Universidad de las Ciencias Informáticas

Facultad 3

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

Título: Diseño e Implementación del módulo Despacho de Tripulantes del Sistema de Gestión Integral Aduanera.

Autor: Maydel Vázquez Fernández

Tutores: Ing. Iliana de la Rosa Zayas Frías

Ing. Lázaro Manuel Gil Martínez

Ciudad de La Habana, Junio 2011
"Año del 52 Aniversario del Triunfo de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser 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 _____.

Maydel Vázquez Fernández

_____

Firma del Autor

Ing. Lázaro Manuel Gil Martínez

_____

Firma del Tutor

Ing. Iliana de la Rosa Zayas Frías

_____

Firma del Tutor

"... He cometido errores, pero ninguno estratégico, simplemente táctico (...) No tengo ni un átomo de arrepentimiento de lo que hemos hecho en nuestro país. ..."

Fidel Castro

Agradecimientos

No soy muy buena para estas cosas pero no puedo dejar de mencionar a muchas personas que de una forma u otra colaboraron con este trabajo e hicieron posible que mi sueño se convirtiera en realidad.

Quisiera agradecerle a mi mamá que depositó toda su confianza en mí y gracias a ella soy la persona que soy hoy, gracias mami, eres una persona muy especial en mi vida.

A mi abuela que más que mi abuela ha sido mi segunda madre, gracias por estar ahí siempre que te necesité.

A mi padrasto que ha sido el padre que nunca tuve.

A mi familia que siempre estuvo ahí para apoyarme y ayudarme a elegir el camino, a mis tíos, mis primos a todos que los quiero mucho y no sé que hubiera sido de mi sin ustedes.

A mis compañeros del aula que aunque siempre estuvimos discutiendo, han sido como mi segunda familia y fueron partícipes de mi victoria.

A mis amigas Lili, Mariem, Odelkis, no sé que hubiera sido de mi en esta universidad sin ustedes, ustedes que más que mis amigas son mis hermanas, gracias por estar en los momentos buenos y en los malos.

Quisiera agradecer también a mis tutores Lázaro e Iliana, gracias por soportarme día a día y gracias por ayudar a convertirme en lo que soy, siempre los tendré presente.

Gracias a todos, Adrian, Ricardo, Elizabeth, Ramses, Yaksel, Gretter, JJ, discúlpenme si me queda alguien sin mencionar, los quiero mucho a todos.

Todo en la vida merece un sacrificio y me siento muy contenta por haber alcanzado este triunfo y quisiera dedicárselo a una persona muy especial que aunque no está conmigo en estos momentos, creo que le hubiera gustado mucho verme triunfar en la vida, esa persona es mi abuelo que me dio todo su esfuerzo y dedicación para hacerme una mujer de bien.

A mi familia por apoyarme siempre…

El objetivo fundamental de la Aduana General de la República (AGR) es el control de personas, artículos o mercancías y tripulantes 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. En este trabajo se hace referencia al Despacho con Carácter no Comercial (DNC) para el cual se desarrolla el siguiente sistema con el objetivo de informatizar dicha información para lograr una mejor organización y control del despacho de tripulantes que será el módulo sobre el que se está trabajando.

Para darle solución a las necesidades del cliente se desea realizar el diseño e implementación de un módulo que cumpla con dichos requerimientos. Para ello se utilizaron las siguientes herramientas como medio de realización: Visual Paradigm para el modelado del diseño, Eclipse para la implementación además de utilizar metodología RUP de conjunto con la notación BPMN logrando con ellos darle respuesta a los requisitos del cliente, dándole una solución completa y obteniendo un sistema que cumpla con lo planteado anteriormente.

Introducción.

Desde el triunfo de la revolución, Cuba ha seguido un camino destinado a incorporar la informática y las nuevas tecnologías de la información y las comunicaciones (TIC) a la sociedad. Como parte de esta estrategia se ha desarrollado todo un proceso de informatización que tiene como objetivo elevar la eficiencia y calidad de los servicios que brinda en varios sectores de la sociedad cubana, como es el sector Aduanero.

Las Aduanas a nivel mundial constituyen barreras que protegen al país del intercambio comercial de los productos procedentes del exterior. Cuando se habla de comercio internacional necesariamente implica la circulación de bultos, personas, mercancías y tripulantes. La capacidad de mover artículos a través de la frontera de forma rápida y segura a costos razonables hace que un país entre en competencia y se impulse hacia un mayor desarrollo. A su vez este tráfico puede llevar a diversos riesgos que pueden comprometer la seguridad del propio país, la economía nacional y la salud de sus pobladores. Por ello las aduanas son muy importantes ya que juegan un rol crítico en la efectividad de los controles que aseguren las recaudaciones, en el cumplimiento de la legislación nacional y en garantizar la protección y seguridad de la sociedad.

La Aduana General de la República de Cuba (AGR) hace cumplir estas leyes y regulaciones que forman parte del sistema de control estatal y actúa en función de ello para garantizar la seguridad y protección de la sociedad socialista y de la economía nacional, así como de la seguridad de todos los que habiten en la isla. Entre las operaciones que realiza la aduana cubana, se encuentran las que no poseen carácter comercial, las cuales se registran dentro del Despacho no Comercial (DNC) donde se lleva el control de las importaciones y exportaciones de un conjunto de personas que son tratadas por la Aduana como tripulantes. Los tripulantes cubanos de buques y aeronaves que cubren rutas internacionales, se rigen para sus actividades por disposiciones diferentes a las del resto de los viajeros, aunque están sujetos a las mismas regulaciones en cuanto a artículos no permitidos o que necesitan autorización para su importación o exportación.

Actualmente la aduana cuenta con un sistema para la gestión de sus operaciones, el Sistema Único de Aduana (SUA), dividido en diferentes módulos, entre los cuales se encuentra el de Despacho No Comercial, en el que se registran, entre otras cosas, las actividades de importación y exportación que realizan los tripulantes. El SUA cuenta con características que en el momento de su creación, le brindó una solución efectiva a la AGR, pero con el desarrollo del país, el número de personas circulantes en la frontera ha aumentado en gran medida y la agilidad en el desarrollo de los despachos aduaneros se hace indispensable para viabilizar el tráfico de personal en los puertos y aeropuertos, esto provoca que exista poca flexibilidad y agilidad en el desarrollo del Despacho de Tripulantes.

De igual forma la AGR ha ido incorporando nuevas normativas y modificaciones de antiguas regulaciones que con el desarrollo de un nuevo sistema supone menor costo, tiempo y recursos antes que darle soporte al sistema anterior SUA, además que este sistema no es flexible ante estas normativas, por otra parte esta aplicación maneja un gran volumen de información manipulada por cada módulo por separado sin lograr una integración, lo que provoca ambigüedad en los datos que quedan registrados por lo que es muy engorroso en el momento de ser consultada u obtener alguna información, por ello requiere ser centralizada, de ahí que la aduana, en función de mejorar el desarrollo de sus operaciones, plantea la necesidad de implementar un nuevo sistema para agilizar la gestión integral aduanera (GINA) logrando así mayor agilidad en los procesos que realiza la misma, alcanzando un mayor control de los tripulantes dentro del Despacho no Comercial.

A partir de la situación planteada se identifica:

Problema a resolver:

¿Cómo obtener un producto que agilice el Despacho de Tripulantes para el Sistema de Gestión Integral de Aduanas?

Objeto de estudio:

Los sistemas de la Aduana General de la República de Cuba.

Campo de acción:

Las tecnologías y herramientas del despacho de tripulantes en la Aduana General de la República de Cuba.

Como idea a defender para esta investigación se plantea que:

Si se desarrolla un sistema para el Despacho de Tripulantes permitirá que se agilicen los procesos del Despacho no Comercial relativos al control de tripulantes en el mismo.

Objetivo general:

Realizar el diseño y la implementación del módulo Despacho de Tripulantes del Sistema de Gestión Integral de Aduanas a partir de la solución modelada para agilizar los procesos.

Objetivos específicos:

- Realizar el diseño del módulo Despacho de Tripulantes.
- Realizar la implementación del módulo Despacho de Tripulantes.
- Validar la solución.

Tareas a realizar:

- Diseñar la Base de Datos.
- Obtener el modelo de diseño.
- Obtener modela de clase de acceso a datos.
- Implementar interfaces visuales.
- Implementar los requisitos funcionales.

Métodos utilizados:

Métodos teóricos.

Permiten estudiar las características del objeto de investigación que no son observables directamente, facilitan la construcción de modelos e hipótesis de investigación y crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad, contribuyendo al desarrollo de las teorías científicas y para su ejecución se apoyan en el proceso de análisis y síntesis.

Posibilitan el conocimiento del estado del arte del fenómeno, su evolución en una etapa determinada, su relación con otros fenómenos, así como su aislamiento como objeto estudiado.

Analítico- Sintético.

Constituyen dos procesos cognitivos que cumplen funciones muy importantes en la investigación científica. El análisis es una operación intelectual que posibilita descomponer mentalmente un todo

complejo en sus partes y cualidades. 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 y posibilita descubrir relaciones y características generales entre los elementos de la realidad.

El análisis y la síntesis no existen independientemente uno del otro. En realidad el análisis se produce mediante la síntesis: el análisis de los elementos de la situación problemática se realiza relacionando estos elementos entre sí y vinculándolos con la situación problema como un todo. A su vez, la síntesis se produce sobre la base de los resultados dados previamente por el análisis.
La unidad dialéctica existente entre las operaciones de análisis y síntesis supone que en el proceso de la investigación científica una u otra pueden predominar en una determinada etapa, atendiendo a la tarea cognoscitiva que esté realizando el investigador. Este método se utiliza para analizar la bibliografía para realizar el estudio del estado del arte.

Método Modelación.

El modelo científico es un instrumento de la investigación de carácter material o teórico, creado por los científicos para reproducir el fenómeno que se está estudiando. El modelo 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. La modelación es justamente el proceso mediante el cual se crea modelos con vistas a investigar la realidad.

En este trabajo este método es muy importante ya que ayuda a la construcción de software y a la creación del modelo relacionado con este sistema que representa la realidad aduanera donde se desarrolla. Basados en él se llevan a cabo los artefactos del sistema: diagramas de clases del diseño y diagramas de secuencia, donde se reproduce la realidad que existe en la actualidad en la Aduana General de la República de Cuba.

Estructura del Documento

Capítulo 1: En este capítulo se realiza un estudio de los sistemas aduaneros a nivel mundial y nacional, además de analizar y definir las metodologías y herramientas que se utilizan para la realización del trabajo.

Capítulo 2: Descripción de la solución propuesta, se le da forma para que soporte todos los requisitos y restricciones, consolidando una arquitectura que sirva de base para el desarrollo y la implementación de los requisitos modelados para darle solución a los requerimientos, cumpliendo con las necesidades del cliente.

Capítulo 3: Validación de la solución dándole cumplimiento exitoso a los requerimientos.

Capítulo 1. Fundamentación teórica.

1.1 Introducción.

A través de este capítulo se podrá conocer el estado del arte, donde se han analizado varios sistemas existentes en Cuba y el resto del mundo que son similares al que se desea implantar en la Aduana General de la República de Cuba. Además se describen las metodologías, patrones y herramientas a utilizar para el desarrollo del mismo.

1.2 Ámbito Internacional.

Con el desarrollo de las tecnologías de la Informática y las Comunicaciones (TIC) a nivel mundial, se pretende que las aduanas de todos los países requieran de estas tecnologías, incluyendo el comercio electrónico para lograr una racionalización, simplificación, reducción o eliminación de todas las medidas o barreras que obstaculicen el comercio internacional. Con este fin, la Organización Mundial de Aduanas (OMA) ha preparado directrices detalladas para lograr la informatización de las Aduanas, para ello requiere el uso de sistemas aduaneros informatizados, que provean la transmisión electrónica previa de la información a las aduanas y el intercambio electrónico de información a efectos de exportación y de importación.

1.2.1 Sistema Informático María (SIM).

MARIA surge por un acuerdo entre la aduana argentina y la aduana francesa en el año 1989. El objetivo del SIM es crear un mecanismo de consulta a destinaciones de importaciones y exportaciones 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. Cabe destacar que este sistema cubre aproximadamente el 90 % del total de las importaciones y exportaciones, ya que el resto aún se encuentra comprendido en el sistema DUA (Documento Único Aduanero) (1).

Este sistema combina bases de datos, registros, visualizaciones y declaraciones electrónicas, que utiliza la aduana para realizar el control y la estadística de las declaraciones de importaciones y exportaciones.

Los registros efectuados en el sistema por los usuarios tienen carácter de declaraciones juradas o declaraciones sumarias y los datos incluidos deben ser respaldados por la documentación correspondiente con el fin de evitar grandes sanciones y perjuicios.

Entre las funciones del SIM se encuentra la de analizar y verificar la información ingresada, con la intensión de controlar el valor de las mercaderías declaradas con fines fiscales. A su vez aporta a los agentes de aduana información de las operaciones declaradas para la posterior verificación física de las mercaderías y su documentación, con el propósito de verificar la veracidad y coherencia de lo declarado.

Actualmente el SIM ofrece las siguientes ventajas a los exportadores e importadores:

- Acceso a la información, reglamentación y requisitos para formalizar las operaciones ante la Aduana, previo a la registración de las mismas.
- Acceso a la información referente al estado de tramitación y observaciones de las declaraciones sumarias en el sistema.
- Acceso a la información del estado de pago de los incentivos a las exportaciones.
- Autoliquidación de los derechos a la exportación y de los derechos, tasa e impuestos a la importación.
- Acceso al estado de habilitación de los operadores.
- En caso de tener habilitada la planta, poder realizar operaciones de consolidado y precintado de exportación, ingresando la información desde la fábrica del exportador.
- En caso de tener habilitada la planta, poder realizar toda la tramitación de importaciones en el establecimiento del importador bajo la figura de aduana domiciliaria.
- Acceder a los estados de cuentas ante la Dirección General de Aduanas (DGA).
- Indirectamente ofrece la información estadística del comercio exterior argentino.

Su arquitectura informática de la década del 80 lo limita ante los actuales progresos de transmisión de información, incorporación de documentos electrónicos, generación de documentos electrónicos, operatividad, análisis de la información recibida bajo múltiples parámetros, interconexión con las nuevas tecnologías de escaneos, obtención de imágenes y lectura de codificación de barras, auditoría, control del

sistema y niveles de seguridad de la integridad de la información, aplicados por sistemas aduaneros de tecnología actual(2).

1.2.2 Sistema Aduanero Automatizado SIDUNEA.

Los objetivos del Sistema Automatizado de Datos Aduaneros (SIDUNEA) son:

- Modernizar las aduanas mediante la informatización de la mayoría de los trámites que hay que realizar en las fronteras con el fin de acelerar el proceso de despacho de aduanas.
- Reforzar la gestión y el control de las aduanas proporcionando a los gobiernos datos estadísticos precisos y actuales sobre las operaciones aduaneras y el comercio exterior útiles para las políticas fiscales y comerciales.

El SIDUNEA abarca todo el procedimiento de declaración y tramitación, incluyendo las mercancías y el tránsito. Sus sofisticadas herramientas engloban todos los procedimientos de inspección convencionales y pueden asignar las mercancías declaradas a un "canal de control" (verde para el despacho de mercancías sin examen; amarillo para el control de documentos antes del despacho de mercancías; rojo para la inspección material de las mercancías antes de realizar el despacho de aduanas; o azul para indicar que las mercancías serán entregadas después de ser sometidas a una auditoría de las autoridades aduaneras posterior al despacho de aduanas). El sistema utiliza medios audiovisuales, imágenes analizadas y dispositivos inalámbricos, que permiten a los funcionarios de aduanas acceder a distancia de manera inmediata a las bases de datos de información y control(3).

El Sistema Aduanero Automatizado (SIDUNEA) es una herramienta informática para el control y administración de la gestión aduanera, desarrollada por la conferencia de las Naciones Unidas sobre el Comercio y Desarrollo (UNCTAD), la misma permite realizar un seguimiento automatizado de las operaciones aduaneras y controlar efectivamente la recaudación de los impuestos verificando automáticamente los registros y contabilizando todo lo relativo a cada declaración, con la mínima intervención del factor humano.

Se puede configurar de acuerdo a las características de cada régimen aduanero, al arancel nacional y a la legislación de cada país, además de implementar los estándares internacionales para procesar los datos

de comercio exterior ya acordados por la OMA y por la Organización Internacional para la Estandarización (ISO).

Gracias a la tecnología 100% JAVA, SIDUNEA permite el uso de tarjetas Inteligentes con procesadores y tecnología JAVA para controlar los accesos al sistema y los pagos electrónicos. Así mismo, su aplicabilidad vía Internet, permite acceder al sistema a través de dispositivos inalámbricos, donde quiera que se encuentre y cuando más se necesite. También es resistente a las caídas de las telecomunicaciones, es compatible con la mayoría de los RDBMS (JDBC) e independiente de las plataformas (equipos) y de los sistemas de base de datos. Además cuenta con:

- Modernos conceptos de seguridad (PKI) o Conceptos DOM (Document Object Model), XML.
- Directorios de mensajes XML estándar que hacen posible la cooperación internacional entre sistemas y la creación de la red Customs Global o Protocolo REWI extendido, TCP/IP.
- Interfaz de usuario amigable (WYSIWYG).
- Extensión e implementación dinámica.
- Adaptabilidad (según el número de operaciones).
- Aspectos de seguridad ya integrados.
- Funciones especiales como Multiidioma, Gestión, Propiedad de documentos, Auditoría e Historización.

El utilizar la arquitectura cliente-servidor, lo convierte en un programa que se instala en la computadora y a la vez es una red de información nacional y mundial, lo que permitirá trabajar con o sin conexión a la red. Genera datos estadísticos sobre comercio exterior y permite el intercambio electrónico de datos entre comerciantes y la aduana. Cuenta con directorios de mensajes XML, estándar que hace posible la cooperación internacional entre sistemas; con una interfaz de usuario amigable y con funciones especializadas como: multiidioma, gestión, propiedad de documentos y auditoría.

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

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

- Cobrar correctamente los impuestos y tasas.
- Detectar los errores en los valores de la declaración.
- Monitorear el pago de los impuestos.
- Evitar la evasión de impuestos.
- Minimizar el contrabando.
- Crear incentivos para el declarante.
- Administrar efectivamente el proceso de despacho.
- Poner en práctica un esquema de garantía con la modalidad de pago anticipado, para facilitar el comercio y asegurar el cobro de los derechos aduaneros.
- Controlar la ruta de comercio por medio de las oficinas de despacho de mercancía de cada aduana(4).

1.3 Ámbito Nacional.

El CADI comenzó a implementar e implantar en el Sistema de Órganos Aduaneros (SOA), varios sistemas que aunque desplegados en diferentes plataformas, resolvían y gestionaban procesos o partes de ellos y comenzaron a manifestarse resultados satisfactorios. Ejemplos de estos sistemas lo constituyen, por mencionar algunos: SADEM, SACOM, SAPIA y SADONCE. Todos estos sistemas funcionan de forma independiente, convirtiéndose ésta en la principal limitante del proceso ya que al no poder retroalimentarse, muchas veces la información se hacía redundante.

1.3.1 SADONCE.

El Sistema Automatizado de Despacho y Operaciones No Comerciales (SADONCE) es utilizado para el desarrollo del Despacho sin Carácter Comercial en la AGR. El mismo permite entre sus principales opciones, el registro de las declaraciones de importación y exportación, su modificación y cancelación, además del registro de las operaciones no comerciales sin declaración de documentos. Controla las importaciones y exportaciones sobre los artículos que son prohibidos y calcula de manera automática los montos a pagar ante la aduana, gestionando el pago de servicios y derechos. El sistema es capaz de generar también una serie de reportes estadísticos de gran importancia para varios departamentos de la aduana.

Unidos al SADONCE se desarrollaron además para el control de las importaciones de tripulantes y colaboradores dos sistemas independientes que brindan varias funcionalidades, en el caso de los tripulantes (pilotos, marineros, aeromozas, capitanes, entre otros): el registro ya sea con o sin pre-despacho de la declaración de importación, su cancelación y su modificación, la devolución de artículos decomisados o retenidos, y el registro de multas aplicadas, además de aportar una serie de reportes estadísticos a partir de la gestión, para cada tripulante registrado, de las importaciones que ha hecho, los valores de importación acumulados, las multas que le han sido registradas, así como los artículos que ha importado. En el caso de los colaboradores: el sistema permite el registro de las declaraciones de importación, así como su cancelación y modificación, su principal función es llevar un control sobre los colaboradores y los artículos que estos ingresan al país, además de brindar también reportes estadísticos(5).

1.3.2 Sistema Único de Aduanas

El Sistema Único de Aduanas (SUA) automatiza el procesamiento informativo referente a todas las operaciones que conforman los diferentes procesos, ya sea de Medios de Transporte Internacional, Importaciones y Exportaciones con y sin carácter comercial, Bultos Postales, Viajeros y las Tablas de Control en un ambiente WEB, brindando todas las facilidades de estos servicios a los usuarios(6).

En el mismo, todos los módulos validan y controlan las entradas de datos contra los nomencladores y clasificadores que se diseñaron, con vista a tener organizada la información y así asegurar la consistencia de los datos.

En el proceso de despacho con carácter no comercial el SUA implementa el control de las importaciones, captando y registrando las declaraciones, logrando la total validación de los datos con el fin de llevar una estadística de los artículos que entran en el país y de los montos a pagar en dependencia de la operación que se esté realizando. La principal problemática que se presenta es que a pesar de haberse agrupado en un solo sistema las funcionalidades de los que le dieron precedencia todavía se sigue contemplando el procedimiento separado por tipos de personas y operaciones por carga, habiendo quedado dividido en 4 submódulos fundamentales: Tripulantes, Colaboradores, Cargas y Pasajeros. Cada uno de estos

submódulos utiliza una base de datos diferente y en ocasiones las funciones que se realizan en cada uno son repetitivas, o sea en los cuatro se realiza la misma función lo que con informaciones diferentes.

2.1 Herramientas escogidas para desarrollar este trabajo.

2.1.1 Herramienta CASE Visual Paradigm.

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML(7).

Algunas características de Visual Paradigm:

- Es profesional.
- Es amigable.
- Contiene facilidades para redactar especificaciones de casos de uso del sistema.
- Sincronización entre diagramas de entidad-relación y diagramas de clases.
- Generación de código / Ingeniería inversa.
- Soporte de UML versión 2.1.
- Interoperabilidad con otras aplicaciones.

2.2 Lenguaje de programación a utilizar.

En los últimos tiempos han surgido variedades de lenguaje de programación cada vez con más facilidades para el desarrollador y el educador, como el java, php 5, .NET, que son tecnologías más amigables en el momento de desarrollar aplicaciones de todas las envergaduras, pero para que un lenguaje atraiga a desarrolladores, vendedores y educadores debe resolver un problema particular tan bien que sea capaz de superar la resistencia de adoptarlo.

La programación orientada a objetos (POO) ha tenido un desarrollo de cerca de 30 años; es una técnica que se enfoca en los datos (objetos) y en la manera de llegar a ellos (interfaces), no en las herramientas

que se utilizan para manejarlos, a fin de crear “módulos” que interactúen entre sí para lograr el objetivo del programa.

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina.

2.2.1 PHP.

PHP (acrónimo de Hipertexto Preprocessor) es un lenguaje "del lado del servidor" (esto significa que PHP funciona en un servidor remoto que procesa la página Web antes de que sea abierta por el navegador del usuario) especialmente creado para el desarrollo de páginas Web dinámicas.
Puede ser incluido con facilidad dentro del código HTML, y permite una serie de funcionalidades tan extraordinarias que se ha convertido en el favorito de millones de programadores en todo el mundo(8).
Características de PHP:

- Gratuito. Al tratarse de software libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre.
- Versatilidad. PHP puede usarse con la mayoría de sistemas operativos, ya sea basados en UNIX (Linux, Solares, FreeBSD...), como con Windows, el sistema operativo de Microsoft.
- Enorme eficiencia. Con escaso mantenimiento y un servidor gratuito (en este caso, Apache), puede soportar sin problema millones de visitas diarias.
- Sencilla integración con múltiples bases de datos. Esencial para una página Web verdaderamente dinámica, es una correcta integración con base de datos. Aunque MySQL es la base de datos que mejor trabaja con PHP, puede conectarse también a PostgreSQL, Oracle, dbm, filePro, interbasem o cualquier otra base de datos compatible con ODBC.

Partiendo de que el sistema a implementar tiene que integrarse al GINA que es una aplicación PHP, se decidió que para el problema a resolver y con la idea de desarrollar una aplicación lo más libre de licencias posible y valorando que PHP cubre todas las necesidades se puede aceptar este lenguaje como el ideal para el desarrollo, lenguaje definido en la línea base de la arquitectura del GINA del que este trabajo está integrado y por el cual se rige.

2.2.2 Framework a utilizar.

Un Framework es un conjunto de clases o estructuras que implementan los componentes de una aplicación genérica, así como también componentes concretos que cumplen a cabalidad tareas concretas. Para desarrollar programas completos, los desarrolladores buscan e instancian los componentes apropiados.

Realmente no existe una definición oficial de Framework, pero todos los autores coinciden en la utilización de un tema común: la reutilización. Una definición dada por R. E. Johnson, and B. Foote en 1988 en su publicación “Designing Reusable Classes”.

"Un Framework es un conjunto de clases que personifican un diseño abstracto para soluciones de una familia de problemas relacionados"

2.2.2.1 Symfony Project.

Symfony es un Framework para PHP5 es compatible con la mayoría de gestores de bases de datos, MySQL, PostgreSQL, Oracle y SQLServer de Microsoft, entre otros en dependencia del tipo de abstracción de la Base de Datos que se utilice.

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

Características de Symfony

- Proyecto de Symfony es un framework muy amplio, e incluye un verdadero ORM, de nombre Propel, que es otro proyecto de código abierto y, probablemente, una de las mejores soluciones ORM para PHP.

- Incluye Creole para la capa de abstracción de base de datos y Mojavi para la capa Modelo-Vista-Controlador.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicación empresarial y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros(9).

2.2.3 Framework de JavaScript, ExtJS.

Ext.JS es un Framework de JavaScript que permite realizar aplicaciones Web enriquecidas basándose en tecnología AJAX, JSON, DHTML y DOM. Ext.2.0 está patentado bajo licencia LGPL lo que posibilita su uso para aplicaciones empresariales privativas de código cerrado.

Ext.JS brinda la posibilidad de utilizar un gran número de componentes visuales que mejoran considerablemente la calidad de las aplicaciones. Brinda la posibilidad de validaciones de formularios de todo tipo, basándose en expresiones regulares y tipos de datos. Trae implícitos componentes como vista en arboles, arrastrado y soltado, cambio de tamaño de imágenes, rejillas, paginado, agrupado de objetos, tabs, asistentes, entre otros muchos.

La utilización de un Framework de JavaScript como Ext.JS facilita la separación de las capas de la vista con la del controlador desde el punto de vista productivo ya que el código utilizado en la primera es

solamente JavaScript y no es necesario utilizar ningún tipo de código PHP, así los desarrolladores pueden centrarse más en el aprendizaje de un solo lenguaje. Además al soportar serialización de objetos mediante tecnología JSON permite que los datos enviados desde el controlador como respuesta a la vista contengan solo las propiedades de dichos objetos, pero no el comportamiento, minimizando los posibles errores de programación y los accidentes de que los objetos sean modificados erróneamente desde la vista.

2.2.4 Oracle.

El SGBD Oracle, fabricado por Oracle Corporation, 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 bases de datos 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.

Los tipos objeto de Oracle son tipos de datos definidos por el usuario que permiten modelar entidades complejas del mundo real en una estructura que trata cada entidad como una unidad atómica simple en la base de datos. A partir de la versión 10 del año 2004, se añade a los servidores la capacidad de funcionar según el paradigma de “Grid” (o rejilla) y se ofrecen mejoras en la administración e integración de algunos elementos que previamente no funcionaban correctamente juntos(10).

2.2.5 Elipse.

Es un entorno de desarrollo integrado, de código abierto y multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. No es más que un entorno de desarrollo integrado (IDE) en el que se encuentran todas las herramientas y funciones necesarias para el trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar.

Ventajas:

- El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la Plataforma de Cliente rico, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no.
- La arquitectura plug-in permite escribir cualquier extensión deseada en el ambiente, como sería gestión de la configuración. Se provee soporte para Java y CVS en el SDK de Eclipse. Y no tiene por qué ser usado únicamente para soportar otros lenguajes de programación.
- La definición que da el proyecto Eclipse acerca de su Software es: "una especie de herramienta universal, un IDE abierto y extensible para todo y nada en particular"(10).

3.1Patrones de Diseño.

3.1.1 Patrones GRASP.

Patrones de Software para la Asignación General de Responsabilidad. En este caso las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento:

- **Conocer:** atributos, relaciones con otros objetos.
- **Hacer:** crear un objeto, hacer un cálculo, iniciar una acción en otros objetos, controlar y coordinar actividades en otros objetos.

GRASP posee nueve patrones: Experto, Creador, Alta cohesión, Bajo acoplamiento, Controlador, Polimorfismo, Fabricación Pura, Induración y No Hables con Extraños. De ellos se utilizaron:

Experto: es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para ejecutar la tarea (clase experta). Refuerza el encapsulamiento y esto redunda en bajo acoplamiento.

Creador: ayuda a identificar quién debe ser el responsable de la instanciación o creación de nuevas clases u objetos. La clase podrá crear la nueva instancia si y sólo si tiene en cuenta al menos uno de los siguientes criterios:

- Tiene la información necesaria.

- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.
- Contiene o agrega la clase.

La visibilidad entre la clase creada y la clase creadora es una de las facilidades que se deriva del uso de patrón y además conduce a un bajo acoplamiento, lo cual supone facilidad de mantenimiento, reutilización y claridad.

Alta cohesión: plantea que una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas, indica que la información que almacena una clase debe ser coherente, de manera que todos sus métodos tengan un comportamiento bien definido.

- Las clases se pueden reutilizar con mayor facilidad y flexibilidad.
- Una clase con baja cohesión hace muchas cosas no relacionadas.
- Una clase con alta cohesión hace lo que uno podría esperar que hiciera o hace demasiado trabajo.

Con una alta cohesión dentro de las clases definidas, si el sistema fallara por alguna razón es mucho más fácil encontrar responsabilidades. Por otra parte, se puede lograr que la complejidad sea lo más manejable posible e incrementar la claridad y comprensión del sistema para simplificar su mantenimiento.

Bajo acoplamiento: meta principal que es preciso tener presente durante el diseño ya que constituye un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño, de modo que se mantenga un engranaje pobre entre las clases y objetos (baja dependencia) para reducir el impacto de los cambios y aumentar las posibilidades de reutilización.

Controlador: asigna la responsabilidad a una clase de manejar mensajes correspondientes a eventos en un sistema. Encargada de recibir los datos del usuario y enviarlos a las distintas clases. La aplicación del patrón conlleva a separar la lógica de negocios de la capa de presentación. Esto facilita la centralización de actividades (validaciones, seguridad, entre otras cosas.) Si se aplican estos principios, el controlador no realiza las actividades mencionadas sino que las delega en otras clases con las que mantiene un modelo de alta cohesión.

3.1.2 Patrones GOF1.

Los patrones GOF describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Según su propósito, están clasificados en:

- **Creacionales:** se encargan de la creación de los objetos ayudando a que el sistema sea independiente de la creación, composición y representación de los objetos.
- **Estructurales:** encargados de definir cómo las clases y objetos están compuestos para formar estructuras más complejas. Usan la herencia para componer interfaces u objetos en tiempo de ejecución.
- **Comportamiento:** plantean algoritmos y la asignación de responsabilidades entre objetos. Estos patrones no solo describen clases y objetos sino también describen la comunicación y cooperación entre ellos.

Los patrones GoF utilizados son:

Fachada: patrón estructural que permite proveer una interfaz unificada y sencilla como intermediaria entre un cliente y una interfaz o grupo de interfaces.

Este patrón simplifica el acceso a un conjunto de objetos relacionados mediante una interfaz de alto nivel que esconde la complejidad del sistema. Se utiliza para proporcionar un acceso sencillo a subsistemas muy complejos.

Mediador: patrón de comportamiento que coordina las relaciones entre sus asociados. Permite la interacción de varios objetos, sin generar acoples fuertes en esas relaciones.

La comunicación en la base de datos se puede tornar compleja debido a las dependencias marcadas entre las tablas que la componen, esto resulta engorroso a la hora de acceder a un determinado valor. La solución a este inconveniente viene dada por la utilización de este patrón; específicamente; creando una nueva tabla entre todas las tablas unidas mediante una relación de muchos a muchos. La tabla mediadora posee una relación de uno a muchos con las vinculadas a ella. De esta forma, el comportamiento distribuido entre las clases queda adaptado a las circunstancias y necesidades del diseño(11).

[1] GoF: (Gang of Four) llamados así por los cuatro autores del libro: “Patrones de Diseño” que recoge alrededor de 23 patrones de los más utilizados.

4.1 Metodología que se utiliza.

4.1.1 Proceso Unificado de Desarrollo (RUP).

La metodología de desarrollo a utilizar es RUP, en esta se han unificado técnicas de desarrollo, a través del Lenguaje Unificado de Modelado (UML). En su modelación define como sus principales elementos el quién hace qué, cuándo y cómo lo hace. RUP está preparado para desarrollar grandes y complejos proyectos y es la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

El ciclo de vida de RUP se caracteriza por:

- Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean. Estos guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.
- Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo.
- Iterativo e Incremental: Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Para el caso de una iteración de elaboración, centra su atención en el análisis y diseño, aunque refina los requerimientos y obtiene un producto con un determinado nivel, pero que irá creciendo incrementalmente en cada iteración.

RUP representa un ciclo de desarrollo en la vida de un producto de software con 4 fases:

- La fase de Concepción: Define la visión, los objetivos y el alcance del proyecto, tanto desde el punto de vista funcional como del técnico.
- La fase de Elaboración: Completa el análisis de los Casos de Uso y define la arquitectura del sistema.
- La fase de Construcción: Se obtiene la capacidad operacional inicial del producto.
- La fase de Transición: Inicia con una versión “beta” del sistema y culmina con el sistema en fase de producción.

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. Los cuales son:

Modelamiento del negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, Despliegue, Gestión de configuración y cambios, Gestión de proyectos y por último Entorno, donde los 3 últimos constituyen los flujos de soporte(12).

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

Actualmente en el Departamento de Soluciones para la Aduana se usa la metodología de desarrollo de Software RUP con modificaciones, ya que es un proceso de desarrollo de software configurable que puede ser adaptado a las características propias de cada proyecto. En el proyecto en vez de la utilización del modelo de casos de uso del negocio y los diagramas de actividades, se propone el uso del Modelo de Proceso de Negocio con la notación BPMN (Notación para el Modelado de Procesos de Negocio). Así como para los conceptos del negocio, en vez del Modelo de Objeto del Negocio, se propone confeccionar el Modelo Conceptual. Se corresponde cada requisito funcional a un caso de uso. Se pasará directamente al diseño ya que los analistas no tienen los conocimientos necesarios de los framework que se utilizan en el proyecto, por lo que los artefactos que se generen no ayudarían a los diseñadores, constituyendo una pérdida de tiempo.

Conclusiones Parciales.

A partir del estudio realizado de los sistemas existentes en Cuba y el mundo relacionados con la gestión de los despachos aduaneros, se llegó a la conclusión de la necesidad de crear en Cuba un nuevo sistema para gestionar el Despacho de Tripulantes integrado al GINA. El sistema MARIA es un sistema con el cual se pueden realizar operaciones de importación y exportación. Pero dicha versión necesita ser mejorada ya que este sistema no abarca el control de tripulantes que realiza la AGR, debido a que en otros países no se controla el tráfico de los mismos en la frontera. Por otra parte SIDUNEA, aunque fue el primer sistema informático que utilizó la Aduana General de la República de Cuba para la gestión de procesos aduanales, al ser proveniente de países capitalistas no contempla diferencias entre las operaciones de carácter no comercial y las comerciales, procesos que en el resto del mundo se tratan unidos y que en Cuba tienen un tratamiento diferenciado dado por las características del sistema económico cubano; además que los costos de soporte de esta aplicación son elevados y no rentables para la economía.

Se crearon posteriormente en Cuba sistemas automatizados para la gestión aduanera, divididos e independientes unos de otros, división que ocasionaba problemas para obtener datos estadísticos, decidiéndose desarrollar un sistema automatizado único, que contemplara todos los procesos de la Aduana para implantarse en todas las aduanas del país, este sistema debía responder a los intereses del comercio cubano y satisfacer los requerimientos de cada funcionario, surge así, de esta forma por concepción general e implementado sobre plataforma Web, el Sistema Único de Aduanas (SUA).

A pesar de que este nuevo sistema ha resuelto muchas de las dificultades que se tenían antes y ha aumentado la eficiencia del despacho reduciéndose el tiempo y el costo del mismo, quedaron algunos requerimientos que no se contemplaron por lo que todavía es necesario en ocasiones acudir a los viejos sistemas para obtener datos, generalmente estadísticos.

Luego del estudio de las características principales de las tecnologías antes mencionadas se decide realizar el diseño y la implementacion del módulo Despacho de Tripulantes del subsitema Despacho no Comercial del Sistema de Gestión Integral de Aduanas. Para la realización del sistema se utilizará la herramienta CASE del Visual Paradimg siguiendo la metodología RUP y los estándares establecidos por el proyecto, para el modelado de las clases se utulizará como lenguaje servidor PHP haciendo uso del framework del mismo llamado Symfony y como lenguaje cliente el framework de JavaScrip, ExtJS.
Como herramienta de desarrollo IDE se utilizará el eclipse, el servidor de aplicaciones es Apache y como servidor de control de versiones, el SubVersion.

Capítulo 2. Diseño e Implementación.

2.1 Introducción.

En el presente capítulo se analiza una descripción de los artefactos del diseño que se generan durante el flujo de trabajo. Se realiza un análisis del funcionamiento de los algoritmos más importantes para la implementación del módulo, así como las estructuras de datos utilizadas para manejar los datos en el programa. Se presenta la descripción de las clases fundamentales que definen el comportamiento del sistema con sus atributos y métodos esenciales.

Se propone el diseño y la implementación de un módulo que logre agilizar los procesos referentes a las actividades del despacho de tripulantes, resuelva las dificultades existentes, sea compatible con la tecnología y arquitectura del GINA y brinde una interface agradable y fácil de manipular.

2.2 Modelo del Diseño.

La fase de diseño expande y detalla los modelos de análisis tomando en cuenta todas las implicaciones y restricciones técnicas. 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.

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

La fase del Diseño del Sistema encierra cuatro etapas:

- ***El diseño de los datos.***
- ***El Diseño Arquitectónico.***
- ***El Diseño de la Interfaz.***
- ***El Diseño de Procedimientos.***

El Diseño es la única manera de materializar con precisión los requerimientos del cliente. El Diseño del Software es un proceso y un modelado a la vez. El proceso de Diseño es un conjunto de pasos repetitivos que permiten al diseñador describir todos los aspectos del Sistema a construir(13).

2.3 Requerimientos.

Un requerimiento es una condición o capacidad que necesita el usuario para resolver algún problema o alcanzar un objetivo, es una condición o capacidad que debe cumplir o poseer un sistema o componente del sistema para satisfacer un contrato, estándar, especificación u otro documento formalmente impuesto. La mayoría de los sistemas de Software cuentan con una cantidad inmensa de requisitos, es decir son demasiado grandes por lo que el desglose del mismo lo convierte en un poco más sencillo en el momento de su desarrollo.

El módulo Despacho de Tripulantes perteneciente al Despacho de Tripulante cuenta con siete requisitos funcionales:

- Buscar Tripulantes.
- Buscar Declaración.
- Registrar Pre- Despacho.
- Registrar Despacho.
- Modificar DIT.
- Cancelar DIT.
- Registrar Devolución.

2.2.2 Diagrama de Paquetes.

Un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema.

Los Paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes. Con estas líneas maestras sobre la

mesa, los paquetes son buenos elementos de gestión. Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido.

En la figura 2.1 se muestra el diagrama de paquetes donde se expone cómo interactúa el módulo Despacho de Tripulantes con los diferentes subsistemas del proyecto GINA, así como las librerías y paquetes del mismo.

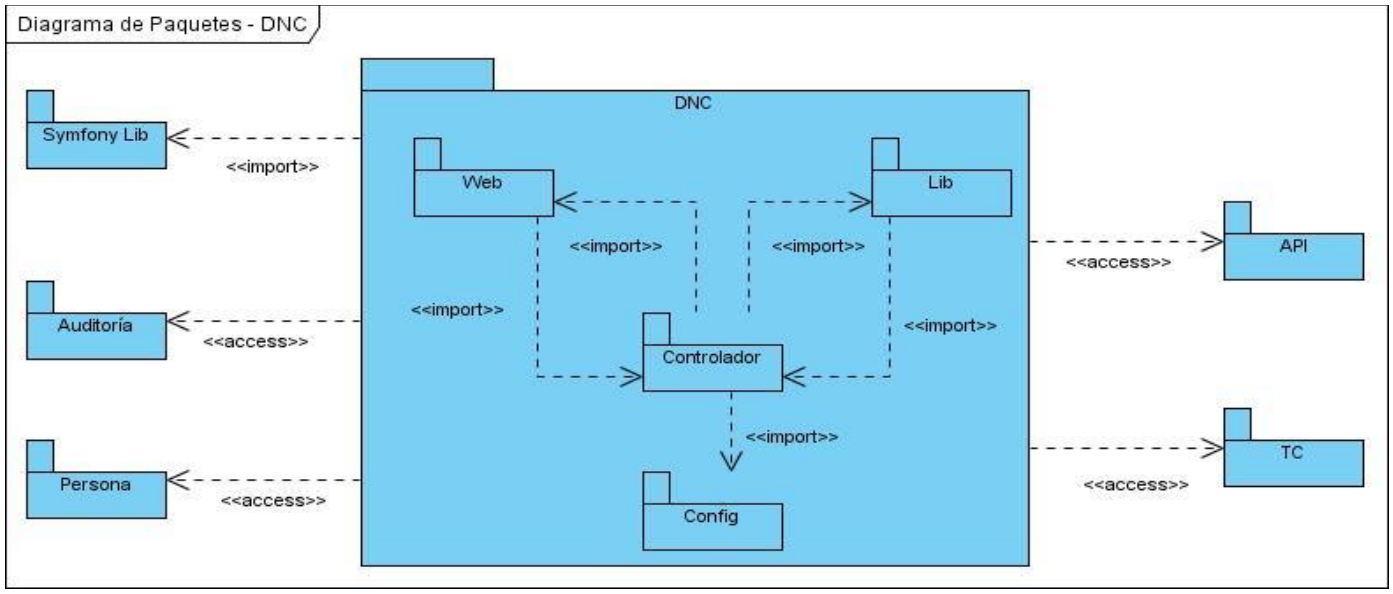

Figura 2.1 Diagrama de Paquetes

Dentro de los subsistemas que se relacionan con el módulo Despacho de Tripulantes se encuentra las Tablas de Control (TC) que es el encargado de administrar las clases que son nomencladoras y comunes para todos los subsistemas del GINA. El subsistema Persona brinda todo tipo de información referente al personal de la AGR. Se utilizan los subsistemas Auditoría para guardar todas las acciones realizadas sobre el módulo que pueden ser auditadas para la aduana, así como la interacción con el núcleo de Symfony y la librería. Dentro del mismo módulo se materializa el vínculo con diferentes paquetes dentro de los que sobresale el WEB encargado de la capa de presentación, el 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 las clases del negocio, los formularios y algunas de las encargadas del mapeo de la base de datos y el paquete Config que es donde se encuentra el view.yml que es el fichero que establece la estructura de las vistas por defecto: el nombre del Layout, las hojas de estilo CCS y los archivos JavaScript.

2.2.3 Diagrama de Clases del Sistema.

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

Un **diagrama de clases** sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso.
Un diagrama de clases está compuesto por los siguientes elementos:

- Clase: atributos, métodos y visibilidad.
- Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

En la figura 2.2 se muestra el diagrama de clases del requisito Registrar Pre- Despacho DIT del módulo Despacho de Tripulantes donde se presenta la interfaz principal del módulo, la cual puede contener uno o muchos formularios, que son los encargados de enviar la información a la clase servidora, esta contiene los atributos comunes y es la responsable de darle solución a las peticiones a través de las funcionalidades y enviar una respuesta a la página cliente. La página servidora interactúa con el modelo para la obtención de información.

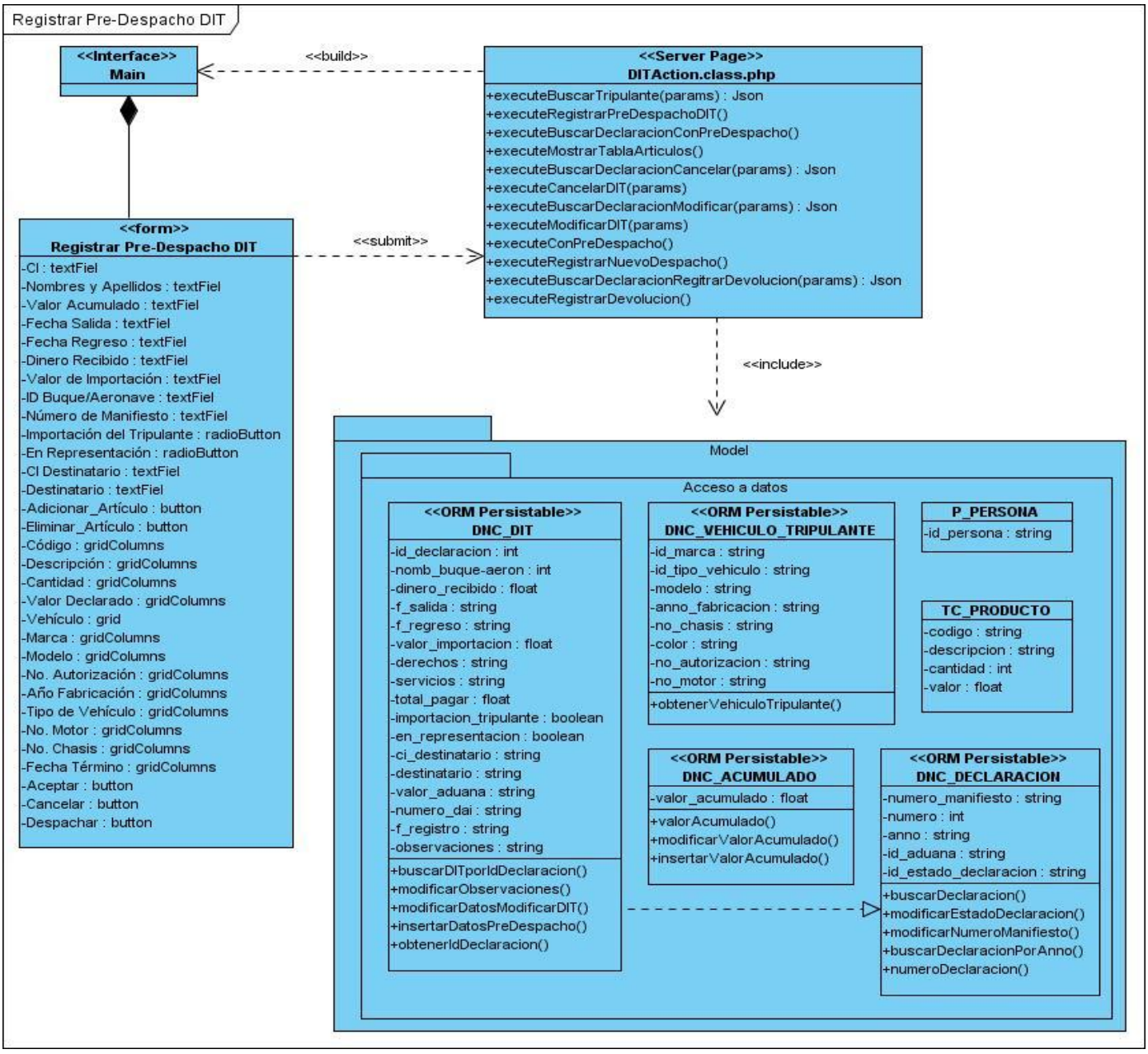

Figura 2.2 Diagrama de Clases

2.2.4 Diagrama de Secuencia.

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso. Mientras que el diagrama de casos de uso permite el modelado de una vista del negocio del escenario, el diagrama de secuencia contiene detalles de

implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos.

Típicamente se examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si se dispone de la descripción de cada caso de uso como una secuencia de varios pasos, entonces se puede "caminar sobre" esos pasos para descubrir qué objetos son necesarios para que se puedan seguir los pasos. Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales. El objetivo del diagrama de secuencia es lograr la expresión escrita de lo que se pretende con la realización del programa o proyecto que se planifica.

En el diagrama de secuencia de la figura 2.3 se representa el requisito Registrar Pre- Despacho DIT, donde se muestra una serie de pasos lógicos a seguir para implementar dicho requisito. El objetivo de este diagrama es realizar todas las validaciones del mismo, además de mantener una secuencia lógica en el momento de realizar el Pre- Despacho incluye también actividades que identifican llamadas a funcionalidades.

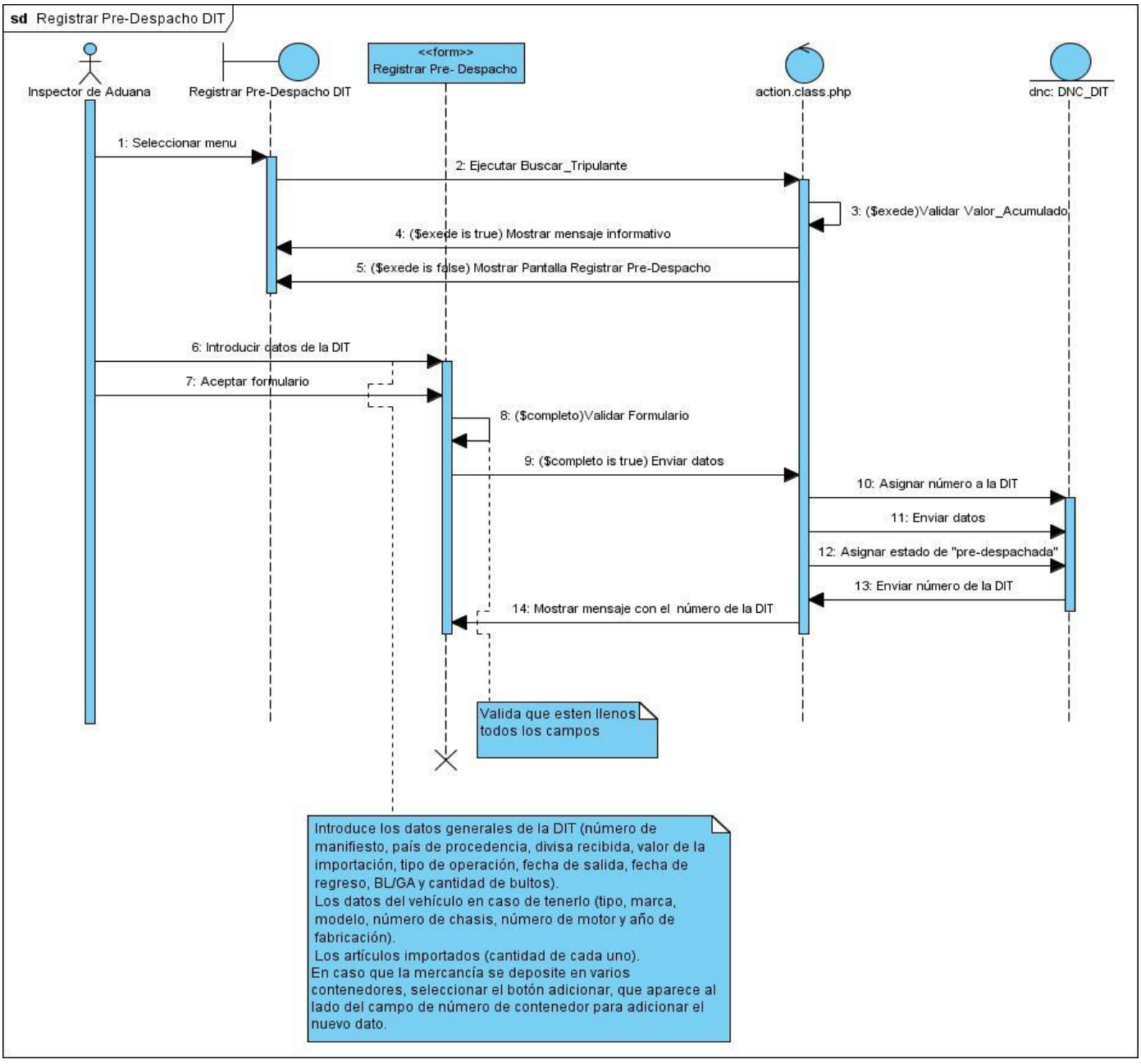

Figura 2.3 Diagrama de Secuencia.

2.2.5 Modelo de Base de Datos.

El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en sub-problemas y se resuelve cada uno de

estos sub-problemas independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico.

El diseño conceptual: parte de las especificaciones de requisitos de usuario y su resultado es el esquema conceptual de la base de datos. Un esquema conceptual es una descripción de alto nivel de la estructura de la base de datos, independientemente del Sistema Gestor de Base de Datos (SGBD) que se vaya a utilizar para manipularla.

El diseño lógico: parte del esquema conceptual y da como resultado un esquema lógico. Un *esquema lógico* es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD.

El diseño físico: parte del esquema lógico y da como resultado un esquema físico. Un *esquema físico* es una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos(14).

2.2.5 Modelo Entidad- Relación

En la figura 2.4 se muestra el diagrama entidad-relación que fue generado para el sistema el cual contiene las tablas que guardarán la información del Despacho de Tripulante.
Las tablas de color amarillo son las Tablas de Control (TC) las cuales son nomencladoras, contribuyen al completamiento de la información, permitiendo el acceso de todos los subsistemas que conforman el GINA.

Las tablas de color gris son las más importantes arquitectónicamente debido a que en ellas se almacenan los datos principales del Despacho de Tripulantes.

Todas en su conjunto conforman un modelo de entidad-relación, normalizado en Tercera Forma Normal (3FN), según la propuesta original de Codd (1972), que plantea que una relación está en tercera forma normal si todos los atributos de la relación dependen funcionalmente sólo de la clave, y no de ningún otro atributo.

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

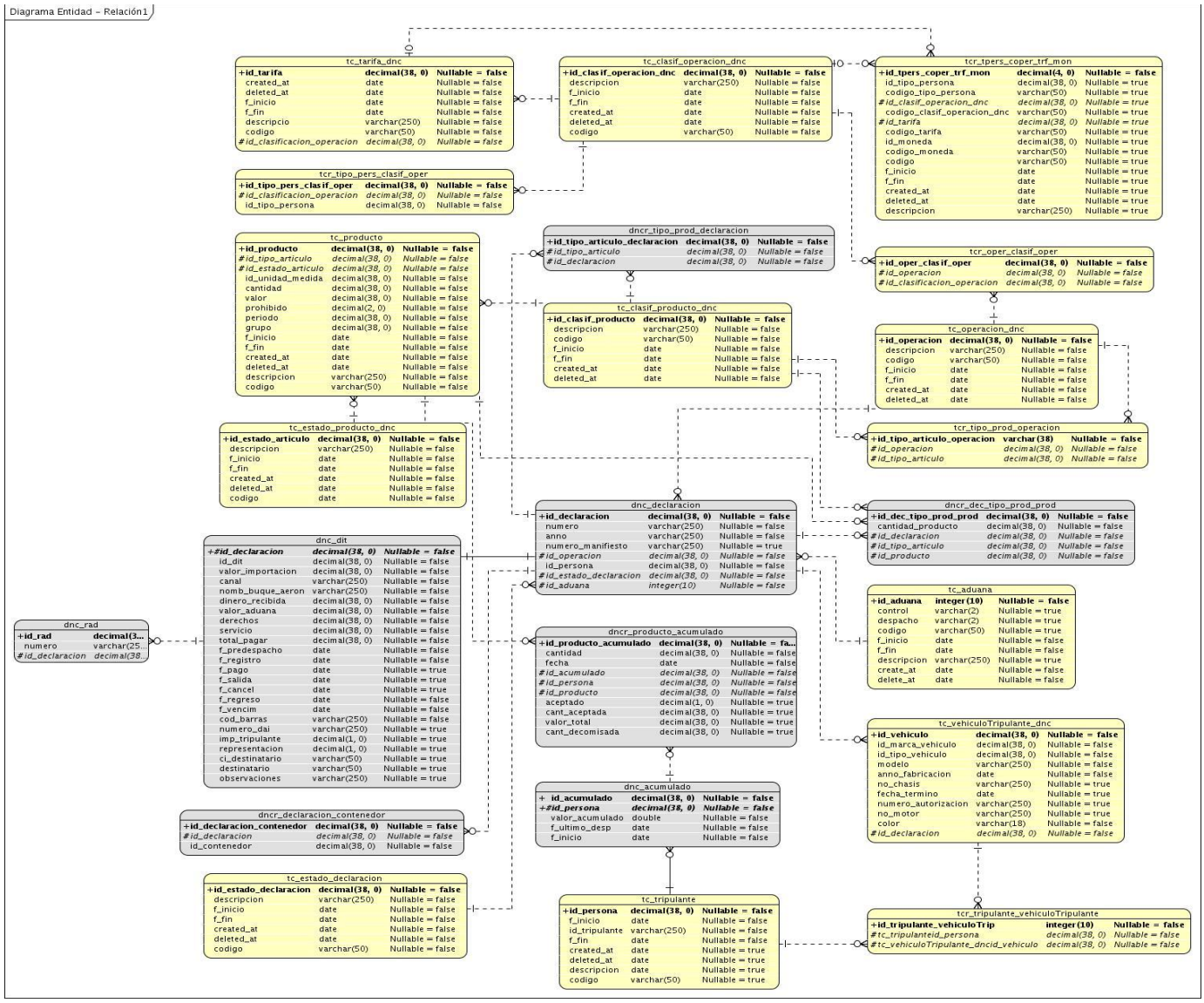

Figura 2.4 Modelo de Datos

2.3 Modelo de Implementación.

En el modelo de implementación se recogen, mediante diferentes artefactos, todas las acciones que se llevarán a cabo en este proceso. Se realiza en una sucesión de pasos pequeños y manejables de manera incremental distribuyendo por todo el sistema los diferentes componentes y organizándolos de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados.

El flujo de trabajo de Implementación tiene como objetivo definir la organización del código, la implementación de los elementos de diseño en términos de ficheros fuentes, binarios y ejecutables, para poder integrar los diferentes componentes de desarrolladores o equipos y generar un ejecutable entregable o producto final.

2.3.1 Tratamiento de Errores.

El tratamiento de errores es un aspecto importante para toda aplicación. Muchos son los errores cometidos por los usuarios, unos por accidentes y otros no tan accidentales. En el módulo Despacho de Tripulantes se controlan información importante que influye en la toma de decisiones valiosas para el Despacho No Comercial realizado por la AGR, es por eso que se realizan varios tipos de validaciones de errores.

En symfony la validación en el servidor es obligatoria para no corromper la base de datos con datos incorrectos. La validación en el lado del cliente es opcional, pero mejora enormemente la experiencia de usuario.

Las validaciones del negocio son las más importantes de todas, estas se realizan por medio de los formularios. Se encargan de controlar el flujo correcto de los datos introducidos, para evitar consecuencias alarmantes. Se lleva a cabo en las diferentes clases por medio del lenguaje PHP. En caso de encontrarse casos de este tipo, se le informa al usuario los datos que están erróneos para que pueda corregirlos.

Las validaciones realizadas en la vista juegan también un papel importante, estas se realizan por medio de expresiones regulares. Todas tienen como función principal evitar que introduzcan tipos de datos incorrectos en los diferentes campos, logrando así evitar ataques contra el sistema por esta vía.

2.3.2 Comunicación entre Capas.

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura 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.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. La Figura 2.5 ilustra el funcionamiento del patrón MVC. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación(15).

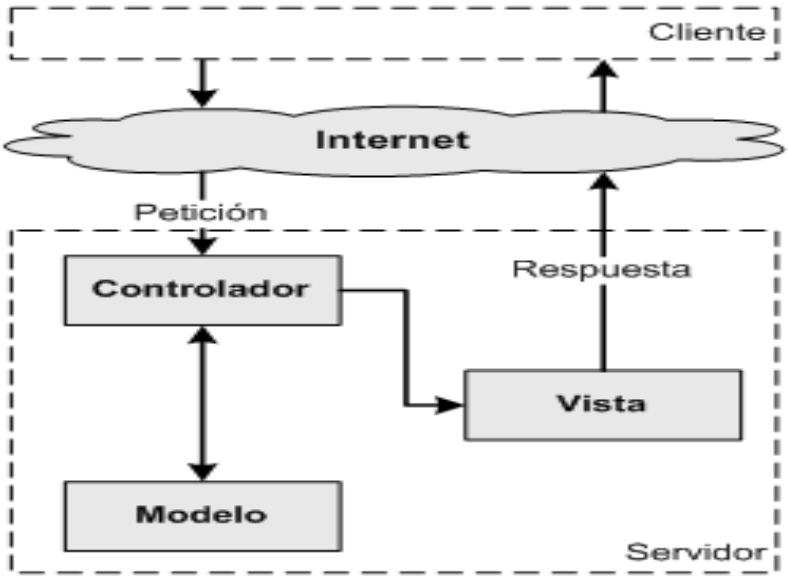

Figura 2.5 Patrón MVC.

2.3.3 Prototipo de Clases.

Es necesario partir de que Javascript es un lenguaje interpretado por los navegadores en el cliente, y no comprende el paradigma de la programación orientada a objetos. Esto nos lleva a que los conceptos de clase, componente y objeto, no son manejados explícitamente por este lenguaje, sino que son simulados a partir de la propiedad prototype. Esta propiedad permite gestionar los atributos y métodos de un objeto, brindando un comportamiento similar a una clase en la programación orienta a objetos.

El framework Ext.JS 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.

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 (Ver figura 2.6).

Cada lenguaje de programación orientado a objetos promueve viablemente una manera de invocar al constructor y demás métodos de la clase base, para esto Javascript implanta el uso de la propiedad superclass, la cual establece las relaciones entre los métodos de las clases hijas y los de las clases bases(16).

```
//Nombre del paquete donde se incluyen los componetes
  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 2.6 Modelo de clases para implementar los componentes visuales.

2.3.4 Petición AJAX.

Ajax es una técnica de desarrollo web para crear aplicaciones interactivas, las cuales se ejecutan en el cliente, mientras se mantiene la comunicación asíncrona con el servidor en segundo plano (ver figura 2.7). De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.

Ajax permite que las páginas web respondan de forma más rápida mediante el intercambio en segundo plano de pequeñas cantidades de datos con el servidor, por lo que no es necesario recargar la página entera cada vez que el usuario realiza un cambio. El objetivo es aumentar la interactividad, la rapidez y la usabilidad de la página.

Ajax depende de XMLHttpRequest, un objeto JavaScript cuyo comportamiento es similar a un frame oculto, cuyo contenido se puede actualizar realizando una petición al servidor y se puede utilizar para manipular el resto de la página web. Se trata de un objeto a muy bajo nivel, por lo que los navegadores lo tratan de forma diferente y el resultado es que se necesitan muchas líneas de código para realizar peticiones Ajax a mano. Afortunadamente, Prototype encapsula todo el código necesario para trabajar con Ajax y proporciona un objeto Ajax mucho más simple y que también utiliza Symfony. Este es el motivo por el que la librería Prototype se carga automáticamente cuando se utiliza un helper de Ajax en la plantilla. Las interacciones de Ajax están formadas por tres partes: el elemento que la invoca (un enlace, un formulario, un botón, un contador de tiempo o cualquier otro elemento que el usuario manipula e invoca la acción), la acción del servidor y una zona de la página en la que mostrar la respuesta de la acción. Se pueden crear interacciones más complejas si por ejemplo la acción remota devuelve datos que se procesan en una función JavaScript en el navegador del cliente (15).

```
Ext.Ajax.request({
 //dirección de la página donde será enviada la petición
   url: 'foo.php',
 //respuesta si la petición fue satisfactoria
   success: someFn,
 //respuesta si la petición no fue satisfactoria
   failure: otherFn,
   headers: {
        'my-header': 'foo'
   },
 //parámetros que serán enviados
   params: { foo: 'bar' }
});
```

Figura 2.9 Ejemplo de peticiones AJAX.

Figura 2.8 Diagrama de Peticiones AJAX.

El diagrama de la figura 2.8 muestra un ejemplo de peticiones AJAX donde el cliente envía 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 el JSon que es el encargado del intercambio de datos entre la página cliente y la servidora.

2.3.5 Formato JSon.

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

El formato JSON es el más adecuado para la respuesta del servidor cuando la acción Ajax debe devolver una estructura de datos a la página que realizó la llamada de forma que se pueda procesar con JavaScript. Este mecanismo es útil por ejemplo cuando una sola petición Ajax debe actualizar varios elementos en la página.

JSON se ha convertido en un estándar en el desarrollo de aplicaciones web. Los servicios web proponen la utilización de JSON en vez de XML para permitir la integración de servicios en el navegador del usuario en vez de en el servidor. El formato JSON es seguramente la mejor opción para el intercambio de información entre el servidor y las funciones JavaScript.

Desde la versión 5.2 de PHP existen dos funciones, json_encode() y json_decode(), que permiten convertir un arreglo PHP en un arreglo JSON y viceversa. Estas funciones facilitan la integración de los arreglos JSON (y de Ajax en general) y permiten escribir código PHP nativo más fácil de leer(15).

```
$resultado = array("titulo" => "Resultado a mostrar");
return $this->renderText(json_encode($resultado));
```

json_decode

Decodifica un string JSON. Retorna el valor codificado de *json* en un tipo PHP apropiado. Los valores *true*, *false* y *null* (case-insensitive) se retornan como **TRUE**, **FALSE** y **NULL** respectivamente. **NULL** se retorna si no se puede decodificar *json* o si los datos superan el límite de profundidad de recursión indicado por *depth*. En la figura 2.9 se muestra un ejemplo de json_decode y el resultado obtenido.

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

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

?>
```

Resultado Obtenido:

```
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)
}
```

Figura 2.9 Ejemplos de json_decode().

json_encode

Retorna la representación JSON del valor dado. Si no hay error, retorna un string codificado en JSON. En la figura 2.10 se muestra un ejemplo de json_encode y el resultado obtenido.

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

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

Resultado Obtenido:

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

Figura 2.10 Ejemplos de json_encode()

2.3.6 Action y Clases del Modelo.

Las bases de datos son relacionales. PHP 5 y Symfony están orientados a objetos. Para acceder de forma efectiva a la base de datos desde un contexto orientado a objetos, es necesaria una interfaz que traduzca la lógica de los objetos a la lógica relacional, esta interfaz se llama ORM (*object-relational mapping*) o *"mapeo de objetos a bases de datos"*, y está formada por objetos que permiten acceder a los datos y que contienen en sí mismos el código necesario para hacerlo.

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. Para crear el modelo de objetos de datos que utiliza Symfony, se debe traducir el modelo relacional de la base de datos 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 (15).

El esquema se utiliza para construir las clases del modelo que necesita la capa del ORM. Al analizar el esquema se generan las clases base del modelo:

- BaseArticulo.php
- BaseArticuloPeer.php
- BaseComentario.php
- BaseComentarioPeer.php

Además, se crean las verdaderas clases del modelo de datos:

- Articulo.php
- ArticuloPeer.php
- Comentario.php

- ComentarioPeer.php

Las clases con nombre Base del son las que se generan directamente a partir del esquema. Nunca se deberían modificar esas clases, porque cada vez que se genera el modelo, se borran todas las clases.
Por otra parte, las clases de objetos propias heredan de las clases con nombre Base. Estas clases no se modifican cuando se ejecuta la tarea propel:build-model, por lo que son las clases en las que se añaden los métodos propios. Las clases de tipo *"peer"*, son clases que tienen métodos estáticos para trabajar con las tablas de la base de datos. Proporcionan los medios necesarios para obtener los registros de las tablas. Sus métodos devuelven normalmente un objeto o una colección de objetos de la clase objeto relacionada.

La tarea del action.class.php es crear e iniciar variables, realizar cálculos complejos, comprobaciones de datos que provienen de la Base de Datos, los cuales se obtienen a través de las clases peer. Esta obtención de datos se realiza a través de la clase Criteria() la cual se utiliza para definir consultas sin utilizar SQL.

Conclusiones Parciales.

En este capítulo se han descrito los componentes ejecutables involucrados a través de algunos diagramas como el diagrama de paquetes, el diagrama de secuencia de una de las funcionalidades y un diagrama de clases principal. También se describió alguna de las técnicas, funciones y estándares que se llevarán a cabo para realizar la implementación de dicho sistema.

Capítulo 3. Validación de la solución.

3.1 Introducción.

El desarrollo de sistemas de software lleva implícito una serie de actividades en las cuales los fallos y errores son frecuentes desde el momento inicial. Es por ello que el desarrollo de software debe ir acompañado de una o varias actividades que garanticen la calidad. Las pruebas de software son un elemento crítico para la garantía de la calidad y representa una revisión final de las especificaciones, del diseño y del código. El éxito de las mismas puede mejorar la percepción de calidad del usuario final y lograr su satisfacción. Tienen como objetivo principal asegurar que el software cumpla con las especificaciones requeridas y eliminar los posibles defectos que pudiera tener. De manera específica tiene como propósito:

- Realizar la validación del software desarrollado, determina si el software satisface los requisitos.
- Realizar la verificación del software desarrollado, determina si los productos de una fase satisfacen las condiciones de la misma.

Es preciso aclarar que las pruebas no aseguran la ausencia de defectos en el software, pero sí detectan la mayor cantidad de defectos posibles para su debida corrección.
En este capítulo se especifica el conjunto de validaciones y pruebas que evalúan la calidad del sistema, así como el grado de cumplimiento a las necesidades del cliente o usuario.

3.2 Pruebas.

Una vez generado el código fuente, el software debe ser probado para descubrir (y corregir) el máximo de errores posibles antes de su entrega al cliente. El objetivo es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores. Para esto se aplican las técnicas de pruebas del software. Estas técnicas facilitan una guía sistemática para diseñar pruebas que comprueben la lógica interna de los componentes software y verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y rendimiento.

3.2.1 Objetivos de las pruebas.

En un excelente libro sobre las pruebas del software, Glen Myers [MYE79] establece varias normas que pueden servir acertadamente como objetivos de las pruebas:

1. La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
2. Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
3. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Si la prueba se lleva a cabo con éxito (de acuerdo con el objetivo anteriormente establecido), descubrirá errores en el software. Como ventaja secundaria, la prueba demuestra hasta qué punto las funciones del software parecen funcionar de acuerdo con las especificaciones y parecen alcanzarse los requisitos de rendimiento. Además, los datos que se van recogiendo a medida que se lleva a cabo la prueba proporcionan una buena indicación de la fiabilidad del software y, de alguna manera, indican la calidad del software como un todo. Pero, la prueba no puede asegurar la ausencia de defectos; solo puede demostrar que existen defectos en el software.

Los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

3.2.2 Métricas utilizadas para validar el diseño

Las métricas de software son una medida cuantitativa que permite a los desarrolladores tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software.(17)

La métrica empleada está diseñada para evaluar los siguientes atributos de calidad:

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

La métrica escogida como instrumento para evaluar la calidad del diseño descrito anteriormente y su relación con los atributos de calidad es la siguiente:

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Tabla 1: Atributos de calidad evaluados por la métrica TOC

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Atributo	Categoría	Promedio
Responsabilidad	Baja	<= Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2* Promedio
Complejidad de implementación	Baja	<= Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2* Promedio
Reutilización	Baja	>2* Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Tabla 2: Criterios de evaluación para la métrica TOC.

Resultados obtenidos de la aplicación de la métrica TOC:

Tras los resultados obtenidos de la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 74% de las clases empleadas en el sistema poseen entre 1 y 10 procedimientos (Ver Figura 3.1) lo que conlleva a evaluaciones positivas de un 84% de los atributos de calidad involucrados (responsabilidad, complejidad de implementación y reutilización).

Clases	Cantidad de Métodos	Responsabilidad	Complejidad	Reutilización
action	12	Media	Media	Media
DncDeclaracionPeer	5	Baja	Baja	Alta
DncDitPeer	5	Baja	Baja	Alta
DncrProductoAcumuladoPeer	6	Baja	Baja	Alta
DncAcumulado	3	Baja	Baja	Alta
DncRad	1	Baja	Baja	Alta
DncVehiculoTripulante	1	Baja	Baja	Alta
sfAuxiliarDNCTC	18	Alta	Alta	Baja

Tabla 3: Instrumento de evaluación de la métrica TOC

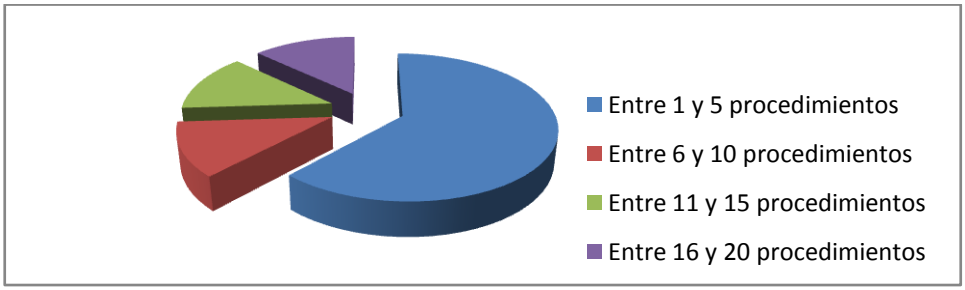

Figura 3.1: Gráfico en % del total de procedimientos por clases

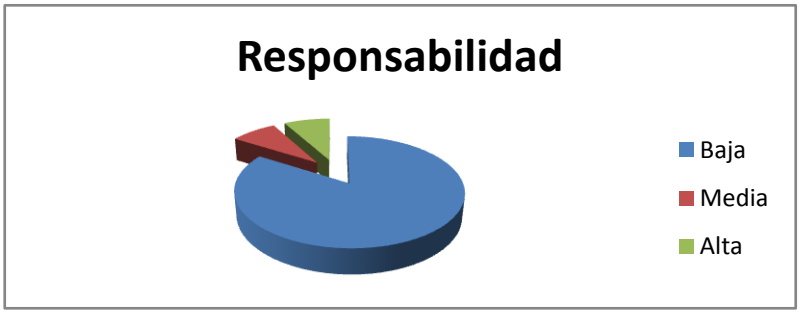

Figura 3.2: Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad.

Figura 3.3 Resultados de la evaluación de la métrica TOC para el atributo Complejidad.

Figura 3.4: Resultados de la evaluación de la métrica TOC para el atributo Reutilización.

3.2.3 Prueba de Caja Blanca.

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que (1) garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; (2) ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; (3) ejecuten todos los bucles en sus límites y con sus límites operacionales; y (4) ejerciten las estructuras internas de datos para asegurar su validez.

Prueba del camino básico.

La *prueba del camino básico* es una técnica de prueba de caja blanca propuesta inicialmente por Tom McCabe [MCC76]. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida corno guía para la

definición de un *conjunto básico* de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Pasos para aplicar esta técnica:

1. Usando el diseño o el código como base, se dibuja el correspondiente grafo de flujo.
2. Determinar la complejidad ciclomática del grafo de flujo resultante.
3. Determinar un conjunto básico de caminos linealmente independientes.
4. Preparar los casos de prueba que forzarán la ejecución de cada camino del conjunto básico.

A continuación se expone cómo se obtuvo el diseño de los casos de prueba de caja blanca aplicando la técnica del camino mínimo.

Se toma como funcionalidad de muestra “Devolver Artículos”. Este requisito se realiza con el objetivo de registrar los productos que le son devueltos a un tripulante luego de una acción de decomiso o retención. De cada producto devuelto se registra la cantidad devuelta del mismo y el valor dado por la aduana para su despacho.

```
public function executeRegistrarDevolucion(){
      $gridArticulos = $this->getRequestParameter('nombArticulos'); //1
      $idDeclaracion = $this->getRequestParameter('idDeclaracion'); //1
      $idPersona = $this->getRequestParameter('idPersona'); //1
      $derechos = $this->getRequestParameter('derechos'); //1
      $totalpagar = $this->getRequestParameter('totalpagar'); //1
      $valorAduana = $this->getRequestParameter('valorAduana'); //1

      if((empty($derechos))||(empty($totalpagar))||(empty($valorAduana))) //2
          {
            return $this->renderText("{'success':false, 'msg': 'Debe realizar la liquidacion'}"); //3
        }

      try{
           $articulos = json_decode($gridArticulos); //4
           foreach ($articulos as $articulo) //5
              {
            $cantDevuelta = $articulo->cantDevuelta; //6
            $valor = $articulo->valorTotal; //6
            $idProduc = $articulo->idProducto; //6

            DncrProductoAcumuladoPeer::devolverProductoAcumlado($idPersona,$idProduc,$cantDevuelta,$valor); //6
            DncDitPeer::modificarConPreDespacho($idDeclaracion,$derechos,$totalpagar,$valorAduana); //6
            }

            return $this->renderText("{'success':true, 'msg': 'Se ha realizado la operaci&oacute;n con &eacute;xito'}")
      }catch (Exception $error) //8
            {
            return $this->renderText("{'success':false, 'msg':".$error->getMessage()."}"); //9
          }

      } //10
```

Figura 3.5: Código fuente de la funcionalidad Devolver Artículos

Variables empleadas en el procedimiento:

- $gridArticulos: Arreglo de objetos donde, en cada posición, se encuentran los datos de los artículos importados por el tripulante.
- $idDeclaracion: Variable entera que define el identificador de la declaración.
- $idPersona: Variable entera que define el identificador de la persona, en este caso del tripulante.

- $derechos: Variable que contiene el importe que debe pagar el tripulante por concepto de aranceles aduaneros.
- $totalPagar: Variable que define el importe total que debe pagar el tripulante.
- $valorAduana: Variable que define el valor que tiene para la aduana la mercancía importada por el tripulante.

A partir del código del procedimiento, se obtiene el siguiente grafo de flujo:

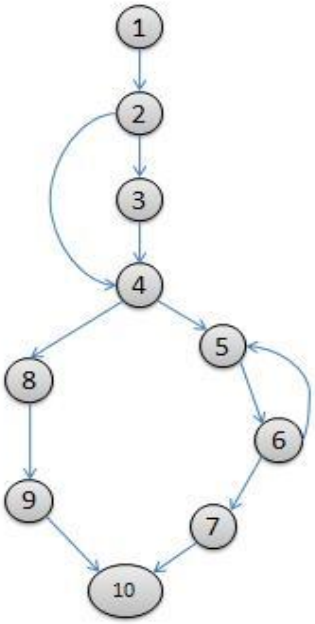

Figura 3.6: Grafo de flujo asociado al algoritmo executeRegistrarDevolución

Cálculo de la complejidad ciclomática del grafo anterior:

El cálculo de la complejidad ciclomática se calcula de 3 formas, en las cuales el resultado debe ser el mismo:

1. V(G) = A – N + 2 Siendo A la cantidad de aristas y N la cantidad de nodos
   V(G) = 12 – 10 + 2
   V(G) = 4

2. V(G) = P + 1 Siendo P el número de nodos predicado, en este caso los nodos predicando son el 2, el 4 y el 6.
   V(G) = 3 + 1
   V(G) = 4
3. V(G) = R Siendo R el número de regiones del grafo
   V(G) = 4

El cálculo realizado mediante las fórmulas anteriores arrojan una complejidad ciclomática de valor 4, lo cual define que existen cuatro posibles caminos por donde el flujo puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Seguidamente es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución.

Camino # 1: 1-2-3-4-5-6-7-10

Camino # 2: 1-2-3-4-8-9-10

Camino # 3: 1-2-4-8-9-10

Camino # 4: 1-2-5-6-7-10

Después de determinar los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico teniendo en cuenta las siguientes variables de chequeo:

Caso de Prueba para el camino # 1

Descripción	Los datos de entrada no están vacíos, contienen información de la declaración, la persona y los productos para registrar los artículos que se devuelven.
Condición de ejecución	$derecho = null
Entrada	$gridArticulos=[{"idProducto":"1","codigo":"10","descripcion":"TV PANTALLA PLANA","cantidadImportada":"1","valorDado":"356","tipo":"DECOMISADO","cantDevuelta":null,"cantDecomisada":1,"valorTotal":50},{"idProducto":"2","codigo":"11","descripcio

	n":"ROPA","cantidadImportada":"6","valorDado":"534","tipo":"DECOMISADO","cantDevuelta":null,"cantDecomisada":2,"valorTotal":10}] $idDeclaracion = 1 $idPersona = 1 $derechos = 0 $totalpagar = 0 $valorAduana = 0
Resultados esperados	{'success':false, 'msg': 'Debe realizar la liquidacion'} $gridArticulos=[{"idProducto":"1","codigo":"10","descripcion":"TV PANTALLA PLANA","cantidadImportada":"1","valorDado":"356","tipo":"LIBERADO","cantDevuelta":1,"cantDecomisada":null,"valorTotal":null},{"idProducto":"2","codigo":"11","descripcion":"ROPA","cantidadImportada":"6","valorDado":"534","tipo":"LIBERADO","cantDevuelta":2,"cantDecomisada":null,"valorTotal":null}] $valorAduana = 60 $derechos = 15 $servicios = 2 $totalpagar = 17 {'success':true, 'msg': 'Se ha realizado la operación con éxito'}

Caso de Prueba para el camino # 2

Descripción	Los datos de entrada no están vacíos, contienen información de la declaración, la persona y los productos para registrar los artículos que se devuelven.
Condición de ejecución	$derecho = null $valorAduana = null

	$totalPagar = null
Entrada	$gridArticulos=[{"idProducto":"1","codigo":"10","descripcion":"TV PANTALLA PLANA","cantidadImportada":"1","valorDado":"300","tipo":"DECOMISADO","cantDevuelta":null,"cantDecomisada":1,"valorTotal":50},{"idProducto":"2","codigo":"11","descripcion":"ROPA","cantidadImportada":"2","valorDado":"60","tipo":"DECOMISADO","cantDevuelta":null,"cantDecomisada":2,"valorTotal":10}] $idDeclaracion = 3 $idPersona = 16 $derechos = 0 $totalpagar = 0 $valorAduana = 0
Resultados esperados	$derechos = 0 $totalpagar = 0 $valorAduana = 0 {'success':false, 'msg': 'Debe realizar la liquidacion'} ("{'success':false, 'msg':".$error->getMessage()."}

Caso de Prueba para el camino # 3

Descripción	Los datos de entrada no están vacíos, contienen información de la declaración, la persona y los productos para registrar los artículos que se devuelven.
Condición de ejecución	Los datos registrados en el gridArticulos, o al menos uno de ellos, es registrado como un dato tipo string.
Entrada	$gridArticulos=[{"idProducto":"1","codigo":"10","descripcion":"ROPA","cantidadImportada":"20","valorDado":"200","tipo":"DECOMISADO","cantDevuelta":null,"cantDecomisada":20,"valorTotal":null}] $idDeclaracion = 7

	$idPersona = 29 $derechos = 0 $totalpagar = 0 $valorAduana = 0
Resultados esperados	("{'success':false, 'msg':".$error->getMessage()."}

Caso de Prueba para el camino # 4

Descripción	Los datos de entrada no están vacíos, contienen información de la declaración, la persona y los productos para registrar los artículos que se devuelven.
Condición de ejecución	$derecho != null $valorAduana != null $totalPagar != null
Entrada	$gridArticulos=[{"idProducto":"1","codigo":"10","descripcion":"TV PANTALLA PLANA","cantidadImportada":"3","valorDado":"500","tipo":"DECOMISADO","cantDevuelta":null,"cantDecomisada":3,"valorTotal":null} $idDeclaracion = 55 $idPersona = 16 $derechos = 0 $totalpagar = 0 $valorAduana = 0
Resultados esperados	$valorAduana = 60 $derechos = 15 $servicios = 2 $totalpagar = 17 {'success':true, 'msg': 'Se ha realizado la operación con éxito'}

Después de aplicar los casos de prueba diseñados y obtener los resultados esperados en cada uno de ellos, se pudo comprobar que el método funciona correctamente, garantizando la ejecución, por lo menos una vez, de cada sentencia del programa.

3.2.4 Prueba 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 que los métodos de caja blanca. 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 bases de datos externas, (4) errores de rendimiento y (5) errores de inicialización y de terminación. A diferencia de la prueba de caja blanca, que se lleva a cabo previamente en el proceso de prueba, la prueba de caja negra tiende a aplicarse durante fases posteriores de la prueba. Las pruebas se diseñan para responder a las siguientes preguntas:

- ¿Cómo se prueba la validez funcional?
- ¿Cómo se prueba el rendimiento y el comportamiento del sistema?
- ¿Qué clases de entrada compondrán unos buenos casos de prueba?
- ¿Es el sistema particularmente sensible a ciertos valores de entrada?
- ¿De qué forma están aislados los límites de una clase de datos?
- ¿Qué volúmenes y niveles de datos tolerará el sistema?
- ¿Qué efectos sobre la operación del sistema tendrán combinaciones específicas de datos?

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(17).

En la tabla 4, se muestra el caso de prueba del requisito “Registrar pre-despacho de DIT”, el cual describe cada uno de los escenarios que pueden existir ante las posibles acciones realizas por el usuario. De manera similar se realizaron los casos de prueba al resto de los requisitos.

Nombre del requisito	Descripción de la funcionalidad	Escenarios de prueba	Flujo central
Registrar pre-despacho de DIT	Permite que se registren los datos de la DIT para ser pre-despachada.	EC 1.1: Registrar pre-despacho de DIT	- Se ejecuta el requisito “Buscar Tripulante”. - Muestra la pantalla para captar los datos de la DIT, donde ya aparecen los datos del tripulante: nombre y apellidos, carnet de identidad y valor acumulado.
		EC 1.1.2: Introduce los datos generales de la DIT: Número de manifiesto País de procedencia/destino Dinero recibido Valor de la importación Tipo de operación Fecha de salida Fecha de regreso Id buque/aeronave Los artículos importados (cantidad y valor de cada uno).	- Oprime el botón “Aceptar”. - Valida que se llenen todos los campos y guarda los datos. - Le asigna un número consecutivo a la declaración. - Guarda los datos de la declaración y le asigna el estado de “Pre-despachada”. - Muestra mensaje con el número asignado a la declaración registrada. - En caso de oprimir el botón “Despachar”, se ejecuta el requisito “Registrar Despacho de DIT”.

		EC 2.1: Error en Datos	- Muestra mensaje indicando los errores cometidos.

Tabla 4: Caso de prueba del Requisito Registrar Pre-Despacho DIT

ID del escenario	Escenario	Número de manifiesto	País de procedencia	Dinero recibido	Valor de la importación	Tipo de operación	Respuesta del sistema	Resultado de la prueba
EC 1.1	Registrar pre-despacho de DIT	V	V	V	V	V	Muestra mensaje con el número asignado a la declaración registrada.	Se muestra un mensaje donde aparece el número asignado a la DIT.
		V	V	I	I	V	No permite que se inserten datos diferentes a números enteros o decimales.	Permite la entrada de cualquier tipo de datos.
		I	V	I	V	I	Muestra un mensaje de error indicando que los datos son incorrectos.	Se muestra un mensaje indicando que debe corregir los datos señalados.
		N/A	V	N/A	V	N/A	Muestra un mensaje de error indicando que hay campos vacíos	Se muestra un mensaje indicando que debe llenar todos los campos.
		V	N/A	V	N/A	V	Muestra un mensaje de error indicando que hay campos vacíos	Se muestra un mensaje indicando que debe llenar todos los campos.
EC 2.1	Error en Datos	V	V	V	V	V	Muestra mensaje con el número asignado a la	Se muestra un mensaje con un número que no es

							declaración registrada.	consecutivo a la última declaración registrada.
		NA	V	N/A	V	V	Muestra un mensaje de error indicando que hay campos vacíos.	Se muestra un mensaje indicando que hay campos vacíos.
		V	I	V	I	I	Muestra un mensaje de error indicando que los datos no son correctos.	Se muestra un mensaje de error indicando que los datos no son correctos.

Tabla 5: Ejecución del caso de prueba

ID del escenario	Escenario	Fecha de salida	Fecha de regreso	Id Buque/ Aeronave	Respuesta del sistema	Resultado de la prueba
EC 1.1	Registrar pre-despacho de DIT	V	V	V	Muestra mensaje con el número asignado a la declaración registrada.	Se muestra un mensaje con el número de la declaración registrada.
		V	I	V	Muestra un mensaje de error indicando que los datos no son correctos. Debe validar que la fecha de regreso sea posterior a la fecha de salida.	No se realiza la validación correspondiente a las fechas de salida y regreso.
		I	V	I	Muestra un mensaje de error indicando que los datos son incorrectos.	Se muestra un mensaje de error indicando que los campos no son correctos.

		N/A	V	N/A	Muestra un mensaje de error indicando que hay campos vacíos.	Se muestra un mensaje indicando que los campos no pueden estar vacíos.
EC 2.1	Error en Datos	V	V	V	Muestra mensaje con el número asignado a la declaración registrada.	Se muestra un mensaje con el número de la declaración registrada.
		V	I	I	Muestra un mensaje de error indicando que los datos son incorrectos.	Se muestra un mensaje de error indicando que los campos no son correctos.
		I	V	I	Muestra un mensaje de error indicando que los datos son incorrectos.	Se muestra un mensaje de error indicando que los datos no son correctos.
		N/A	V	N/A	Muestra un mensaje de error indicando que hay campos vacíos.	Se muestra un mensaje indicando que los campos no pueden estar vacíos.

Tabla 6: Continuación de la tabla 5

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.]

Elemento	No.	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Resp. equipo de desarrollo
Iteración # 1									
Datos insertados.	1	Validación de datos insertados.	No se valida que la fecha de regreso sea posterior a la fecha de salida.	Pruebas internas.	X		Se debe incorporarla validación correspondiente.	PD 06/06/2011	
Datos inserta	2	Validación de datos	No valida que entren	Pruebas	X		Se debe incorporar	PD	

dos.		insertados.	símbolos diferentes de la “,”.	internas.			la validación correspondiente.	06/06/2011	
Iteración # 2									
Asignar número a la declaración.	3	Número consecutivo incorrecto.	El número de la declaración no es consecutivo a la última declaración registrada	Pruebas internas.	X		Arreglar función para mostrar el consecutivo que corresponda.	PD 08/06/2011 RA 06/06/2011	Corrección de la NC 1 y 2 detectadas el 06/06/2011.
Iteración # 3									
Asignar número a la declaración.	4	Número consecutivo incorrecto.						RA 08/06/2011	Corrección de la NC 3 detectada el 08/06/2011.

Tabla 7: Registro de defectos y dificultades detectadas

RA: Resuelta.

PD: Pendiente.

Conclusiones Parciales.

La calidad del módulo implementado fue la premisa fundamental en el desarrollo de este capítulo. En ese sentido se efectuaron pruebas de caja blanca y caja negra, de las cuales se obtuvo los casos de prueba que ayudan a comprobar el correcto funcionamiento del producto desarrollado.

En general, los resultados obtenidos fueron favorables, desde el punto de vista funcional todos los requisitos realizan las funcionalidades requeridas y responden a las necesidades del cliente con calidad y eficiencia.

Conclusiones.

Con la culminación del presente trabajo de diploma se contribuye a la agilización de los procesos aduanales para mejorar las operaciones importación y exportación de los tripulantes. Por otra parte se les dio cumplimiento a los objetivos específicos propuestos:

- Se elaboró el marco teórico de la investigación a partir del estudio del estado del arte; el mismo evidenció la carencia de una solución informática capaz de responder a las necesidades y requerimientos de la economía cubana. Por otra parte, se analizaron diferentes tecnologías, lenguajes y herramientas indispensables para llevar a cabo el desarrollo de la investigación.
- Se realizó el diseño y la implementación de los componentes, con el objetivo de erradicar los problemas de los sistemas existentes y fusionar sus mejores prácticas teniendo en cuenta las necesidades del cliente.
- Finalmente, se aplicaron pruebas exhaustivas al software implementado, las cuales demostraron el cumplimiento de diferentes atributos de calidad, lo que demuestra que los componentes cumplen satisfactoriamente con los requisitos definidos por los clientes.

Recomendaciones

Se recomienda en el presente trabajo, considerando que se han cumplido los objetivos trazados:

- Continuar realizando pruebas de calidad a los componentes para garantizar su buen funcionamiento y certificarlos.
- Realizar el despliegue de la aplicación en varias entidades aduaneras para comprobar si cumple desde el punto de vista tecnológico con las expectativas del cliente.

Referencias Bibliográficas.

1. WRIGHT, C. *Comercio Exterior* Disponible en: http://cristianwright.blogspot.com/2008/09/el-sistema-informatico-maria.html. .
2. NICODEMOS, S. B. *Sistema María, una necesaria evolución constante. .* Disponible en: http://www.caei.com.ar/es/programas/comercio/6.pdf.
3. *Conferencia de las Naciones Unidas sobre Comercio y Desarrollo*. Disponible en: http://www.unctad.org/templates/Page.asp?intItemID=4140&lang=3.
4. ARELLANO, A. A. F. *Conociendo al SIDUNEA - Sistema Aduanero Automatizado* Disponible en: http://www.gestiopolis.com/canales2/economia/sidunea.htm.
5. MEDEROS, Y. F. y HERNÁNDEZ, A. M. *"Sistema de despacho no comercial".* "Universidad de las Ciencias Informáticas". 2007.
6. *Sitio Web de la Aduana Cubana*. Disponible en: https://sua.aduana.cu/sua.php.
7. ORG. *Free Download Manager* Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.
8. PÉREZ, M. D. *Software para el tratamiento inteligente de datos sobre patentes.* 2008, Disponible en: http://bvs.sld.cu/revistas/aci/v17_5_08/aci06508.htm.
9. ECURED. *Conocimientos con todos y para todos.* 2007, Disponible en: http://www.ecured.cu/index.php/Symfony.
10. ECURED. *Conocimiento con todos y para todos.* 2007, Disponible en: http://www.ecured.cu/index.php/Oracle.
11. ECURED. Conocimientos con todos y para todos. 2007, nº Disponible en: http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_y_arquitectura.
12. JACOBSON y BOOCH, I. *El Proceso Unificado de Desarrollo de Software.* Madrid: 2000, Disponible en: http://www.unilibro.es/find_buy_es/libro/addison_wesley_iberoamericana_espana_s_a_/el_proceso_unificado_de_desarrollo_de_software.asp?sku=224141&idaff=0.

13. MARTÍNEZ, G. M. *Análisis y Diseño de Sistemas* Disponible en: http://www.monografias.com/trabajos5/andi/andi.shtml.

14. CAMPOS. *Base de Datos* Disponible en: http://www.buenastareas.com/ensayos/Base-De-Datos-Introduccion/548070.html.

15. POTENCIER, F. y ZANINOTTO, F. *Symfony la guía definitiva* 2008,

16. MARTÍNEZ, L. M. G. *Componentes Visuales que extienden la arquitectura de dominio específico del GINA.* UCI, 2010.

17. PRESSMAN. Ingeniería del Software un enfoque práctico. 2002, nº

Glosario de Términos.

AGR: Aduana General de la República de Cuba.

Decomiso: Pena que consiste en la incautación por parte del Estado de mercancías o instrumentos causa de delito.

Despacho: Un despacho aduanero es una destinación que tiene por objeto someter al control del servicio aduanero una operación. En la misma se liquidan los tributos pertinentes y la aduana da su conformidad para el embarque en caso de una exportación, y su retiro en caso de una importación.

DIT: Declaración de Importación de Tripulantes. Documento que contiene la información de los tripulantes y de los productos importados por él.

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.

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

Pre-Despacho: Acción que se realiza en la aduana de registrar una DIT y guardarla con un estado de "Pre-Despachada". Anterior a realizar el despacho.

Retención: Retener una mercancía, por algún motivo aduanero, la cual se guarda bajo custodia de aduana hasta que el interesado gestione su devolución.

Tripulante: Se consideran tripulantes todas aquellas personas que al arribar o salir un vehículo del territorio Nacional, por cualquier lugar habilitado para operaciones aduaneras, se encuentran a bordo del mismo, prestando servicios en calidad de empleados del transportador.

Anexos

Anexo 1. Diagrama de Clases del requerimiento Buscar Declaración.

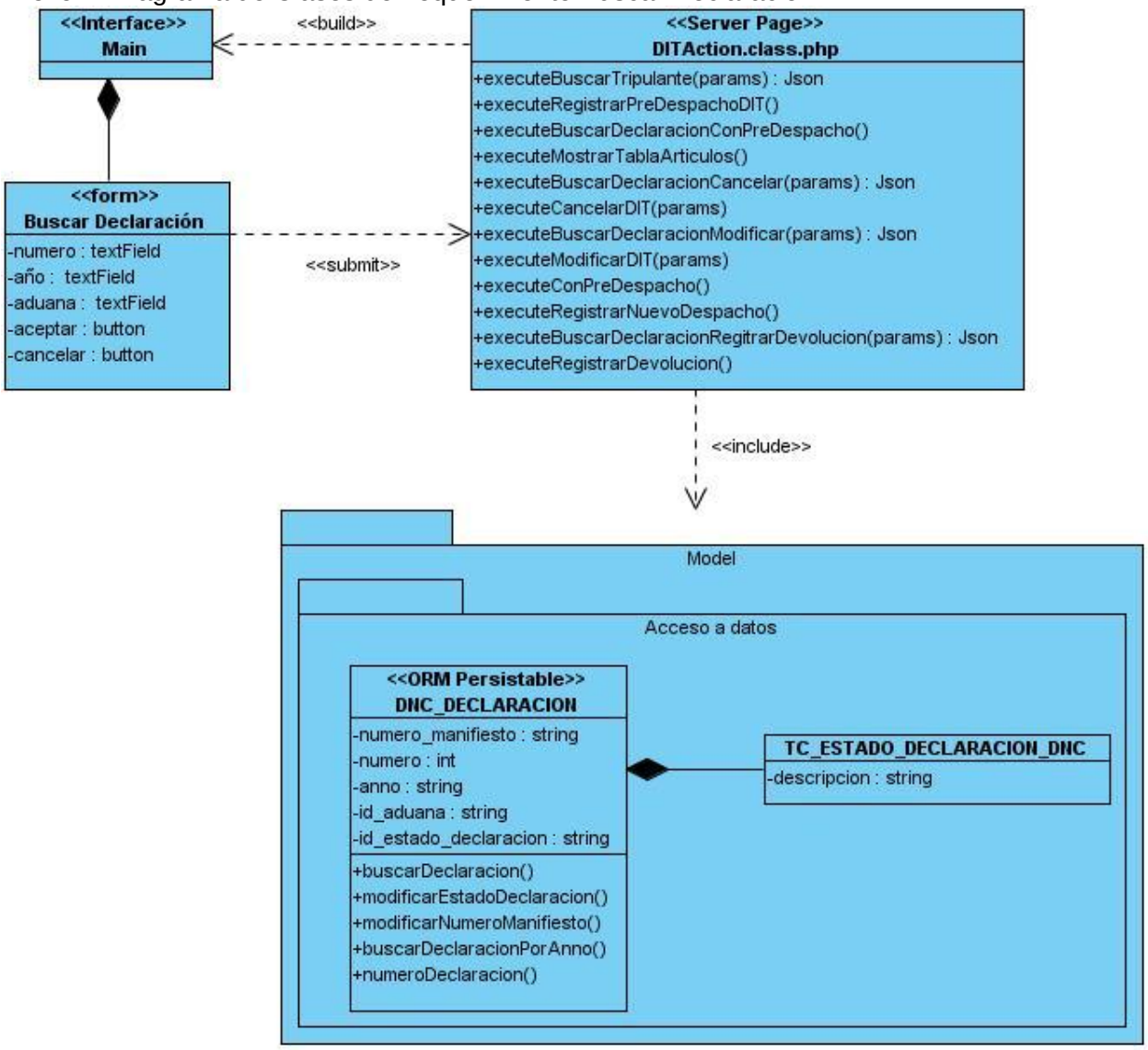

Figura 1: Buscar Declaración

Anexo 2. Diagrama de Clases del requerimiento Buscar Tripulante.

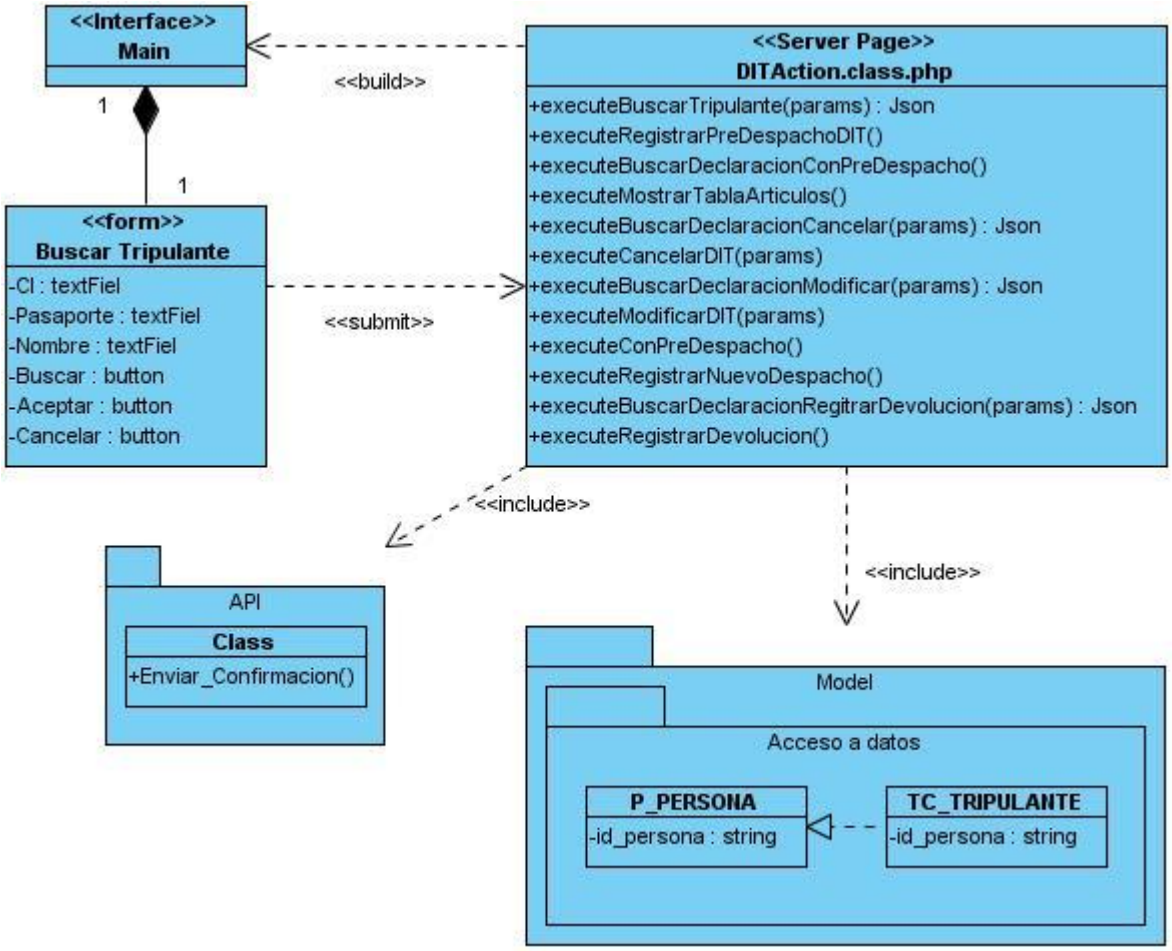

Figura 2: Buscar Tripulante.

Anexo 3. Diagrama de Clases del Requerimiento Registrar Despacho.

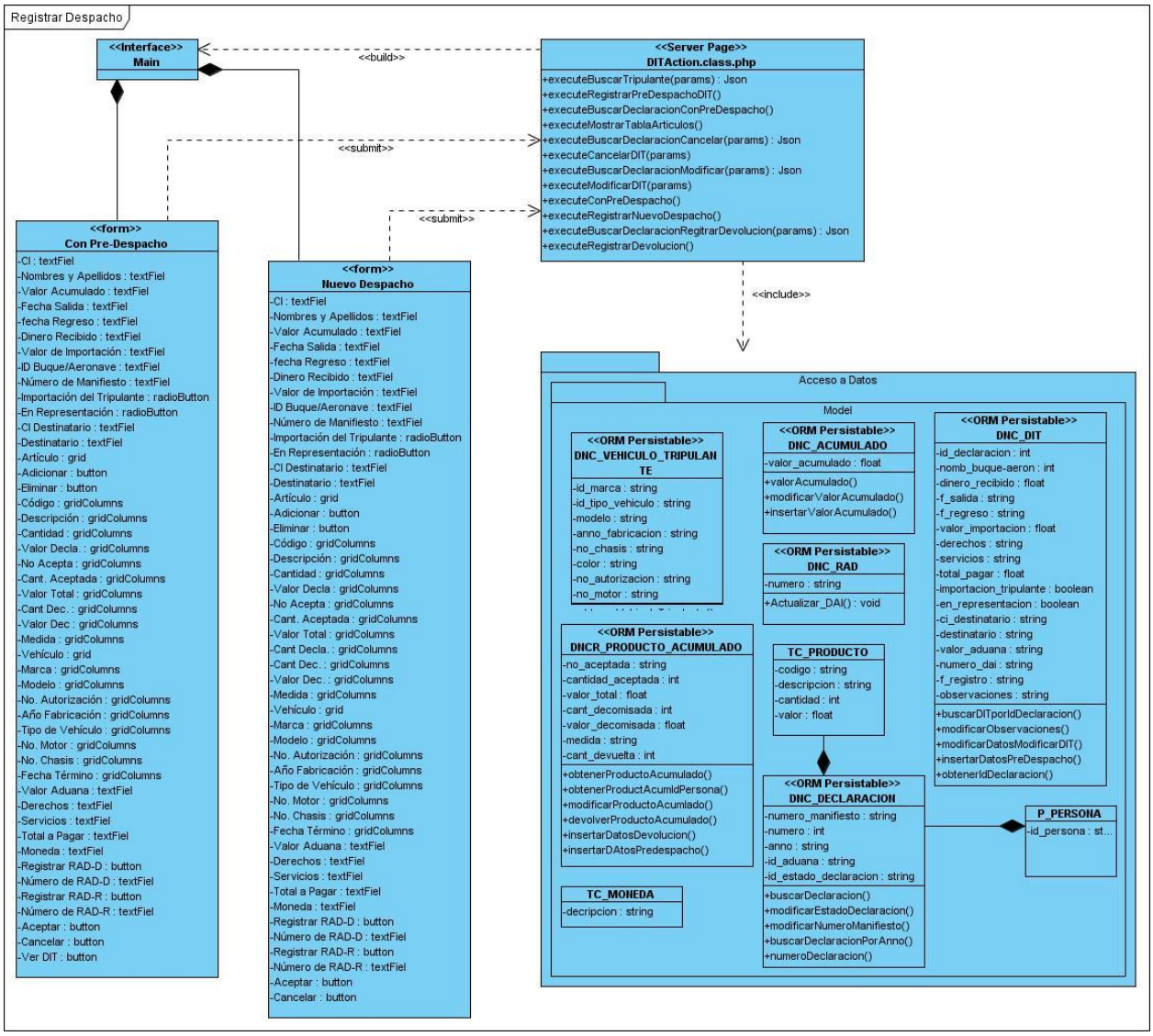

Figura 3: Registrar Despacho.

Anexo 4. Diagrama de Clases del Requerimiento Registrar Devolución.

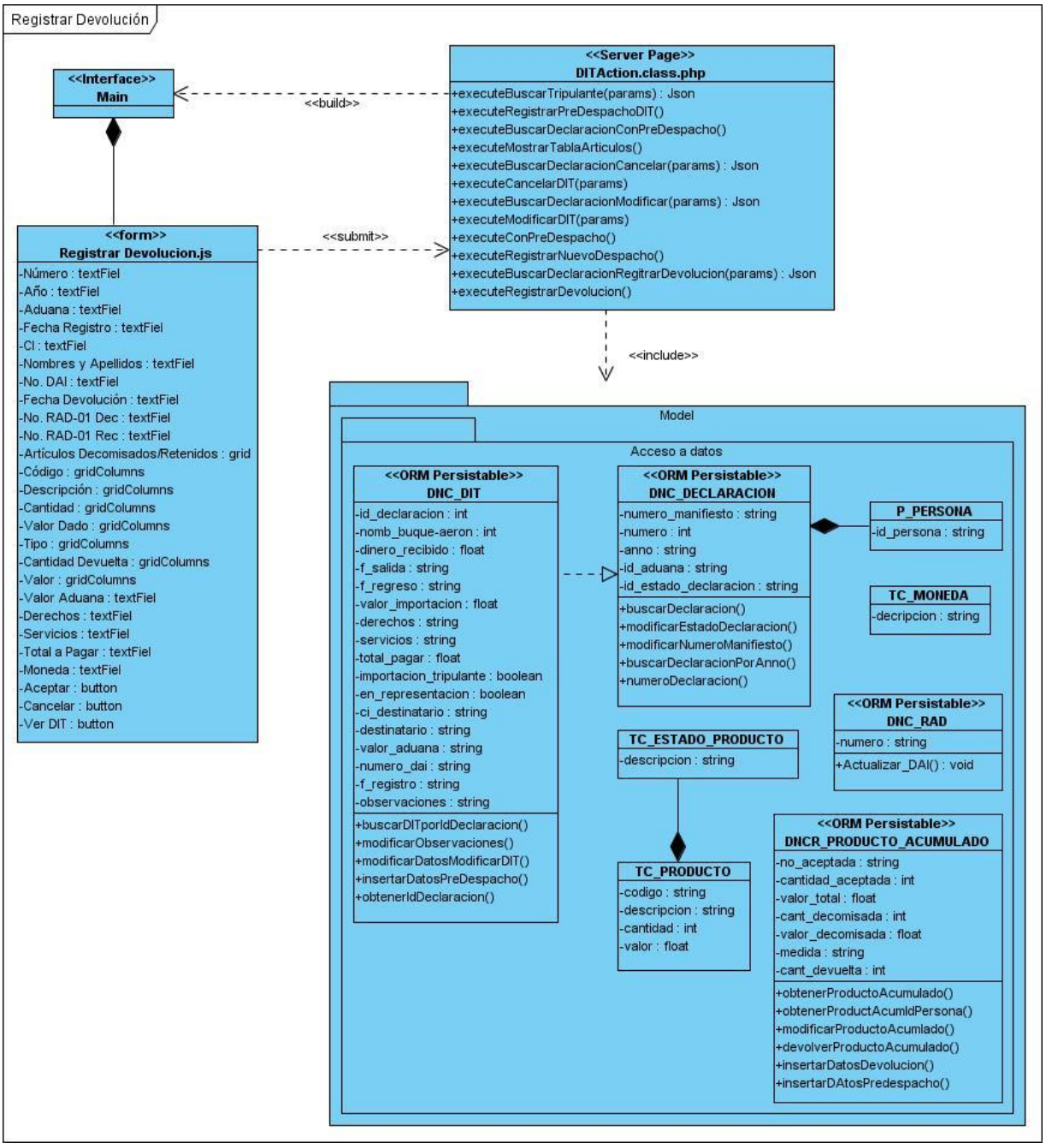

Figura 4: Registrar Devolución.

Anexo 5. Diagrama de Clases del Requerimiento Modificar DIT.

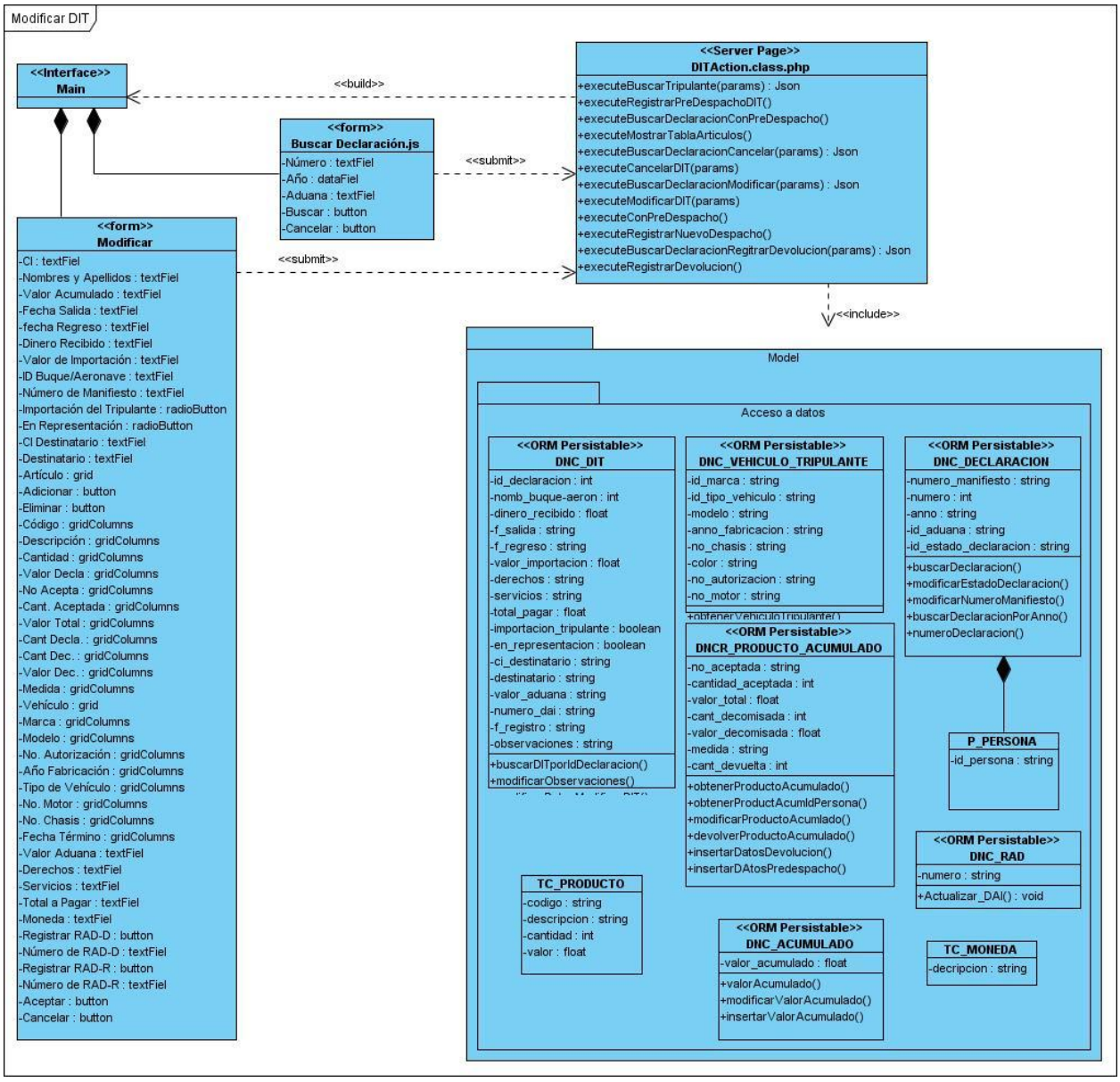

Figura 5: Modificar DIT.

Anexo 6. Diagrama de Clases del Requerimiento Cancelar DIT.

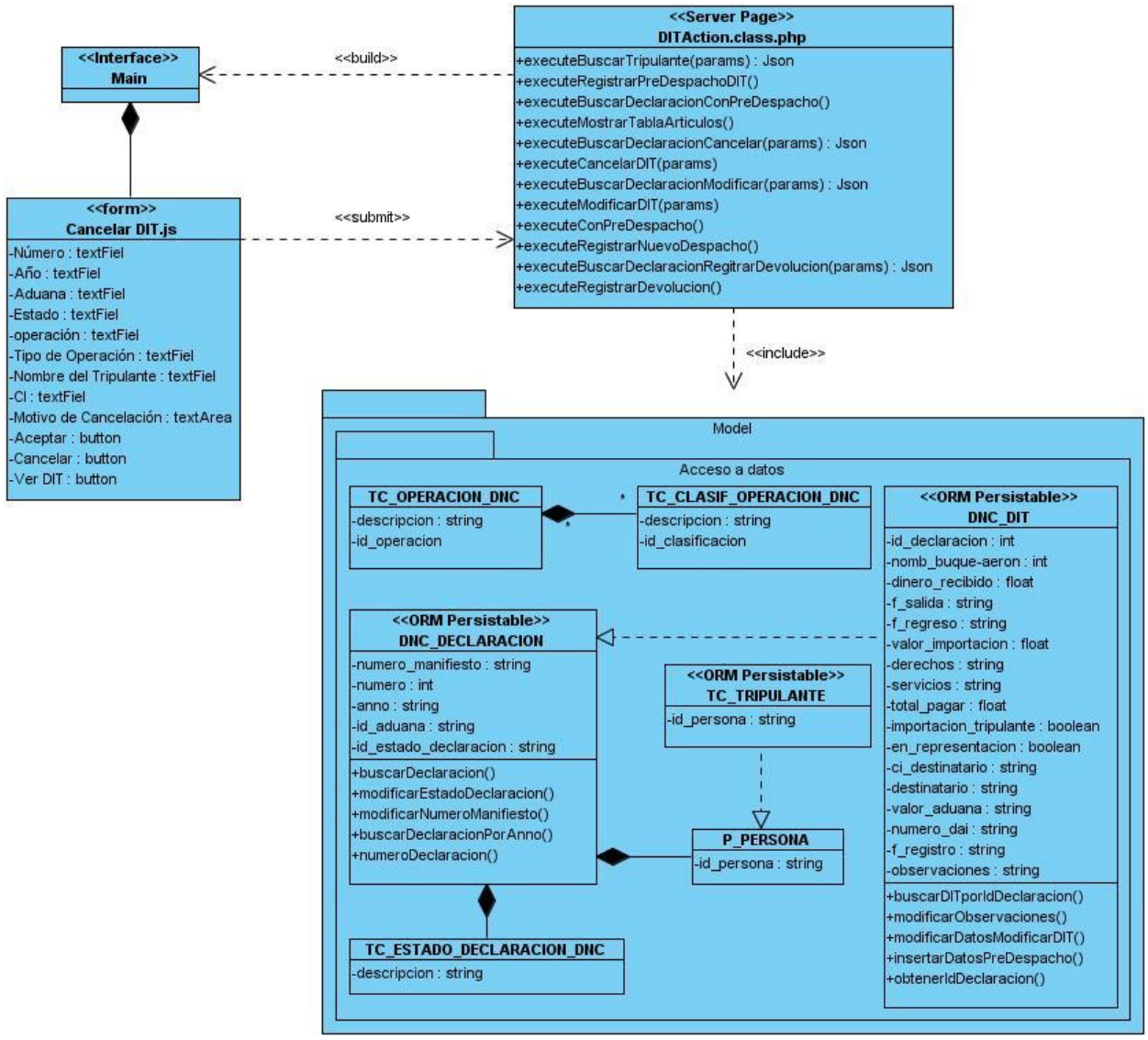

Figura 6: Cancelar DIT.

Anexo 7. Diagrama de Secuencia, Buscar Declaración.

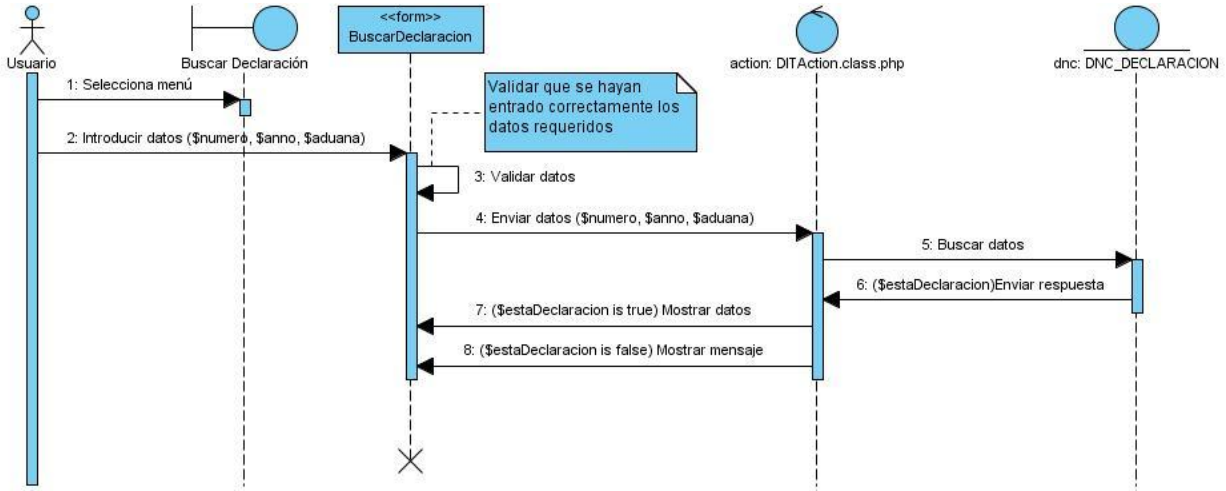

Figura 7: Buscar Declaración.

Anexo 8. Diagrama de Secuencia, Buscar Tripulante.

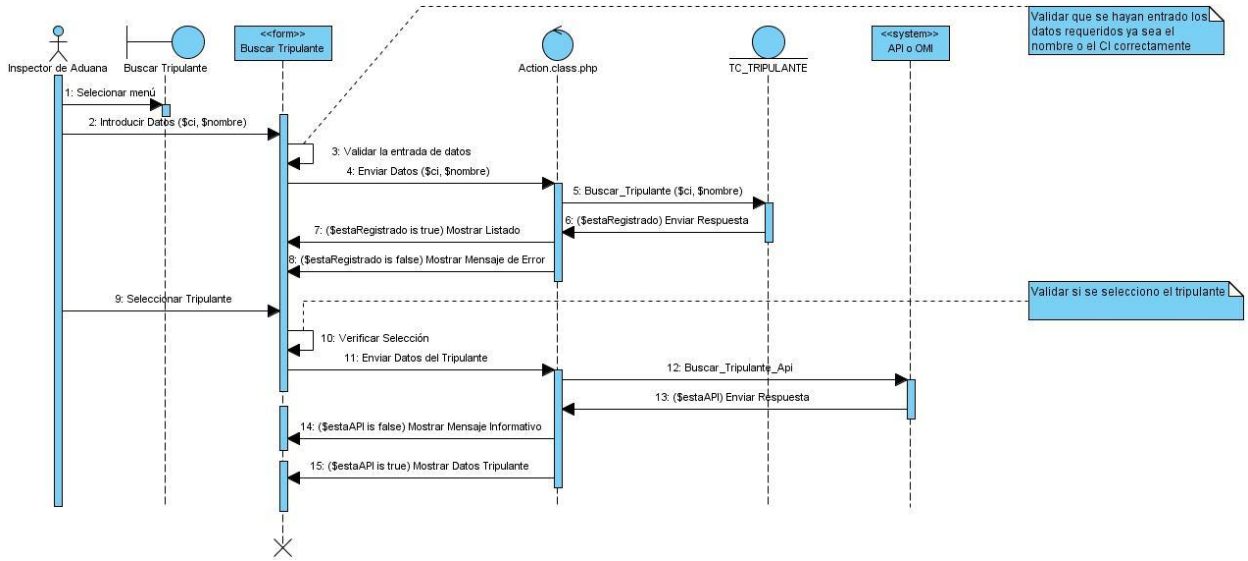

Figura 8: Buscar Tripulante.

Anexo 9. Diagrama de Secuencia, Registrar Despacho con Pre-Despacho.

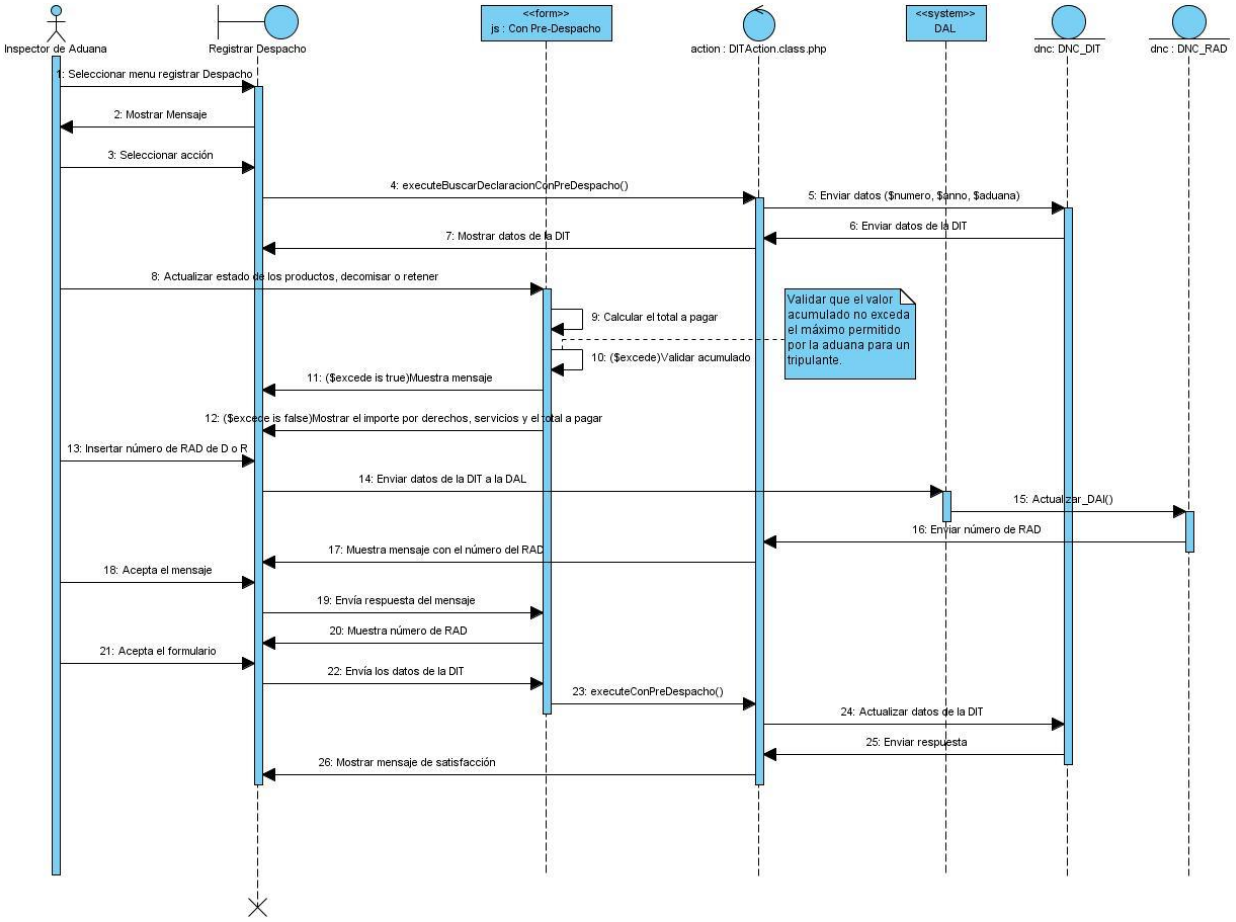

Figura 9: Registrar Despacho con Pre-Despacho.

Anexo 10. Diagrama de Secuencia, Registrar Nuevo Despacho.

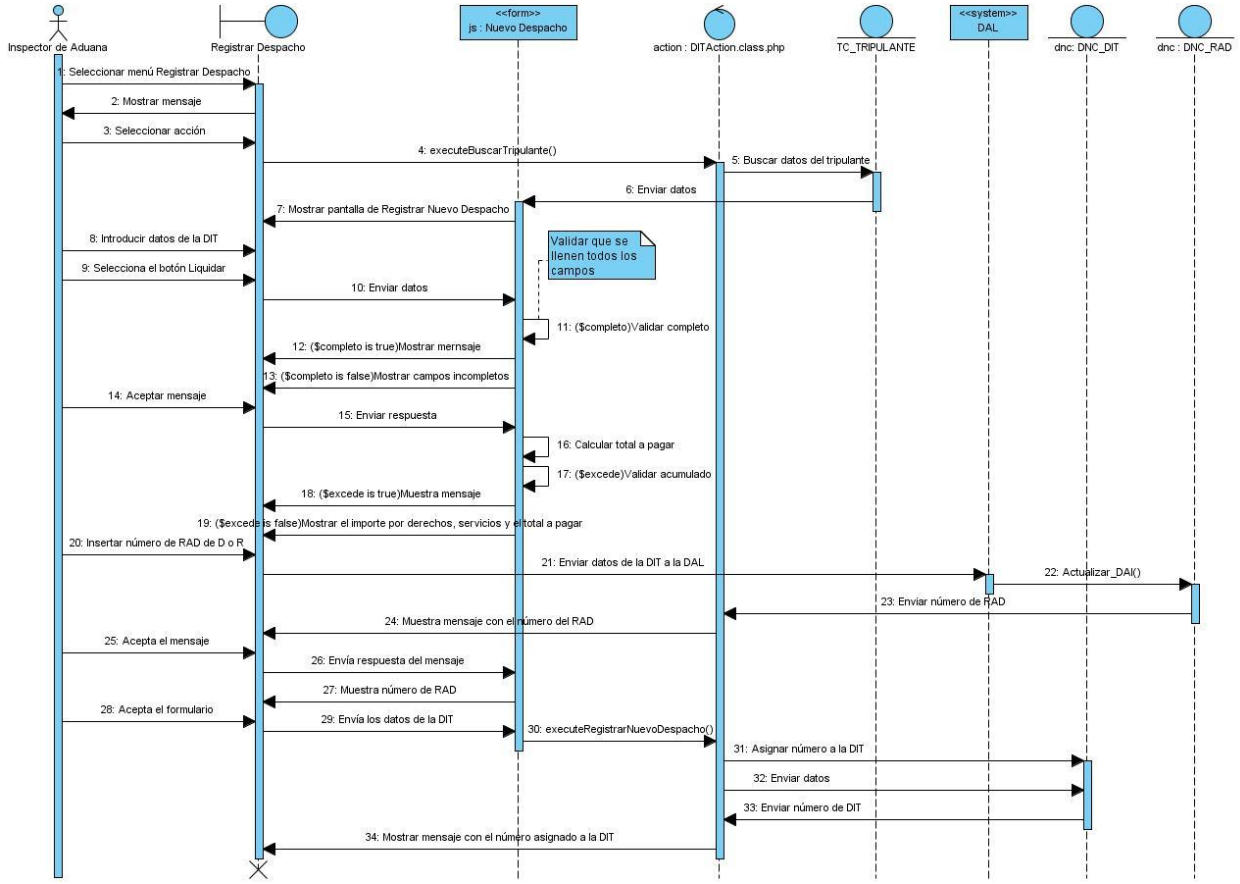

Figura 10: Registrar Nuevo Despacho.

Anexo 11. Diagrama de Secuencia, Registrar Devolución.

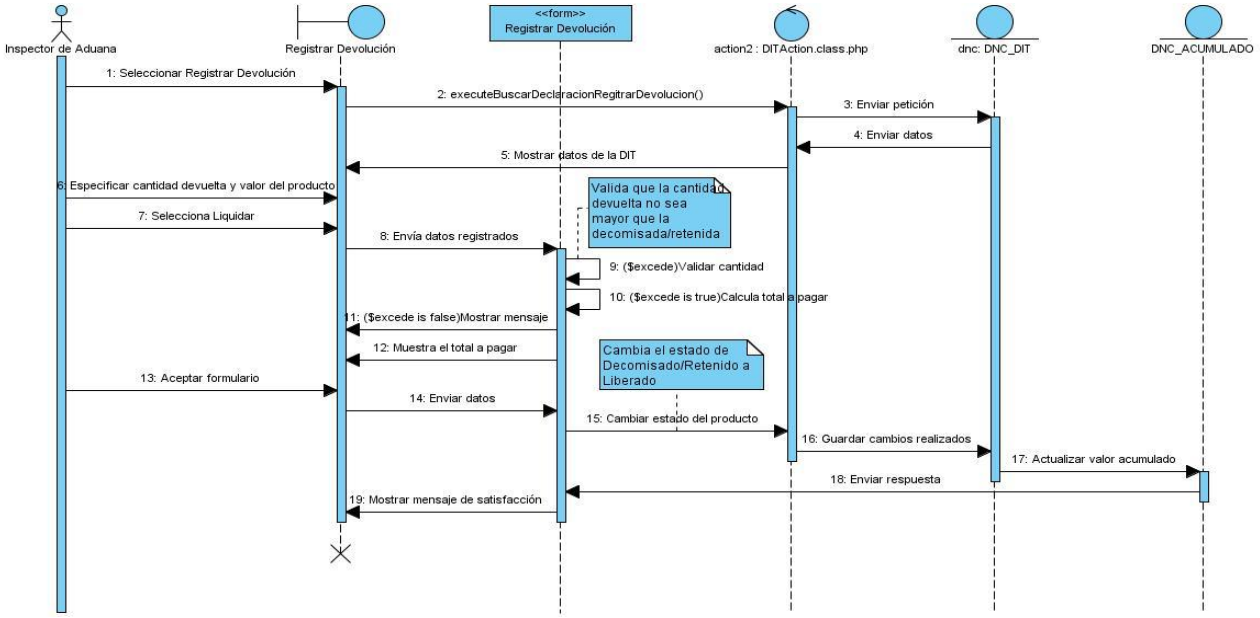

Figura 11: Registrar Devolución.

Anexo 12. Diagrama de Secuencia, Modificar DIT.

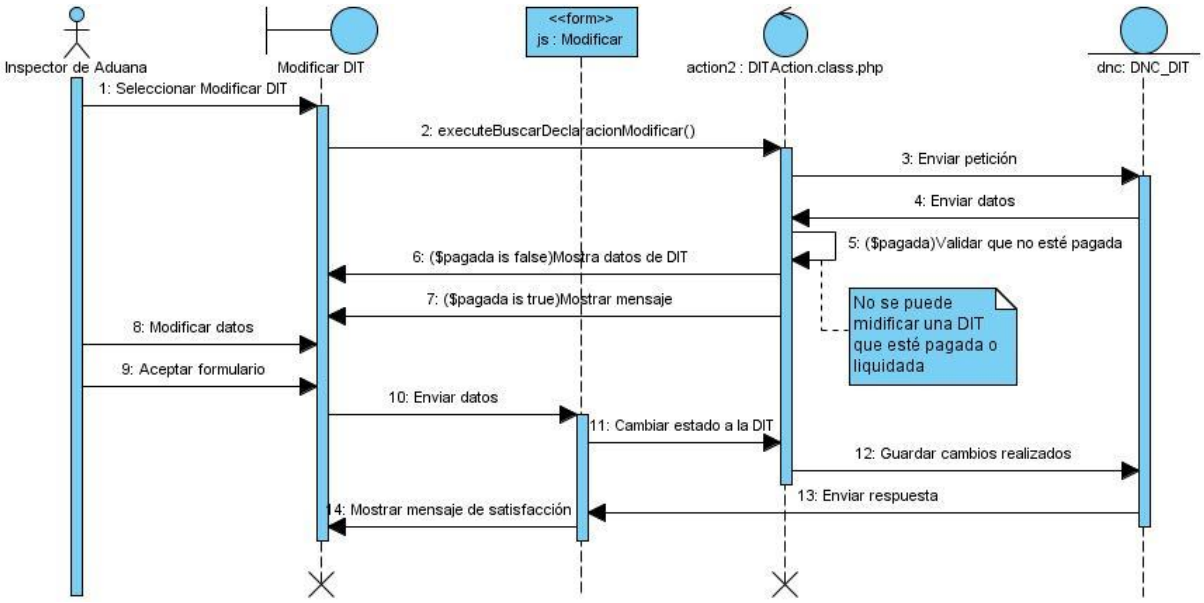

Figura 12: Modificar DIT.

Anexo 13. Diagrama de Secuencia, Cancelar DIT.

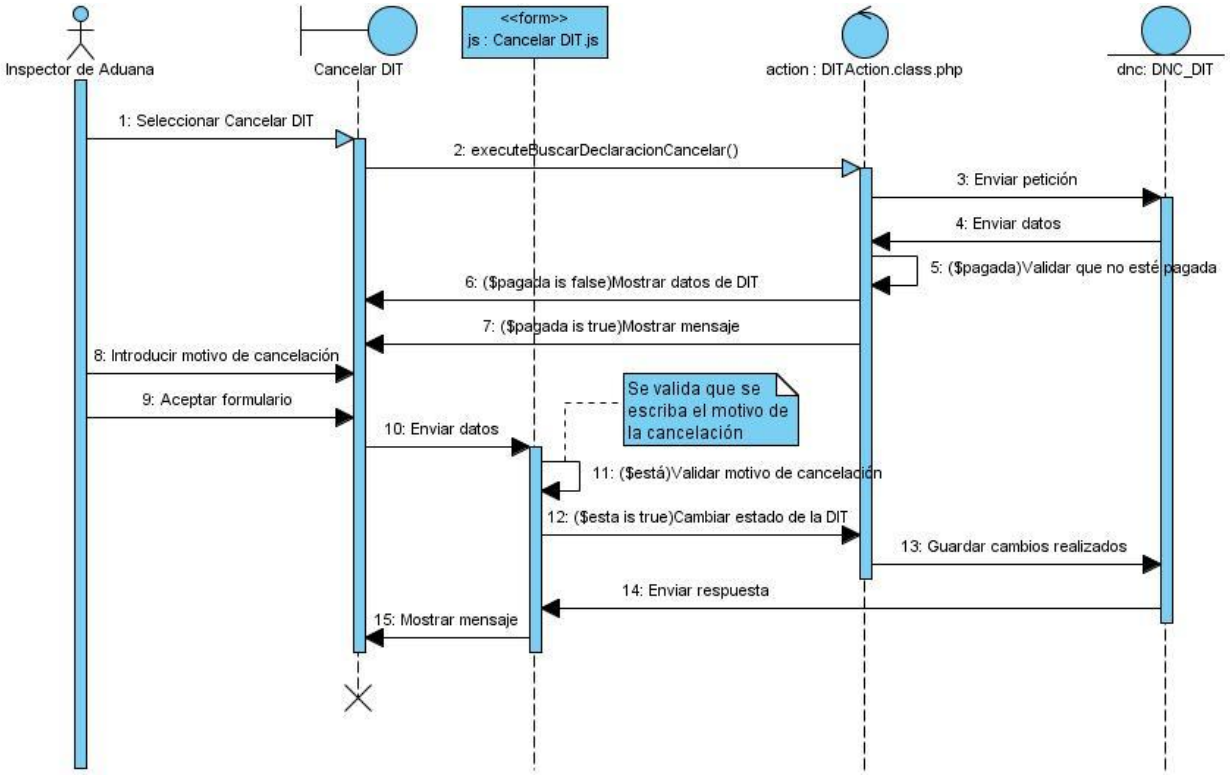

Figura 13: Cancelar DIT.