Universidad de las Ciencias Informáticas

Facultad 4

Tema: Diseño e Implementación del Módulo Medios de Transporte Internacional.

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

Autor: Roberto Carlos Garcia Andino.

Tutor: Ing. Alain Eduardo Rodríguez Arias.

Ciudad de la Habana, junio 2009

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

<Nombre y apellidos del o de los autores>

_____

Firma de Autor

<Nombre y apellidos del o de los tutores>

_____

Firma de Tutor

AGRADECIMIENTOS

*A mi familia, que me apoyó en todo momento.
*A Angel Gustavo Suárez Braña.
*A todos los profesores que me prepararon para servir a mi patria.
*A la Revolución Cubana que por su carácter Socialista nos brindó el derecho de prepararnos como profesionales universitarios.
* A los compañeros, profesores y amigos que compartieron su tiempo conmigo y me ayudaron para la realización de este trabajo.

DEDICATORIA

Este trabajo va dedicado a mis padres Carlos García Flores y Liliana Andino Elliot, a mis hermanos Albin Ariel y Jesús Arias, a mí querida abuela Emelina Flores que siempre confió en mí y a todos los colegas de 5 importantes años que cambiaron mi vida.

Roberto Carlos

"No hay ciencias aplicadas si no hay ciencia que aplicar" (Castro Díaz – Balart, 2001)

Bernardo Houssay

RESUMEN

En la Aduana General de la República se desarrollan varios procesos, entre ellos el Control y Despacho de los medios de transporte internacional, este proceso tiene como misión garantizar que el control y despacho de los medios de trasporte internacional cumpla con los requisitos y formalidades establecidas por la misma. Sin embargo, mediante este proceso se manejan grandes cantidades de datos de forma manual, lo que provoca que se prolongue considerablemente el procesamiento de los mismos. El presente trabajo ofrece un diseño e implementación como solución que puede cumplir con las necesidades reales de los clientes y usuarios finales y con los requisitos de la Aduana General de la República, para lo cual se presentan las clases del diseño web, diagramas de secuencia y despliegue y se pasa a implementar la solución propuesta, guiada por un estándar de codificación, diagrama de componentes y los artefactos que se generan como salida de la etapa de diseño. Para desarrollar las tareas mencionadas fue necesario guiar el proceso a través de una metodología de desarrollo de software y hacer uso de herramientas ya definidas para generar los artefactos que propone dicha metodología. Fue necesario también estudiar los artefactos generados de la etapa de análisis y la documentación que fue entregada por los clientes.

PALABRAS CLAVE AGR (Aduana General de la República de Cuba), MTI (Medios de Trasporte Internacional), SUA (Sistema Único de Aduanas).

TABLA DE CONTENIDOS

INTRODUCCIÓN

En la aduana general de la república de Cuba y en específico en las Aduanas Marítimas y Aéreas se realiza el proceso de control y despacho de los medios de transporte internacional, tanto para la entrada como para la salida de los mismos de las aduanas correspondientes. El inspector de aduana es el encargado de realizar este proceso, en el cual se engloban todas las acciones referentes al arribo y despacho tanto de entrada como de salida y a la prestación de servicios a los medios de transporte internacional que arriben a las aduanas marítimas y aéreas de nuestro país.

En la actualidad existen varios problemas que afectan las condiciones laborales para realizar un control y despacho con calidad y confiabilidad, pues no existe una entrega de información adelantada de despacho que cumpla con los requisitos precisados por la Aduana, lo cual provoca que el tiempo de procesamiento de los documentos ya sean manifiestos, declaraciones, notificaciones, certificados, autorizaciones, solicitudes, modelos, listados, entre otros, sea prolongado y que el control de las operaciones en las que intervienen los medios de transporte internacional sea más tedioso.

Para suplir estas necesidades un equipo de desarrollo del proyecto Sistema Único de Aduanas (SUA) comete un análisis profundo de las necesidades actuales de la Aduana General de la República (AGR) para el control y despacho de los Medios de Transporte Internacional (MTI), análisis que fue validado y arrojó como resultado los artefactos necesarios para alimentar el proceso de diseño, como fueron: requerimientos funcionales, diagrama de Casos de Uso y descripción de Casos de Uso.

A partir de esta situación, se ha tomado como **problema a resolver:** ¿cómo diseñar e implementar un subsistema de despacho de buques y aeronaves para el Sistema Único de Aduanas?

Basado en esto, se define como **objeto de estudio** los procesos: diseño e implementación de aplicaciones de gestión de información y despacho de Medios de Transporte Internacional y como **campo de acción** el subsistema de despacho de los MTI para el SUA de la Aduana General de la República de Cuba.

Para dar solución al problema se propone como **objetivo general**:

- Diseñar e implementar un sistema informático que automatice el proceso de control y despacho de buques y aeronaves.

Las **tareas de investigación** planificadas para dar solución al problema y cumplimiento al objetivo planteado son:

- Realizar un estudio sobre las tecnologías y herramientas a utilizar.
- Revisión bibliográfica de la documentación relacionada con el Módulo Medios de Transporte Internacional.
- Revisión de los casos de uso del sistema.
- Diseñar e implementar el módulo para la gestión del proceso de despacho de buques y aeronaves

Este trabajo está estructurado en tres capítulos, tal y como se describe a continuación:

Capítulo 1: Se enuncian los principales conceptos relacionados con el sistema a diseñar. Se realiza un estudio de los sistemas de información, un estudio del arte y se hace referencia a la metodología de desarrollo de software así como de las herramientas y el gestor de base de datos seleccionados para realizar este trabajo.

Capítulo 2: En este capítulo se realiza el diseño de la solución, se muestran los diagramas de clases del diseño con estereotipos Web, así como los diagramas de secuencia. También se construye el modelo físico de datos y el diagrama de despliegue, se especifica además la solución propuesta en el marco de la arquitectura definida para el Sistema Único de Aduanas.

Capítulo 3: Se expone el estándar de codificación a utilizar así como el diagrama de componentes y se procede a implementar la solución propuesta.

CAPÍTULO 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En el presente capítulo se realiza una revisión de los conceptos a los cuales se harán referencia en todo el contexto del trabajo, como es el caso de los términos MTI y SUA. Se brinda información acerca del proceso de control y despacho de los Medios de Transporte Internacional y se hace referencia a las herramientas que utilizaremos en el desarrollo del trabajo, además del paradigma de programación a seguir, base de datos y *framework** a utilizar.

**framework: en el desarrollo de software, es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado.*

1.2. Definiciones.

1.2.1. MTI

Medio de Transporte Internacional (MTI), es uno de los subsistemas con que cuenta el Sistema Único de Aduana (SUA), en el cual se realiza, específicamente, el control y despacho de los medios de transporte internacional, que son los buques, yates y las aeronaves.

1.2.2. SUA

La Aduana General de la República (AGR) cuenta en estos momentos con el Sistema Único de Aduanas (SUA) conformado por diferentes procesos. El SUA tiene como objetivo automatizar 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 y Viajeros y las Tablas de Control en ambiente WEB, debido a las facilidades que brindan estos servicios a los usuarios. SUA es un sistema en el cual todos los

módulos validan y controlan las entradas de datos contra los nomencladores y clasificadores (cien aproximadamente) que se diseñaron oportunamente, los cuales facilitan la flexibilidad del sistema además de tener organizada la información y así asegurar la consistencia de los datos.

1.2.3. Proceso de control y despacho de los medios de transporte internacional.

El control y despacho de los medios de transporte internacional, es el conjunto de acciones mediante las cuales se tramita ante la Aduana, por el mando del buque, la aeronave o su consignatario, la documentación que declara la carga, provisiones, pasajeros y tripulantes que transportan los mismos. El proceso de control se realiza tanto a la entrada como a la salida de los buques y aeronaves en las Aduanas Marítimas y Aéreas.

1.3. Las metodologías de desarrollo de software

Las metodologías de desarrollo de software surgieron a raíz de la necesidad de controlar y documentar proyectos cada vez más complejos, impulsadas principalmente por instituciones económicamente importantes, con requisitos sumamente estrictos de seguridad y fiabilidad en sistemas. La implantación de una metodología es necesaria si se quieren gestionar adecuadamente los proyectos.

El término de metodología se define como un conjunto de métodos eficientes, orientados a conseguir un objetivo propuesto. Son un conjunto de procesos que organizados dan una secuencia de pasos a seguir para obtener los hitos propuestos y finalmente el producto final. Como no existe una metodología de software universal, esta debe guiar a la organización en el desarrollo de sus objetivos. Por ello, la organización, su estructura, canales de información, recursos humanos y físicos, deben ser los adecuados y garantizar el correcto funcionamiento de los procesos y métodos definidos.

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar, inspirados por otras disciplinas de la ingeniería. En dependencia del capital humano del que se disponga, del tamaño del software que se quiera construir y el marco de tiempo de realización, se utilizará una metodología u otra. Sin embargo, hay algo que se debe tomar en cuenta: Las características de cada proyecto (equipo de desarrollo o recursos) exigen que la metodología sea configurable y flexible a sus necesidades.

Existen diversas propuestas metodológicas para llevar a cabo un proyecto de software. Estas propuestas se encuentran divididas en dos grupos fundamentales:

- las metodologías tradicionales o pesadas, las cuales se centran fundamentalmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán.
- las metodologías ágiles, las cuales se centran fundamentalmente en el factor humano y en el producto de software, dando mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas.

1.3.1. El Proceso Unificado de Desarrollo de software (RUP).

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Procesos, es un producto de Racional Software Corporación (IBM). Se caracteriza por ser:

- ✓ Iterativo e incremental, lo cual permite dividir el proyecto en pequeños subproyectos para desarrollarlo en distintas etapas e iteraciones que resultan en un incremento del producto.
- ✓ Centrado en la arquitectura, lo que permite organizar o estructurar el sistema en sus partes más relevantes e ir refinando esta estructura progresivamente
- ✓ Guiado por los casos de uso, el cual es uno de los métodos más utilizados y efectivos para reflejar los requisitos. Estos no solo sirven para especificar los requisitos, ellos son los encargados de guiar el ciclo de vida del proyecto.

Incluye artefactos que son los productos tangibles del proceso, como por ejemplo, el modelo de casos de uso, el código fuente y roles[2]. En la literatura se plantea que RUP es un proceso de desarrollo de software que describe un conjunto de actividades para transformar los requerimientos del cliente en un Sistema de Software (JACOBSON, 2000). Tiene como objetivo asignar tareas y responsabilidades para producir software de alta calidad, buscando satisfacer las necesidades de los clientes, ajustándose al presupuesto y a los tiempos estimados. RUP define “un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto” (JACOBSON, 2000).Utiliza el Leguaje Unificado de Modelado (Unified Modeling Lenguaje, UML), el cual los autores han seleccionado para visualizar, especificar

y construir los artefactos del sistema automatizado a definir. RUP divide el proceso de desarrollo en ciclos, teniendo un producto al final de cada uno, y estos se dividen en fases que finalizan con un hito donde se debe tomar una decisión importante:

Inicio: Determinar la visión del proyecto.
Elaboración: Determinar la arquitectura óptima.
Construcción: Obtener la capacidad operacional inicial.
Transmisión: Obtener la liberación del proyecto.

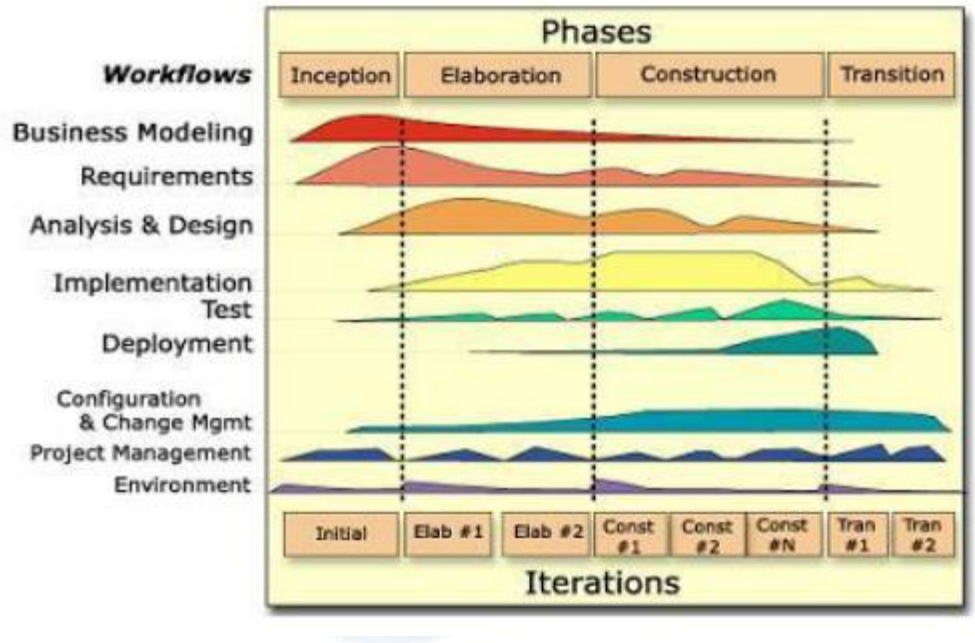


Figura 1. Fases e Iteraciones de la Metodología RUP

Según los criterios analizados, RUP no debe catalogarse como una metodología tradicional o ágil, sino que es una metodología a la medida entre las dos clasificaciones, debido a que tiene bien definido sus procesos orientados a objetos, puede ser menos pesado si se es capaz de aceptar y adaptar a las condiciones esperadas de cada proyecto. Además de traer consigo un grupo de beneficios como la estandarización, facilita el entendimiento por parte de los usuarios del software, que no necesariamente deben tener conocimientos previos sobre el tema y facilita además, el desarrollo del producto a distancia de los clientes.

1.4. Estado actual del Diseño de software

En la actualidad el concepto de diseño es orientado a diferentes aristas como la que plantea Grady Bosch: “El diseño es el proceso de determinar una implementación efectiva y eficiente que realice las funciones y tenga la información del análisis de dominio”, o la planteada por R.S. Pressman: “El diseño del software se encuentra en el núcleo técnico de la ingeniería del software y se aplica independientemente del modelo de diseño de software que se utilice. Una vez que se analizan y especifican los requisitos del software, el diseño de éste es la primera de las tres actividades técnicas - diseño, implementación y pruebas - que se requieren para construir y verificar el mismo…”

En el caso de las metodologías, como por ejemplo RUP, los criterios sobre el diseño de sistemas plantean que es el flujo del trabajo donde el arquitecto y los diseñadores de sistema, describen de una forma más concreta y acorde al tipo de tecnología, el cómo serán implementadas las funcionalidades del sistema en construcción, basándose en los tipos de colaboración, la relación que existe entre las clases, y las responsabilidades de cada unidad de código (RUMBAUGH, y otros, 1998.). El diseño debe ser lo más simple posible, el paradigma KISS ("Keep It Small and Simple" para unos o "Keep it Simple, Stupid" para otros) se lleva hasta las últimas consecuencias; por ejemplo, se hace énfasis en no añadir funcionalidad al diseño del sistema nunca antes de lo necesario, por las sencillas razones de que probablemente ahora mismo no sea lo más prioritario o porque quizás nunca llegue a ser necesaria (FOWLER, 2003).

Algunas empresas, como *DEADELUS*,* plantean que el diseño de sistemas se ocupa de desarrollar las directrices propuestas durante el análisis en términos de aquella configuración que tenga más posibilidades de satisfacer los objetivos planteados tanto desde el punto de vista funcional como del no funcional. El proceso de diseño de un sistema complejo se suele realizar de forma descendente: Diseño de alto nivel (o descomposición del sistema a diseñar en subsistemas menos complejos), diseño e implementación de cada uno de los subsistemas, especificación consistente y completa del subsistema de acuerdo con los objetivos establecidos en el análisis, desarrollo según la especificación, prueba, integración de todos los subsistemas y validación del diseño (DAEDALUS, 2006).

**DEADELUS: 1Empresa de software © DAEDALUS - Data, Decisions and Language, S. A*.

1.5. Aplicación de sistemas informáticos en el mundo.

1.5.1. Sistemas extranjeros.

Sistema Automatizado Aduanero Integral (SAAI)

Sistema automatizado que permite el control de las operaciones aduaneras en las 48 Aduanas de México desde 1993. Su control empieza con la autodeclaración electrónica de *Pedimientos** por parte de los agentes y apoderados aduanales y continúa hasta los procesos de entrada y/o salida de mercancías.

La información es concentrada en un sistema centralizado y se usa por diferentes entidades gubernamentales para generar las estadísticas nacionales de Comercio Exterior.

**Pedimiento: Definiremos al pedimiento aduanal como el documento en el cual se solicita a la autoridad aduanera se permita introducir o extraer mercancías al o del territorio nacional, que cumple con la función de ser una declaración de impuestos y derechos (impuestos al comercio exterior, IVA, IEPS, ISAN, DTA)*

Sistema Informático SOFI.

El Sistema Sofía, implementado por la dirección general de Aduanas del Paraguay tiene sus orígenes en Francia, cuando por el año 1976 se puso en marcha un Sistema Informático llamado SOFIA (Systéme d'Ordinateurs du Fret International Aérien) Sistema de Computación del Flete Internacional Aéreo, que como su nombre lo indica era una aplicación que cubría solamente las importaciones realizadas con flete aéreo.

Más tarde este sistema se perfecciona y pasa, de la importación con flete aéreo, a cubrir las importaciones con todo tipo de fletes, este Sistema se llamó SOFI.

1.5.2. Soluciones cubanas

Sistema automatizado para el control y despacho de Medios de Transporte Internacional.

La solución cubana adoptada para el control y despacho de los MTI está basada en programación estructurada, valido señalar que fue producido en nuestra universidad en 2007 y está integrado al Sistema Único de Aduanas.

1.6. Sistemas de información

Los sistemas de información proveen apoyo para las operaciones o servicios que realizan organizaciones o entidades para la sociedad, más que dirección y funciones administrativas, ellos apoyan actividades y procesos que son la razón para una existencia organizacional. Un sistema puede definirse simplemente como un grupo de elementos interrelacionados o que interactúan conformando un todo unificado. Sin embargo, el siguiente concepto genérico de sistema proporciona un marco más apropiado para describir los sistemas de información. Un sistema es un grupo de componentes interrelacionados que trabajan hacia una meta común, mediante la afectación de entradas y generando salidas en un proceso de transformación organizado. Este tipo de sistema (algunas veces llamado sistema dinámico) tiene los tres componentes con funciones básicas de interacción:

- **La entrada,** comprende la captura y el ensamblaje de elementos que entran al sistema para ser procesados. Por ejemplo, las materias primas, la energía, los datos y el esfuerzo humano, deben asegurarse y organizarse para el procesamiento.

- **El procesamiento,** incluye el proceso de transformación que convierte la entrada en salidas. Por ejemplo, un proceso de manufacturas, el proceso humano de respiración o los cálculos matemáticos.

- **La salida,** abarca la transferencia de elementos que han sido generados por un proceso de transformación hasta su destino final. Por ejemplo, los productos terminados, los servicios humanos y la información gerencial que debe entregarse a sus usuarios humanos.

El papel que juegan los sistemas de información (SI) es esencial en las operaciones eficientes, la gerencia efectiva, el éxito estratégico de empresas y otras organizaciones que deben operar un entorno empresarial global. Por tanto, los SI se han convertido en un área funcional importante de la administración de empresas.

1.6.1. Actividades de los Sistemas de Información:

O’Brien (O´Brien 2001), al igual que otros autores (Stair and Reynolds 2007), plantea que el funcionamiento de estos sistemas se basa en la planificación y el diseño de cuatro tipos de actividades:

Entrada de información: Es la alimentación del sistema o captura de datos de la organización en la que actúa y de su entorno. Constituye un proceso mediante el cual el SI toma los datos que requiere para procesar la información, las entradas pueden ser manuales o automáticas. Las manuales son aquellas que se proporcionan en forma directa por el usuario, mientras que las automáticas son datos o información que provienen o son tomadas de otros sistemas o módulos. Esto último se denomina interfaces automáticas.

Procesamiento de información: Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecidas. Estos cálculos pueden generarse con datos introducidos recientemente en el sistema o bien con datos que están almacenados. Esta característica de los sistemas permite la transformación de datos fuente en información que puede después ser utilizada por la empresa.

Almacenamiento de información: El almacenamiento es una de las actividades o capacidades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardada en la sección o proceso anterior. Esta información suele ser almacenada en estructuras de información denominadas archivos.

Salida de información: Es la capacidad de un sistema para sacar la información procesada o bien datos de entrada al exterior. Las unidades típicas de salida son las impresoras, terminales, diskettes, cintas magnéticas, la voz, los graficadores y los plotters, entre otros. Es importante aclarar que la salida de un Sistema de Información puede constituir la entrada a otro Sistema de Información o módulo.

1.6.2. Tipos de sistemas de información:

Conceptualmente los sistemas de información en el mundo real trabajan en 4 niveles indistintamente, que son, estratégicos, tácticos, técnico-operativos e interinstitucionales además de

que también pueden ser clasificados teniendo en cuenta los principales papeles que cada uno desempeña. Consideremos los siguientes tipos como los más tratados en el mundo empresarial:

a) Sistemas de procesamiento de transacciones.
b) Sistemas de información gerencial.
c) Sistemas de apoyo a decisiones.
d) Sistemas expertos e inteligencia artificial.
e) Sistemas de apoyo a decisiones de grupo.
f) Sistemas de apoyo a ejecutivos.

1.7. Herramientas

1.7.1. Eclipse Plataform 3.3.1.1

Eclipse es principalmente una plataforma de programación, usada para crear entornos integrados de desarrollo del (inglés IDE) de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido". El IDE de Eclipse emplea módulos (del 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. Adicionalmente al permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++, PHP y Python, permite a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistemas de gestión de bases de datos.

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*".

Eclipse dispone de las siguientes características:

- Editor de texto.
- Resaltado de sintaxis.
- Compilación en tiempo real.
- Control de versiones con CVS.
- Integración con Ant.
- Asistentes (*wizards*): para creación de proyectos, clases, etc.

- Integración con Hibernate.

1.7.2. Mozilla Firefox versión 3.0.8

Mozilla Firefox es un navegador web desarrollado por la Corporación Mozilla y un gran número de voluntarios externos. Es un navegador multiplataforma y está disponible en varias versiones de Microsoft Windows, Mac OS X, GNU/Linux y algunos sistemas basados en Unix. Su código fuente es software libre publicado bajo una triple licencia GPL/LGPL/MPL.

Principales características de Mozilla Firefox:

- Excelente navegación por pestañas.
- Marcadores dinámicos.
- Corrector ortográfico in time.
- Gestor de descargas integrado.
- Manejador de gran cantidad de extensiones.
- Herramientas de desarrollo web (un inspector DOM y una consola JavaScript) integradas.
- Bloqueo de pestañas emergentes.
- Personalización de temas.

Las extensiones de Firefox son pequeños programas que se le añaden y que expande sus capacidades, funcionalidad que casi ningún otro navegador posee. El uso de las extensiones posibilita a Firefox debuguear páginas web, manejar imágenes, música, videos y multimedia, entre otros.

Firefox está construido sobre el motor de dibujado Mozilla Gecko, uno de los más rápidos disponibles hoy en día. Cumple perfectamente los estándares del W3C (*del inglés*, *World Wide Web Consortium*) tales como: CSS (*del inglés*, *Cascading Style Sheets*), DOM (*del inglés, Document Object Model*), entre otros. Mientras que navegadores como Opera probablemente fallen en una página dinámica, Mozilla Firefox generalmente tiene éxito.

Actualmente existen numerosos lenguajes de programación web. Los lenguajes de programación del lado del cliente que indican al navegador donde colocar cada texto, imagen o video y la forma que tendrán estos al ser colocados en la página; y los lenguajes del lado del servidor ejecutados en el servidor web justo antes de que se envíe la página a través de Internet al cliente, permitiendo que se muestre en la misma los datos deseados por el servidor.

1.7.2. Visual Paradigm para UML 6.1 Enterprise Edition

Visual Paradigm es una plataforma de modelado poderosa y de fácil uso que soporta el ciclo de vida completo del desarrollo de software de: análisis y diseño orientados a objetos, construcción, prueba y despliegue. Está compuesta por varios productos como: Visual Paradigm for UML, Smart Development Environment, DB Visual ARCHITECT, Business Process Visual ARCHITECT, Visual Paradigm Suite, DB Visual ARCHITECT SQL, Teamwork Server; en sus diferentes ediciones: Enterprise, Professional, Standard, Modeler, Personal y en su versión libre Community. Posee una intuitiva interfaz de usuario. Permite dibujar todos los tipos de diagramas de clases, posibilitando que la creación de los diagramas o representaciones sea mucho más rápida que en otras herramientas CASE.

Visual Paradigm posee elementos que la posicionan entre las herramientas más usadas en el mundo entre ellos: el soporte para UML 6.1, el modelado de procesos de negocio, la administración de requisitos, el modelado de bases de datos con la creación de diagramas Entidad Relación (ER), entre otros.

Es capaz de integrarse con diferentes plataformas de desarrollo de aplicaciones como Visual Studio.Net (Microsoft), Eclipse, NetBeans, JDeveloper, JBuilder, Suntm One, IntelllJ Ideatm, WebLogic Workshoptm; así como con distintos gestores de bases de datos como: MySql, PostgreSQL, MS SQLServer, Informix, DB2, Oracle y otros.

1.7.3. Servidor HTTP Apache

El **servidor HTTP Apache** es un servidor web de código abierto para plataformas Unix, Windows y Macintosh que implementa el protocolo HTTP/1.1(1999) y la noción de sitio virtual. Su nombre se lo debe a Behelendorf quien lo eligió porque quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU. Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. Tiene amplia aceptación en la red desde 1996, alcanzó su máxima cuota de mercado en 2005 siendo el servidor más empleado en un 70% de los sitios web en el mundo.

Entre las ventajas que proporciona están:

- Modular
- Open source (Código abierto)
- Multi-plataforma
- Extensible
- Popular (fácil conseguir ayuda/soporte)

1.7.4. Embarcadero ERStudio 7.5

Asistente fácil de usar para la construcción de esquemas XML directamente desde un modelo de datos físico o lógico, permitiendo a los arquitectos de datos incorporar los mismos estándares en el desarrollo SOA(Microsoft, 2006) que ellos utilizaron previamente para la construcción de las bases de datos.

Posee una utilería para el nombramiento de estándares y un editor para el mapeo de diferentes tipos de datos que automáticamente transforma el modelo de metadatos, permitiendo a los arquitectos de datos ser más productivos para mover modelos entre las capas de los diseños lógicos y físicos.

1.8. Paradigma de programación

1.8.1. Programación Orientada a Objetos (POO)

Tal como los paradigmas afectan la vida de las personas en muchos sentidos, los paradigmas de programación afectan en buena medida la forma de programar, y por tanto de desarrollar software de muchos ingenieros y programadores. Entre los diferentes paradigmas que existen, encontramos el Paradigma de La Programación Orientada a Objetos (POO) el cual se enfoca en la identificación de entidades (en el sistema a considerar), su estructura, clasificación y comportamiento dentro del sistema, para lo cual es necesario identificar:

- Objetos
- Clases
- Abstracción
- Encapsulación

- Modularidad
- Jerarquización
- Tipificado
- Concurrencia y Persistencia

Por lo tanto la POO no es en sí una forma de programar, no es un lenguaje de programación, más bien es todo un Paradigma que se orienta a la localización de entidades, su clasificación y su funcionamiento dentro del sistema analizado.

1.8.2. Comparación del paradigma de la POO vs otros paradigmas

El paradigma de la programación orientada a objetos difiere de las tradiciones históricas de la programación procedimental, esta última se centra en los datos y procedimientos, sin limitaciones acerca de los procedimientos que pueden actuar sobre dichos datos.

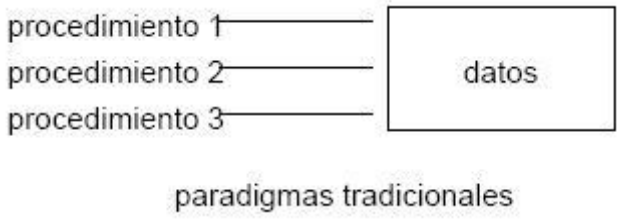


Fig. 1.1

En el paradigma de la orientación a objetos los programas son conjuntos de una única entidad básica, el objeto, el cual combina los datos con los procedimientos que actúan sobre ellos, además los objetos reciben las peticiones e interactúan enviando mensajes a cada uno de los demás.

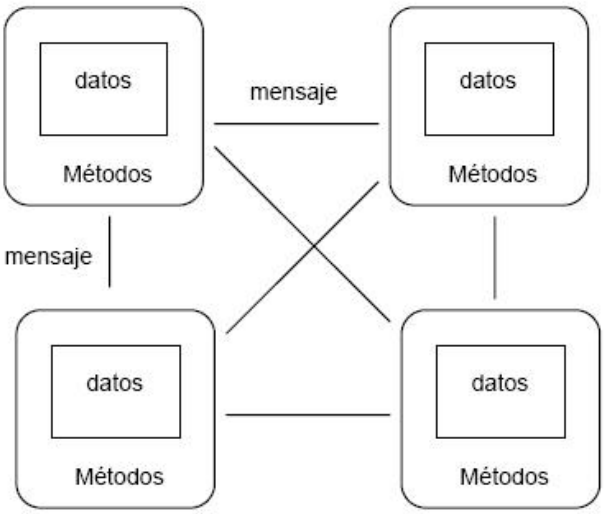


Fig. 1.2

- Con el paradigma procedimental es más difícil modelar el mundo real.
- Con el paradigma orientado a objetos se beneficia al desarrollo del software proporcionando una forma natural de modelar el fenómeno del complejo mundo real. Esto ofrece realmente un modelo más natural del mundo real.
- Las características comunes entre dos entes quedan explícitas en la POO, mientras que en la Programación tradicional no es así.
- Los lenguajes orientados a objetos benefician el desarrollo de software modular, esto quiere decir que el programador no necesita examinar todo el código para ver si los cambios locales causarán problemas en cualquier otra parte del sistema. Por el contrario, con los lenguajes tradicionales, siempre existe la posibilidad de que una actualización de una subrutina o elemento de datos afecte a alguna rutina que se halle físicamente muy alejada de la actualización.
- El apartado anterior implica que se facilita la reutilización del código, creando una biblioteca de clases que están perfectamente probadas.

- La herencia, que no tienen los lenguajes tradicionales, es una de las características más importantes y más potentes de la programación orientada a objetos, le permite a los programadores crear nuevas clases programando solamente las diferencias con las clases "padres".
- El encapsulamiento facilita la comunicación entre programadores.

1.9. Sistemas gestores de base de datos

La propia evolución de la informática en todos sus aspectos fue abaratando costes y simplificando procesos que de forma manual resultaban muy engorrosos; el servidor de Base de Datos es una muestra de uno de estos trabajos para evitar el desgaste del hombre, este nos permite tener almacenada nuestra información manteniendo su integridad y protección para la manipulación de la misma. **Sistema Gestor de base de Datos (SGDB)**

Un Sistema Gestor de Base de Datos es el software que posibilita la gestión de la información por parte de los usuarios, almacenada en una o varias Bases de Datos que constituyen un conjunto exhaustivo y no redundante de datos, estructurados de forma independiente a su implantación o utilización en la máquina, accesible en tiempo real y compatible con usuarios recurrentes y sus respectivas necesidades de información. Su objetivo fundamental es el de suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, de manera que no sea necesario conocer el modo de almacenamiento de ellos en la PC, ni el método de acceso empleado.

Un SGBD tiene los siguientes objetivos específicos:

- Independencia de los datos y los programas de aplicación
- Minimización de la redundancia
- Integración y sincronización de las bases de datos
- Integridad de los datos
- Seguridad y protección de los datos
- Facilidad de manipulación de la información
- Control centralizado

1.9.1. MySQL

MySQL, es un sistema para la administración de bases de datos relacional (RDBMS) rápido y sólido creado por la empresa sueca MySQL AB. Está disponible desde 1996, pero su nacimiento se remonta a 1979. Ha obtenido el galardón Choice Award del Linux Journal Readers en varias ocasiones.

Es un sistema gestión de base de datos multiusuario y de subprocesamiento múltiple. Utiliza SQL el lenguaje estándar para la consulta de bases de datos utilizado en todo el mundo. MySQL es licenciado bajo la GNU GPL (*del inglés, GNU General Public License*). MySQL AB distribuye una versión comercial, que en lo único que se diferencia de la versión libre, es en el soporte técnico que se ofrece y la posibilidad de integrar este gestor en un software propietario, ya que de otra manera, se vulneraría la licencia GPL.

Se puede utilizar en una gran cantidad de sistemas Unix diferentes así como en Microsoft Windows. Presenta APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl. El sistema de privilegios y contraseñas es muy flexible y seguro, permite verificación basada en el host. La interfaz para el conector ODBC (MyODBC) proporciona a MySQL soporte para programas clientes que usen conexiones ODBC.

1.9.2. PostgreSQL

PostgreSQL es un sistema gestor de bases de datos objeto-relacionales (ORDBMS) libre, desarrollado originariamente en el Departamento de Ciencias de Computación de la Universidad de California en Barkeley.

Soporta casi toda la sintaxis SQL y ofrece muchas características modernas tales como: consultas complejas, integridad referencial (foreign keys), triggers, vistas (views), integridad transaccional, control de concurrencia multi-versión. También soporta almacenamiento de objetos grandes (imágenes, sonido y video).

Ofrece otras funcionalidades importantes tales como:

- **Savepoints:** permite hacer un roll back sin tener que repetir la transacción entera.
- **Point in Time Recovery:** permite salvar el estado de la DB en momentos concretos, para su posterior recuperación.

- **Tablespaces:** permite destinar discos físicos a un índice o a una tabla concreta.
- **Improved Memory and I/O:** optimizaciones en la velocidad de ejecución y en el consumo de memoria de la aplicación.

Cuenta con herramientas gráficas como PgAdmin, phpPgAdmin, las cuales hacen que la administración de la base de datos sea sencilla.

Debido a su licencia libre PostgreSQL puede ser utilizado, modificado y distribuido por todo el mundo de forma gratuita para cualquier fin, ya sea privado, comercial o académico.

1.9.3. Oracle

Oracle es un manejador de base de datos relacional que hace uso de los recursos del sistema informático en todas las arquitecturas de hardware, para garantizar su aprovechamiento al máximo en ambientes cargados de información.

Es el conjunto de datos que proporciona la capacidad de almacenar y acudir a estos de forma recurrente con un modelo definido como relacional. Además es una suite de productos que ofrece una gran variedad de herramientas. Oracle es el mayor y más usado Sistema Manejador de Base de Dato Relacional (RDBMS) en el mundo.

1.10. Lenguajes.

1.10.1. PHP 5

Es un lenguaje para programar scripts del lado del servidor, gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. PHP es un lenguaje encapsulado dentro de los documentos HTML de forma que se pueden introducir instrucciones PHP dentro de las páginas, debido a esto, el diseñador gráfico de la Web puede trabajar de forma independiente al programador. Una Web dinámica con PHP contiene una serie de documentos de este tipo que el servidor Apache interpreta proporcionando al cliente documentos HTML.

¿Por qué utilizar PHP?

- PHP corre en (casi) cualquier plataforma utilizando el mismo código fuente.

- La sintaxis de PHP es similar a la del C.
- Puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix, PostgreSQL, y otros muchos.
- Se pueden hacer grandes cosas con pocas líneas de código.
- El código PHP es mucho más legible que el de otros lenguajes.
- Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor, acceso a base de datos, encriptación, envío de correo y otros.
- Al poderse encapsular dentro de código HTML se puede recoger el trabajo del diseñador gráfico e incrustar el código PHP posteriormente.
- Es multiplataforma, funciona en todas las plataformas que soporten Apache.
  Es software libre. Se puede obtener en la Web y su código está disponible bajo la licencia GPL**.**

1.10.2. JavaScript

Brendan Eich, un programador que trabajaba en Netscape, adaptó otras tecnologías existentes (como ScriptEase) al navegador Netscape Navigator 2.0, que iba a lanzarse en 1995 y denominó inicialmente a su lenguaje LiveScript. Posteriormente, Netscape firmó una alianza con Sun Microsystems para el desarrollo del nuevo lenguaje de programación. Justo antes del lanzamiento Netscape decidió cambiar el nombre por el de JavaScript.

JavaScript se utiliza fundamentalmente para el desarrollo de páginas web dinámicas; es un lenguaje interpretado, es decir, que no requiere compilación y su sintaxis es semejante a la de los lenguajes Java y C.

El lenguaje está basado en objetos y no orientado a estos, el código se integra junto a HTML brindando diferentes efectos para la interacción con los usuarios. Los navegadores más populares como Internet Explorer, Mozilla Firefox, Opera, Netscape, entre otros interpretan el código JavaScript integrado dentro de las páginas web.

1.11. Frameworks

Los Frameworks son desarrollados con el objetivo de brindarles a los programadores y diseñadores una mejor organización y estructura a sus proyectos.

Los Frameworks ayudan en el desarrollo de software, proporcionan una estructura definida la cual ayuda a crear aplicaciones con mayor rapidez. Ayuda a la hora de realizar el mantenimiento del sitio gracias a la organización durante el desarrollo de la aplicación.

Los Frameworks simplifican el desarrollo de las aplicaciones mediante la automatización de muchas de las tareas comunes. Además, un Frameworks proporciona estructura al código fuente, forzando al programador a crear código más legible y más fácil de mantener.

1.11.1. Symfony: Framework para el desarrollo de aplicaciones Web con PHP.

Symfony fue diseñado con el objetivo de optimizar la creación de las aplicaciones web, con el uso de sus características. Es un enorme conjunto de herramientas y utilidades que simplifican el desarrollo de las aplicaciones web. Posee una librería de clases que permiten reducir el tiempo de desarrollo. Está desarrollado en PHP5, se puede utilizar en plataformas Unix, Linux y Windows, e incorpora el patrón MVC para separar las distintas partes que forman una aplicación web, soporta AJAX, plantillas y un gran número de bases de datos. Symfony es uno de los frameworks PHP más populares entre los usuarios y las empresas, ya que permite que los programadores sean mucho más productivos a la vez que crean código de más calidad y más fácil de mantener. Symfony es maduro, estable, profesional y está muy bien documentado.

1.11.2. ExtJS 2.2.1

ExtJS es una librería de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Originalmente construida como una extensión de la biblioteca YUI (*del inglés*, *Yahoo User Interface*).

Permite construir aplicaciones complejas en Internet, incluye componentes de interfaz de usuario de alto performance y personalizable, un modelo de componentes extensibles y un manejador de layouts similar al que provee Java Swing.

EXTJS incluye conexión asincrónica mediante Ajax con el servidor web, permitiendo que la aplicación web no esté sujeta a la acción del usuario y dando la libertad de cargar información sin que el cliente se dé cuenta.

1.12. CONCLUSIONES PARCIALES.

En este capítulo se demuestra la importancia que tiene para la Aduana cubana un software que ayude al despacho de los MTI. Además se estudian los sistemas extranjeros que trabajan con un objetivo similar, de los cuales se toman ideas que se implantarán en la solución propuesta, de forma tal que el producto cuente con opciones similares a las de sus competidores en el mundo. Como herramienta de modelado se utilizará el Visual Paradigm, se implementará mediante los lenguajes de programación web: JavaScript y PHP, el servidor web será Apache, el navegador Mozilla Firefox y el sistema gestor de base de datos Oracle.

CAPÍTULO 2

CAPÍTULO 2: DISEÑO DE LA SOLUCIÓN

2.1. Introducción

En el presente capítulo se describe el diseño del módulo Medios de Transporte Internacional para el Sistema Único de Aduanas. En el mismo se abordan los patrones de diseño utilizados para la solución de la aplicación, se presentan los diagramas de clases del diseño con estereotipos Web. Se presenta el diseño de la base de datos del sistema y como se adapta el módulo MTI a la arquitectura definida para el SUA.

Los autores I. Jacobson, G. Booch y J. Rumbaugh en el libro El Proceso Unificado de Desarrollo de Software, plantean: “En el diseño modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos incluyendo los requisitos no funcionales y otras restricciones que se le suponen.” (JACOBSON, 2000)

2.2. Patrones de diseños utilizados

El desarrollo del sistema utilizando el framework Symfony proporciona ventajas significativas para los desarrolladores de software. El marco de trabajo mencionado es capaz de fusionar buenas prácticas de trabajo por sí mismo, de forma que los desarrolladores no tengan que preocuparse por implementar varios de los patrones de diseño y arquitectónicos más utilizados en la actualidad, ya que el mismo framework los implementa.

2.2.1. Patrones GRASP implementados.

Creador

La clase actions.class.php contiene las acciones definidas para el módulo Medios de Transporte Internacional y es en ella misma donde se ejecutan las funciones que hacen al sistema funcional. En esta clase las acciones se encargan de crear los objetos de las clases que representan las

entidades, evidenciando de este modo que la clase actions.class.php es el "*creador*" de las entidades.

Controlador

Todas las peticiones Web son manejadas por un solo controlador frontal (despachoMTI_dev.php), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

Experto

Se evidencia este patrón puesto que Propel es la librería externa que utiliza Symfony para realizar su capa de abstracción al modelo de datos, encapsulando toda la lógica de los datos y generando las clases con funcionalidades comunes de las entidades. Por tanto cada clase creada por Propel a partir de una entidad es experta en manejar su información.

Alta Cohesión

Symfony permite la asignación de responsabilidades con alta cohesión, por ejemplo la clase actions.class.php tiene la responsabilidad para definir las acciones sobre las plantillas y colabora con otras para realizar diferentes operaciones y crear objetos, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas, proporcionando que el software sea flexible frente a grandes cambios.

2.2.2. Patrones GOF utilizados.

Singleton (Instancia única) Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Es el caso del controlador frontal, donde hay una llamada a la función sfContext::getInstance() que garantiza que siempre se acceda a la misma instancia.

Decorator (Decorador): Añade funcionalidad a una clase, dinámicamente. El archivo layout.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout decorando la misma.

Abstract Factory (Fábrica abstracta) Se utiliza este patrón al trabajar con objetos de distintas familias de manera que no se mezclen entre sí, haciendo transparente el tipo de familia concreta

que se esté usando. Cuando el framework necesita, por ejemplo crear un nuevo objeto, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

2.2.3. Patrón Modelo Vista Controlador (MVC).

El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador. Modelo: Es la representación de la información que maneja la aplicación. El modelo en sí son los datos puros que puestos en contexto del sistema proveen de información al usuario y a la aplicación misma.

Vista: Es la representación del modelo en forma gráfica, disponible para la interacción con el usuario. En el caso de una aplicación Web, la "Vista" es una página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

Controlador: Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el modelo en caso de ser necesario.

Contenido de cada capa en Symfony:

La capa del Modelo:

- Abstracción de la base de datos
- Acceso a los datos

La capa de la Vista:

- Vista
- Plantilla
- Layout

La capa del Controlador:

- Controlador frontal
- Acción

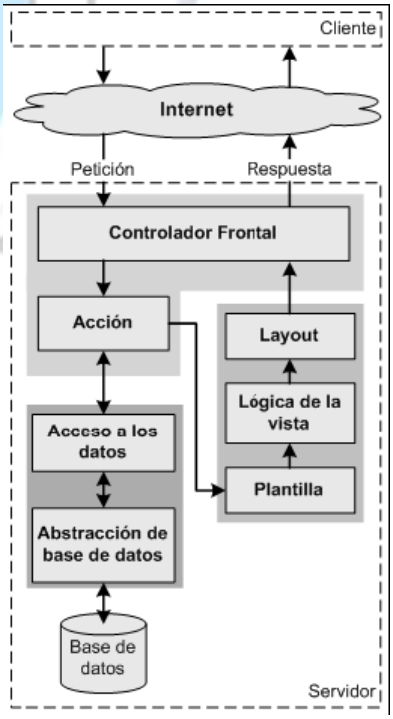


Fig. 2.1 Flujo de trabajo de Symfony

Ventajas y desventajas del MVC

Entre las principales ventajas se encuentran:

- La separación del Modelo de la vista, es decir, separar los datos de la representación visual de los mismos.
- Es mucho más sencillo agregar múltiples representaciones de los mismos datos.
- Facilita agregar nuevos tipos de datos según sea requerido por la aplicación ya que son independientes del funcionamiento.
- Facilita el mantenimiento en caso de errores.
- Ofrece maneras más sencillas para probar el correcto funcionamiento del sistema.

Como desventajas

- La separación de conceptos en capas agrega complejidad al sistema.
- La cantidad de archivos a mantener y desarrollar se incrementa considerablemente.
- El aprendizaje del patrón es más lento que otros modelos más sencillos.

2.3. Clases de diseño.

"Una clase del diseño es una abstracción sin costuras de una clase o construcción similar en la implementación del sistema…" (JACOBSON, 2000)

2.3.1. Extensiones para el diseño Web.

Las extensiones UML para el diseño Web, exponen una solución para el modelado de diagramas de clases de diseño sobre tecnologías Web. A continuación quedan presentados los siguientes estereotipos a utilizar (Macías, y otros):

Estereotipos para las clases		
Estereotipos	**Imagen**	**Descripción**
Página Servidora		Representa una página Web dinámica que contiene el código ensamblado por el servidor cada vez que se solicita. Típicamente, una página servidora contiene scripts que se ejecutan en el servidor y que actúan recíprocamente con los recursos del lado del servidor estos son: las bases de datos, los componentes de la lógica del negocio, sistemas externos, y así sucesivamente.

Página Cliente		Representa una instancia de una página cliente. Es una página Web con formato HTML. Las páginas cliente son interpretadas y mostradas por los navegadores del cliente y además pueden contener scripts que se interpretan en el navegador.
Formulario		Representa un formulario (elemento encargado de realizar envíos a las páginas servidoras)
Estereotipos para las relaciones entre las clases		
<<link>>		Representa un apuntador desde una "página cliente" hacia una "página cliente" o "página servidora". Corresponde directamente con una etiqueta <a> (ancla) de HTML.
<<submit>>		Esta relación siempre se da entre un "formulario" y una "página servidora", por supuesto, la "página servidora" procesa los datos que el "formulario" le envía.
<<build>>		Sirve para identificar cual(es) "página(s) servidora(s)" son responsables de de la creación de una "página cliente". Una "página servidora" puede crear varias "páginas cliente", pero una "página cliente" sólo puede ser creada por una sola "página servidora". Esta relación siempre es unidireccional.
<<redirect>>		Esta es también una relación unidireccional que indica que una página Web redirige hacia otra.
<<include>>		Asociación direccional desde una "página servidora" a otra "página servidora" o "página cliente". Esta asociación indica que las páginas incluidas son procesadas mientras que la página se ensambla

Tabla 2.1. Estereotipos y especificaciones de estereotipos para diseño Web.

Relaciones que se establecen entre las clases que conforman la extensión UML para Web.

Hasta/Desde	Página Servidora	Página Cliente	Formulario
Página Servidora	<<redirect>>	<<build>>,<<redirect>>	---
Página Cliente	<<link>>,<<redirect>>	<<link>>,<<redirect>>	contiene
Formulario	<<submit>>	Agregado por	---

Tabla 2.2. Relaciones entre clases que conforman la extensión UML para diseño Web.

2.3.2. Clases del diseño con estereotipos Web.

Una *clase de diseño* “completa”, es aquella suficientemente detallada que sirve como base para generar código fuente. Una clase de diseño es una clase cuya especificación es completa hasta un nivel que se pueda implementar. (Chuecko, 2008)

Caso de Uso: Captar Mensaje

La utilización del framework Symfony y el uso de las librerías ExtJS permiten desarrollar un diseño con una estructura similar para varios casos de uso. La página servidora actions.class.php se encargará de las peticiones que realiza el controlador frontal; luego estas se redireccionan al Layout. El Layout carga en el cuerpo la plantilla (en el presente caso por ser la plantilla una página en blanco no se presenta en el diagrama), luego se construye la página cliente que importa todas las interfaces de usuario del módulo, estas interfaces están definidas en el fichero de configuración view.yml y se encuentran representadas dentro del paquete Interfaces JS. Ver Figura 2.6.

En el caso de uso Captar Mensaje se tiene una relación de agregación entre la página cliente “CP_CaptarMensaje” y los formularios “frm_captarMsgBuque” y “frm_captarMsgAeronave”, que contienen los componentes necesarios para la solución. Luego los datos entrados son enviados al servidor y recibidos por las funciones de la clase “MTI_Actions.class.php”, las cuales mediante las clases del paquete de Acceso a Datos son capaces de registrar y obtener información. Ver Figura 2.2.

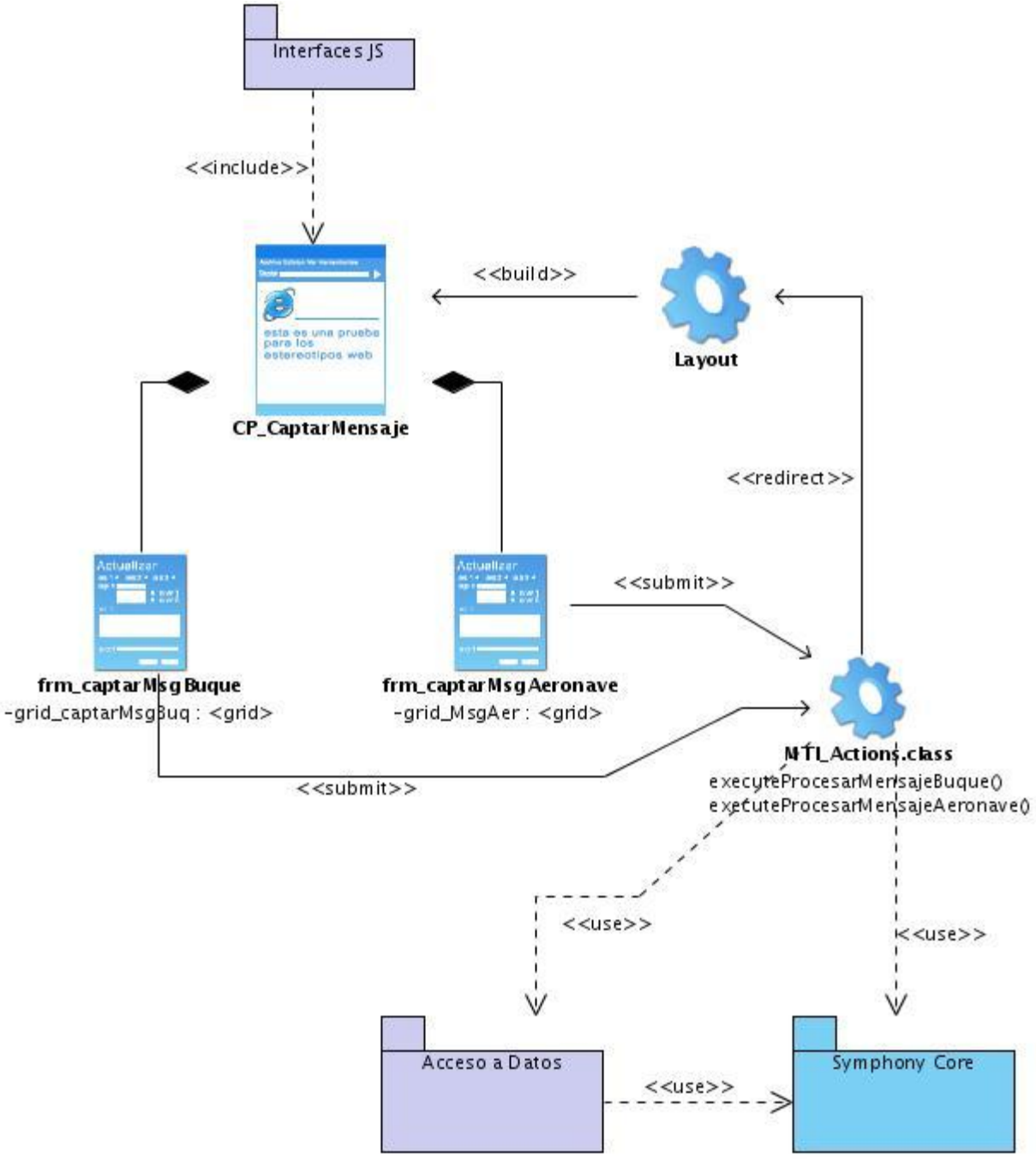


Figura 2.2. Diagrama de clases para el CU Captar mensaje.

Caso de Uso: Registrar Arribo.

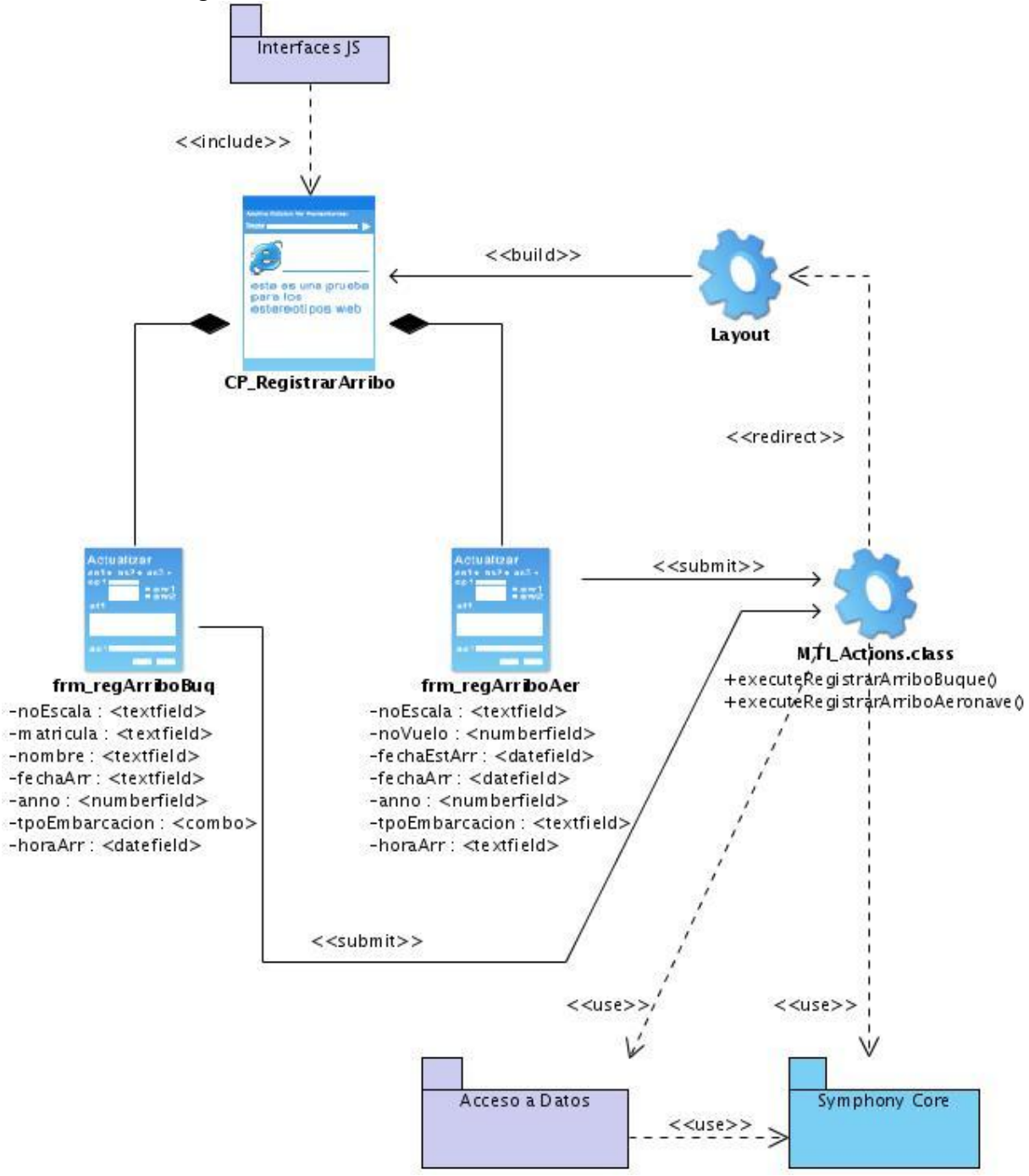


Figura 2.3. Diagrama de clases para el CU Registrar Arribo.

Caso de Uso: Servicios Prestados.

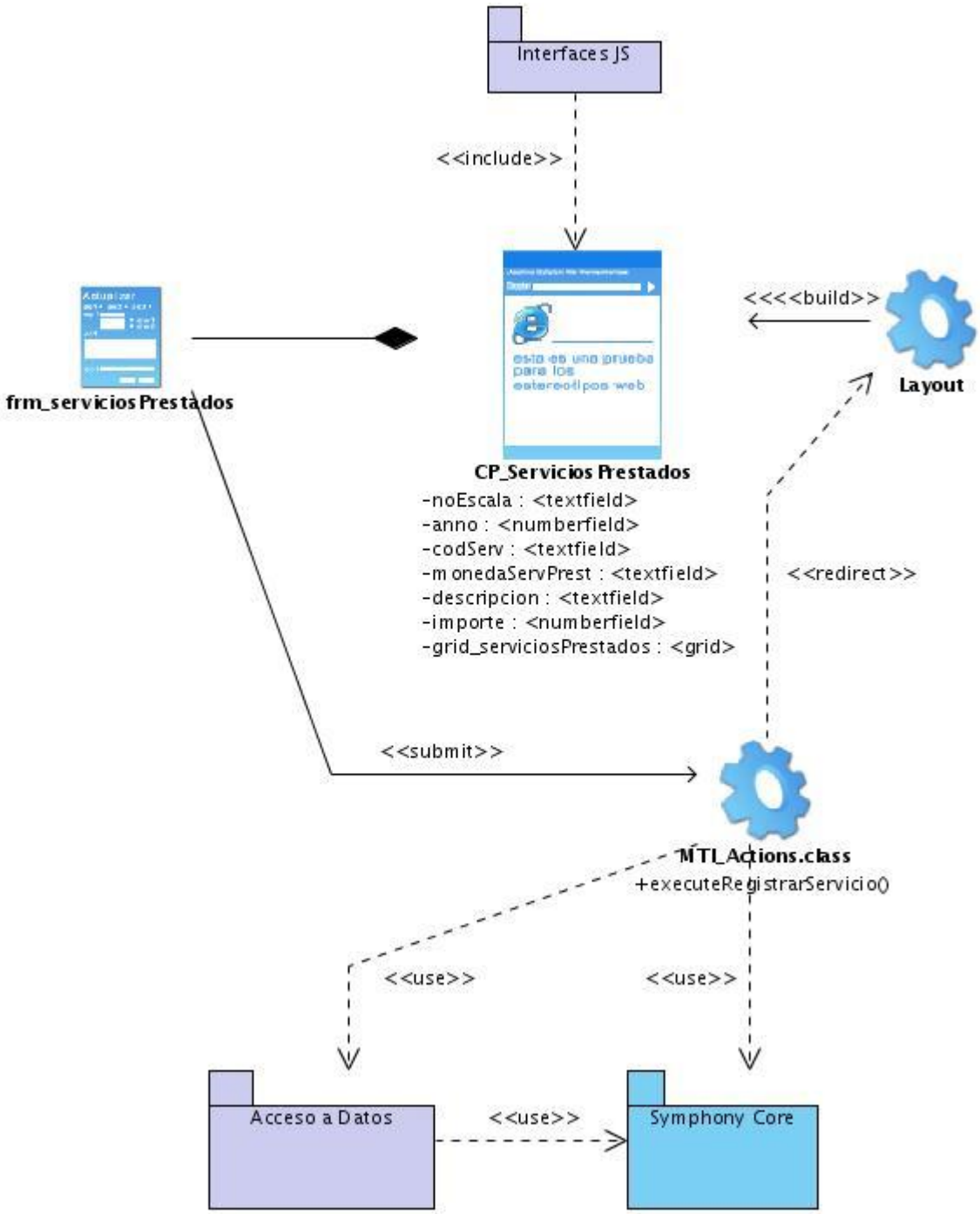


Figura 2.4. Diagrama de clases para el CU Servicios Prestados.

Caso de Uso: Realizar despacho.

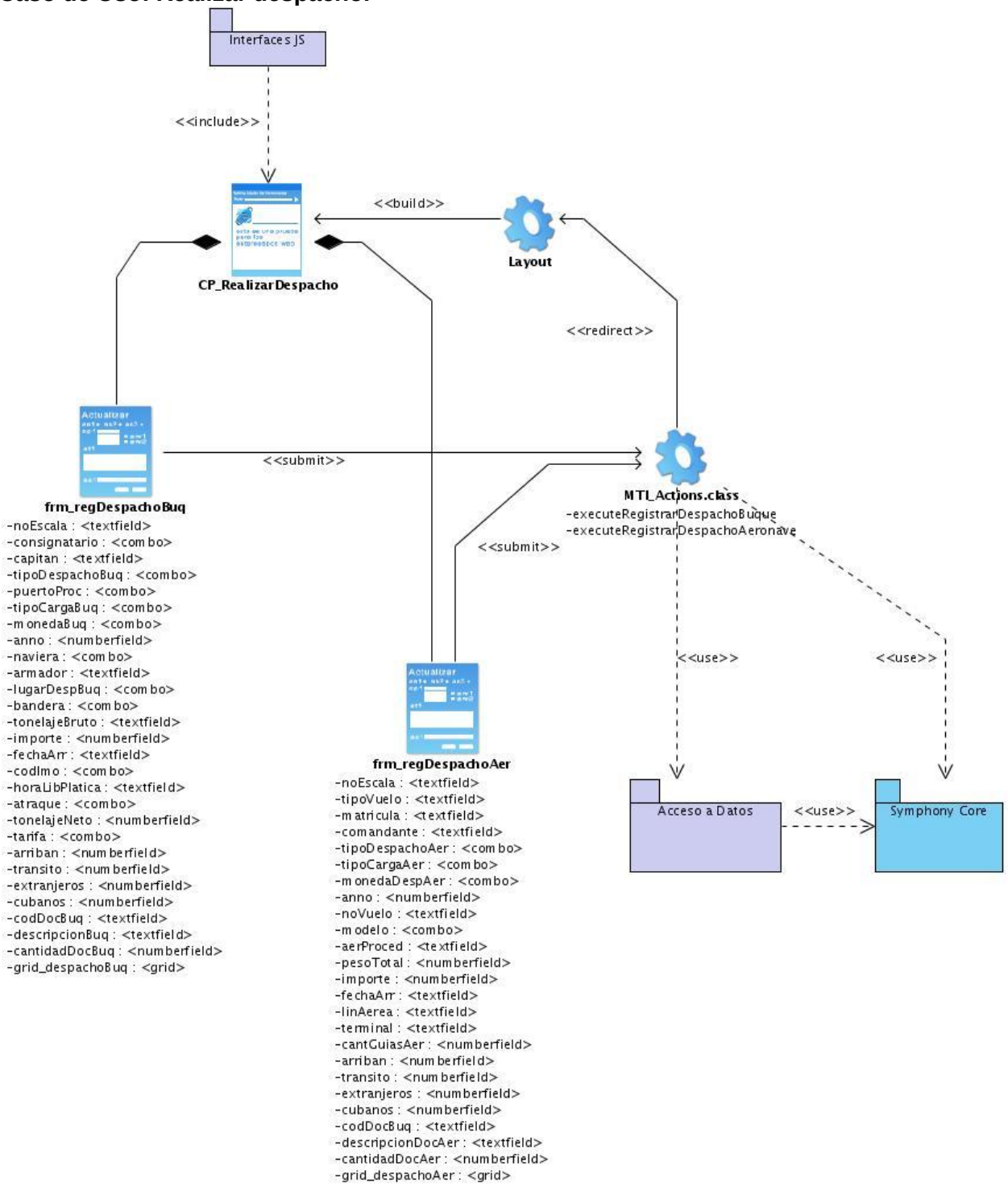


Figura 2.5. Diagrama de clases para el CU Realizar despacho.

2.3.3. Paquetes de interfaces JS y acceso a datos.

El paquete “Interfaces JS” contiene los códigos Java Script que son ensamblados por el navegador una vez que la página cliente se está interpretando. Ver Figura 2.6.

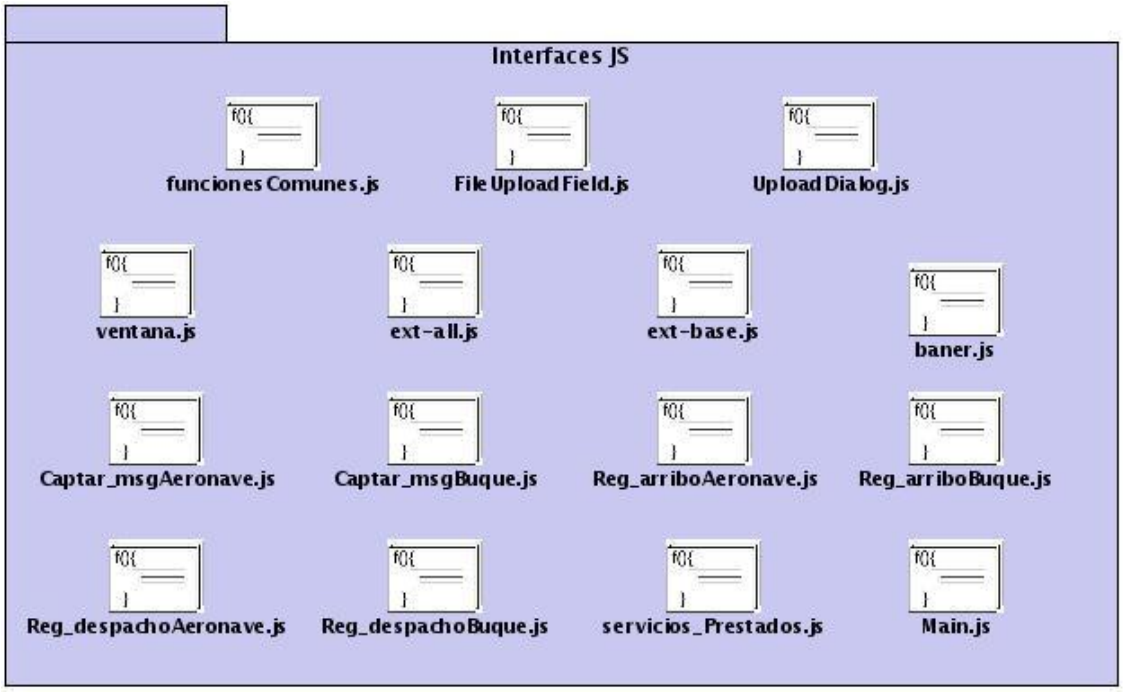


Figura 2.6. Paquete de interfaces JS.

Los script Ext-base.js y Ext-all.js representan el núcleo de las librerías ExtJS utilizadas, y los otros constituyen las interfaces del módulo MTI, además de las funciones comunes para las mismas que se encuentran en el fichero Funciones Comunes.js.

El paquete “Acceso a Datos” contiene tres paquetes en su interior Objetos, Objetos Peer y Propel. Ver Figura 2.7. Dentro de estos paquetes están las clases necesarias para efectuar la conexión y el intercambio de información de la aplicación con la base de datos.

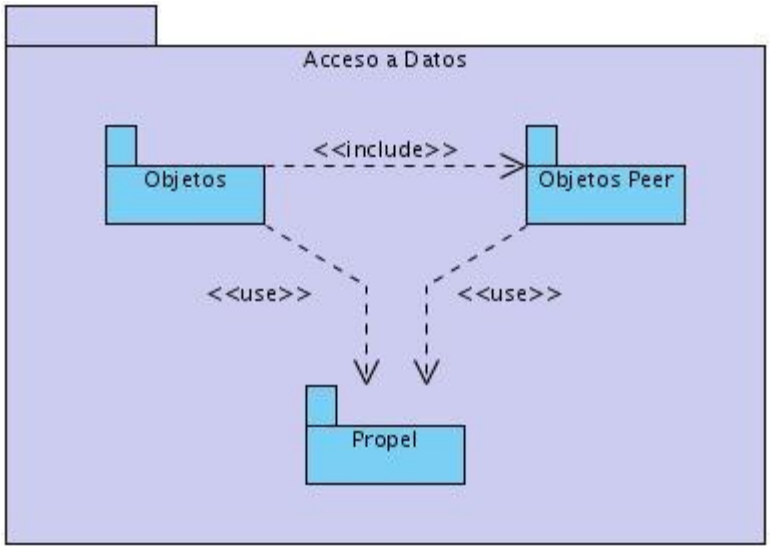


Figura 2.7. Paquete de acceso a datos.

El paquete Propel representa la capa de abstracción de objetos relacional utilizada por Symfony, implementa una de las mejores capas para la abstracción a bases de datos disponible en PHP5, y se encuentra completamente integrado al framework. Por su parte el paquete Objetos tiene las clases lógicas de la aplicación que contienen los atributos y métodos necesarios para que mediante el uso de Propel se permita la inserción y actualización de la información persistente, Ver Figura. 2.8.

Por cada tabla de la base de datos, existen 2 clases que pertenecen al paquete Objetos, por ejemplo: la clase MtiArribo hereda de BaseMtiArribo, de esta forma se logran los objetivos (inserción y actualización) sólo modificando la clase hija, ya que la clase base tiene las funcionalidades para el acceso a la base de datos con el uso del paquete Propel e incluyendo el paquete de Objetos Peer para el cual la nomenclatura de su clase homóloga sería “Base<clase>Peer”, para el caso específico tratado sería BaseMtiArriboPeer.

Las clases del paquete de Objetos Peer contienen los métodos y atributos para efectuar consultas a la base de datos, al igual que en paquete de Objetos por cada tabla en la base de datos se presentan dos clases nomencladas de la siguiente forma “<clase>Peer” y “Base<clase>Peer”, para el ejemplo tratado anteriormente las clases serían: MtiArribo y BaseMtiArriboPeer. Con esta estructura solo se modificará la clase hija y se podrán utilizar todos los métodos y atributos de la clase padre para lograr las consultas. Ver Figura. 2.9.

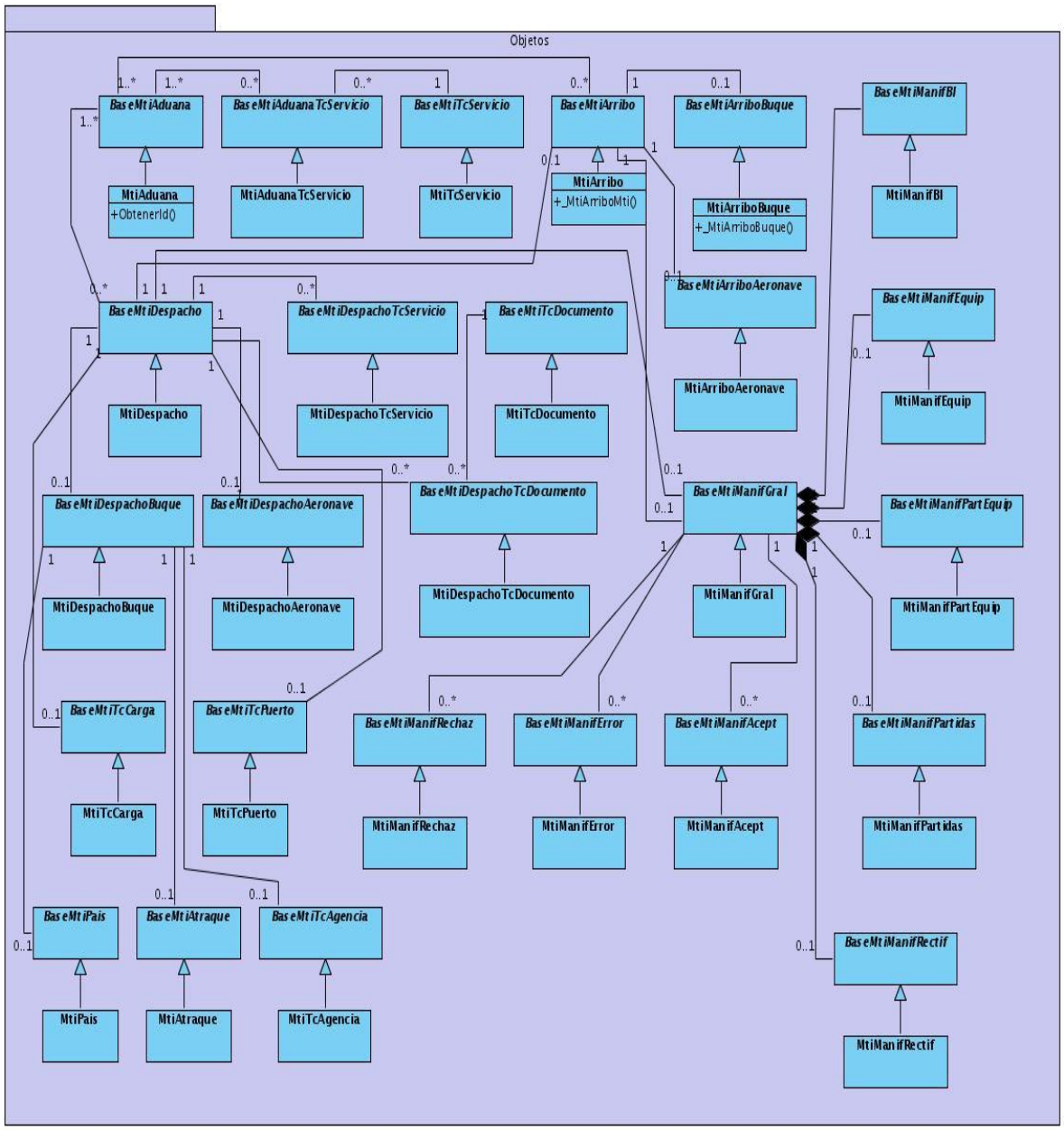


Figura 2.8. Paquete de Objetos.

Figura 2.9. Paquete de Objetos Peer

2.4. Diagramas de secuencia

El diagrama de secuencia muestra las interacciones entre objetos en un sistema ordenadas en secuencia temporal(Vilas, 2001). Los diagramas de secuencia, formalmente diagramas de traza de eventos o de interacción de objetos, se utilizan con frecuencia para validar los casos de uso y documentar el diseño desde el punto de vista de los casos de uso observando qué mensajes se envían a los objetos, componentes o casos de uso. Ayudan a comprender los cuellos de botella potenciales, para así poder eliminarlos.

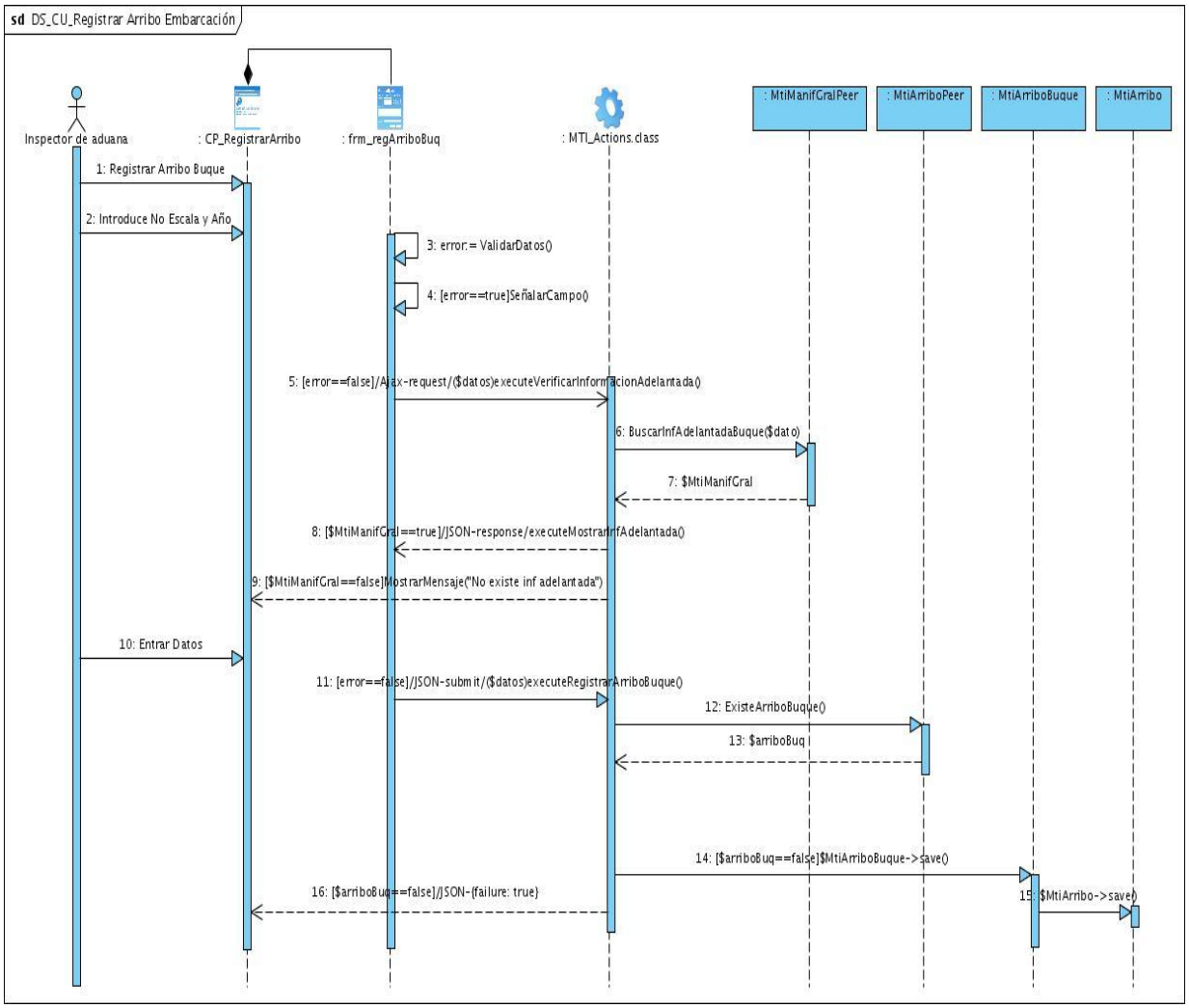


Figura 2.10. Diagrama de secuencia registrar arribo embarcación.

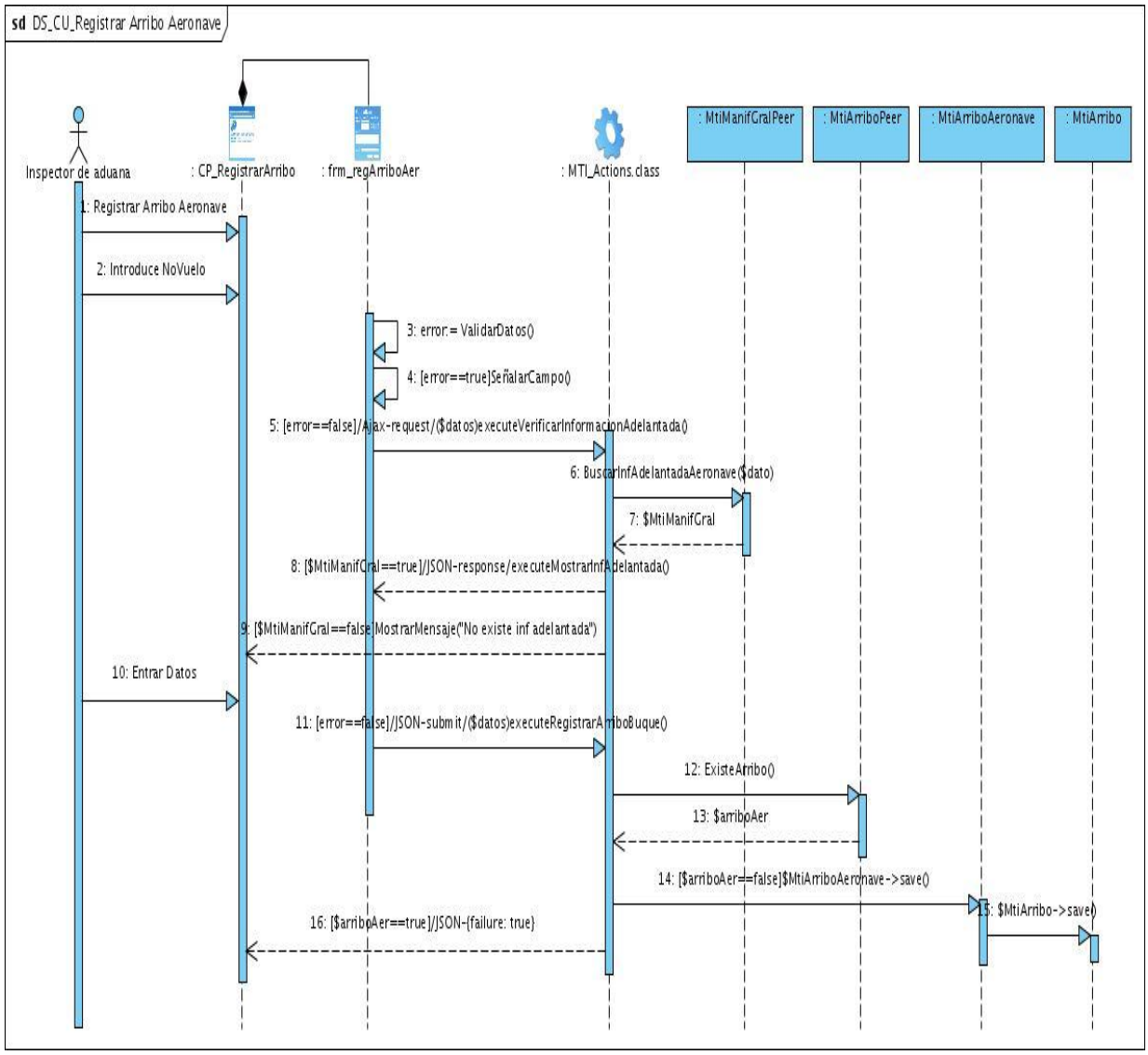


Figura 2.11. Diagrama de secuencia registrar arribo aeronave

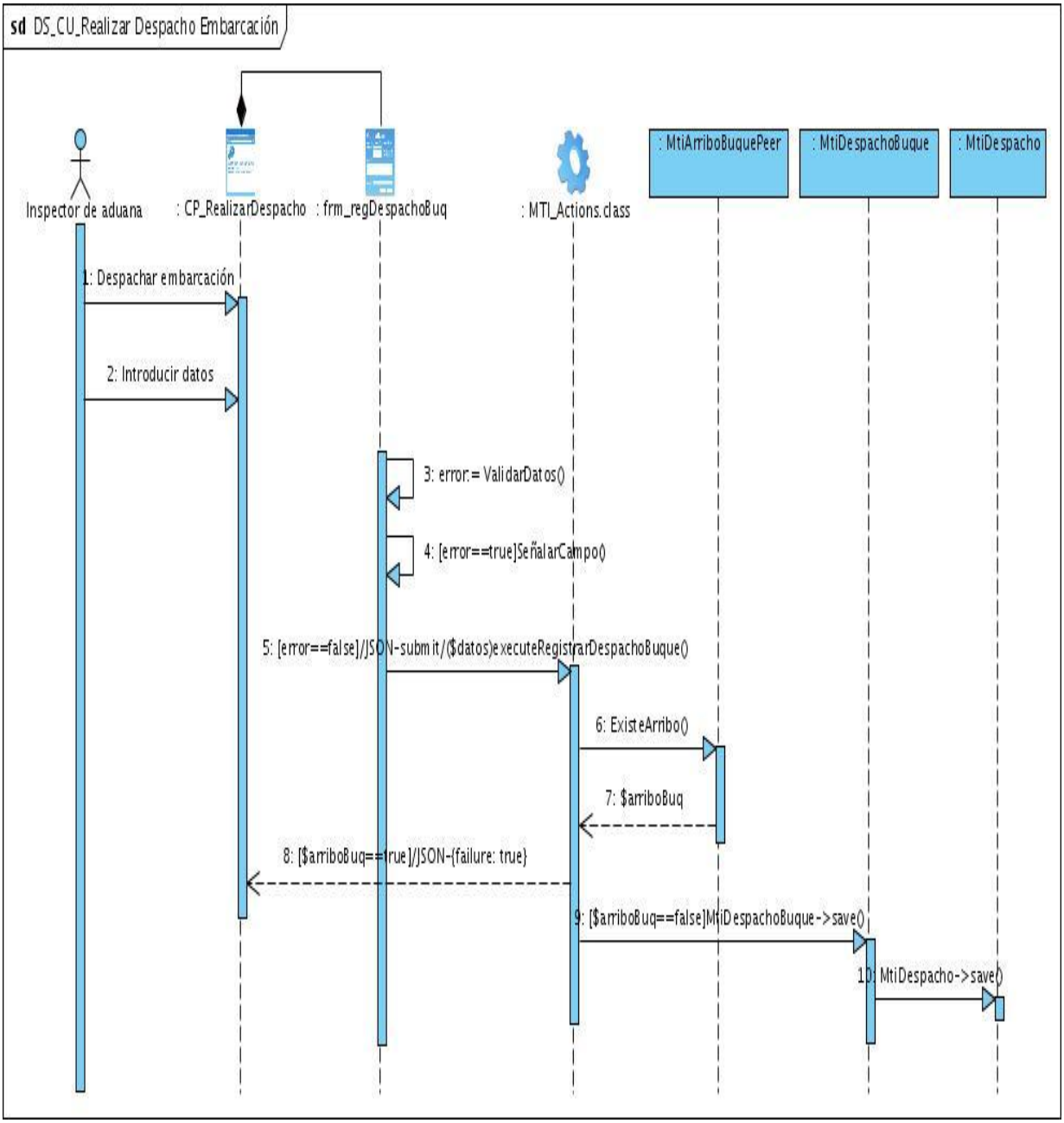


Figura 2.12. Diagrama de secuencia realizar despacho embarcación.

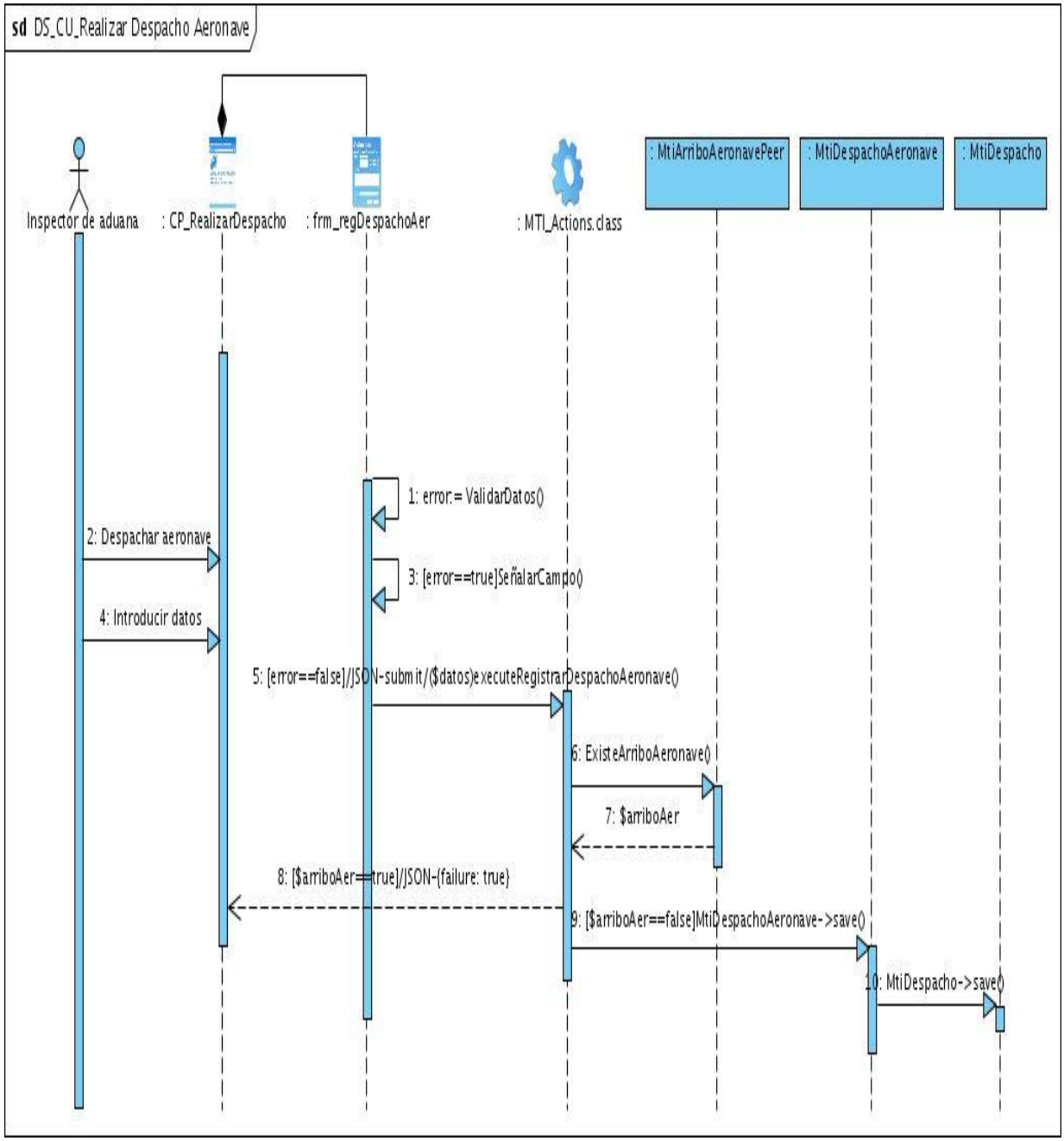


Figura 2.13. Diagrama de secuencia realizar despacho aeronave.

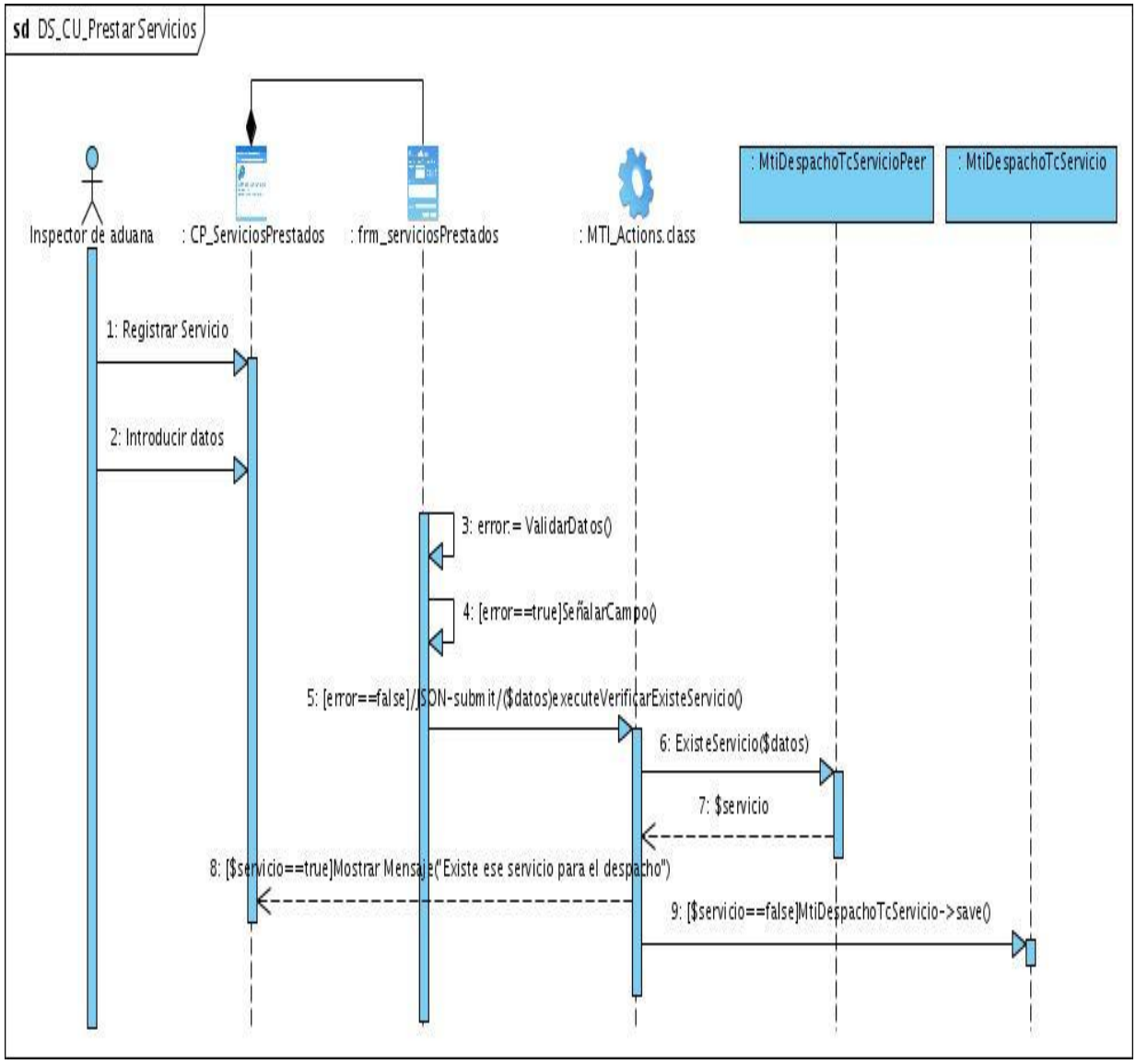


Figura 2.14. Diagrama de secuencia prestar servicios.

2.5. Diseño de la base de datos.

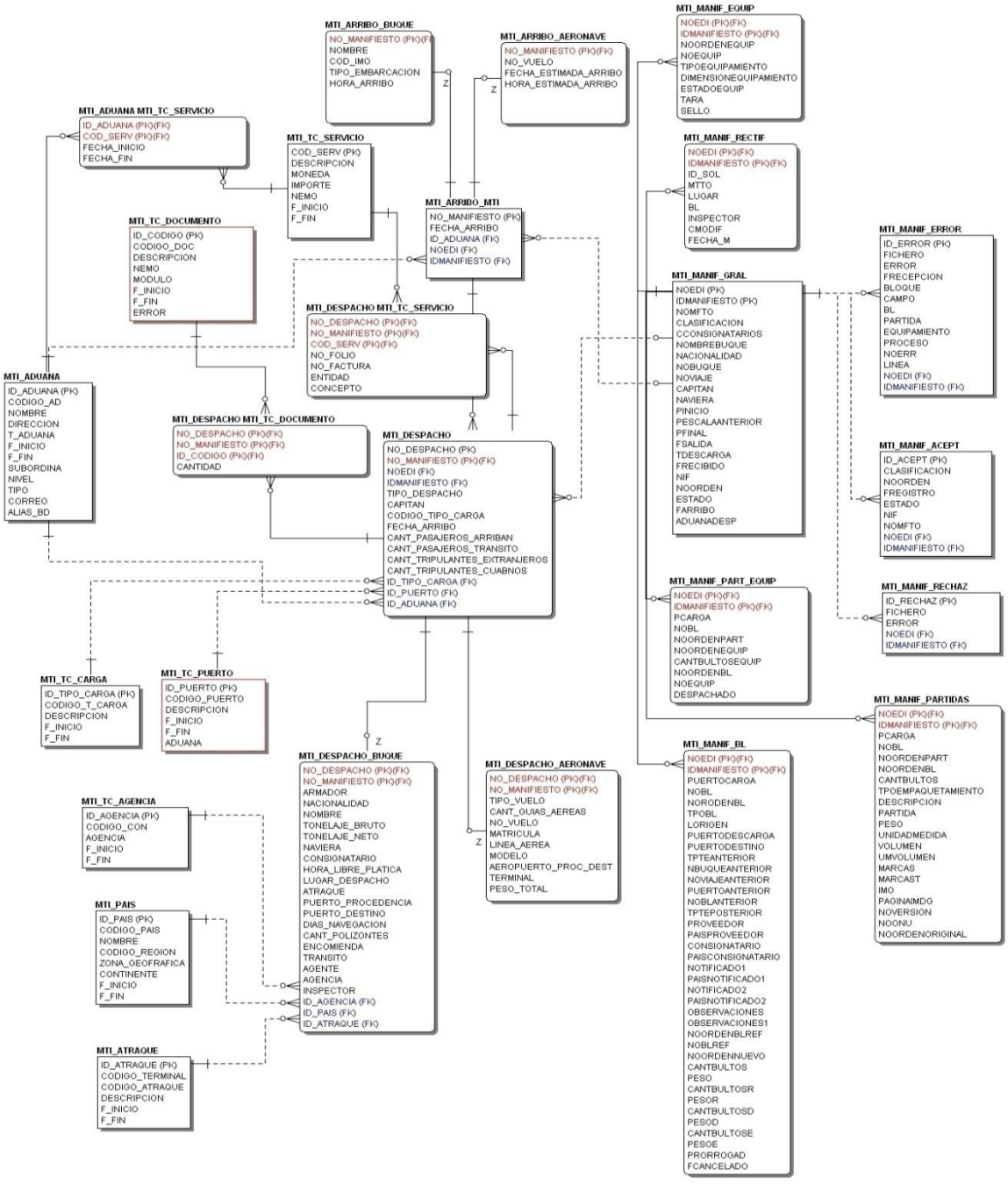


Figura 2.15. Modelo de base de datos para el despacho de los Medios de Transporte Internacional.

2.6. Diagrama de despliegue.

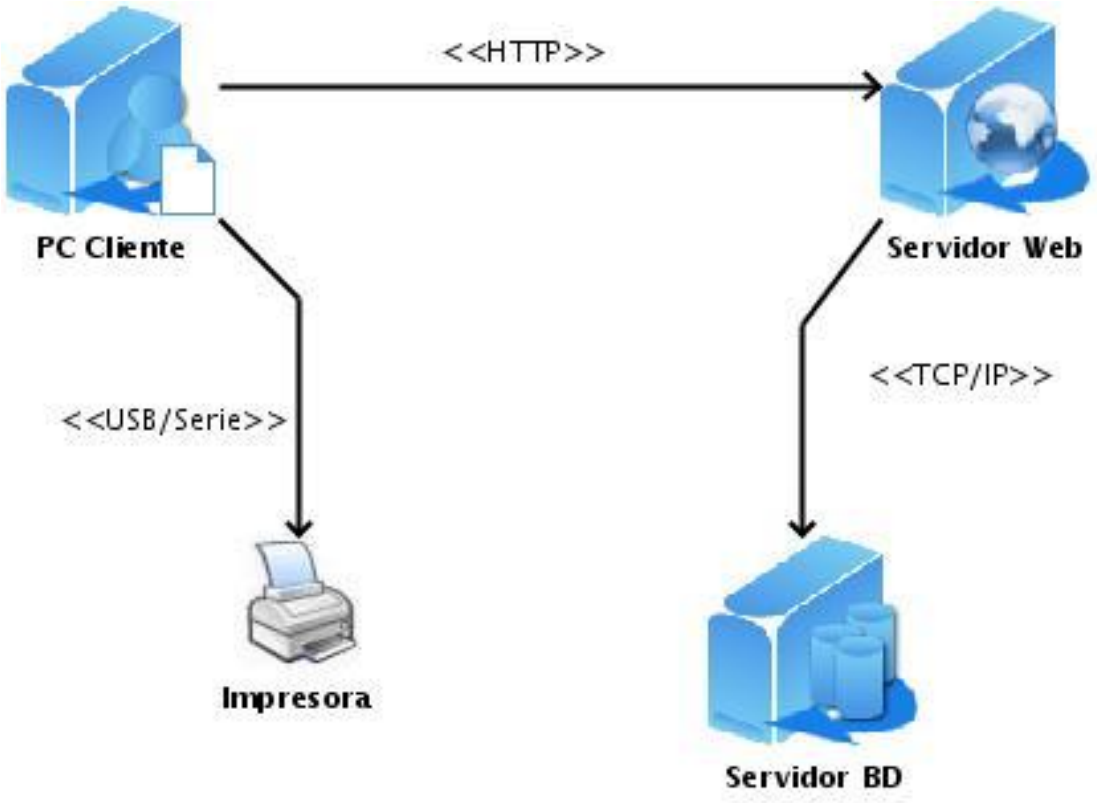


Figura 2.16. Diagrama de despliegue.

2.7. El módulo “Medios de Trasporte Internacional” orientado a la arquitectura del SUA.

La arquitectura* moldea la forma que debe tener el sistema, los casos de uso deben encajar completamente con la arquitectura así como la arquitectura debe permitir el desarrollo de los casos de uso. En la figura 2.17 muestra como se integra el módulo Medios de Transporte Internacional al Sistema Único de Aduanas. Primero hay que especificar que el mismo es parte del subsistema Despacho, este subsistema se encuentra definido dentro de las aplicaciones como “despachoMTI” y dentro se encuentra el módulo llamado “MTI”.

**arquitectura: Conjunto de elementos de programación adecuadamente estructurados dentro de un sistema, con el fin de crear una programación lógica y fiable para el diseño de aplicaciones.*

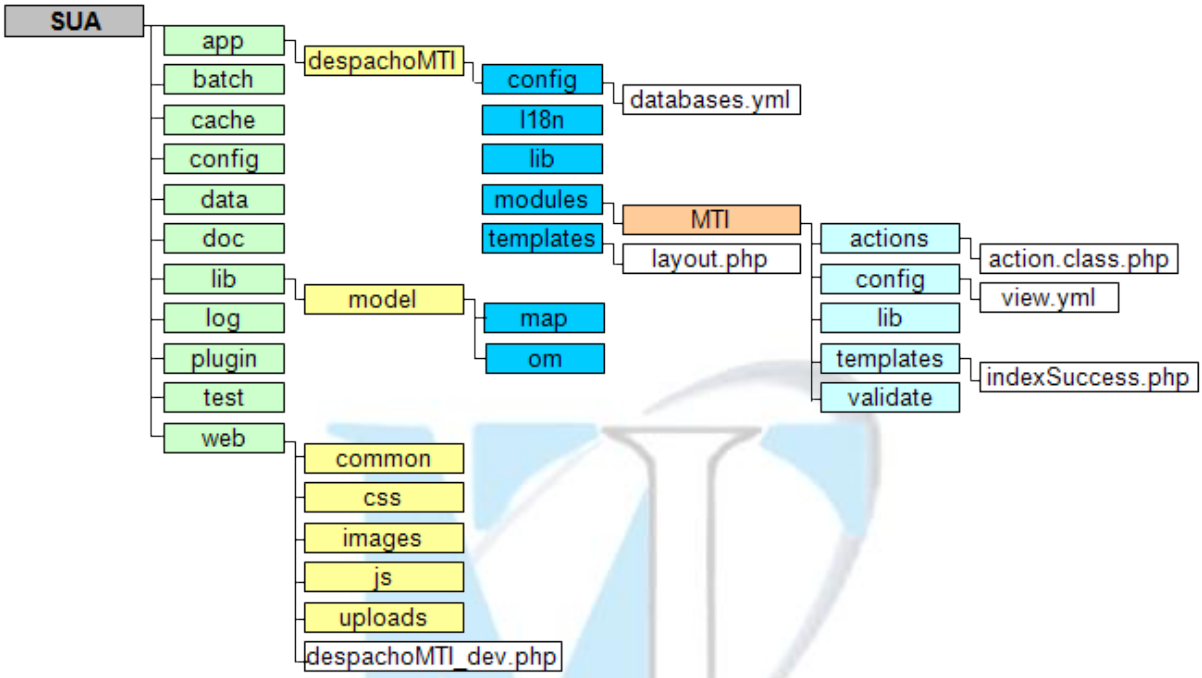


Figura 2.17. Despacho de los Medios de Transporte Internacional en el SUA.

El módulo Medios de Transporte Internacional (MTI) consta de cinco carpetas, estas son: *actions*, *config*, *lib*, *templates* y *validate*. Dentro de "actions" se encuentra el fichero *action.class.php*, donde se definen todas las acciones con que contará el módulo. En el interior de la carpeta "config" el archivo de configuración *view.yml* y dentro de templates se encuentra el fichero *indexSuccess.php* el cual representa la plantilla, que aunque es una página en blanco es importante definirla para que la aplicación funcione correctamente.

Dentro de la aplicación "despachoMTI" se encuentran cinco carpetas, en una de ellas, específicamente en "config" está archivo el *databases.yml*, utilizado para la conexión a la base de datos que será manejado en el siguiente capítulo.

La carpeta "templates" contiene el archivo *layout.php* encargado del diseño visual de todo el subsistema. En la dirección "SUA/lib/model/" se encuentran las clases definidas para el modelo de la aplicación que será generado en el capítulo 3.

En la carpeta Web, se encuentran cinco carpetas, “common”, “css”, “images”, “js” y “uploads”, además del fichero “despachoMTI_dev.php” que representa el controlador frontal de la aplicación en desarrollo. La carpeta “common” almacena ficheros con funciones y variables comunes para ser utilizadas por los ficheros javascript que se encuentran en la carpeta “js”.
Las carpetas “images”, “css” y “js” contienen las imágenes, hojas de estilo y ficheros javascript propios de la aplicación y las utilizadas por ExtJS respectivamente.

2.8 CONCLUSIONES PARCIALES

A lo largo de este capítulo se obtienen los artefactos del diseño de la aplicación. Se realizaron las clases del diseño con estereotipos web, diagrama de secuencia, diseño de la base de datos, el diagrama de despliegue que muestra las relaciones físicas entre los componentes hardware y software en el sistema final y la arquitectura del SUA aplicada al módulo. Los cuales servirán de entrada para el flujo de implementación en el siguiente capítulo.

CAPÍTULO 3

CAPÍTULO 3: IMPLEMENTACIÓN DE LA SOLUCIÓN

3.1. Introducción

Para la implementación de la solución se debe señalar que se realizará a manera prototipo (piloto) y se contempla dentro de los alcances de la tesis; buscando que dicho prototipo cumpla con las metas propuestas a lo largo de esta tesis a manera funcional; es decir, que cumpla con realizar las funciones u operaciones planteadas sin buscar necesariamente presentar un acabado final.

Es meritorio destacar que para la implementación del prototipo se decidió utilizar como plataforma al sistema Debian, en su versión 5.0.1 Lenny buscando una pronta migración a software libre.

Teniendo en cuenta lo anterior, se explicará detalladamente la implementación de la solución propuesta.

3.2. Estándar de codificación

Actualmente se hallan estándares de codificación para la mayoría de los lenguajes de programación existentes. El uso de ellos partiendo de las convenciones definidas permite una mejor comunicación entre los programadores creando condiciones para la reusabilidad y mantenimiento de los sistemas.

Para definir el estilo de codificación a seguir en la aplicación se utilizó la notación estándar establecida para aplicaciones desarrolladas en PHP (PHP Coding Standard), que mayormente está basada en el estándar de código para aplicaciones en C++ (C++ Coding Standard) [COD03].

Las etiquetas de apertura y cierre del lenguaje serán de la forma <?php ?>, ya que siempre están disponible en cualquier configuración.

Para nombrar las variables se seguirá la regla de escribir los identificadores con letras minúsculas y en español, no se usará separador entre palabras, sino que las siguientes palabras se escribirán con letra inicial mayúscula, tratando de usar nombres sugerentes a la acción de la variable.

Todos los campos id van a comenzar con el identificador (id) seguido del nombre del campo. Ejemplo id_noEscala. Los arreglos empezarán con el identificador array y las palabras no se separarán. Ejemplo: arrayBuscar.

En el caso de las clases se pondrá delante el identificador del módulo. Ejemplo: MtiNombreClase y para los métodos las palabras continuas deben comenzar con mayúsculas. Ejemplo: ObtenerInfAdelantada.
Para comentar el código se utilizará, en el caso de una línea, al final de la misma el carácter "//"y seguido el comentario y en el caso de un bloque se utilizarán los caracteres "/* */". Se usará una identificación en el código de cuatro espacios para facilitar la lectura de éste.
Las llaves se usarán poniendo la llave inicial en una línea para ella sola, y en su respectiva columna la llave final también en una línea.
Los nombres de las tablas se escribirán en mayúsculas con el identificador del módulo separado por línea abajo "_". (Ejemplo: MTI_ArriboBuque)

3.2. Implementando Symfony

Antes de empezar a desarrollar aplicaciones Symfony, es necesario instalar los archivos y librerías que componen el framework. Como Symfony está programado con PHP y sus creadores han sido muy cuidadosos, Symfony funciona igual de bien en cualquier sistema operativo (Windows, Mac OS X y Linux). Por este motivo, la instalación es idéntica en cualquier sistema operativo y tan sólo es necesario modificar las rutas de los directorios y algún otro detalle menor(Eguíluz).
Symfony se puede instalar de tres formas diferentes:

1. Para probar Symfony lo más rápido posible para ver sus posibilidades, es recomendable implementar el **entorno de pruebas o *sandbox***.
2. Si se conoce Symfony y se quiere instalar de la mejor forma posible para desarrollar proyectos con el mismo framework, se debe realizar la **instalación mediante PEAR**, que además es la instalación más común.
3. Para usuarios muy avanzados, si se desea tener un control absoluto sobre la versión de Symfony, y hacer cambios en el código fuente del framework y además que cada proyecto disponga de su propia versión de Symfony, se debe realizar la **instalación mediante el repositorio Subversion**.

Para implementarlo se eligió la segunda opción teniendo en cuenta que la versión del framework a utilizar será la 1.0.19, para ello se siguen los siguientes pasos:

- Instalar pear desde una consola con derechos administrativos (`#aptitude install php-pear`)

- Especificarle a pear el proxy a través del cual navegamos (`pear config-set http_proxy http://usuario:password@10.0.0.1:8080`)
- Conectar con el proyecto Symfony (`pear channel-discover pear.symfony-project.com`)
- Instalar Symfony (`pear install symfony/symfony-1.0.19`)

Además se pueden listar los paquetes disponibles del framework ejecutando el comando (`pear remote-list -c symfony`)

3.2.1. Implementando el proyecto, la aplicación y el módulo.

Symfony considera un proyecto como "*un conjunto de servicios y operaciones disponibles bajo un determinado nombre de dominio y que comparten el mismo modelo de objetos*"(Potencier, y otros, 2008)

Dentro de un proyecto symfony, las operaciones se agrupan de forma lógica en aplicaciones las que están formadas por uno o varios módulos, a su vez, un módulo representa a una página web o grupo de ellas con un propósito estrechamente relacionado.

Para crear el proyecto, la aplicación y el módulo, al tratarse de una plataforma como Debian podemos hacerlo de la siguiente forma: se crea la carpeta del proyecto (`/var/www# mkdir SUA`) y se ejecutan los comandos (`/var/www/SUA# symfony init-project SUA`) para crear la estructura raíz del proyecto ver Tabla3.1, (`/var/www/SUA# symfony init-app despachoMTI`) para crear los subdirectorios de la aplicación ver Tabla 3.2 y (`/var/www/SUA# symfony init-module despachoMTI MTI`) para crear el módulo ver Tabla 3.3.

Para ver los resultados de los comandos anteriores ver Anexo 1.

```
apps/
  frontend/
  backend/
batch/
cache/
config/
data/
sql/
doc/
lib/
  model/
log/
plugins/
test/
unit/
functional/
web/
  css/
  images/
  js/
  uploads/
```

Tabla 3.1. Estructura del directorio raíz del proyecto

```
apps/
  despachoMTI/
config/
i18n/
lib/
modules/
templates/
  layout.php
```

Tabla 3.2. Estructura de los subdirectorios de la aplicación

```
apps/
  despachoMTI/
modules/
  MTI/
    actions/
      actions.class.php
    config/
    lib/
    templates/
      indexSuccess.php
    validate/
```

Tabla 3.3. Estructura de los subdirectorios del módulo

3.2.2. Implementando el esquema y el modelo.

En orden de crear las clases que representen las tablas y relaciones de la base de datos (Ver Figura 2.15) es necesario crear una representación XML de la misma y generar el modelo de dicha representación.

Symfony nos propone una ventaja a la hora de manejar este XML, para ello es necesario editar los ficheros *databases.yml* y *propel.ini* (ver Anexo 2).

Ya editados correctamente dichos archivos se puede ejecutar el comando (`symfony propel-build-schema xml`) para generar el esquema que representará las tablas de la base de datos en XML y que será almacenado en el archivo *schema.xml*.

Si se generó correctamente (Véase Anexo 2) se procede a generar el modelo, no sin antes recordar que al usar Oracle se debe especificar los campos de las tablas que serán autoincrementables (ver Anexo 2)

Editado el fichero *schema.yml* se procede a generar el modelo de datos ejecutando el comando (`symfony propel-build-model`). Arrojando como resultado las clases de los Paquetes de Objetos y Peer. Figura 2.8 y 2.9.
Véase Anexo 2.

3.2.3. Implementando ExtJS para Symfony

Un programa muy poderoso con una interfaz de usuario pobremente elaborada tiene poco valor para los usuarios inexpertos, de ahí que una de las partes más importantes del desarrollo de todo sistema es la creación de una interfaz que posibilite una comunicación lo más fácil y cómoda posible entre el sistema y el usuario.
Por lo anterior expuesto se decide usar ExtJS para desarrollar el sistema que propiciará dicha comunicación.
Teniendo en cuenta que el proyecto está desarrollándose con symfony se deben seguir una serie de pasos para unir estos dos frameworks.

- Copiar los componentes de ExtJS dentro de nuestro proyecto. Ver Anexo 3
- Editar el fichero *view.yml* del módulo. Ver Anexo 3

Cometido esto se puede comenzar a utilizar ExtJS en el marco del proyecto.

3.3. Implementando una petición Ajax

En varios casos de la aplicación es necesario consultar datos que se encuentran almacenados en el servidor de datos, ya sea para ser mostrados ó para advertir al usuario de forma transparente; ejecutando peticiones Ajax en el cliente puede materializarse este procedimiento, es decir, mientras se mantiene una comunicación *asíncrona** con el servidor en segundo plano. Ver Figura 3.1

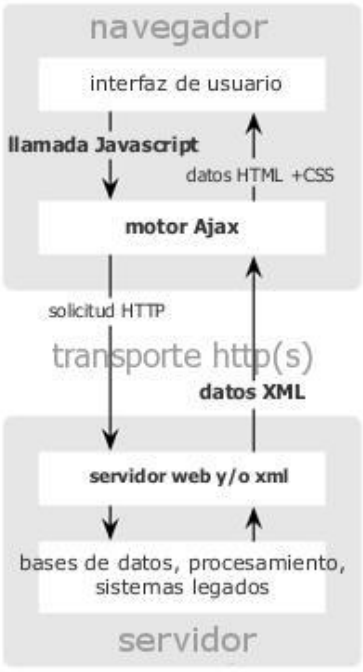


Figura 3.1. Modelo Ajax para aplicaciones web

De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en la aplicación.

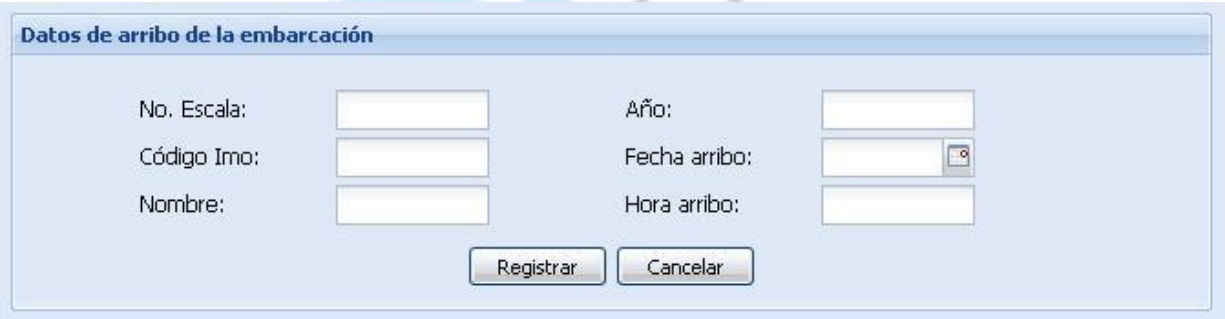


Figura 3.2. Registrar arribo de embarcación

Para el escenario *registrar arribo de embarcación* del caso de uso *registrar arribo* se necesita consultar si existe información adelantada luego de entrar los datos "No. Escala" y "Año" (ver Figura 3.2), para ello se crea la petición Ajax (ver Tabla 3.4) agregando un objeto *listeners.*

```
{xtype: 'numberfield',
 name: 'anno',
 tabIndex: 2,
 fieldLabel: 'A&ntilde;o',
 allowBlank: false,
 listeners: {
   'blur': function(obj, record, index){
             Ext.Ajax.request({
                  params: {
                    anno: obj.value,
                    noEscala: Ext.getCmp('noEscala').getValue()
                  },
                  url: 'MTI/buscarInformacionAdelantadaBuque',
                  success: function(resp){
                    var r = Ext.decode(resp.responseText);
                    if(r.estructura[0]){
                      Ext.getCmp('nombreBuq').setValue(r.estructura[0].nombreBuq);
                      Ext.getCmp('fechaArr').setValue(r.estructura[0].fechaArr);
                    }
                  },
                  failure: {}
   });
 }
}
```

Tabla 3.4. Creando petición Ajax

Objeto que será implementado en el campo 'anno' de la Tabla 3.4 y estará a la escucha del evento 'blur' para ejecutar la petición con los parámetros 'anno' y 'noEscala', la 'url' que recibirá dicha petición, en este caso 'MTI/buscarInformacionAdelantedaBuque', quien procesará los parámetros arrojando como resultado un objeto *JSON**, además de las funciones 'succes' y 'failure' que serán ejecutadas en caso de éxito ó fracaso de la petición respectivamente.

**JSON: (Notación de objeto JavaScrip) es un ligero formato de intercambio de datos, fácil de leer y escribir por los humanos y de generar y analizar por las máquinas.*(Crockford, 2006)

3.4. Implementando la interfaz de usuario

Recordando lo hablado en la sección 3.2.3 sobre las interfaces de usuario, se procede a implementar una interfaz visual lo más cómoda, profesional y funcional posible.

Figura 3.3. Interfaz principal

En la interfaz controladora ó principal se implementa en menú que dará paso a la ejecución de cada uno de los Casos de Uso del sistema(Correa, y otros, 2008) ver Anexo 4.

Para el escenario Registrar arribo de embarcación del Caso de Uso Registrar Arribo haciendo uso de Ajax ver Tabla 3.4 se tiene en cuenta el principio de usabilidad de tener siempre informado al usuario sobre el estado del sistema, así como para el resto de los Casos de Uso, por ejemplo, dejando un mensaje de que ocurrieron los cambios. Ver Figura 3.4.

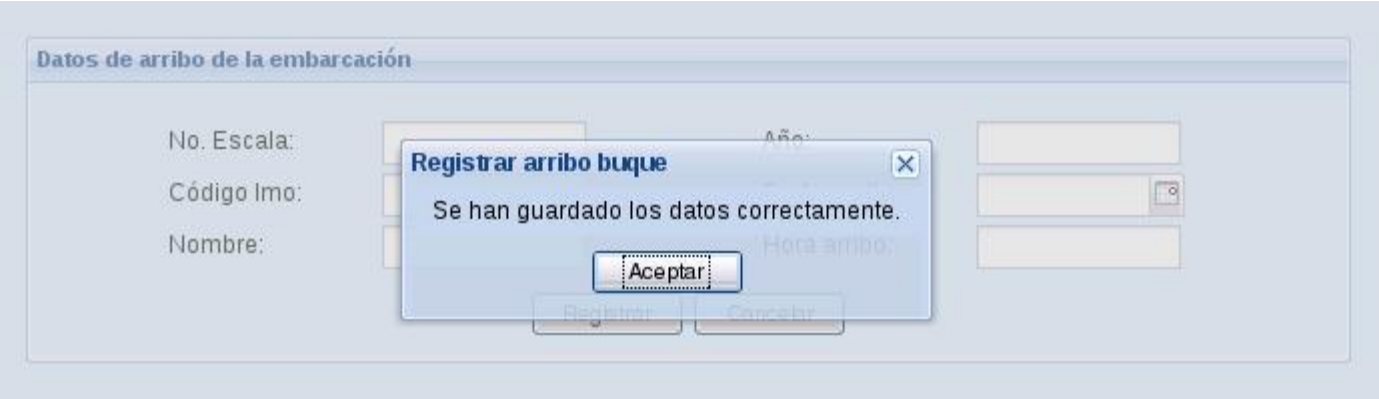


Figura 3.4. Información al usuario

Además se probaron otras alternativas para destacar los cambios, como es la animación en el centro de la pantalla cuando se está registrando en la base de datos.

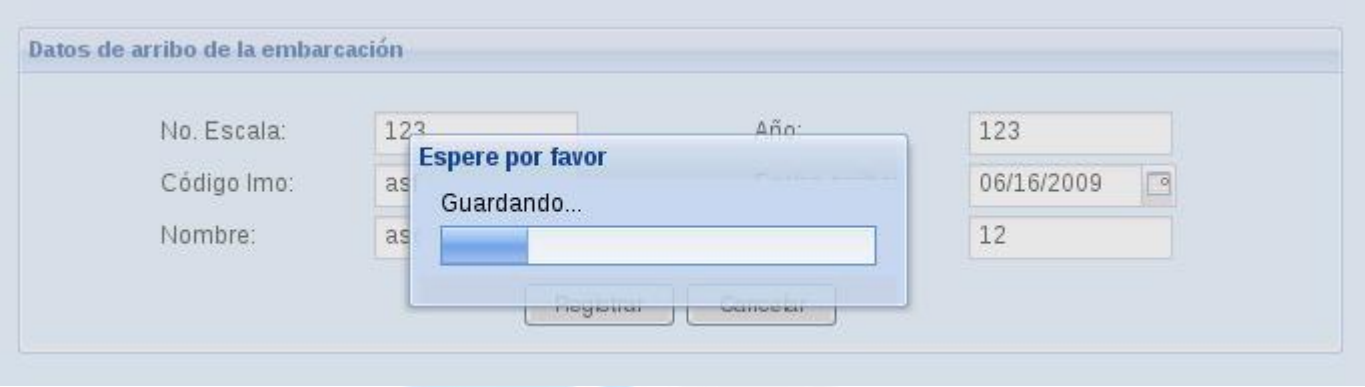


Figura 3.5. Enviando datos

3.5. Implementando un objeto JSON

JSON es una estructura de datos universal que se puede representar a través de una serie de pares desordenada de nombre / valor, que comienza con ‘**{**‘ (corchete izquierdo) y termina con ‘**}**’ (corchete derecho). Cada nombre es seguido por **:** (dos puntos) y la pareja nombre/valor separadas por (**,**) coma. Ver Figura 3.6.

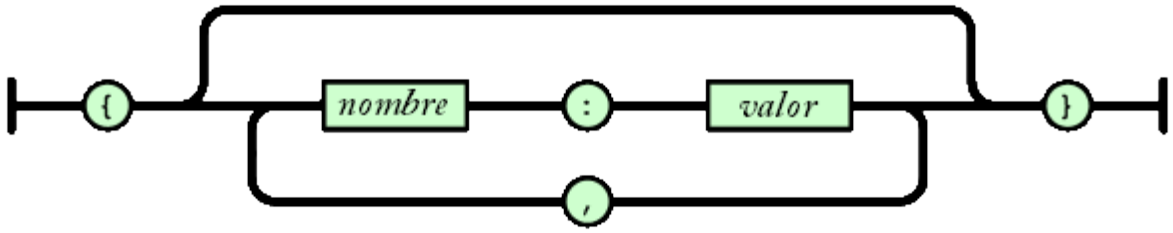


Figura 3.6. Estructura de un JSON

Para el caso específico tratado en la Sección 3.3 es necesario conformar un objeto JSON para la comunicación entre php y javascript, el mismo es implementado en la clase “MtiManifGralPeer”, ver Tabla 3.5.

```
1public static function ObtenerInfAdelantada($manif)
2 {
3   $criter = new Criteria();
4   $criter->add(MtiManifGralPeer::NOMFTO,$manif);
5   $result = self::doSelect($criter);
6   $cantidad = count($result);
7   $json = "{ count: {$cantidad}, estructura: [";
8   $cont = 0;
9   $value = new MtiManifGral();
10   foreach($result as $value){
11    $json .= "{nombreBuq: '{$value-
>getNombrebuque()}',fechaArr:'{$value-> getFarribo()}'}";
12    if(($cantidad - 1) !== $cont){
13      $json .= ",";
14    }
15    $cont++;
16   }
17   $json .= "]}";
18   return $json;
19 }
```

Tabla 3.5. Creando un JSON

El método “ObtenerInfAdelantada($manif)”, obtiene como parámetro un manifiesto, por el cual se crea un criterio de búsqueda y se ejecuta la consulta en la línea 5, ésta a su vez arroja un resultado en la variable “$result”, luego se crea la estructura del objeto JSON en la línea 7 y mediante un ”foreach” se van recogiendo los resultados de la consulta en el objeto “$value” y agregándose al objeto $json, luego se

cierra el objeto siguiendo su estructura (Ver Figura 3.6) y se retorna como resultado de la llamada a este método.

3.6. Flujo de comunicación entre las capas MVC

En el capítulo 2, sección 2.2.3 se detalla el patrón Modelo, Vista, Controlador; a continuación veremos cómo se implementa dicho patrón en el caso específico de Registrar el arribo de una embarcación haciendo uso de ExtJS y Symfony (ver Tabla 3.6)

```
/**
 * @author RCGA
 */
MtiRegArriboBuque = function(){
    MtiRegArriboBuque.superclass.constructor.call(this, {
        .
        .
        .
        items: [{
            xtype: 'form',
            title: 'Datos de arribo de la embarcaci&oacute;n',
            id: 'id_frm_regArriboBuq',
            url: 'MTI/registrarArriboBuque',
        .
        .
        .
        buttons: [{
                id: 'id_registrar',
                text: 'Registrar',
                handler: function(n){
                  if (Ext.getCmp('id_frm_regArriboBuq').form.isValid()) {
                        Ext.getCmp('id_frm_regArriboBuq').form.submit({
                            waitMsg: 'Guardando...',
                            waitTitle: 'Espere por favor',
                    success: function(){
                     SuccessMensaje('Registrar arribo buque');
                     Ext.getCmp('id_frm_regArriboBuq').form.reset();
                    },
                    failure: function(){
                     FailureMessage('Registrar arribo buque','Ya se le
asign&oacute arribo a esa embarcaci&oacute;n.');
                    }
        .
        .
        .
       }]
    });
}
Ext.extend(MtiRegArriboBuque, Ext.Panel);
```

Tabla 3.6. Enviando datos al controlador.

Con el uso de ExtJS se crea la clase “Reg_arriboBuque” encargada de recoger los datos del arribo, validarlos y enviarlos en forma de objeto JSON (ver Figura 3.6) a la clase controladora, quien procesará los datos en la acción “executeRegistrarArriboBuque()”, especificada en la propiedad ‘url’ del formulario “id_frm_regArriboBuq” siguiendo el estándar de Symfony para referenciar una acción que sería: nombre de la acción que procesa los datos con la primera letra en minúsculas y el nombre del módulo en mayúsculas delante, separados por ‘/’. Ejemplo: MTI/registrarArriboBuque.

Al recibir los datos en el controlador se procede a la comunicación entre las capas del Controlador y el Modelo que gracias a la implementación del ORM, que está dada por el envío de objetos que contienen la información necesaria para manejar las peticiones del usuario.

Para el caso de Registrar Arribo de Buque Ver Anexo 5

3.8. Diagrama de componentes

Los diagramas de componentes muestran como el sistema está dividido en componentes y las dependencias entre ellos. Proveen una vista arquitectónica de alto nivel del sistema y ayudan a los desarrolladores a visualizar el camino de la implementación, permitiendo tomar decisiones respecto a las tareas de implementación y las habilidades requeridas.

A continuación se presenta el diagrama de componentes del subsistema SUA y las relaciones entre sus componentes; la saeta discontinua indica una dependencia ej.: el componente “despachoMTI_dev.php” utiliza al componente “actions.class.php” y así sucesivamente. Ver Figura 3.7.

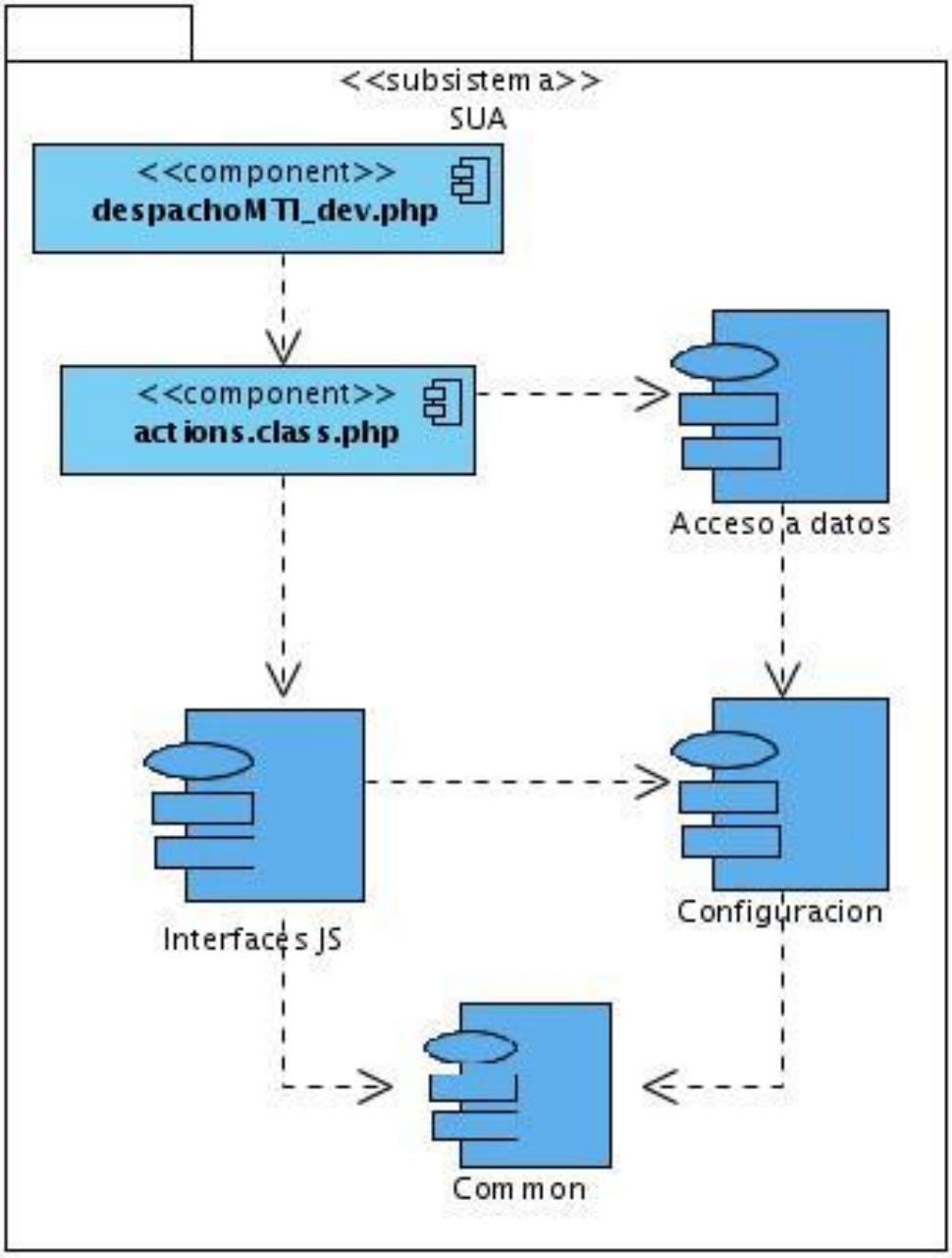


Figura 3.7. Subsistema SUA

Para el resto de los paquetes de componentes remitirse al Anexo 6.

Conclusiones

Una vez concluido el presente trabajo se puede afirmar que se logró alcanzar exitosamente el objetivo trazado. Guiado por la metodología RUP y con el uso de las herramientas definidas fue posible obtener los artefactos necesarios para modelar la solución. Se trazaron una serie de tareas detallando su cumplimiento a lo largo del trabajo, las mismas fueron: revisión bibliográfica de la documentación relacionada con el Módulo Medios de Transporte Internacional obtenida del proceso de análisis, revisión de los Casos de Uso del sistema, diseño y por último implementación del subsistema Medios de Transporte Internacional; logrando solucionar el problema que dio origen a la realización de este trabajo.

Recomendaciones

Por la importancia que tiene el correcto funcionamiento de la implementación de la aplicación para el Sistema Único de Aduanas, se recomienda:

- Avanzar a la fase de pruebas del Módulo Medios de Transporte Internacional.
- Migrar la base de datos a PostgreSQL.

Referencias bibliográficas

1. **Castro Díaz – Balart, Fidel. 2001.** *Ciencia, innovación y futuro.* La Habana : Instituto Cubano del libro, 2001.
2. **Chuecko. 2008.** Workflow de Diseño, Clases de Diseño, Interfaces, Diagrama. *Taringa.net.* [Online] 08 27, 2008. http://www.taringa.net/posts/info/1492028/Workflow-De-Dise%C3%B1o,-Clases-de-Dise%C3%B1o,-Interfaces,-Diagrama.html.
3. **Correa, Ivette and Trainchet, Yosvany. 2008.** *Análisis del módulo de Medios de Transporte Internacional.* Ciudad de la Habana : s.n., 2008.
4. **Crockford. 2006.** JSON.org. *JSON.org.* [Online] July 2006. http://www.ietf.org/rfc/rfc4627.txt?number=4627.
5. **DAEDALUS. 2006.** Diseño de Sistemas. [Online] 2006. http://www.daedalus.es/AreasISDiseno-E.php.
6. **Eguíluz, Javier.** Symfony.es. *Symfony.es.* [Online] http://www.symfony.es/documentacion/instalacion/.
7. **FOWLER, M. 2003.** La nueva metodología. [Online] 2003. http://www.programacionextrema.org/articulos/newMethodology.es.html.
8. **Gutiérrez, Enrique.** eSemanal en Internet. [Online] [Cited: Abril 3, 2009.] http://www.esemanal.com.mx/articulos.php?id_sec=2&id_art=5532.
9. **JACOBSON, I. 2000.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000. 84-7829-036-2.
10. **Macías, Charlie and Orozco, Sergio.** Uso de UML en aplicaciones Web: páginas y relaciones. *Milestone.* [Online] [Cited: 05 15, 2009.] http://www.milestone.com.mx/articulos/uso_de_uml_en_aplicaciones_web.htm.
11. **Microsoft. 2006.** *La Arquitectura Orientada a Servicios (SOA) de Microsoft.* 2006.
12. **Potencier, Fabien and Zaninotto, François. 2008.** *Guía definitiva de Symfony.* 2008.
13. **PRESSMAN, R. S. 1998.** *Ingenieria de software. Un enfoque practico. Mc Graw Hill.* 1998.
14. **1999.** RFC 2616. *RFC 2616.* [Online] 6 1999. http://www.ietf.org/rfc/rfc2616.txt.
15. **RUMBAUGH, J. and I. JACOBSON, et al. 1998..** *The Unified Modeling Language. Reference Manual.* Massachusetts : ADDISON-WESLEY, 1998. 0-201-30998-X.

16. **Usolab Consultoría S.L. "Especialistas en usabilidad web". 2005.** Usolab Consultoría S.L. *www.usolab.com.* [Online] Noviembre 2005. http://www.usolab.com/articulos/desafios_interfaz_web_2.php.
17. **Vilas, Ana Fernandez. 2001.** tvdi.det.uvigo. [Online] Marzo 20, 2001. http://tvdi.det.uvigo.es/~avilas/UML/node42.html.

Glosario de términos

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

SUA: Sistema Único de Aduanas.

MTI: Medios de Transporte Internacional.

RUP: Proceso Unificado Racional (***R****ational* ***U****nified* ***P****rocess* en inglés) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

UML: Unified **M**odeling **L**anguage. Lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

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

Software libre: El software libre es la libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.

XML: Siglas en inglés de *E****x****tensible* ***M****arkup* ***L****anguage* (lenguaje de marcas extensibles), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C)

Ajax: Acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (javascript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o **RIA** (Rich Internet Applications)

JSON: (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

SOA: La Arquitectura Orientada a Servicios del inglés (Service Oriented Architecture) es una filosofía de diseño que permite un mejor alineamiento de las Tecnologías de Información (IT) con las necesidades de negocio.

URL: acrónimo de ***U****niform* ***R****esource* ***L****ocator*, es decir, localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos.

Anexo 1. Resultados satisfactorios de:

Crear el proyecto

```
>> dir+      /var/www/SUA/doc
>> dir+      /var/www/SUA/lib
>> dir+      /var/www/SUA/lib/model
>> dir+      /var/www/SUA/log
>> dir+      /var/www/SUA/web
>> dir+      /var/www/SUA/web/js
>> dir+      /var/www/SUA/web/css
>> file+     /var/www/SUA/web/css/main.css
>> file+     /var/www/SUA/web/.htaccess
>> file+     /var/www/SUA/web/robots.txt
>> dir+      /var/www/SUA/web/images
>> dir+      /var/www/SUA/web/uploads
>> dir+      /var/www/SUA/web/uploads/assets
>> dir+      /var/www/SUA/apps
>> dir+      /var/www/SUA/data
>> dir+      /var/www/SUA/data/sql
>> dir+      /var/www/SUA/data/model
>> dir+      /var/www/SUA/test
>> dir+      /var/www/SUA/test/unit
>> dir+      /var/www/SUA/test/functional
>> dir+      /var/www/SUA/test/bootstrap
>> file+     /var/www/SUA/test/bootstrap/unit.php
>> file+     /var/www/SUA/test/bootstrap/functional.php
>> dir+      /var/www/SUA/batch
>> dir+      /var/www/SUA/cache
>> dir+      /var/www/SUA/config
>> file+     /var/www/SUA/config/databases.yml
>> file+     /var/www/SUA/config/config.php
>> file+     /var/www/SUA/config/propel.ini
>> file+     /var/www/SUA/config/rsync_exclude.txt
>> file+     /var/www/SUA/config/schema.yml
>> file+      /var/www/SUA/config/properties.ini
>> file+      /var/www/SUA/symfony
>> dir+       /var/www/SUA/plugins
>> tokens     /var/www/SUA/config/propel.ini
>> tokens     /var/www/SUA/config/properties.ini
>> tokens     /var/www/SUA/config/propel.ini
>> tokens     /var/www/SUA/config/config.php
>> chmod 777 /var/www/SUA/cache
>> chmod 777 /var/www/SUA/log
>> chmod 777 /var/www/SUA/web/uploads
>> chmod 777 /var/www/SUA/symfony
>> chmod 777 web/uploads/assets
```

Crear la aplicación

```
>> dir+      /var/www/SUA/apps/despachoMTI/lib
>> file+     /var/www/SUA/apps/despachoMTI/lib/myUser.class.php
>> dir+      /var/www/SUA/apps/despachoMTI/i18n
>> dir+      /var/www/SUA/apps/despachoMTI/config
>> file+     /var/www/SUA/apps/despachoMTI/config/routing.yml
>> file+     /var/www/SUA/apps/despachoMTI/config/factories.yml
>> file+     /var/www/SUA/apps/despachoMTI/config/filters.yml
>> file+     /var/www/SUA/apps/despachoMTI/config/cache.yml
>> file+     /var/www/SUA/apps/despachoMTI/config/i18n.yml
>> file+     /var/www/SUA/apps/despachoMTI/config/config.php
>> file+     /var/www/SUA/apps/despachoMTI/config/settings.yml
>> file+     /var/www/SUA/apps/despachoMTI/config/app.yml
>> file+     /var/www/SUA/apps/despachoMTI/config/security.yml
>> file+     /var/www/SUA/apps/despachoMTI/config/view.yml
>> file+     /var/www/SUA/apps/despachoMTI/config/logging.yml
>> dir+      /var/www/SUA/apps/despachoMTI/modules
>> dir+      /var/www/SUA/apps/despachoMTI/templates
>> file+     /var/www/SUA/apps/despachoMTI/templates/layout.php
>> tokens    /var/www/SUA/apps/despachoMTI/config/settings.yml
>> file+     /var/www/SUA/web/index.php
>> file+     /var/www/SUA/web/despachoMTI_dev.php
>> tokens    /var/www/SUA/web/despachoMTI_dev.php
>> tokens    /var/www/SUA/web/index.php
>> chmod 777 /var/www/SUA/cache
>> chmod 777 /var/www/SUA/log
>> chmod 777 /var/www/SUA/web/uploads
>> chmod 777 /var/www/SUA/symfony
>> chmod 777 web/uploads/assets
>> dir+      /var/www/SUA/test/functional/despachoMTI
```

Crear el módulo

```
>> dir+      /var/www/SUA/apps/despachoMTI/modules/MTI/lib
>> dir+      /var/www/SUA/apps/despachoMTI/modules/MTI/config
>> dir+      /var/www/SUA/apps/despachoMTI/modules/MTI/actions
>> file+     /var/www/SUA/apps/despachoMTI/m...s/MTI/actions/actions.class.php
>> dir+      /var/www/SUA/apps/despachoMTI/modules/MTI/validate
>> dir+      /var/www/SUA/apps/despachoMTI/modules/MTI/templates
>> file+     /var/www/SUA/apps/despachoMTI/m.../MTI/templates/indexSuccess.php
>> file+     /var/www/SUA/test/functional/despachoMTI/MTIActionsTest.php
>> tokens    /var/www/SUA/test/functional/despachoMTI/MTIActionsTest.php
>> tokens    /var/www/SUA/apps/despachoMTI/m...s/MTI/actions/actions.class.php
>> tokens    /var/www/SUA/apps/despachoMTI/m.../MTI/templates/indexSuccess.php
```

Anexo 2. Creando el esquema y el modelo

Editando *databases.yml* y *propel.ini*

```
# gedit /var/www/SUA/config/databases.yml
>Editar la siguiente línea

dsn:          oracle://despacho_mti:despacho@CADI.UCI.CU/

>Guardar cambios


# gedit /var/www/SUA/config/propel.ini
>Editar las siguientes líneas

propel.database            = oracle
propel.database.createUrl  = oracle://despacho_mti:despacho@CADI.UCI.CU/
propel.database.url        = oracle://despacho_mti:despacho@CADI.UCI.CU/
propel.builder.addIncludes = false
propel.builder.addComments = false
propel.builder.addBehaviors = false

>Guardar cambios
```

Generando el esquema

```
# symfony propel-build-schema xml

Buildfile: /usr/share/php5/symfony/lib/vendor/propel-generator/build.xml
[resolvepath] Resolved /var/www/SUAbad/config to /var/www/SUAbad/config

propel-project-builder > check-project-or-dir-set:

propel-project-builder > check-project-set:

propel-project-builder > set-project-dir:

propel-project-builder > check-buildprops-exists:

propel-project-builder > check-buildprops-for-propel-gen:

propel-project-builder > check-buildprops:

propel-project-builder > configure:
     [echo] Loading project-specific props from /var/www/SUAbad/config/propel.ini
 [property] Loading /var/www/SUAbad/config/propel.ini

propel-project-builder > creole:
    [phing] Calling Buildfile '/usr/share/php5/symfony/lib/vendor/propel-
generator/build-propel.xml' with target 'creole'
 [property] Loading /usr/share/php5/symfony/lib/vendor/propel-
generator/./default.properties

propel > creole:
     [echo] +-----------------------------------------------+
     [echo] |                                               |
     [echo] | Generating XML from Creole connection !       |
     [echo] |                                               |
     [echo] +-----------------------------------------------+
[propel-creole-transform] Propel - CreoleToXMLSchema starting
[propel-creole-transform] Your DB settings are:
[propel-creole-transform] driver : (default)
[propel-creole-transform] URL : oracle://despacho_mti:despacho@CADI.UCI.CU/
[propel-creole-transform] DB connection established
[propel-creole-transform] Processing database
[propel-creole-transform] Processing table: MTI_MANIF_ACEPT
[propel-creole-transform] Processing table: MTI_ADUANA
[propel-creole-transform] Processing table: MTI_MANIF_EQUIP
[propel-creole-transform] Processing table: MTI_ARRIBO_AERONAVE
[propel-creole-transform] Processing table: MTI_ADUANA_TC_SERVICIO
[propel-creole-transform] Processing table: MTI_DESPACHO_TC_SERVICIO
[propel-creole-transform] Processing table: MTI_TC_PUERTO
[propel-creole-transform] Processing table: MTI_DESPACHO_AERONAVE
[propel-creole-transform] Processing table: MTI_MANIF_BL
[propel-creole-transform] Processing table: MTI_ARRIBO_MTI
[propel-creole-transform] Processing table: MTI_MANIF_GRAL
[propel-creole-transform] Processing table: MTI_ATRAQUE
[propel-creole-transform] Processing table: MTI_DESPACHO
```

```
[propel-creole-transform] Processing table: MTI_TC_SERVICIO
[propel-creole-transform] Processing table: MTI_DESPACHO_TC_DOCUMENTO
[propel-creole-transform] Processing table: MTI_DESPACHO_BUQUE
[propel-creole-transform] Processing table: MTI_MANIF_ERROR
[propel-creole-transform] Processing table: MTI_ARRIBO_BUQUE
[propel-creole-transform] Processing table: MTI_MANIF_PART_EQUIP
[propel-creole-transform] Processing table: MTI_MANIF_RECHAZ
[propel-creole-transform] Processing table: MTI_MANIF_RECTIF
[propel-creole-transform] Processing table: MTI_PAIS
[propel-creole-transform] Processing table: MTI_TC_AGENCIA
[propel-creole-transform] Processing table: MTI_TC_CARGA
[propel-creole-transform] Processing table: MTI_TC_DOCUMENTO
[propel-creole-transform] Processing table: MTI_MANIF_PARTIDAS
[propel-creole-transform] Writing XML to file: /var/www/SUAbad/config/schema.xml
[propel-creole-transform] Propel - CreoleToXMLSchema finished

BUILD FINISHED

Total time: 5 minutes  56.16 seconds
```

Editando el fichero *schema.yml*

```
# gedit /var/www/SUA/config/schema.xml

<database name="propel" package="lib.model" defaultIdMethod="native"
noXsd="true">

>Modificar los campos de las tablas que tengan una seguencia:

<column name="ID_ACEPT" type="NUMERIC" size="22" primaryKey="true"
autoIncrement="true"/>
```

Generando el modelo

```
# symfony propel-build-model

>> schema    converting "/var/www/SUAbad/config/schema.yml" to XML
>> schema    putting /var/www/SUAbad/config/generated-schema.xml
Buildfile: /usr/share/php5/symfony/lib/vendor/propel-generator/build.xml
[resolvepath] Resolved /var/www/SUAbad/config to /var/www/SUAbad/config

propel-project-builder > check-project-or-dir-set:

propel-project-builder > check-project-set:

propel-project-builder > set-project-dir:

propel-project-builder > check-buildprops-exists:

propel-project-builder > check-buildprops-for-propel-gen:

propel-project-builder > check-buildprops:

propel-project-builder > configure:
     [echo] Loading project-specific props from
/var/www/SUAbad/config/propel.ini
 [property] Loading /var/www/SUAbad/config/propel.ini

propel-project-builder > om:
    [phing] Calling Buildfile '/usr/share/php5/symfony/lib/vendor/propel-
generator/build-propel.xml' with target 'om'
 [property] Loading /usr/share/php5/symfony/lib/vendor/propel-
generator/./default.properties

propel > check-run-only-on-schema-change:

propel > om-check:

propel > om:
     [echo] +-------------------------------------------+
     [echo] |                                           |
     [echo] | Generating Peer-based Object Model for    |
     [echo] | YOUR Propel project! (NEW OM BUILDERS)!   |
     [echo] |                                           |
     [echo] +-------------------------------------------+
[phingcall] Calling Buildfile '/usr/share/php5/symfony/lib/vendor/propel-
generator/build-propel.xml' with target 'om-template'
 [property] Loading /usr/share/php5/symfony/lib/vendor/propel-
generator/./default.properties
propel > om-template:
[propel-om] Target database type: oracle
[propel-om] Target package: lib.model
[propel-om] Using template path: /usr/share/php5/symfony/lib/vendor/propel-
generator/templates
```

```
[propel-om] Output directory: /var/www/SUAbad
[propel-om] Processing: generated-schema.xml
[propel-om] Processing: schema.xml
[propel-om] Processing Datamodel : JoinedDataModel
[propel-om]   - processing database :
[propel-om]   - processing database : propel
[propel-om]  + MTI_MANIF_ACEPT
[propel-om]         -> BaseMtiManifAceptPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiManifAcept [builder: SfObjectBuilder]
[propel-om]         -> MtiManifAceptMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiManifAceptPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiManifAcept [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_ADUANA
[propel-om]         -> BaseMtiAduanaPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiAduana [builder: SfObjectBuilder]
[propel-om]         -> MtiAduanaMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiAduanaPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiAduana [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_MANIF_EQUIP
[propel-om]         -> BaseMtiManifEquipPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiManifEquip [builder: SfObjectBuilder]
[propel-om]         -> MtiManifEquipMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiManifEquipPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiManifEquip [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_ARRIBO_AERONAVE
[propel-om]         -> BaseMtiArriboAeronavePeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiArriboAeronave [builder: SfObjectBuilder]
[propel-om]         -> MtiArriboAeronaveMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiArriboAeronavePeer [builder:
SfExtensionPeerBuilder]
[propel-om]         -> MtiArriboAeronave [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_ADUANA_TC_SERVICIO
[propel-om]         -> BaseMtiAduanaTcServicioPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiAduanaTcServicio [builder: SfObjectBuilder]
[propel-om]         -> MtiAduanaTcServicioMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiAduanaTcServicioPeer [builder:
SfExtensionPeerBuilder]
[propel-om]         -> MtiAduanaTcServicio [builder:
SfExtensionObjectBuilder]
[propel-om]  + MTI_DESPACHO_TC_SERVICIO
[propel-om]         -> BaseMtiDespachoTcServicioPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiDespachoTcServicio [builder: SfObjectBuilder]
[propel-om]         -> MtiDespachoTcServicioMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiDespachoTcServicioPeer [builder:
SfExtensionPeerBuilder]
[propel-om]         -> MtiDespachoTcServicio [builder:
SfExtensionObjectBuilder]
[propel-om]  + MTI_TC_PUERTO
```

```
[propel-om]         -> BaseMtiTcPuertoPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiTcPuerto [builder: SfObjectBuilder]
[propel-om]         -> MtiTcPuertoMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiTcPuertoPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiTcPuerto [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_DESPACHO_AERONAVE
[propel-om]         -> BaseMtiDespachoAeronavePeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiDespachoAeronave [builder: SfObjectBuilder]
[propel-om]         -> MtiDespachoAeronaveMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiDespachoAeronavePeer [builder:
SfExtensionPeerBuilder]
[propel-om]         -> MtiDespachoAeronave [builder:
SfExtensionObjectBuilder]
[propel-om]  + MTI_MANIF_BL
[propel-om]         -> BaseMtiManifBlPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiManifBl [builder: SfObjectBuilder]
[propel-om]         -> MtiManifBlMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiManifBlPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiManifBl [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_ARRIBO_MTI
[propel-om]         -> BaseMtiArriboMtiPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiArriboMti [builder: SfObjectBuilder]
[propel-om]         -> MtiArriboMtiMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiArriboMtiPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiArriboMti [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_MANIF_GRAL
[propel-om]         -> BaseMtiManifGralPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiManifGral [builder: SfObjectBuilder]
[propel-om]         -> MtiManifGralMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiManifGralPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiManifGral [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_ATRAQUE
[propel-om]         -> BaseMtiAtraquePeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiAtraque [builder: SfObjectBuilder]
[propel-om]         -> MtiAtraqueMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiAtraquePeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiAtraque [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_DESPACHO
[propel-om]         -> BaseMtiDespachoPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiDespacho [builder: SfObjectBuilder]
[propel-om]         -> MtiDespachoMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiDespachoPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiDespacho [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_TC_SERVICIO
[propel-om]         -> BaseMtiTcServicioPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiTcServicio [builder: SfObjectBuilder]
[propel-om]         -> MtiTcServicioMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiTcServicioPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiTcServicio [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_DESPACHO_TC_DOCUMENTO
```

```
[propel-om]         -> BaseMtiDespachoTcDocumentoPeer [builder:
SfPeerBuilder]
[propel-om]         -> BaseMtiDespachoTcDocumento [builder: SfObjectBuilder]
[propel-om]         -> MtiDespachoTcDocumentoMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiDespachoTcDocumentoPeer [builder:
SfExtensionPeerBuilder]
[propel-om]         -> MtiDespachoTcDocumento [builder:
SfExtensionObjectBuilder]
[propel-om]  + MTI_DESPACHO_BUQUE
[propel-om]         -> BaseMtiDespachoBuquePeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiDespachoBuque [builder: SfObjectBuilder]
[propel-om]         -> MtiDespachoBuqueMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiDespachoBuquePeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiDespachoBuque [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_MANIF_ERROR
[propel-om]         -> BaseMtiManifErrorPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiManifError [builder: SfObjectBuilder]
[propel-om]         -> MtiManifErrorMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiManifErrorPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiManifError [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_ARRIBO_BUQUE
[propel-om]         -> BaseMtiArriboBuquePeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiArriboBuque [builder: SfObjectBuilder]
[propel-om]         -> MtiArriboBuqueMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiArriboBuquePeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiArriboBuque [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_MANIF_PART_EQUIP
[propel-om]         -> BaseMtiManifPartEquipPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiManifPartEquip [builder: SfObjectBuilder]
[propel-om]             -> MtiManifPartEquipMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiManifPartEquipPeer [builder:
SfExtensionPeerBuilder]
[propel-om]         -> MtiManifPartEquip [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_MANIF_RECHAZ
[propel-om]         -> BaseMtiManifRechazPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiManifRechaz [builder: SfObjectBuilder]
[propel-om]         -> MtiManifRechazMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiManifRechazPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiManifRechaz [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_MANIF_RECTIF
[propel-om]         -> BaseMtiManifRectifPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiManifRectif [builder: SfObjectBuilder]
[propel-om]         -> MtiManifRectifMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiManifRectifPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiManifRectif [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI PAIS
```

```
[propel-om]         -> BaseMtiPaisPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiPais [builder: SfObjectBuilder]
[propel-om]         -> MtiPaisMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiPaisPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiPais [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_TC_AGENCIA
[propel-om]         -> BaseMtiTcAgenciaPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiTcAgencia [builder: SfObjectBuilder]
[propel-om]         -> MtiTcAgenciaMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiTcAgenciaPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiTcAgencia [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_TC_CARGA
[propel-om]         -> BaseMtiTcCargaPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiTcCarga [builder: SfObjectBuilder]
[propel-om]         -> MtiTcCargaMapBuilder [builder: SfMapBuilderBuilder]
[propel-om]         -> MtiTcCargaPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiTcCarga [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_TC_DOCUMENTO
[propel-om]         -> BaseMtiTcDocumentoPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiTcDocumento [builder: SfObjectBuilder]
[propel-om]         -> MtiTcDocumentoMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiTcDocumentoPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiTcDocumento [builder: SfExtensionObjectBuilder]
[propel-om]  + MTI_MANIF_PARTIDAS
[propel-om]         -> BaseMtiManifPartidasPeer [builder: SfPeerBuilder]
[propel-om]         -> BaseMtiManifPartidas [builder: SfObjectBuilder]
[propel-om]         -> MtiManifPartidasMapBuilder [builder:
SfMapBuilderBuilder]
[propel-om]         -> MtiManifPartidasPeer [builder: SfExtensionPeerBuilder]
[propel-om]         -> MtiManifPartidas [builder: SfExtensionObjectBuilder]

BUILD FINISHED

Total time: 1.7112 second
>> file-     /var/www/SUAbad/config/generated-schema.xml
```

Anexo 3. Implementando ExtJS para usarlo con Symfony

Copiar componentes

```
/var/www/SUA/web# mkdir js
/var/www/SUA/web# mkdir images
/var/www/SUA/web# mkdir css
/var/www/SUA/web# cd css
/var/www/SUA/web/css# mkdir ext
/var/www/SUA/web/css# cp home/treboR/Desktop/ext2.2.1/resources/*
/var/www/SUA/web/css
/var/www/SUA/web/css# cd ..
/var/www/SUA/web/# cd js
/var/www/SUA/web/js# mkdir ext
/var/www/SUA/web/js# cp home/treboR/Desktop/ext2.2.1/adapter/ext/ext-
base.js /var/www/SUA/web/js
/var/www/SUA/web/js# cp home/treboR/Desktop/ext2.2.1/ext-all.js
/var/www/SUA/web/js
/var/www/SUA/web/js# cp home/treboR/Desktop/ext2.2.1/ext-all-debug.js
/var/www/SUA/web/js
/var/www/SUA/web/js# cp home/treboR/Desktop/ext2.2.1/source/locale/ ext-
lang-sp.js /var/www/SUA/web/js
/var/www/SUA/web/js# cp /var/www/SUA/apps/despachoMTI/config/view.yml
/var/www/SUA/apps/despachoMTI/modules/MTI/config
```

Editando el fichero *view.yml* del módulo

```
# gedit /var/www/SUA/apps/despachoMTI/modules/MTI/config/view.yml

default:
  http_metas:
    content-type: text/html

  metas:
    title:        symfony project
    robots:       index, follow
    description:  symfony project
    keywords:     symfony, project
    language:     en

  stylesheets:    [ext/css/ext-all, ext/css/file-upload]

  javascripts:    [ext/ext-base, ext/ext-all, ext/ext-all-debug,
ext/locale/ext-lang-sp, common/ventana, common/UploadDialog,
common/funcionesComunes, common/FileUploadField, grid, Captar_msgBuque,
Captar_msgAeronave, baner, Reg_arriboBuque, Reg_arriboAeronave,
Prestar_ServBuque, Prestar_ServAeronave, servicios_Prestados,
Reg_despachoBuque, Reg_despachoAeronave, demanda, cosas, distribucion,
distribucio, probaractualizar, versubmit, mostrardistribucion, Main]

  has_layout:     on
  layout:         layout

>Guardar cambios
```

Anexo 4. Interfaces de usuario.

Controlador principal

```
/**
 * @author RCGA
 */
Ext.BLANK_IMAGE_URL = '../css/ext/images/default/s.gif';
Ext.onReady(function(){

    Ext.QuickTips.init();

    /**
     *       Panel Principal de la Aplicacion con Menu
     */
    var mainPanel = new Ext.Panel({
        id: 'id_mainPanelExt',
        layout: 'card',
        region: 'center',
        hideBorders: true,
        frame: true,
        autoScroll: true,
        activeItem: 0,
        items: [{
            frame: true
        }],
        tbar: [{
            text: 'Captar Mensaje',
            menu: [{
                text: 'Mensaje Buque',
                handler: function(){
                    Ext.getCmp('id_mainPanelExt').getLayout().setActiveItem(0);
                    var des = Ext.getCmp('captar_msgBuque');
                    if (des)
                        des.destroy();
                    var msgBuq = new Captar_msgBuque();
                    Ext.getCmp('id_mainPanelExt').add(msgBuq);
                    Ext.getCmp('id_mainPanelExt').doLayout();

Ext.getCmp('id_mainPanelExt').getLayout().setActiveItem('captar_msgBuque');
                }
            }, {
                text: 'Mensaje Aeronave',
                handler: function(){
                    Ext.getCmp('id_mainPanelExt').getLayout().setActiveItem(0);
                    var per = Ext.getCmp('captar_msgAeronave');
                    if (per)
                        per.destroy();
                    var msgAer = new Captar_msgAeronave();
                    Ext.getCmp('id_mainPanelExt').add(msgAer);
                    Ext.getCmp('id_mainPanelExt').doLayout();
Ext.getCmp('id_mainPanelExt').getLayout().setActiveItem('captar_msgAeronave');
                }
            }]
        }, '-', {
            text: 'Arribo',
            menu: [{
                text: 'Embarcaci&oacute;n',
```

```
handler: function(){
                Ext.getCmp('mainPanelExt').getLayout().setActiveItem(0);
                var serv = Ext.getCmp('Prestar_ServBuque');
                if (serv)
                    serv.destroy();
                var prestServ = new Prestar_ServBuque('margin-left: 15%; margin-
top: 1%; ');
                Ext.getCmp('mainPanelExt').add(prestServ);
                Ext.getCmp('mainPanelExt').doLayout();

Ext.getCmp('mainPanelExt').getLayout().setActiveItem('Prestar_ServBuque');
            }
                    },{
                           text: 'Aeronave',
                           handler: function(){

      Ext.getCmp('mainPanelExt').getLayout().setActiveItem(0);
                                  var serv = Ext.getCmp('Prestar_ServAeronave');
                                  if (serv)
                                         serv.destroy();
                                  var prestServ = new Prestar_ServAeronave('margin-
left: 15%; margin-top: 1%; ');
                                  Ext.getCmp('mainPanelExt').add(prestServ);
                                  Ext.getCmp('mainPanelExt').doLayout();

      Ext.getCmp('mainPanelExt').getLayout().setActiveItem('Prestar_ServAeronave');
                           }
                    }]

        }, '-']
    });

    /**
     * Viewport del módulo
     */
    var viewport = new Ext.Viewport({
        layout: 'border',
        region: 'center',
        hideBorders: true,
        items: [Banner, mainPanel]
    });

});
```

Anexo 5. MVC

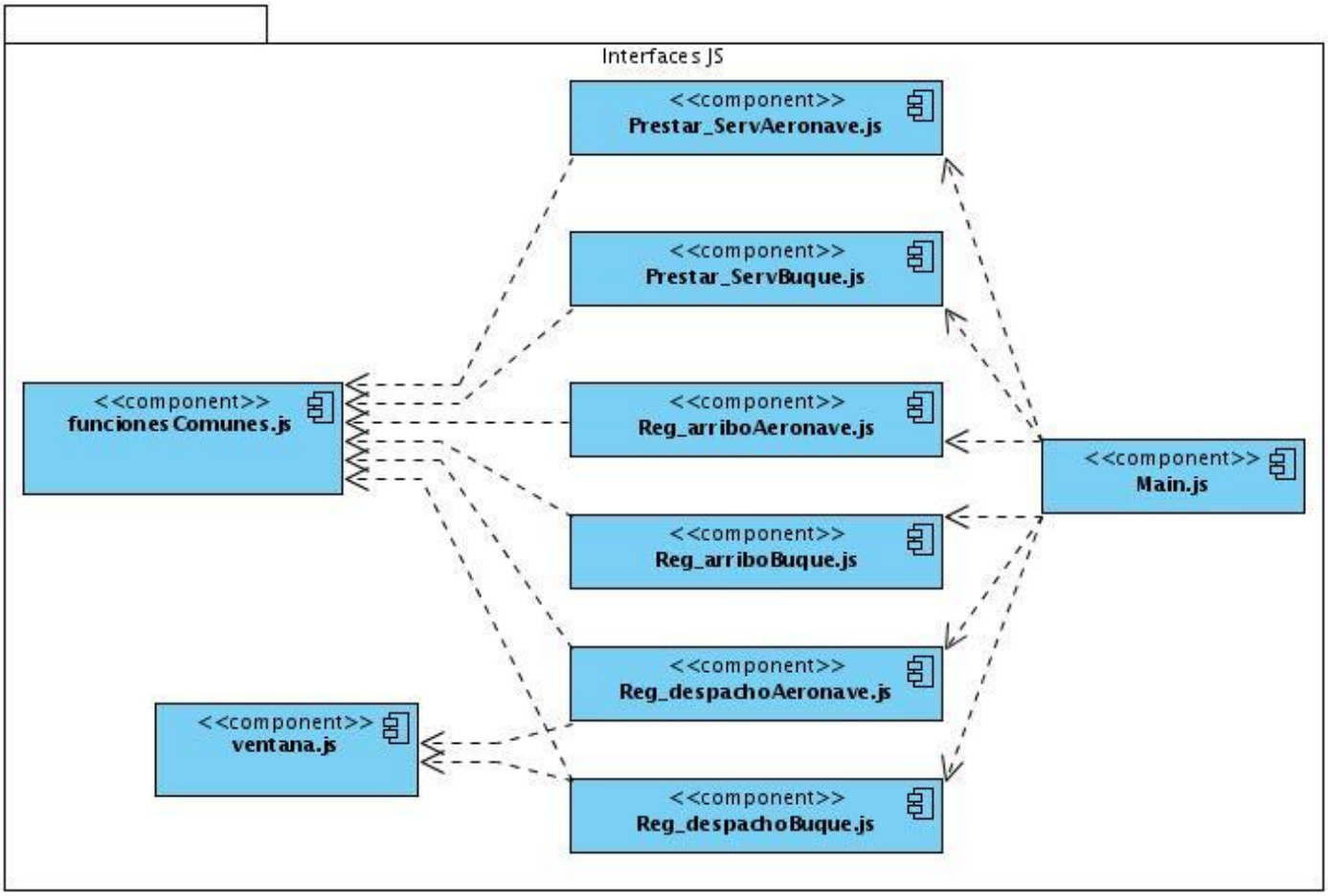
El controlador

```
<?php

/**
 * MTI actions.
 *
 * @package    SUA
 * @subpackage MTI
 * @author     Roberto Carlos Garcia Adino
 * @version    SVN: $Id: actions.class.php 2692 2006-11-15 21:03:55Z fabien $
 */
class MTIActions extends sfActions
{
  /**
   * Executes index action   *
   */
  public function executeIndex()
  {

  }
  .
  .
  .
  public function executeRegistrarArriboBuque(){
       $anno = $this->getRequestParameter('anno');
       $noEscala = $this->getRequestParameter('noEscala');
       $codImo = $this->getRequestParameter('codImo');
       $fechaArr = $this->getRequestParameter('fechaArr');
       $horaArr = $this->getRequestParameter('horaArr');
       $nombre = $this->getRequestParameter('nombre');
       $noManif = $noEscala."/".$anno;
       $idAduana = MtiAduana::ObtenerId("003");
/* los métodos _MtiArriboBuque y _MtiArriboMti funcionan como si fueran constructores
parámetros
 * los parámetros son los siguientes:
 * $noManif,$fechaArr,$idAd,$noEdi,$idManif,$codImo,$horaArr,$nombre,$tEmbarcacion
*/
       if(MtiArriboBuquePeer::ExisteArribo($noManif)!=true){
              $arribo = new MtiArriboBuque();
              $arribo-
>_MtiArriboBuque($noManif,$fechaArr,$idAduana,NULL,NULL,$codImo,$horaArr,$nombre,'yat

              $arribo->save();
              return $this->renderText("{success:true}");
       }
       else
       {
              return $this->renderText("{failure:true}");
       }
  }
  .
  .
  .
}
```

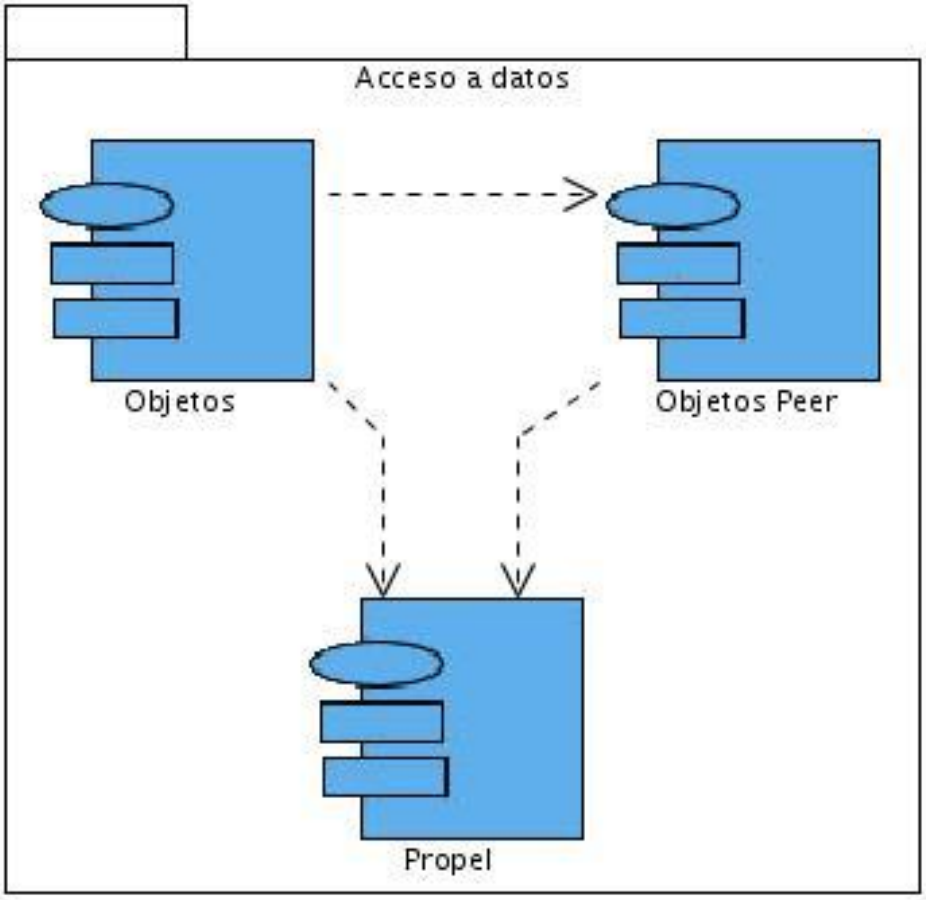
El modelo

```
<?php

/**
 * Subclass for representing a row from the 'MTI_ARRIBO_BUQUE' table.
 *
 *
 *
 * @package lib.model
 */
class MtiArriboBuque extends BaseMtiArriboBuque
{
	public function
_MtiArriboBuque($noManif,$fechaArr,$idAd,$noEdi,$idManif,$codImo,$horaArr,$nombre,$tE
	{
		$arribo = new MtiArriboMti();
		$arribo->_MtiArriboMti($noManif,$fechaArr,$idAd,$noEdi,$idManif);
		$this->setMtiArriboMti($arribo);
		$this->setCodImo($codImo);
		$this->setHoraArribo($horaArr);
		$this->setNombre($nombre);
		$this->setTipoEmbarcacion($tEmbarcacion);
	}
}

<?php

/**
 * Subclass for performing query and update operations on the 'MTI_ARRIBO_BUQUE' tabl
 *
 *
 *
 * @package lib.model
 */
class MtiArriboBuquePeer extends BaseMtiArriboBuquePeer
{
	/*
	 * Si existe el arribo para ese manifiesto, devuelve false
	*/
	public static function ExisteArribo($manif)
	{
		$criter = new Criteria();
		//print_r($manif);
		$criter->add(MtiArriboBuquePeer::NO_MANIFIESTO,$manif);
		$respuesta = self::doSelect($criter);
		if($respuesta){
			return true;
		}
		else{
			return false;
		}
	}

}
```

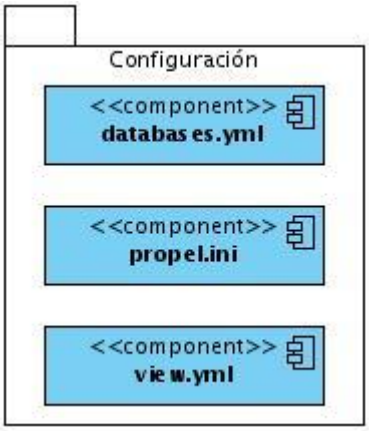
Anexo 6. Paquetes de componentes



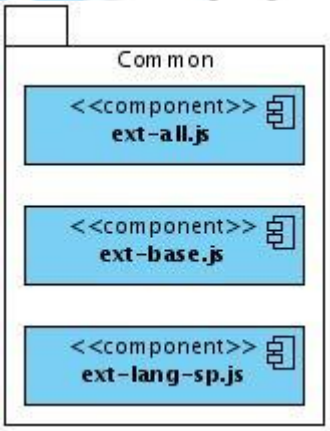
Paquete de componentes Interfaces JS



Paquete de componentes de acceso a datos



Paquete de componentes de configuración



Paquete de componentes comunes