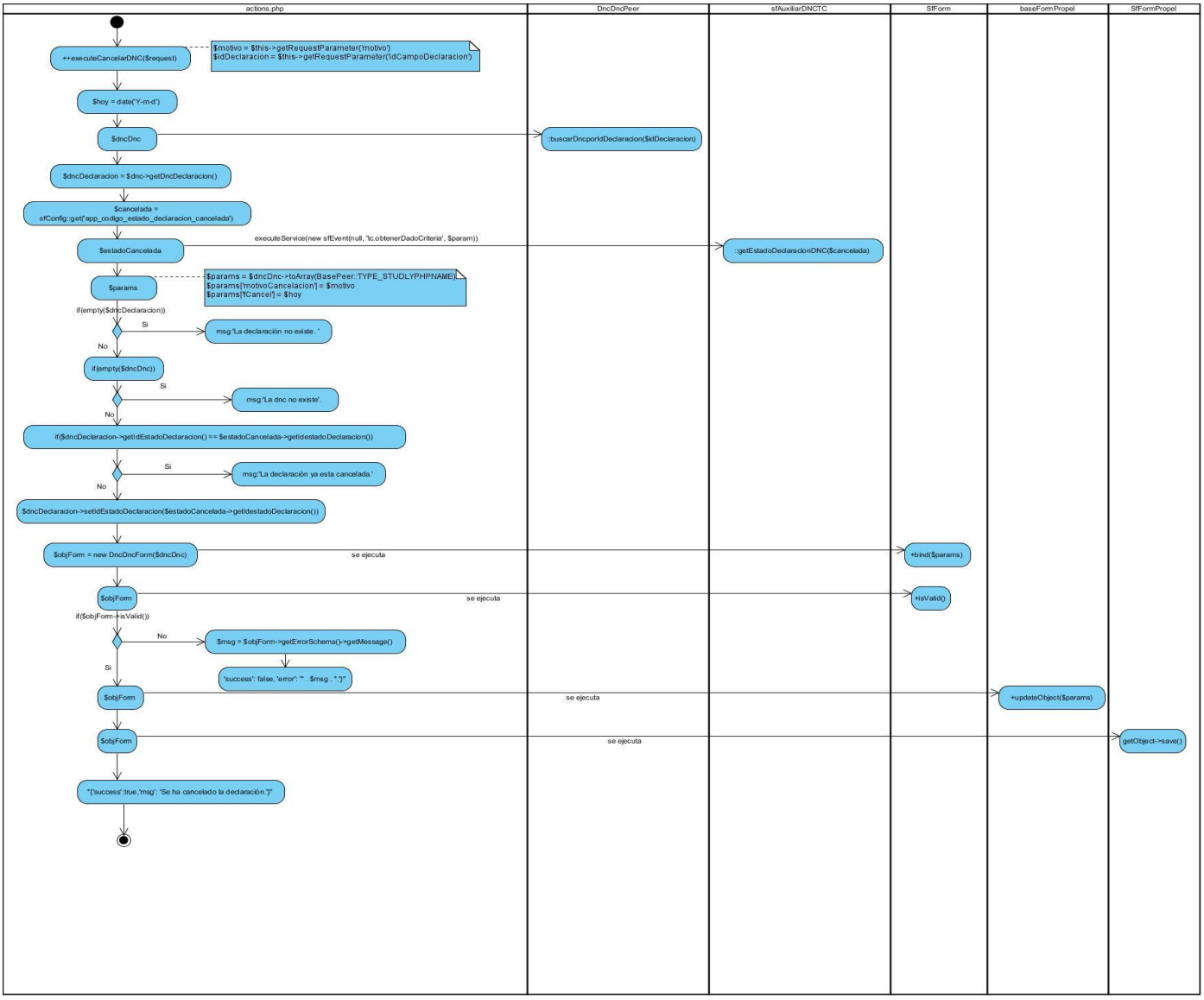


Figura 2.2. Diagrama de secuencia orientado a actividades del requisito Cancelar DNC. (Capa negocio)

En la figura 2.3 se representa el requisito Cancelar DNC orientado a la vista, donde se muestra cómo el actor comienza el flujo, introduciendo el motivo por el cual desea cancelar la declaración y el identificador obtenido de la declaración asociada, haciendo un submit al actions y este devolviendo un mensaje indicando que la declaración ha sido cancelada correctamente.

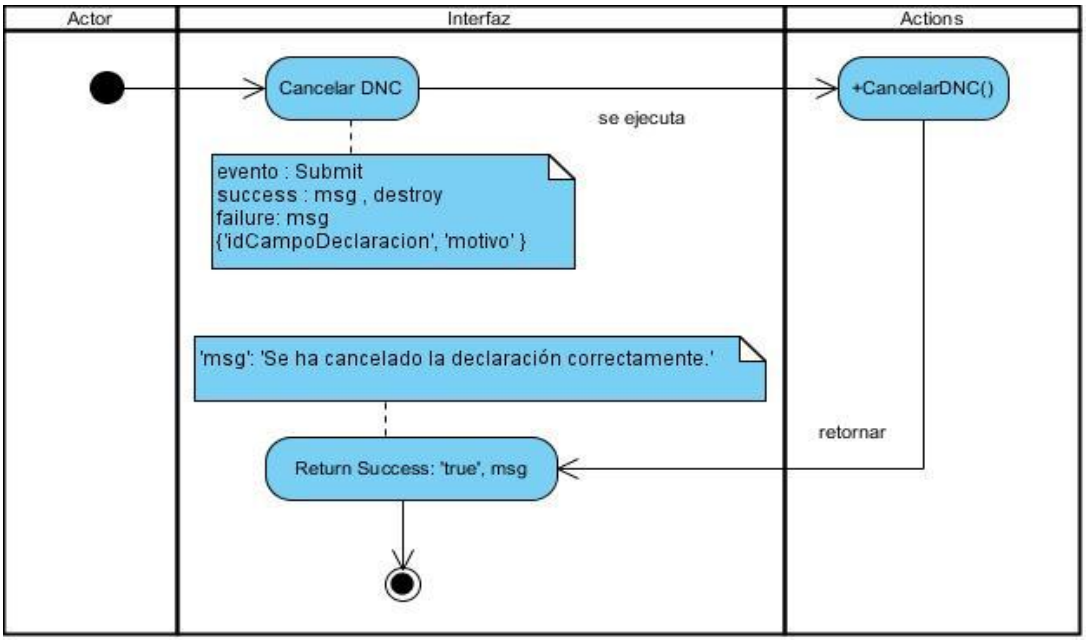


Figura 2.3. Diagrama de secuencia orientado a actividades del requisito Cancelar DNC. (Capa interfaz)

Para ver los diagramas de secuencia orientado a actividades correspondientes a los demás requisitos, dirigirse a la sección de “Anexos”, y para una mejor especificación, dirigirse al documento Modelo de Diseño v2.0 en el Repositorio del Proyecto de Aduana.

2.1.4 Diagrama de clases persistentes

La figura 2.4 muestra el diagrama de diseño de las clases persistentes del sistema, sirve como una primera aproximación al diseño definitivo del modelo de datos.

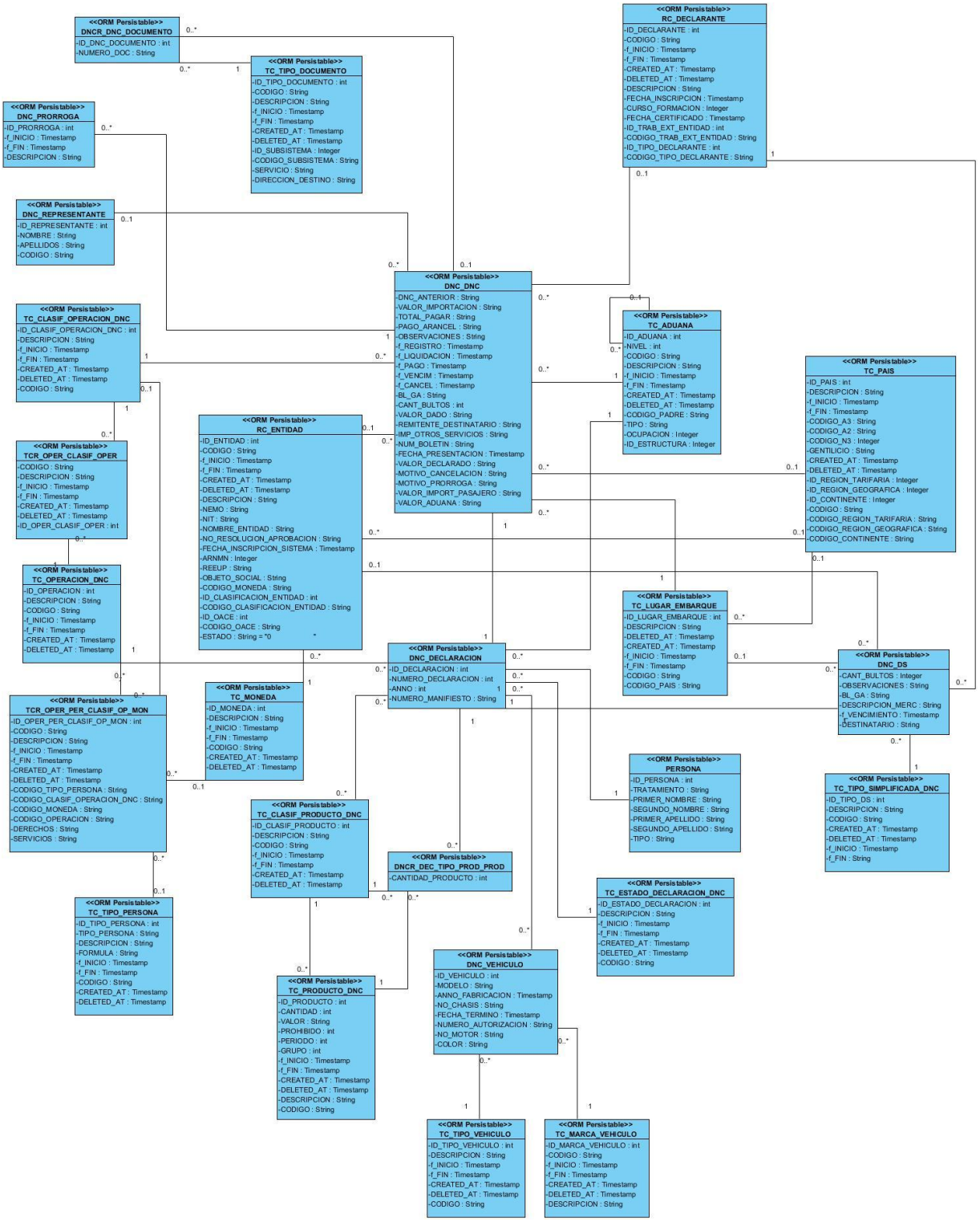

<<ORM Persistable>>
DNCR_DNC_DOCUMENTO
-ID_DNC_DOCUMENTO : int
-NUMERO_DOC : String
<<ORM Persistable>>
TC_TIPO_DOCUMENTO
-ID_TIPO_DOCUMENTO : int
-CODIGO : String
-DESCRIPCION : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-ID_SUBSISTEMA : Integer
-CODIGO_SUBSISTEMA : String
-SERVICIO : String
-DIRECCION_DESTINO : String
<<ORM Persistable>>
RC_DECLARANTE
-ID_DECLARANTE : int
-CODIGO : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-DESCRIPCION : String
-FECHA_INSCRIPCION : Timestamp
-CURSO_FORMACION : Integer
-FECHA_CERTIFICADO : Timestamp
-ID_TRAB_EXT_ENTIDAD : int
-CODIGO_TRAB_EXT_ENTIDAD : String
-ID_TIPO_DECLARANTE : int
-CODIGO_TIPO_DECLARANTE : String
<<ORM Persistable>>
DNC_PRORROGA
-ID_PRORROGA : int
-f_INICIO : Timestamp
-f_FIN : Timestamp
-DESCRIPCION : String
<<ORM Persistable>>
DNC_REPRESENTANTE
-ID_REPRESENTANTE : int
-NOMBRE : String
-APELLIDOS : String
-CODIGO : String
<<ORM Persistable>>
DNC_DNC
-DNC_ANTERIOR : String
-VALOR_IMPORTACION : String
-TOTAL_PAGAR : String
-PAGO_ARANCEL : String
-OBSERVACIONES : String
-f_REGISTRO : Timestamp
-f_LIQUIDACION : Timestamp
-f_PAGO : Timestamp
-f_VENCIM : Timestamp
-f_CANCEL : Timestamp
-BL_GA : String
-CANT_BULTOS : int
-VALOR_DADO : String
-REMITENTE_DESTINATARIO : String
-IMP_OTROS_SERVICIOS : String
-NUM_BOLETIN : String
-FECHA_PRESENTACION : Timestamp
-VALOR_DECLARADO : String
-MOTIVO_CANCELACION : String
-MOTIVO_PRORROGA : String
-VALOR_IMPORT_PASAJERO : String
-VALOR_ADUANA : String
<<ORM Persistable>>
TC_ADUANA
-ID_ADUANA : int
-NIVEL : int
-CODIGO : String
-DESCRIPCION : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-CODIGO_PADRE : String
-TIPO : String
-OCUPACION : Integer
-ID_ESTRUCTURA : Integer
<<ORM Persistable>>
TC_PAIS
-ID_PAIS : int
-DESCRIPCION : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CODIGO_A3 : String
-CODIGO_A2 : String
-CODIGO_N3 : Integer
-GENTILICIO : String
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-ID_REGION_TARIFARIA : Integer
-ID_REGION_GEOGRAFICA : Integer
-ID_CONTINENTE : Integer
-CODIGO : String
-CODIGO_REGION_TARIFARIA : String
-CODIGO_REGION_GEOGRAFICA : String
-CODIGO_CONTINENTE : String
<<ORM Persistable>>
TC_CLASIF_OPERACION_DNC
-ID_CLASIF_OPERACION_DNC : int
-DESCRIPCION : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-CODIGO : String
<<ORM Persistable>>
RC_ENTIDAD
-ID_ENTIDAD : int
-CODIGO : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-DESCRIPCION : String
-NEMO : String
-NIT : String
-NOMBRE_ENTIDAD : String
-NO_RESOLUCION_APROBACION : String
-FECHA_INSCRIPCION_SISTEMA : Timestamp
-ARNMN : Integer
-REEUP : String
-OBJETO_SOCIAL : String
-CODIGO_MONEDA : String
-ID_CLASIFICACION_ENTIDAD : int
-CODIGO_CLASIFICACION_ENTIDAD : String
-ID_OACE : int
-CODIGO_OACE : String
-ESTADO : String = "0 "
<<ORM Persistable>>
TCR_OPER_CLASIF_OPER
-CODIGO : String
-DESCRIPCION : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-ID_OPER_CLASIF_OPER : int
<<ORM Persistable>>
TC_OPERACION_DNC
-ID_OPERACION : int
-DESCRIPCION : String
-CODIGO : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
<<ORM Persistable>>
TC_LUGAR_EMBARQUE
-ID_LUGAR_EMBARQUE : int
-DESCRIPCION : String
-DELETED_AT : Timestamp
-CREATED_AT : Timestamp
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CODIGO : String
-CODIGO_PAIS : String
<<ORM Persistable>>
DNC_DECLARACION
-ID_DECLARACION : int
-NUMERO_DECLARACION : int
-ANNO : int
-NUMERO_MANIFIESTO : String
<<ORM Persistable>>
DNC_DS
-CANT_BULTOS : Integer
-OBSERVACIONES : String
-BL_GA : String
-DESCRIPCION_MERC : String
-f_VENCIMIENTO : Timestamp
-DESTINATARIO : String
<<ORM Persistable>>
TCR_OPER_PER_CLASIF_OP_MON
-ID_OPER_PER_CLASIF_OP_MON : int
-CODIGO : String
-DESCRIPCION : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-CODIGO_TIPO_PERSONA : String
-CODIGO_CLASIF_OPERACION_DNC : String
-CODIGO_MONEDA : String
-CODIGO_OPERACION : String
-DERECHOS : String
-SERVICIOS : String
<<ORM Persistable>>
TC_MONEDA
-ID_MONEDA : int
-DESCRIPCION : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CODIGO : String
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
<<ORM Persistable>>
PERSONA
-ID_PERSONA : int
-TRATAMIENTO : String
-PRIMER_NOMBRE : String
-SEGUNDO_NOMBRE : String
-PRIMER_APELLIDO : String
-SEGUNDO_APELLIDO : String
-TIPO : String
<<ORM Persistable>>
TC_TIPO_SIMPLIFICADA_DNC
-ID_TIPO_DS : int
-DESCRIPCION : String
-CODIGO : String
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-f_INICIO : Timestamp
-f_FIN : String
<<ORM Persistable>>
TC_CLASIF_PRODUCTO_DNC
-ID_CLASIF_PRODUCTO : int
-DESCRIPCION : String
-CODIGO : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
<<ORM Persistable>>
DNCR_DEC_TIPO_PROD_PROD
-CANTIDAD_PRODUCTO : int
<<ORM Persistable>>
TC_ESTADO_DECLARACION_DNC
-ID_ESTADO_DECLARACION : int
-DESCRIPCION : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-CODIGO : String
<<ORM Persistable>>
TC_TIPO_PERSONA
-ID_TIPO_PERSONA : int
-TIPO_PERSONA : String
-DESCRIPCION : String
-FORMULA : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CODIGO : String
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
<<ORM Persistable>>
TC_PRODUCTO_DNC
-ID_PRODUCTO : int
-CANTIDAD : int
-VALOR : String
-PROHIBIDO : int
-PERIODO : int
-GRUPO : int
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-DESCRIPCION : String
-CODIGO : String
<<ORM Persistable>>
DNC_VEHICULO
-ID_VEHICULO : int
-MODELO : String
-ANNO_FABRICACION : Timestamp
-NO_CHASIS : String
-FECHA_TERMINO : Timestamp
-NUMERO_AUTORIZACION : String
-NO_MOTOR : String
-COLOR : String
<<ORM Persistable>>
TC_TIPO_VEHICULO
-ID_TIPO_VEHICULO : int
-DESCRIPCION : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-CODIGO : String
<<ORM Persistable>>
TC_MARCA_VEHICULO
-ID_MARCA_VEHICULO : int
-CODIGO : String
-f_INICIO : Timestamp
-f_FIN : Timestamp
-CREATED_AT : Timestamp
-DELETED_AT : Timestamp
-DESCRIPCION : String

Figura. 2.4. Diagrama de clases persistentes.

2.1.5 Modelo Entidad-Relación

Es necesario tener una BD diseñada y actualizada correctamente para tener acceso a la información exacta para el desarrollo y uso del software, ya que en esta es donde se encuentra concentrada y almacenada toda la información referente al sistema a desarrollar.

El diseño de una BD es un proceso complejo que abarca decisiones a muy distintos niveles. Para llevar a cabo la implementación de la BD hay que tener en cuenta todas las fases de diseño de esta, como diseño conceptual, que es donde se crea el modelo Entidad-Relación (ER); diseño lógico, se transforman las entidades y relaciones del modelo anterior en tablas y en el diseño físico ya se implementan las tablas de la BD con sus características y almacenamiento interno.(DATE, 2001)

En la figura 2.5 se muestra el Diagrama Entidad-Relación(ER) que fue generado para guardar toda la información referente al módulo Despacho de Cargas, donde las clases de color carmelita son las arquitectónicamente significativas, ya que en estas se almacena la información referente al módulo, las amarillas son las nomencladoras, de estas solo se puede obtener información, no se pueden modificar, al igual que las verdes.

Este modelo ER esta normalizado en Tercera Forma Normal (3FN), donde todos los atributos de la relación dependen funcionalmente sólo de la clave, y no de ningún otro atributo.(DATE, 2001)

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.(VÁZQUEZ, 2011)

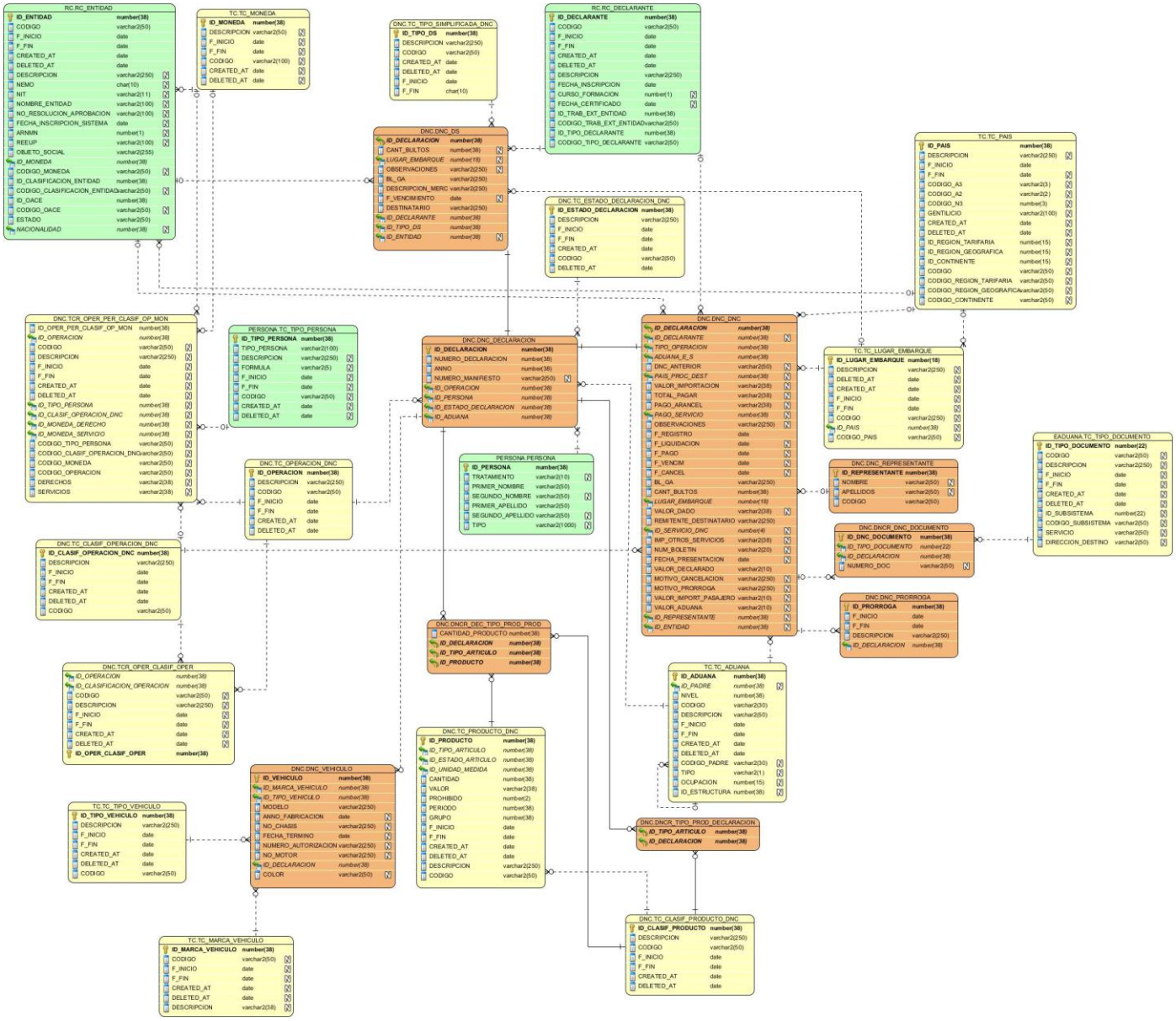


Figura 2.5. Diagrama Entidad-Relación.

2.2. *Validación del diseño.*

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

(PRESSMAN, 2005)

Métricas orientadas a clases.

Se sabe que la clase es la unidad principal de todo sistema Orientado a Objetos. Esto implica que las medidas y métricas para una clase individual, la jerarquía y las colaboraciones sean sumamente valiosas para un ingeniero de software que tenga que estimar la calidad de un diseño. (PRESSMAN, 2005)

Métricas propuestas por Lorenz y Kidd

Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones para una clase individual, y promedian los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos de clase, examinan la cohesión y asuntos relacionados con el código, y las métricas orientadas a valores externos examinan el acoplamiento y la reutilización.(PRESSMAN, 2005)

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

- ✓ **Tamaño Operacional de Clase**
- ✓ **Relaciones entre clases**

Tamaño Operacional de Clase (TOC): Esta métrica es muy usada por diseñadores de software, el tamaño general de una clase puede medirse determinando las siguientes medidas:

- El total de operaciones (operaciones tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase.
- El número de atributos (atributos tanto heredados como privados de la instancia), encapsulados por la clase.

Si los valores son grandes para TOC indican que la clase debe tener bastante responsabilidad. Esto reduciría la reutilización de la clase y complicaría la implementación y las pruebas.

Cuanto menor sea el valor promedio para el TOC, mayor la posibilidad de reutilizarse.(LORENZ y KIDD, 1994)

Parámetros de calidad	Valores grandes de TOC
Responsabilidad	La clase tiene bastante responsabilidad.
Reutilización	Reduce la reutilización de la clase.
Implementación	Complica la implementación.
Complejidad de las pruebas	Hace compleja las pruebas del sistema.

Tabla 1. Parámetros de calidad para valores grandes de TOC.

Medidas o umbrales aplicados.

TOC	Umbral
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Tabla 2. Umbrales para TOC.

Medidas para las clases arquitectónicamente significativas.

No.	Clase	Atributos	Operaciones	Tamaño
1	DncDeclaracion	8	5	Pequeño
2	DncDnc	40	7	Grande
3	DncDs	19	6	Medio

4	DncRepresentante	4	3	Pequeño
5	DncrDocumento	4	2	Pequeño
6	DncProrroga	3	2	Pequeño
7	DncrTipoProdDeclaracion	2	2	Pequeño
8	DncrDecTipoProdProd	4	2	Pequeño
9	DncVehiculo	11	2	Pequeño

Tabla 3. Clases de la capa de negocio a las que se les aplicó la métrica TOC.

Se les aplicó la métrica de TOC a un total de 9 clases, para un total de 95 atributos y 31 operaciones. De las clases analizadas se obtuvo un total de 7 clases de tamaño pequeño, 1 de tamaño medio y 1 clase grande.

Llevando los resultados obtenidos a un gráfico de por ciento se obtiene que del total de clases analizadas existe un 78% de clases de tamaño pequeño, un 11% de tamaño medio y un 11% de tamaño grande, como se muestra en la tabla 4.

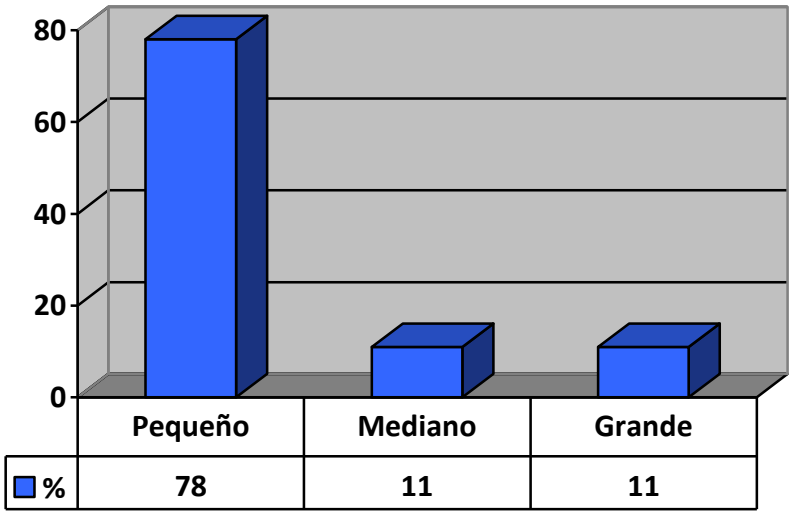


Tabla 4. Por ciento de clases por tamaño.

Después de mostrados los datos en las tablas anteriores se puede apreciar que la mayoría de las clases se clasifican en pequeñas, una mediana, y una grande, lo que implica un resultado positivo según los parámetros de calidad propuestos para esta métrica.

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

Parámetros de calidad	Valores grandes de RC
Acoplamiento	Implica aumento del acoplamiento de la clase.
Complejidad de mantenimiento	Implica aumento de la complejidad del mantenimiento de la clase.
Reutilización	Implica disminución en el grado de reutilización de la clase.

Tabla 5. Parámetros de calidad para valores grandes de RC.

Para un total de 9 clases arquitectónicamente significativas se obtuvo tras aplicar la métrica valores promedio de 4 asociaciones de uso por clase. (Tabla 6)

Total de clases	Promedio asociaciones de uso
9	4

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

En la tabla 7 se muestran los valores de los umbrales para esta métrica basada en el promedio de asociaciones de uso por clases.

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

Tabla 7. Valores de los umbrales para la métrica RC.

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

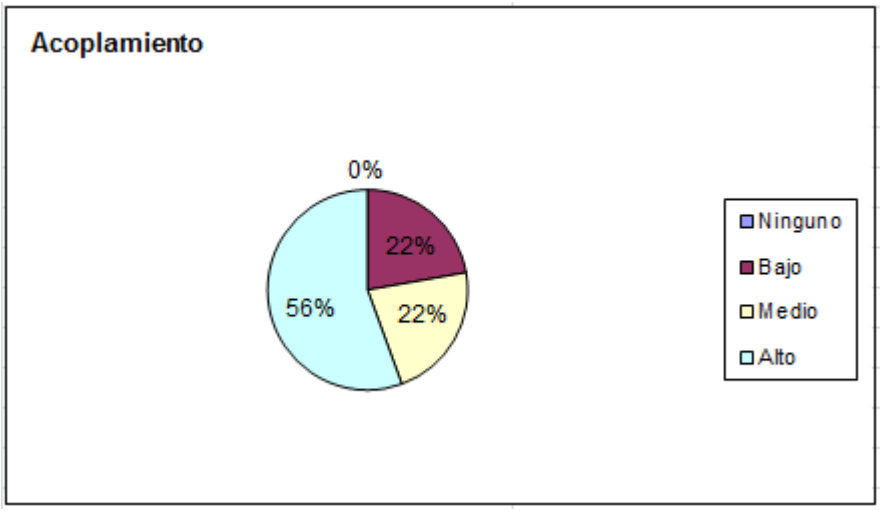


Figura 2.6.Representación de las clases según el acoplamiento.

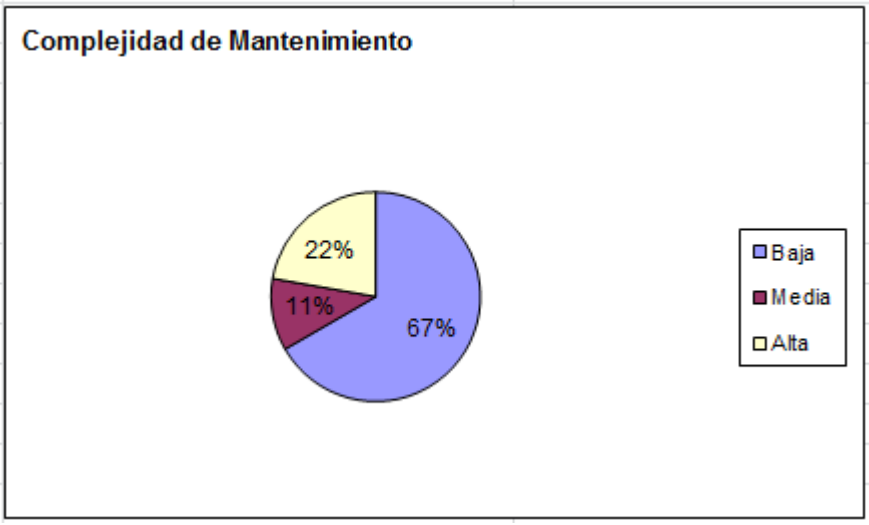


Figura 2.7. Representación de las clases según la complejidad de mantenimiento.

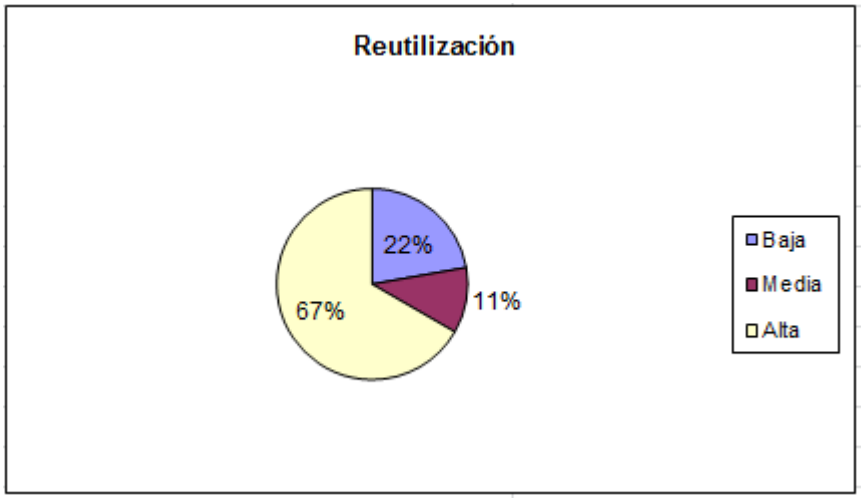


Figura 2.8. Representación de las clases según la reutilización.

La evaluación de los resultados obtenidos durante la aplicación de la métrica RC, demuestra que el diseño asumido tiene una calidad aceptable, a pesar de que hay un acoplamiento alto, producto a las relaciones entre clases, hay solo un 22% de complejidad de mantenimiento alto, y un 67% de alta reutilización, lo cual es un factor aceptable desde el punto de vista de la implementación del software. Estos resultados de los atributos de calidad se suman a los obtenidos por las pruebas de TOC y demuestran el uso de un buen diseño de software, lo que se puede proceder a la implementación.

2.3. *Modelo de implementación*

El objetivo del modelo de implementación, es traducir el modelo de diseño en diferentes componentes ejecutables de la aplicación a desarrollar. El modelo de implementación es una correspondencia directa de los modelos de diseño y de despliegue, en este flujo de trabajo se implementan las clases y objetos en ficheros fuente, ejecutables y demás. El resultado final de este flujo de trabajo es un sistema ejecutable.

Codificación de patrones

Patrón Arquitectónico MVC

Symfony está basado en el patrón MVC, que está formado por tres niveles. A continuación se muestra un ejemplo de cómo se evidencia el patrón arquitectónico MVC en el requisito “Presentar DNC” de la aplicación.

La vista se encarga de obtener y enviar la información necesaria hacia el controlador a través de peticiones.

```
this.form.submit({
    waitTitle: 'Espere por favor',
    waitMsg: 'Enviando...',
    url: 'noComercial/PresentarDNC',
    params: {
        idCampoDeclaracion: this.idDeclaracion
    },
```

Figura 2.6 Submit desde la interfaz PresentarDNC.js. (Vista)

Luego el controlador se encarga de obtener los datos enviados desde la vista y según la petición, realiza la acción correspondiente, la cual accediendo al modelo devolverá la respuesta determinada.

```
public function executePresentarDNC(){
    $idDeclaracion = $this->getRequestParameter('idCampoDeclaracion');

    hoy = date('Y-m-d');
    $dncDnc = DncDncPeer::buscarDncPorIdDeclaracion($idDeclaracion);
    $dncDeclaracion = $dncDnc->getDncDeclaracion();

    $params = $dncDnc->toArray(BasePeer::TYPE_STUDLYPHPNAME);
    $params['fechaPresentacion'] = $hoy;

    $presentada = sfConfig::get('app_codigo_estado_declaracion_presentada');
    $estadoPresentada = sfAuxiliarDNCTC::getEstadoDeclaracionDNC($presentada);

    $dncDeclaracion->setIdEstadoDeclaracion($estadoPresentada->getIdEstadoDeclaracion());

    $objForm = new DncDncForm($dncDnc);
    $objForm->bind($params);
    if($objForm->isValid()){
        $objForm->save();
        return $this->renderText("{'success':true,'msg': 'Se ha presentado la DNC correctamente.'}");
    }
    else{
        $msg = $objForm->getErrorSchema()->getMessage();
        return $this->renderText("{ 'success': false, 'error': '" . $msg . ".'}");
    }
}
```

Figura 2.7 Acción correspondiente a la petición. (Controlador)

```
public static function buscarDncPorIdDeclaracion($idDeclaracion){
    return self::retrieveByPK($idDeclaracion);
}
```

Figura 2.8 Función “buscarDncPorIdDeclaracion”. (Modelo)

Patrones del diseño

Singlenton (Instancia Única): En el app.yml de la carpeta config, de la aplicación dnc, se crean variables globales, con el objetivo de que si los valores asignados a las variables cambian, no sea necesario modificar el código, solo se cambiarían los valores; estas se crean con los diferentes estados que va a tener la declaración. En la figura 2.9 se evidencia este patrón.

change.log | schema.yml | schema.yml | actions.class.php | app.yml

```
1  # default values
2  all:
3    codigo_estado_declaracion_cancelada: C
4    codigo_estado_declaracion_pagada: P
5    codigo_estado_declaracion_registrada: R
6    codigo_estado_declaracion_liquidada: L
7    codigo_estado_declaracion_presentada: S
8    codigo_estado_declaracion_modificada: M
9    codigo_estado_declaracion_devuelta: D
```

Figura.2.9. Variables globales.

Decorator (Decorador): En el view.yml, de la carpeta config, del módulo noComercial, se van a escribir todas las rutas de los ficheros CSS y JS que se van a usar en el módulo. En la figura 2.10 se muestra el ejemplo.

```
all:
  metas:
    title:        Despacho Sin Car&aacute;cter Comercial

  stylesheets:    [ext/css/ext-all, %SF_GINA-STYLES%]
  javascripts:    [ext/ext-base, ext/ext-all, ext/locale/ext-lang-es, ext/ux/CheckColumn]


indexSuccess:

  javascripts:
    -  %SF_GINA-ALL%
    -  dnc/noComercial/windows
    -  dnc/noComercial/DNC
    -  dnc/noComercial/RegistrarDNC
    -  dnc/noComercial/LiquidarDNC
    -  dnc/noComercial/ModificarDNC
    -  dnc/noComercial/PresentarDNC
    -  dnc/noComercial/CancelarDNC
    -  dnc/noComercial/RegistrarDS
    -  dnc/noComercial/Prorroga
    -  dnc/noComercial/main


  has_layout:     on
  layout:         layout
```

Figura. 2.10. Clases a pintar en la vista.

Creador: En las clases Peer, se crean los objetos de dichas entidades. En la figura 2.11 se muestra como se crea una prórroga.

```php
<?php

class DncProrrogaPeer extends BaseDncProrrogaPeer
{
    public static function registrarProrroga($idDeclaracion, $antiguaFechaVencimiento,
                                             $nuevaFechaVencim, $motivoProrroga){
        $objForm = new DncProrrogaForm();
            $obj = array(
                'fInicio' => $antiguaFechaVencimiento,
                'fFin' => $nuevaFechaVencim,
                'descripcion' => $motivoProrroga,
                'idDeclaracion' => $idDeclaracion
            );
            $objForm->bind($obj);
            if(!$objForm->isValid()){
                throw new sfDatabaseException($objForm->getErrorSchema()->getMessage());
            }
            $objForm->save();
        }
```

Figura. 2.11. Funcionalidad “Registrar Prórroga”. (Modelo)

Diagrama de Componentes

Un Diagrama de componentes ilustra los fragmentos de software que conformarán un sistema. Este tiene un nivel de abstracción más elevado que un diagrama de clases, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, como también un componente puede comprender una gran porción de un sistema. (ARQUITECT, 2007)

En la figura 2.12 se muestra el Diagrama de componentes del módulo, este está compuesto por el componente interfaces, incluyendo los archivos JS con los que el usuario va a interactuar; dentro de DNC se encuentra la capa de abstracción de Symfony, compuesta por las clases de acceso a datos y del negocio, aquí se almacenan las generadas por Propel; el actions que va a controlar el flujo entre estos; en Configuración se encuentra el view.yml, que es el fichero que establece las configuraciones de los ficheros CSS y JS, el fichero database.yml va a contener las configuraciones de acceso a la BD y en services.yml, se van a encontrar todos los servicios, además la relación con los diferentes subsistemas sfRcIntegracionComun, sfPersonaPlugin y TC de los cuales se va a obtener información, así como las librerías de Symfony.

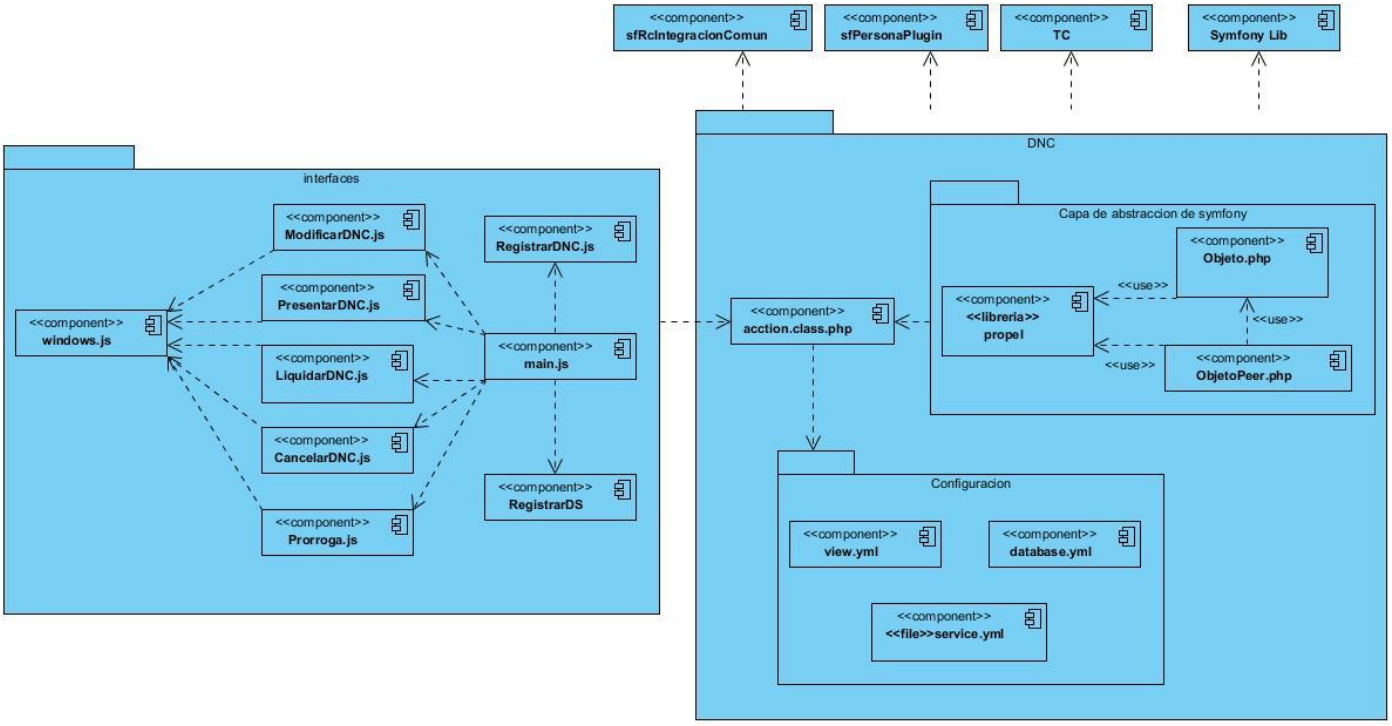


Figura 2.12. Diagrama de Componentes.

Prototipo de clases

Para el marco arquitectónico de los sistemas que se desarrollan en el Departamento de Soluciones para la Aduana se ha definido un framework de dominio específico construido a partir de la vinculación de los frameworks de desarrollo Symfony, de PHP, para el caso del controlador y ExtJS, de JS; para la capa de presentación, así como Propel para la persistencia de los datos.

Producto a que JS es un lenguaje interpretado por los navegadores en el cliente, y no comprende el paradigma de la POO, los conceptos de clase, componente y objeto son simulados a través de la propiedad prototype. Esta propiedad permite gestionar los atributos y métodos de un objeto similar en la POO. Ver figura 2.13.

El framework ExtJS maneja internamente la propiedad prototype, y propone su propia estructura para crear funciones que serán manejadas como clases, y que incluyen simuladamente el concepto de herencia en la implementación de la función *Ext.extend()*. De esta forma el framework permite crear nuevas funciones que heredan el comportamiento de otras funciones, por lo cual pueden ser denominados indistintamente los conceptos de clase, herencia, polimorfismo, atributo e instancia.(POTENCIER y ZANINOTTO, 2008)

Al igual que la función Ext.extend(), Ext.JS brinda las funciones Ext.ns() y Ext.reg(), para crear paquetes de clases y registrarlas para ser reconocidas por el framework respectivamente. Los componentes visuales a desarrollar se ajustarán a un modelo estructural de clase para su implementación, cumpliendo así el estándar propuesto por Ext.JS para la creación de nuevas clases.

```
//Nombre del paquete donde se incluyen los componentes
Ext.ns("Package");

//Nombre del componente dentro del paquete
Package.NameComponent = Ext.extend(

    //Nombre de la clase padre, ejemplo: Ext.Panel, Ext.form.Field
    Ext.form.Field,

    //Cuerpo de la nueva clase
    {
        //Atributos de la clase
        attribute1: 'default value',

        //Constructor de la clase
        initComponent: function(){
            //Cuerpo del constructor

            //Inicialización del constructor en la clase padre
            Package.NameComponent.superclass.initComponent.call(this);
        },

        //Métodos de la clase con sus respectivos argumentos
        method1: function(agrs0, agrs1, agrsN){

        }
});

//Se registra el componente para ser reconocido por Ext JS
Ext.reg('XTypeComponent', Package.NameComponent);
```

Figura 2.13. Modelo de clases para implementar los componentes visuales.

Petición AJAX-Submit

AJAX permite realizar una comunicación asíncrona entre la vista y el servidor en un segundo plano, de forma que se puedan hacer cambios sobre una página sin necesidad de actualizarse completamente. Submit es otra petición, pero esta si actualiza y envía la página completamente.

Para la comunicación asíncrona entre el modelo y la vista se utiliza JSON (JS Object Notation).

JSON es un formato sencillo para intercambiar datos. Consiste básicamente en un array asociativo que se utiliza para incluir información del objeto. JSON ofrece 2 grandes ventajas para las interacciones AJAX: es muy fácil de leer en JS y puede reducir el tamaño en bytes de la respuesta del servidor. (POTENCIER y

ZANINOTTO, 2008)

El formato JSON es el más adecuado para la respuesta del servidor, se realiza una petición, esta debe devolver una estructura de datos a la página que realizó la llamada de forma que se pueda procesar con JS. Este mecanismo es útil por ejemplo cuando una sola petición Ajax debe actualizar varios elementos en la página o cuando se hace un Submit enviando todos los datos de la página. (POTENCIER y ZANINOTTO, 2008)

El diagrama de la figura 2.14 muestra un ejemplo de peticiones tanto AJAX como Submit donde el cliente hace la petición al servidor, este recibe los datos y envía la respuesta al cliente ya sea satisfactoria o no, todo este proceso lo realiza mediante JSON que es el encargado del intercambio de datos entre la página cliente y la servidora.

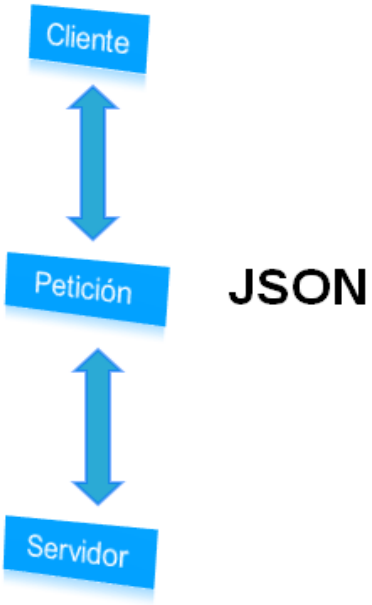


Figura 2.14 Peticiones.

Desde la versión 5.2 de PHP existen dos funciones, json_encode() y json_decode(), que permiten convertir un array PHP en un JSON y viceversa. Estas funciones facilitan escribir código PHP nativo más fácil de leer.(POTENCIER y ZANINOTTO, 2008)

Json_decode

Decodifica un string json, convierte un string codificado en json en una variable php.

```php
<?php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

var_dump(json_decode($json));
var_dump(json_decode($json, true));

?>
```

El resultado sería:

```
object(stdClass)#1 (5) {
    ["a"] => int(1)
    ["b"] => int(2)
    ["c"] => int(3)
    ["d"] => int(4)
    ["e"] => int(5)
}

array(5) {
    ["a"] => int(1)
    ["b"] => int(2)
    ["c"] => int(3)
    ["d"] => int(4)
    ["e"] => int(5)
}
```

Json_encode

Retorna un string con la representación del valor JSON.

```php
<?php
$arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);

echo json_encode($arr);
?>
```

El resultado sería:

```
{"a":1,"b":2,"c":3,"d":4,"e":5}
```

Actions y clases del modelo

Las BD son relacionales. PHP 5 y Symfony están Orientados a Objetos, por lo que el acceso y modificación de los datos se realiza mediante objetos, de esta forma nunca se accede de forma explícita a la BD. Para acceder de forma efectiva a la BD desde un contexto Orientado a Objetos, es necesaria una interfaz llamada Modelo de Objeto Relacional (ORM) que traduzca la lógica de los objetos a la lógica relacional, dicha interfaz está formada por objetos que permiten acceder a los datos y contienen en sí mismo, el código necesario para hacerlo. (POTENCIER y ZANINOTTO, 2008)

En las aplicaciones creadas con Symfony, el acceso y modificación de los datos almacenados en la BD se realiza mediante objetos, para garantizar esto Symfony utiliza Propel como ORM y Propel utiliza PDO[3] (PHP Data Objects) como capa de abstracción de BD. Estos dos componentes han sido desarrollados por el equipo de Propel también es un proyecto de Software Libre. Estos elementos combinados permiten tener un acceso de forma rápida a la información de la BD.

La principal ventaja que aporta el ORM es la reutilización, permitiendo llamar a los métodos de un objeto de datos desde varias partes de la aplicación e incluso desde diferentes aplicaciones. La capa ORM también encapsula la lógica de los datos y además la utilización de objetos en vez de registros y de clases en vez de tablas En la solución se utiliza Propel como capa de persistencia del modelo físico de datos. (POTENCIER y ZANINOTTO, 2008)

Para crear el modelo de objetos de datos que utiliza Symfony, se debe traducir el modelo relacional de la BD a un modelo de objetos de datos. Para realizar ese mapeo o traducción, el ORM necesita una descripción del modelo relacional, que se llama "esquema" (schema). En el esquema se definen las tablas, sus relaciones y las características de sus columnas.

El esquema se utiliza para construir las clases del modelo que necesita la capa del ORM, Symfony como framework de desarrollo, permite una abstracción de la BD a partir de la generación de cinco clases por cada entidad del modelo físico. A partir de la entidad objeto del modelo físico, el ORM del framework genera las siguientes clases:

BaseObjeto: Implementa todas las funciones nativas que se pueden efectuar sobre un objeto, tales como

[3] *Es una extensión que provee una capa de abstracción de acceso a datos para PHP 5*

los métodos get () y set () de los atributos de esta. Dado que su clase hija (Objeto) hereda estas funcionalidades básicas, también pueden ser utilizadas en cualquier parte del código que se desee escribir sobre ella.

Objeto: Esta clase aparece declarada pero con su cuerpo de código en blanco, se utiliza generalmente para agregar funcionalidades extras en caso necesario.

BaseObjetoPeer: No se deben modificar las funciones estáticas que se implementan en esta clase básica, pues se utilizan para realizar consultas y actualizaciones en las operaciones dirigidas a la entidad física correspondiente.

ObjetoPeer: Esta clase extiende de BaseObjetoPeer, se utiliza de forma similar a la clase Objeto, aunque está destinada fundamentalmente para desarrollar funciones estáticas que responden a reportes de búsqueda u actualización de información.

ObjetoMapBuilder: Estas clases representan mapas estáticos de las entidades del modelo físico de la BD, utilizadas por Propel en tiempo de ejecución para la construcción de consultas SQL[4] dinámicas partiendo de cláusulas o criterios predefinidos por las clases del núcleo del framework.

Las clases con nombre Base son las que se generan directamente a partir del esquema, no se deben modificar porque cada vez que se genera un nuevo modelo, se sobrescriben.

Las clases del modelo heredan de las clases con nombre Base, estas clases no se modifican cuando se genera el modelo, por lo que en esas si se pueden crear métodos propios de ella. Las clases de tipo “Peer”, son las que tienen métodos estáticos para trabajar con las tablas de la BD, estas permiten obtener los registros de las tablas, y sus métodos devuelven normalmente un objeto o una colección de objetos de la clase objeto relacionada.

La función de la clase action.class.php es realizar funciones, cálculos complejos, devolver objetos o arreglos de objetos, realizar comprobaciones de datos que provienen de la BD, las cuales se obtienen a través de las clases Peer. Esta obtención de datos se realiza a través de la clase Criteria ().

Clase Criteria

[4] Lenguaje de Consulta Estructurado, utilizado para acceder a los registros de una base de datos.

La clase Criteria permite generar reportes mucho más potentes que una simple consulta SQL, utilizando esta clase se optimiza el código SQL para la BD, todos los valores pasados a Criteria se filtran antes de ser insertados en el código SQL, para evitar problemas de SQL injection, devolviendo un arreglo de objetos y no un resultset12.

Tratamiento de errores

El tratamiento de errores es muy importante para que un software alcance el éxito, ya que son muy frecuentes en la programación. Existen dos tipos de errores: errores en tiempo de compilación (corregibles antes de hacer funcionar el programa), y errores en tiempo de ejecución, más complicados de encontrar, puesto que el error puede depender de las acciones del usuario y manifestarse solo ocasionalmente.

El módulo Despacho de Cargas del subsistema DNC gestiona información muy importante para la Aduana por lo que hay que realizar una serie de validaciones con el objetivo de que los errores no ocurran.

Usando Symfony es obligatoria la validación en el lado del servidor, con el objetivo de no corromper la BD con datos incorrectos, la validación del lado del cliente es opcional, y debe realizarse de forma manual con JS.

Las validaciones realizadas en la vista juegan un papel muy importante, ya que estas evitan que se introduzcan datos incorrectos en los diferentes campos, logrando así evitar ataques contra el sistema por esta vía.

Las validaciones de negocio son las más importantes, estas se realizan por medio de los formularios. Se encargan de controlar el flujo correcto de los datos introducidos.

Conclusiones Parciales

En este capítulo se han descrito un grupo de componentes ejecutables a través de diferentes diagramas, como el Diagrama de paquetes, Diagrama de clases persistentes, DER, los diferentes Diagramas de secuencias orientado a actividades, y el Diagrama de componentes. También se describieron algunas funciones y técnicas para llevar a cabo la implementación del sistema; a través de estos artefactos se le dio cumplimiento a los objetivos específicos: Realizar el Diseño y Realizar la Implementación del módulo Despacho de Cargas.

Capítulo 3: Validación del módulo

El desarrollo de sistemas de software lleva implícito una serie de actividades de los cuáles, los fallos y los errores son frecuentes desde el momento inicial.

Como parte del proceso de desarrollo de este, la fase de pruebas añade valor al producto que se maneja, en el desarrollo del software todos los programas tienen errores y la fase de pruebas tiene como objetivo específico encontrarlos.

Pruebas:

Las pruebas de software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y la codificación. Los procesos que verifican y revelan la calidad de un producto de software, son utilizados para identificar los posibles fallos de la implementación, calidad o la usabilidad de una aplicación informática.(PRESSMAN, 2005)

El objetivo de las pruebas es detectar un error no descubierto hasta entonces, y este tiene éxito si lo descubre.

El software debe probarse desde dos perspectivas diferentes:

- ✓ La lógica interna del programa se comprueba utilizando técnicas de diseño de casos de pruebas de "caja blanca"
- ✓ Los requisitos del software se comprueban utilizando técnicas de diseño de casos de prueba de "caja negra".

En ambos casos se intenta encontrar el mayor número de errores con la menor cantidad de esfuerzo y tiempo.(PRESSMAN, 2005)

3.1. *Pruebas Unitarias*

Al desarrollar un nuevo software, la primera etapa de pruebas a considerar es la etapa de pruebas unitarias o también llamada pruebas de caja blanca (White Box), estás pruebas también son llamadas pruebas modulares ya que permiten determinar si un módulo del programa está listo y correctamente terminado.

Las pruebas unitarias aseguran que un único componente de la aplicación produce una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular. Las pruebas unitarias se encargan de un único caso cada vez, lo que significa que un único método puede necesitar varias pruebas unitarias si su funcionamiento varía en función del contexto.(POTENCIER y ZANINOTTO, 2008)

Symfony incluye su propio framework llamado Lime. Este proporciona el soporte para las pruebas unitarias.

El objeto lime_test dispone de un gran número de métodos para las pruebas, a continuación se muestran algunos métodos utilizados:

ok ($prueba, $mensaje): Si la condición que se indica es true, la prueba tiene éxito mostrando el mensaje.

is($valor1, $valor2, $mensaje): Compara valor1 con valor2 si los dos son iguales la prueba tiene éxito y se muestra el mensaje.

isa_ok ($variable, $tipo, $mensaje): Comprueba si la variable que se le pasa es del tipo que se indica, mostrando un mensaje.

En la siguiente figura se muestra un ejemplo del código de la prueba unitaria realizada a la funcionalidad “insertarRepresentante”:

```
require_once dirname(__FILE__) . '/../../bootstrap/unit.php'; //
$configuration = ProjectConfiguration::getApplicationConfiguration('dnc', 'test', true);
$f = new sfDatabaseManager($configuration);
$test = new lime_test(1, new lime_output_color());

$result = DncRepresentantePeer::insertarRepresentante(123,'Juan','Perez');
$test->ok($result,'Se inserto el representante');
```

Figura 3.1. Prueba unitaria a la funcionalidad “insertarRepresentante”.

Lo primero que se hace es crear la configuración para la conexión a la BD, después se crea un objeto de tipo lime_test donde se le especifica el número de pruebas que se van a ejecutar a la funcionalidad y el segundo parámetro del constructor del objeto lime_test indica el objeto que se utiliza para mostrar los resultados, y luego se escribe el método utilizado.

En la figura se muestra la respuesta arrojada por esta funcionalidad:

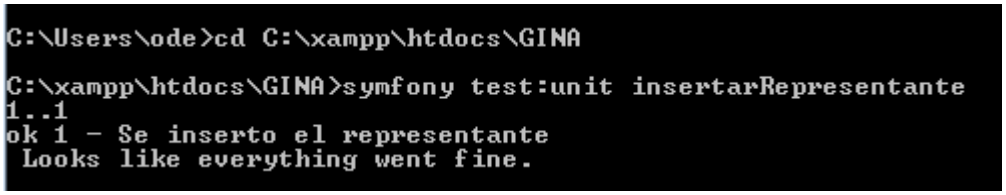


Figura 3.2. Respuesta de la funcionalidad “insertarRepresentante”.

Se le realizó pruebas unitarias a un total de 15 funcionalidades, realizando dos iteraciones, en la primera arrojando 6 no conformidades, estas se resolvieron y ya en la segunda iteración no se detectó ninguna. Para ver los otros Registros de Pruebas Unitarias ir al Repositorio del proyecto de aduana.

3.2. *Pruebas de Caja Negra*

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores. La prueba de caja negra intenta encontrar errores de las siguientes categorías: (1) funciones incorrectas o ausentes, (2) errores de interfaz, (3) errores en estructuras de datos o en accesos a BD externas, (4) errores de rendimiento y (5) errores de inicialización y de terminación.

Como resultado de las pruebas de caja negra se realizó un caso de prueba por cada requisito, en los cuales se detectaron varias no conformidades en revisiones internas realizadas por el equipo de desarrollo, las cuales fueron resueltas. Estas pruebas se realizaron tomando los diseños de casos de pruebas basados en requisitos.

El diseño de casos de pruebas basado en requisitos se realiza a partir de una plantilla definida en el expediente 3.3 del proceso de mejora del nivel 2 de CMMI. Esta plantilla tiene 2 páginas, en la primera se describe qué debe suceder cuando se ejecuta la funcionalidad que se está probando. Para esto se especifican diferentes combinaciones de datos válidos y no válidos para verificar cuál es la respuesta del sistema ante estos tipos de entrada. En la figura 3.4 se muestra el diseño de caso de pruebas del requisito “Registrar Prórroga”.

Condiciones de ejecución

El usuario debe estar autenticado en el sistema.
La declaración debe estar registrada en el sistema.

SC Registrar prórroga

Escenario	Descripción	Número	Año	Aduana	Fecha	Motivo de prórroga	Respuesta del sistema	Flujo central
EC 1.1 Registrar prórroga.	Se registran los datos de una prórroga asignada a una declaración.	V 387	V 2012	V AAI JOSE MARTI	N/A vacío	N/A vacío	El sistema debe mostrar un mensaje indicando que debe de registrarse el motivo y la fecha de la prórroga.	1- Seleccionar del menú la opción Prórrogas. 2- El sistema muestra la pantalla para buscar los datos de la DNC. 3- Introducir número, año y aduana de la DNC. 4- Seleccionar la opción Buscar. 5- Muestra los datos de la DNC especificada. 6- Registrar la nueva fecha de vencimiento y el motivo de la prórroga. 7- Valida que se hayan registrado correctamente todos los datos solicitados y muestra un mensaje indicando que se registró la prórroga, se actualiza la fecha de vencimiento de la declaración.
		V 387	V 2012	V AAI JOSE MARTI	V 05/07/2012	V Motivo de prórroga	El sistema debe mostrar un mensaje donde indique que se ha registrado la prórroga correctamente.	

[Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.]

Figura 3.4. Diseño de casos de prueba del requisito “Registrar Prórroga”.

En la segunda página se describen cada uno de los parámetros de entrada que se deben validar cuando se ejecuta el requisito, mostrando de cada uno su clasificación, si acepta valor nulo y la descripción. Esto se realiza en una tabla como se muestra a continuación en la figura 3.5.

Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Número	Campo texto	No	Número de la DNC (String de números de 5 dígitos).
2	Año	Campo texto	No	Año de la DNC (String de números de 4 dígitos).
3	Aduana	Lista desplegable	No	Lista que contiene el nombre de todas las aduanas.
4	Fecha	Campo fecha	No	Formato de fecha establecido.
5	Motivo	Campo texto	No	Motivo por le cual se le otorga una prórroga (String).

Figura 3.5. Parámetros de entrada al caso de prueba del requisito “Registrar Prórroga”.

Al realizar la primera iteración se registraron las no conformidades detectadas en una plantilla como se muestra en la figura 3.6, las cuales fueron corregidas satisfactoriamente.

												Estado		
Fecha	Turno	Probador	Elemento	Etapa de detecció	No	No conformidad	Aspecto correspondiente	Tipo	Significativa	No Significativa	Recomendación	RA	PD	NP
5/29/2012	mañana	Iliana de la Rosa Zayas	Aplicación	1	1	Cuando se registra la nueva fecha permite que sea menor que la fecha de cumplimiento.	Aplicación	Validación	X				x	

Elemento	Cantidad de NC	Significativas	No significativas	Recomendacion		No correspondencia con el CU	Acciones que no funcionan	Validación	Funcionalidad	Ortografía	Excepciones	Interfaz	Otros
Documenta													
Aplicación	1	1						1					

Figura 3.6.No conformidades del caso de prueba del requisito “Registrar Prórroga”.

Para ver los otros diseños de casos de pruebas y no conformidades detectadas, dirigirse al Repositorio del proyecto de aduana.

Conclusiones Parciales

En el desarrollo del presente capítulo se validó el sistema desarrollado, durante el desarrollo de este se realizaron pruebas unitarias con el objetivo de verificar si un módulo del programa está listo y correctamente terminado. Symfony tiene la ventaja de incluir su propio framework para pruebas, donde este dispone de un gran número de métodos para la realización de estas pruebas, se realizaron dos iteraciones, en la primera se detectaron varias no conformidades y ya en la segunda se verificó que las no conformidades fueron resueltas, además se hicieron pruebas de caja negra para verificar que los requisitos funcionales del software responden a las necesidades del cliente con calidad.

Conclusiones generales

Con la culminación del presente trabajo de diploma se obtuvo un módulo, Despacho de Cargas para el GINA cumpliendo con los requisitos identificados.

- ✓ Por otra parte se le dio cumplimento a los objetivos específicos propuestos, para los cuales se elaboró el marco teórico de la investigación a partir del estudio del arte; el mismo evidenció la necesidad de un sistema informático para controlar el tráfico de mercancías entre aduanas, cumpliendo con los requerimientos del cliente. Por otra parte se analizaron diferentes tecnologías, lenguajes y herramientas indispensables para llevar a cabo el desarrollo de la investigación.

- ✓ Se realizó el diseño e implementación del módulo Despacho de Cargas, erradicando los problemas del sistema existente y fusionando los nuevos requisitos identificados teniendo en cuanta las necesidades del cliente.

- ✓ Finalmente, se aplicaron pruebas unitarias y pruebas de caja negra al software implementado, las cuales demostraron el cumplimiento de diferentes atributos de calidad, lo que demuestra que los componentes cumplen satisfactoriamente con los requerimientos definidos por el cliente.

Glosario de términos

Aduana: Oficina pública de constitución fiscal establecida generalmente en costas y fronteras. Su objetivo es registrar el tráfico internacional de mercancías que se importan o exportan desde un país concreto y cobrar los impuestos que establezcan las aduanas.

Personas Naturales: Es todo ser humano o individuo que nace u obtiene la capacidad legal en la sociedad sin importar edad, sexo o religión.

Personas Jurídicas: Es un ente ficticio que obtiene la capacidad legal porque la ley le asigna poder para contratar y contraer obligaciones con representación de una persona natural.

Plugin: Un Plugin es un programa que puede anexarse a otro para aumentar sus funcionalidades.

Requisitos funcionales: Son los que definen el comportamiento interno del software.

AJAX (JS asíncrono y XML): Es la unión de varias tecnologías de desarrollo Web que buscan crear aplicaciones interactivas, la importancia de ésta técnica está en que al utilizarla, se ejecuta en el lado del cliente, es decir el navegador Web, pero por debajo mantiene una comunicación asíncrona con el servidor.

Importación: Es el transporte legítimo de bienes y servicios nacionales exportados por un país, pretendidos para el uso o consumo interno de otro país.

Exportación: Es cualquier bien o servicio enviado a otra parte del mundo, con propósitos comerciales. La exportación es el tráfico legítimo de bienes y servicios nacionales de un país pretendidos para su uso o consumo en el extranjero.

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

Recomendaciones

Aún con el cumplimiento de los objetivos, durante el desarrollo del mismo han surgido ideas que serían recomendables tener en cuenta para su futuro perfeccionamiento:

- ✓ Integrar el módulo al subsistema de administración del GINA.
- ✓ Desplegar el módulo en la AGR.

Referencias bibliográficas

ACHOUR, M.; FRIEDHELM, B.*, et al. Manual de PHP* Disponible en: http://in2.php.net/distributions/manual/php_manual_es.html.gz.

AFIP, A. D. A. *AFIP Administración Federal* Argentina: [Consultado el: Febrero de 2012]. S.I.M. en Línea. Disponible en: http://www.afip.gov.ar/aduana/sim/descripcion1.asp.

AGR, A. G. D. L. R. D. C. Sistema Único de Aduana 2006 nº

AGR, A. G. D. L. R. D. C. *SITIO WEB DE LA ADUANA CUBANA* Habana: [Consultado el: Enero de 2011]. Disponible en: http://www.aduana.co.cu/.

ALVARADO, D.; JIMENEZ, B.*, et al. El método científico y el método histórico.* Investigativa, Ciencias Sociales, Filosofía y Religión. Ricardo Palma, 2007.

ARQUITECT, E. *Guía de Usuario de Enterprise Architect 7.0* [Consultado el: Mayo de 2011]. Disponible en: http://www.sparxsystems.com.ar/download/ayuda/index.html?packagediagram.htm.

BERTINO, E. y MARTINO, L. *Sistemas de bases de datos orientadas a objetos.* Díaz de Santos ed. 1995, En Díaz de Santos. vol. 1, 278 p. Disponible en: http://books.google.com.cu/books/about/Sistemas_de_bases_de_datos_orientadas_a.html?id=-XohLQySVNMC&redir_esc=y. ISBN 9780201653564.

BUSCHMANN, F. y HENNEY, K. *Pattern-oriented software architecture.* 1996, 487 p. ISBN 0471958897.

CHACON, J. C. R. *Aplicación de la metodología rup para el desarrollo rápido de aplicaciones basado en el estándar j2ee.* Investigativa, Nómina de junta directiva. Universidad de San Carlos de Guatemala, 2006

DATE, C. J. *Introducción a los Sistemas de Bases de Datos.* 7 tima ed. Argentina: 2001 Disponible en: http://books.google.co.ve/books?id=Vhum351T-K8C&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false. ISBN 968-444-419-2.

EGUÍLUZ, J. P. *Introducción a AJAX.* 2007, vol. 1, 241 p. Disponible en: http://www.librosweb.es/ajax/capitulo10/otros_frameworks_importantes.html.

EGUÍLUZ, J. P. *Introducción a CSS.* publicado el: Diciembre 15 de 2008 última actualización: Diciembre 15. vol. 1, Disponible en: http://www.librosweb.es/css.

FALGUERAS; BENET*, et al. Ingeniería del Software.* Editorial UOC ed. 2003, Disponible en: http://www.iberlibro.com/Ingenier%C3%ADa-Software-Falgueras-Benet-Campderrich-Editorial/5821445767/bd. ISBN 9788484297932.

LORENZ, M. y KIDD, J. *Object-Oriented Software Metrics* Hardcover ed. Prentice Hall 1994, ISBN 978-0-13-179292-0.

MOTA, E.; GOMEZ, A.*, et al. Indizen* Madrid: Disponible en: http://www.indizen.com/services/ilabs/herramientas/.

NICODEMOS, S. B. Sistema María, una necesaria evolución constante. 1010, nº p. 6. Disponible en: www.caei.com.ar/es/programas/comercio/6.pdf.

PACHECO, S. R. L.; SOTOMAYOR, G. L.*, et al.* SISTEMA INFORMÁTICO PARA LA GESTIÓN DE LA PLANIFICACIÓN DOCENTE. *PLANIFICACIÓN DOCENTE*, 2010 nº p. 22. Disponible en: http://tallertematico2010.fordes.co.cu/public/site/104.pdf.

PÉREZ, F. U. G. Lenguajes de programación orientada a objetos.. . 2006 nº Disponible en: http://www.articulandia.com/premium/article.php/25-09-2006Lenguajes-de-programacion-orientada-a-objetos.htm.

PINEDA, L. D. L. N.; RODRÍGUEZ, Y. B.*, et al.* SISTEMA INFORMÁTICO PARA LA GESTIÓN Y CONTROL DE LOS DEPÓSITOS DE ADUANA. 2010 nº p. 17. Disponible en: http://semanatecnologica.fordes.co.cu/ocs-2.3.2/public/site/275.pdf.

POTENCIER, F. y ZANINOTTO, F. *Symfony, la guía definitiva.* 2008 vol. 1, Disponible en: http://www.librosweb.es/symfony_1_1.

PRESSMAN, R. S. *Ingeniería del Software Un enfoque practico.* Editado por: Ince, D. Quinta ed. 2005 vol. 11,

SPARX, S. *Enterprise Architect* Disponible en: http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html.

VÁZQUEZ, M. F. *Diseño e Implementación del módulo Despacho de Tripulantes del Sistema de Gestión Integral Aduanera.* Diseño e implementación, CEIGE. Universidad de las Ciencias Informáticas., 2011

ZAYAS, F. I. D. L. R. *Análisis del módulo Despacho de Pasajeros, en la Aduana General de la República.* Analisis, CEIGE. Universidad de la Ciencias Informaticas, 2010