Universidad de las Ciencias Informáticas

Facultad #3

Diseño e implementación del Módulo Despacho de las Embarcaciones de Recreo para el Sistema de Gestión Integral de la Aduana

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor:

Natacha Álvarez Rodríguez

Tutor:

Ing. Manuel R. Almaguer Ochoa

Ciudad de la Habana, Julio del 2011.

"Año 53 de la Revolución".

DECLARACIÓN DE AUTORÍA

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

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

Natacha Álvarez Rodríguez

_____

Firma del Autor

Ing. Manuel R. Almaguer Ochoa

_____

Firma del Tutor

RESUMEN

En la Aduana General de la República se desarrollan varios procesos, entre ellos el control y despacho de las Embarcaciones de Recreo, este proceso tiene como misión que las Embarcaciones de Recreo cumplan con las formalidades correspondientes a su entrada, permanencia y salida de una marina cubana. 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. Porlo que fue necesario, el estudio los artefactos generados en la etapa de análisis, y la documentación entregada por los clientes. Para poder desarrollar el diseño y la implementación se necesitó guiar el proceso de desarrollo del software a través de una metodología, para ello se partió de RUP pero dentro del proyecto se le hicieron algunas modificaciones y adaptaciones, que se ajustaran a las necesidades del proyecto, haciendo uso de alguno de los artefactos puntuales que propone la metodología en la etapa de diseño e implementación, como son, el Modelo de Diseño y el Modelo de Datos, dentro del Modelo de Diseño se generaron los diagramas de clases con estereotipos WEB y los diagramas de secuencia y se pasa a implementar la solución propuesta guiada por un estándar de codificación y se confeccionan los diagramas de componente y despliegue.Se valida el diseño obtenido haciendo uso de diferentes métricas, y se comprueba la solución con la realización de pruebas de caja negra.

Tabla de contenido

TABLA DE CONTENIDO

TABLA DE CONTENIDO

Introducción

La Aduana constituye una de las instituciones administrativas básicas en cualquier país. El Decreto-Ley No.162 de Aduanas del 3 de abril de 1996, en su Capítulo II, artículo 16, inciso a), dispone que la Aduana es la encargada de desarrollar la política y la base reglamentaria y de procedimientos para el ejercicio del control aduanero en la aplicación de los diferentes regímenes aduaneros, propiciando en coordinación con otros organismos vinculados, las medidas de facilitación que contribuyan a la agilización del comercio y del tráfico de viajeros.

También se define a la Aduana como el servicio público encargado de ejecutar el control aduanero aplicable a la entrada, el tránsito, el cabotaje, el trasbordo, el depósito y la salida del territorio nacional de mercancías, viajeros y sus equipajes, bienes y valores sujetos a regulaciones especiales y los medios en que se transporten.(1)

Aduana General de la República de Cuba (AGR), se encuentra enfrascada en la informatización o mejora de sus procesos y sistemas, para brindar a sus obreros una vía más cómoda y segura de trabajo, con el objetivo de ofrecer un servicio de mejor calidad.

La Aduana marítima garantiza que las Embarcaciones de Recreo (ER) cumplan con todas las formalidades correspondientes a su entrada, permanencia y salida de una marina cubana. En la actualidad estas actividades se realizan de forma manual lo que trae consigo una gran cantidad de trabajo que ocasiona, la demora del procesamiento de la información, y dificulta el acceso y revisión de los archivos, lo cual hace que sea más tedioso el control de las operaciones en las que está involucrada una ER, no solo para el inspector de aduana, sino también para el capitán de la misma.

El incremento de la actividad turística y pesquera a través de las fronteras marítimas de nuestro país trae consigo no solo el aumento de las actividades de carácter delictivo sino que el capitán de la ER tenga que esperar a que se realice el despacho de entrada o de salida de otra embarcación. Teniendo en cuenta la necesidad de unos servicios aduaneros más rápidos, se hace indispensable la creación de un sistema totalmente informatizado para gestionar el despacho de las Embarcaciones de Recreo que arriban o parten de las aduanas de Cuba.

Otra de las principales razones que impulsan esta investigación es el costo de licencia o de soporte técnico de las aplicaciones existentes que ofrecen este tipo de soluciones informáticas, en las cuales

se abundará más adelante, pues la actual situación económica de nuestro país, no puede permitirse sumas como esas. Además de esto, ninguno de estos sistemas se ajusta a las Normativas Aduaneras que se establecen en Decreto Ley De Aduanas de nuestra AGR.

Atendiendo a esas necesidades la estudiante Milena Ríos realizó con anterioridad un trabajo de diploma, el cual se enmarcó en el análisis y desarrollo de requerimientos del módulo Embarcaciones de Recreo perteneciente al subsistema Despacho de Medios de Transporte Internacional (MTI) del Sistema de Gestión Integral de la Aduana (GINA) en el cual se recogieron los principales requisitos, aspectos y procesos del negocio que debían ser automatizados, de las aduanas marítimas. El presente trabajo tiene como tarea principal diseñar e implementar un módulo de despacho para las Embarcaciones de Recreo que esté basado en los requerimientos anteriormente documentados y a la vez logre satisfacer todas las necesidades y características del proceso de control aduanero en las marinas cubanas específicamente para las Embarcaciones de Recreo.

Problema a resolver: ¿Cómo desarrollar el Módulo Embarcaciones de Recreo satisfaciendo los requisitos obtenidos en el proceso de análisis?

Objeto de estudio: Los sistemas informáticos para el Control Aduanero de Embarcaciones de Recreo.

Campo de acción: Diseño e implementación del Módulo Embarcaciones de Recreo para el Control Aduanero.

Objetivo general: Realizar el diseño y la implementación del Módulo Embarcaciones de Recreo para el Sistema de Gestión Integral de la Aduana que satisfaga los requisitos obtenidos en el proceso de análisis.

Objetivos específicos:

1. Elaboración de la fundamentación teórica de la investigación.
2. Realizar el diseño de la aplicación y validación del mismo.
3. Implementar la aplicación para crear las funciones definidas en los diagramas de diseño confeccionados.
4. Probar el correcto funcionamiento de la aplicación realizando las pruebas correspondientes.
5. Documentación de la aplicación.

Resultados Esperados:

1. Establecer las pautas a seguir en el diseño de la aplicación a implementar.
2. Diagramas de Interacción entre clases.
3. Diagrama de secuencias para los diferentes escenarios.
4. Código fuente de la aplicación.

Idea a Defender:

Con el diseño y la implementación del módulo Embarcaciones de Recreo se lograrán satisfacer los requisitos obtenidos en el proceso de análisis.

Para lograr llevar a cabo cualquier trabajo de investigación es necesario el uso de Métodos Científicos de Investigación como procedimientos que se utiliza para estudiar la realidad, la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones.

Entre los métodos utilizados en la siguiente investigación están los Métodos Teóricos que crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad, posibilitando el conocimiento del estado del arte, su evolución en una etapa determinada y su relación con otros fenómenos.

Dentro de los **Métodos Teóricos** se utilizó el **Método Analítico – Sintético**: El cual se puede aplicar al estudio de los factores que condicionan la situación problemática, además está presente en todo el desarrollo de la investigación al revelarse cada uno de los elementos que conforman el tema en cuestión, en sus detalles y partes componentes vistos como un todo integral. Permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.

Otros métodos utilizados fueron los **Métodos Empíricos**que permiten extraer de los fenómenos analizados las informaciones que se necesitan sobre ellos a través de técnicas tales como:

Consulta a Expertos, cuyo objetivo fundamental es la riqueza, profundidad y calidad de la información, dicha consulta fue aplicada a algunos especialistas de la Aduana General de la República y algunos profesores vinculados al proyecto Aduana, con el propósito de obtener una información certera acerca del objeto de estudio, pues ellos son los más indicados y conocedores del tema en cuestión.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se realiza una revisión de algunos conceptos fundamentales que se les hará referencia en todo el contexto del trabajo. Se realiza un breve análisis de la metodología de desarrollo de software a seguir así como las mejores prácticas de programación web, 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[1] a utilizar. Todas estas herramientas ya se encuentran definidas dentro del marco arquitectónico definido para los sistemas integrados al Departamento Productivo Sistemas Tributarios y de Aduanas.

1.2 Definiciones

1.2.1 ER

Según el Capítulo 1, articulo 4, de la Resolución 187-2008, se define como Embarcaciones de Recreo (ER), las que, con independencia de su medio de propulsión, son exclusivamente destinadas a la realización de actividades recreativas, turísticas o de descanso y a la pesca no profesional, sin ánimo de lucro.

1.2.2 MTI

Medios de Transporte Internacional, encargados del traslado de mercancías y/o personas de un país a otro y que son sometidos a controles aduaneros,que son los buques, las embarcaciones de recreo y las aeronaves.

1.2.3 GINA

La Aduana General de la República (AGR) cuenta en estos momentos con el Sistema de Gestión Integral de la Aduana (GINA) conformado por diferentes procesos, que tiene como objetivo automatizar

1 Framework: Marco de trabajo en el desarrollo de software.

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. GINA 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.

1.3 Aplicaciones de sistemas informáticos en el mundo para el Control Aduanero de Embarcaciones de Recreo

Seamax: Es un sistema que ofrece servicios de telecontrol, telemedida y telemando. Con la tecnología utilizada por este sistema se benefician sectores como la vigilancia aduanera, el control portuario, equipos de salvamento marítimo, etc. Como objetivo final de este proyecto es la consecuencia de un producto de telecomunicación adecuado a la realidad de las comunicaciones marítimas, que sea un referente a nivel mundial en este sentido.El Proyecto SeaMax surge en el ámbito de la Plataforma Tecnológica TIC VINDEIRA, aglutinando a cinco participantes del grupo de trabajo Tecnologías Radio de dicha plataforma: dos empresas (Quobis Networks SL y Wireless Galicia SL), dos grupos de investigación (Grupo TC-1 de la Universidad de Vigo y Grupo GTEC de la Universidad de Coruña) y el Instituto Tecnológico de Galicia (ITG). (2)

DIANA,Sistema de gestión y ayuda a la navegación en puertos deportivos: Es un sistema español para la gestión y ayuda a la navegación en los distintos puertos deportivos que tiene como objetivo ofrecer a las autoridades portuarias una herramienta que proporcione al navegante información de alta precisión para su acceso al puerto, actualización en tiempo real de los eventos que pongan riesgos potenciales y utilidades que faciliten la venta de servicios. También el sistema Diana se puede integrar con un Sistema de Gestión del Puerto (SGP) ya existente. (3)

Como se puede observar estos sistemas informáticos descritos anteriormente se basan más bien en la comunicación y posicionamiento de las embarcaciones de recreo y no se ajustan a las necesidades y normativas de la AGR, referentes al despacho de entrada, permanencia y salida de la ER de una aduana marítima. Además, están desarrollados con tecnología privativa lo que trae consigo un costo elevado para la adquisición de las licencias y han sido creados para resolver los problemas de entidades específicas.

1.4 Aplicación Web

Una aplicación web es un sistema informático o aplicación informática distribuida que cuenta con una interfaz de usuario que les permite a los clientes acceder a un servidor web a través de internet o de una intranet mediante un navegador web, este sistema de software le permite a los usuarios consultar gran cantidad de datos estructurados o no, procesados y actualizados mediante el navegador. Una de las características principales de estas aplicaciones es su alto grado de interacción con el usuario. Por otro lado, la complejidad de la funcionalidad que proveen dichas aplicaciones puede variar desde las más básicas (navegación/lectura, consultas, modificaciones) hasta servicios muy complejos del estilo transaccionales o de workflow.[2]

1.5 Mejores prácticas en la programación web

A la hora de desarrollar un software siempre los programadores experimentados y los que no lo son tanto pues siguen numerosas prácticas de programación o de reglas empíricas que por lo general son derivadas de las duras lecciones aprendidas. No obstante existen un sin número de prácticas de programación, de las cuales no todas son necesarias de utilizar, solo aquellas que se ajusten a las necesidades y al contexto de lo que se vaya a desarrollar.

Para lograr una buena calidad en la solución de un problema de desarrollo de software determinado, es necesaria la elección correcta de la metodología a seguir y las herramientas a utilizar que permitan la usabilidad, la accesibilidad, la adaptabilidad, la flexibilidad, y una buena configuración y seguridad del sistema.

Alguna de las prácticas de programación más utilizadas a la hora de desarrollar aplicaciones web son:

- ✓ La convención de código.
- ✓ La Programación Orientada a Objeto (POO).
- ✓ El Patrón Modelo - Vista – Controlador (MVC).
- ✓ Una buena estructura de directorios.
- ✓ La reutilización de código.

[2]Workflow: Flujo de Trabajo que hace referencia a la gestión electrónica de procesos de negocios.

Todas estas prácticas fueron tomadas en consideración a la hora de definir el marco arquitectónico de referencia para los sistemas integrados al Departamento Productivo Sistemas Tributarios y de Aduanas,permitiendo la integración de los sistemas que se están desarrollando y los que se creen en un futuro.

1.6 Metodologías de desarrollo de software

La metodología de desarrollo de software se define como “un conjunto de procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software”. (Piattini 1996).

La implantación de una metodología es necesaria si se quieren gestionar adecuadamente los proyectos, estas se encuentran agrupadas en dos grupos:

- ✓ Las *metodologías tradicionales o pesadas*, las cuales están guiadas por una fuerte planificación durante todo el proceso de desarrollo, en la cual se establecen estrictamente las actividades involucradas, los roles definidos, los artefactos que se deben producir, las herramientas y notaciones que serán utilizadas así como el modelado y documentación detallada.
- ✓ Las *metodologías ágiles o ligeras*, las cuales están más orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso. (14)

1.6.1 Metodologías que se utilizan en el mundo para el de diseño de aplicaciones web

Uno de los problemas con los que nos encontramos en el desarrollo de aplicaciones web, es que aún no existe una metodología universalmente aceptada que permita guiar aldesarrollador en el proceso de desarrollo. En cualquier caso existen criterios universales aceptados acerca del desarrollo de software. Por ejemplo según afirman algunos autores, “el modelo de proceso másadecuado para el desarrollo de software es un proceso iterativo e incremental, puesto que adiferencia de otros modelos de proceso, como por ejemplo el modelo en cascada, permite laobtención de diversas versiones del producto de software antes de la entrega final del mismo y ladepuración y validación progresiva del mismo, lo que sin duda redundará en un software mássatisfactorio para usuarios y clientes, (Jacobson et al (2000))”, o según indica Conallen (2000), “con este tipo de proceso es posible añadir o modificar requisitos que no han sido detectados con anterioridad”. Aún no existe ninguna propuesta universalmente aceptada

para el desarrollo Web, pero Fraternali (2000) indica que una posible solución al desarrollo adecuado de aplicaciones Web, sería combinar los ciclos de vida tradicionales con las propuestas de diseño para el desarrollo de aplicaciones hipermedia*.

La práctica ha demostrado que las aplicaciones Webrequieren un tratamiento especial durante el proceso de desarrollo, siendo necesario proponernuevos modelos y técnicas que den soporte al equipo de trabajo asegurando la calidad delproducto resultante, por lo que se han desarrollado técnicas y metodologías asociadas con el desarrollo de sistemas Web como es el caso de la metodología RMM (Relationship Management Methodology), la OOHDM (TheObject-OrientedHypermediaDesignModel), UWE: UML-Based Web Engineering, RNA: Relationship-NavegationalAnalysis, OOH-Method, entre otras.

Estas metodologías y técnicas cada uno de ellas enfocan mayormente su trabajo en la etapa de diseño del ciclo de vida y casi todas comparten las mismas faces de desarrollo en dependencia de las características de estas y las propuestas de solución que ofrezcan, como por ejemplo el diseño conceptual, el diseño abstracto de la Interfaz o UI, la implementación y las pruebas.

No obstante el uso de la Metodología RUP(RationalUnifiedProcess) para desarrollar aplicaciones Web, es igualmente válido ya que ante todo es un software, que se describe mediante funcionalidades, que debe desarrollarse teniendo en cuenta una arquitectura estable, y preferiblemente con un enfoque iterativo. Lo que evidentemente al tener un estilo arquitectónico particular, se deben tener en cuenta algunos métodos diferentes para el desarrollo y la implementación de los mismos; particularmente estas diferencias se ven con mayor claridad en el Flujo de Trabajo de Análisis y Diseño de RUP, ya que es en ese momento cuando se modela cómo internamente debe ser construido el sistema.

1.6.2 RationalUnifiedProcess (RUP)

El Proceso Unificado Racional o RationalUnifiedProcess en inglés es una metodología tradicional o pesada, que data de 1998, exactamente fue lanzada en junio de ese año. RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del leguaje de modelado UML (Lenguaje Unificado de Modelado).
RUP cuenta con cuatro fases de desarrollo que definen su ciclo de vida así como nueve flujos de trabajo a realizar en cada fase del proyecto, que a su vez, se dividen en flujos de trabajo de desarrollo y flujos de trabajo de soporte. (Ver Figura 1)(15)

Faces

- ✓ Inicio: El objetivo es determinar la visión del proyecto y definir lo que se desea realizar.
- ✓ Elaboración: Etapa en la que se determina la arquitectura óptima del proyecto.
- ✓ Construcción: Se obtiene la capacidad operacional inicial.
- ✓ Transmisión: Obtener el producto acabado y definido.

Flujos de trabajo de desarrollo

- ✓ Ingeniería o modelado del negocio: Analizar y entender las necesidades del negocio para el cual se está desarrollando el software.
- ✓ Requisitos: Proveer una base para estimar los costos y tiempo de desarrollo del sistema.
- ✓ Análisis y diseño: Trasladar los requisitos analizados anteriormente, a un sistema automatizado y desarrollar una arquitectura para el sistema.
- ✓ Implementación: Crear software que se ajuste a la arquitectura diseñada y que tenga el comportamiento deseado.
- ✓ Pruebas: Asegurarse de que el comportamiento requerido es correcto y que todo lo solicitado está presente.
- ✓ Despliegue: Producir distribuciones del producto y distribuirlo a los usuarios.

Flujo de trabajo de soporte

- ✓ Configuración y administración del cambio: Guardar todas las versiones del proyecto.
- ✓ Administración del proyecto: Administrar los horarios y recursos que se deben de emplear.
- ✓ Ambiente: Administrar el ambiente de desarrollo del software.

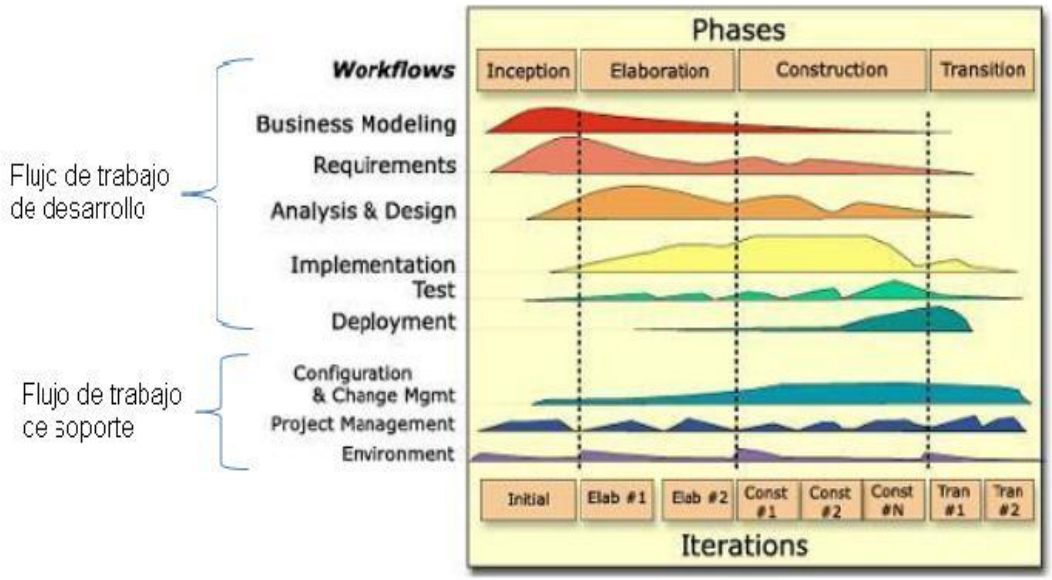


Figura 1 Fases e Iteraciones de la Metodología RUP

RUP presenta 3 características fundamentales que son:

- ✓ *Dirigido por casos de uso*: que inician el proceso de desarrollo proporcionando un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo.
- ✓ *Centrado en la arquitectura*: involucra los aspectos estáticos y dinámicos más significativos del sistema, está relacionada con la toma de decisiones que indican cómo tiene que ser construido el sistema y ayuda a determinar en qué orden.
- ✓ *Iterativo e Incremental*: consta de una secuencia de iteraciones en cascada y cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura, al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

1.6.2.1 Etapa de Diseño y Extensiones para el diseño Web

Como se había planteado anteriormente los sistemas informáticos en la Web requieren de un tratamiento especial y actualmente no existe una metodología aceptada que permita guiar aldesarrollador en el proceso de desarrollo por lo que se han propuesto nuevos modelos y técnicas asociadas con el desarrollo de aplicaciones Web que enfoca mayormente su trabajo en la etapa de

diseño del ciclo de vida. También se plantea que el uso de RUP a pesar de ser una metodología tradicional es válido en el desarrollo de aplicaciones Web.

Para lograr que RUP se adecue a las características de las aplicaciones Web se deben tener en cuenta algunos métodos diferentes, para el desarrollo y la implementación de los mismos, fundamentalmente en la etapa de diseño. Además destacar que en esta etapa modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis (conformado por el modelo conceptual, la realización de caso de uso, entre otros), que proporciona una comprensión detallada de los requisitos e impone una estructura del sistema que debemos esforzarnos por conservar lo más fielmente posible cuando demos forma al sistema.

Los principales artefactos obtenidos en la etapa de diseño son:

1. Modelo de Diseño.
   - ✓ Diagramas de Clases del Diseño y Diagramas de interacción (Colaboración y/o Secuencia).
   - ✓ Paquetes y Subsistemas de Diseño
   - ✓ Clases, interfaces, relaciones, contenidas en los Paquetes y una breve descripción de ellos
2. Modelo de Despliegue.
3. Documento de la Descripción de la Arquitectura.
4. Modelo de Datos.

Los artefactos Modelo de Despliegue y Documento de la Descripción de la Arquitectura en el Proyecto Aduana no son generados por el diseñador sino por el arquitecto de proyecto.

Los artefactos obtenidos en la etapa de diseño deberán adecuarse para que no existan confusiones ya que por ejemplo cuando miramos una clase como una página Web en el modelo, no sabríamos distinguir que atributos, operaciones y a veces relaciones, están activas en el servidor (cuando se está preparando la página) y cuales están activas cuando el usuario está interactuando con la página en el navegador cliente. Por este motivo no podemos mapear una página Web como una clase UML pues no comprenderíamos bien el sistema.

Para resolver este problema los especialistas del Rational decidieron modelar los aspectos del lado del servidor como una clase y los aspectos del lado del cliente como otra clase; para ello se plantearon la

creación de una extensión al modelo de análisis y diseño que permitiera representar el nivel de abstracción adecuado, y la relación con los restantes artefactos de UML.

De forma general los mecanismos de extensión están compuestos por:

1. Estereotipos: nos permite definir un nuevo significado de la semántica para el elemento a modelar.
2. Valores etiquetados: son extensiones de las propiedades o atributos de los elementos a modelar.
3. Restricciones: son reglas que definen la buena forma del modelo.

Estereotipos más usados:

- ✓ *Estereotipos para las clases.*

<<Server Page>> Representa la página Web que tiene código que se ejecuta en el servidor. Este código interactúa con recursos en el servidor.

<<Client Page>> Una instancia de Página Cliente es una página Web, con formato HTML. Mezcla de datos, presentación y lógica.

<<Form>> Colección de elementos de entrada (input boxes, textareas, radio buttons, check boxes y hiddenfields) que son parte de un página cliente. Se relaciona directamente con la etiqueta de igual nombre del HTML.

- ✓ *Estereotipos para las relaciones entre clases.*

<<Build>>Representa una asociación especial que relaciona las páginas cliente con las páginas servidor, de forma general se expresa como que las páginas que se encuentran en el servidor construyen las páginas en el cliente. Debe ser una relación direccional.

<<Link>>Expresa las asociaciones más comunes entre las páginas, en este caso la del hipervínculo; esta asociación siempre se origina desde una página cliente y apunta hacia otra página cliente o una página de servidor.

<<redirect>> La página de servidor puede redireccionar el procesamiento a otra página.

<<include>> Una página de servidor puede incluir a otra página de servidor.

<<Submit>> Es la relación que se crea siempre, entre una página servidor y un formulario, a través de esta relación el formulario manda los valores de sus campos al servidor, para ser procesados por la página servidor.

A modo de conclusión se puede utilizar la siguiente tabla para organizar las combinaciones permitidas para cada estereotipo. (Ver Tabla 1)

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

Tabla 1Relaciones entre clases que conforman la extensión UML para diseño Web

1.7 Lenguaje Unificado de Modelado (UML)

Lenguaje Unificado de Modelado o *UnifiedModelingLanguage*eninglés,es un estándar, no existe otra especificación de diseño orientado a objetos, ya que es el resultado de las tres opciones existentes en el mercado. Su utilización es independiente del lenguaje de programación y de las características de los proyectos.

Este lenguaje de modelado permite visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software, además describe lo que supuestamente hará un sistema, pero no dice cómo implementar dicho sistema.

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común, para todos los desarrollos se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo. (15)

1.8 Técnicas de Programación

La forma en que se especifique y elabore la solución de un problema determinado es a lo que se llama técnica de programación.

Las técnicas de programación son aplicables cuando se desea trasladar a la computadora el funcionamiento de un proceso, o la solución a un problema determinado, para ello se realiza una abstracción, o sea, un modelo simplificado de la realidad tomando los elementos más significativos y transformándolos en variables, de esta forma la computadora podrá entenderlo obteniendo así el resultado que se estaba buscando. (Müller, 1997).

A continuación se mencionan algunas de las técnicas de programación más conocidas.

- ✓ Programación no Estructurada.
- ✓ Programación Procedimental.
- ✓ Programación Modular.
- ✓ Programación Orientada a Objeto (POO).
- ✓ Programación Orientada a Servicios.
- ✓ Programación Orientada a Aspectos (AOP - Aspect-OrientedProgramming).

El uso de una o más de estas técnicas está en dependencia de la complejidad del problema que se necesite resolver, aunque vale la pena aclarar que cada una de estas técnicas usa a sus predecesoras, ya sea de una forma o de otra.

1.8.1 Programación Orientada a Objeto (POO)

La Programación Orientada a Objetos es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. (5)

Dicho paradigma es sin lugar a dudas el más utilizado por las empresas de todo el mundo a la hora de encarar desarrollos de aplicaciones de software, ya que permite representar de manera relativamente simple modelos y realidades muy complejas y esto hace que el software sea más fácil de programar, comprender y mantener.

1.9 Lenguajes de Programación

Actualmente existen diferentes lenguajes de programación para desarrollar en la Web, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas. Desde los inicios de Internet, fueron surgiendo diferentes demandas por los usuarios y se dieron soluciones mediante lenguajes

estáticos. A medida que pasó el tiempo, las tecnologías fueron desarrollándose y surgieron nuevos problemas a dar solución. Esto dio lugar a desarrollar lenguajes de programación para la Web dinámica, que permitieran interactuar con los usuarios y utilizaran sistemas de Bases de Datos.

1.9.1 PHP 5

PHP es el acrónimo de "HipertextPreprocesor" y es un lenguaje script que corre del lado del servidor en la Arquitectura Cliente – Servidor. 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. Es orientado a objetos y está ampliamente difundido en todo el mundo contando con un amplio volumen de documentación.

Ventajasquepresenta:

- ✓ Muy fácil de aprender.
- ✓ Se caracteriza por ser un lenguaje muy rápido.
- ✓ Soporta en cierta medida la orientación a objeto. Clases y herencia.
- ✓ Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- ✓ Capacidad de conexión con la mayoría de los manejadores de base de datos: MysSQL, PostgreSQL, Oracle, MS SQL Server, entre otras.
- ✓ Capacidad de expandir su potencial utilizando módulos.
- ✓ Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones.
- ✓ Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- ✓ Incluye gran cantidad de funciones.
- ✓ No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

1.9.2 JavaScript

JavaScript es un lenguaje de programación interpretado, es decir que no requiere compilación, no es orientado a objeto pero todo su trabajo es basado en objetos.

Se utiliza fundamentalmente para el desarrollo de páginas Web dinámicas y 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.

1.10 Herramientas de desarrollo

1.10.1 Herramientas CASE

Herramientas CASE (ComputerAided Software Engineering, Ingeniería de Software Asistida por Ordenador), se pueden definir como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software (Investigación Preliminar, Análisis, Diseño, Implementación e Instalación). (6)

CASE es también definido como el conjunto de métodos, utilidades y técnicas que facilitan el mejoramiento del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. (6)

1.10.1.1 Visual Paradigm para UML

Es una herramienta que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objeto, construcción, pruebas y despliegue, ayudando así a construir aplicaciones de alta calidad y a un menor coste. (17)

Entre algunas características que se destacan de esta herramienta CASE tenemos:

- ✓ Entorno de creación de diagramas para UML 2.1.
- ✓ Modelado colaborativo con CVS y Subversion.
- ✓ Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- ✓ Generador de informes para generación de documentación.
- ✓ Disponibilidad de múltiples versiones, para cada necesidad.
- ✓ Disponibilidad de integrarse en los principales IDEs, (Netbeans, Eclipse, etc.).
- ✓ Disponibilidad en múltiples plataformas (Windows/Linux/Mac OS X).
- ✓ Soporta una gama de lenguajes en la Generación de Código e Ingeniería Inversa en Java, C++, CORBA IDL, PHP, Esquema de XML, Ada y Python.

1.10.2 Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado o IntegratedDevelopmentEnvironment en inglés, consiste básicamente en un software que previamente ha sido instalado en la máquina del cliente y cuyo principal objetivo es el desarrollo de otro software, permite mantener proyectos informáticos implementados en diferentes lenguajes de programación así como realizar una serie de operaciones básicas sobre ellos.

1.10.2.1 NetBeans IDE 6.8

NetBeans IDE es un reconocido entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. El proyecto de NetBeans está formado por un IDE de código abierto y una plataforma de aplicación que permite a los desarrolladores crear con rapidez aplicaciones Web, empresariales, de escritorio y móviles, utilizando la plataforma Java, así como JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby onRails, Groovy and Grails y C/C++. (7)

El proyecto de NetBeans está apoyado por una comunidad de desarrolladores dinámica y ofrece documentación y recursos de formación exhaustivos, así como una selección diversa de complementos de terceros, además de que es gratuito y sin restricción de uso.

NetBeans IDE 6.8 es el primer IDE en ofrecer compatibilidad para todas las especificaciones de Java EE 6, con compatibilidad mejorada para JSF 2.0/Facelets, Java Persistence 2.0, EJB 3.1 (incluido el uso de EJB en aplicaciones Web), servicios Web RESTful y GlassFish v3. También lo recomendamos para desarrollar con el último JavaFX SDK 1.2.1 y para crear aplicaciones Web con la nueva versión PHP 5.3 o con la estructura Symfony.

Funciones más importante en PHP:

- ✓ Compatibilidad total con PHP 5.3: espacios de nombre, funciones y clausuras lambda, adiciones a la sintaxis: NOWDOC, condiciones ternarias, etiquetas de salto, _callStatic().
- ✓ Compatibilidad con Symfony Framework: proyectos de Symfony, comandos de Symfony, métodos abreviados de teclado, resaltado de sintaxis PHP en archivos YAML.
- ✓ Creación de proyectos PHP desde aplicaciones PHP remotas.
- ✓ PHPUnit, Código de cobertura, mejoras en la integración de FTP/SFTP, exclusión de las carpetas del proyecto PHP de exploraciones e indexaciones.

1.10.3 Framework

Un Framework o marco de trabajo puede definirse como una solución completa que contemplan herramientas de apoyo para el desarrollo y/o implementación de una aplicación. Además representa a una estructura de software compuesta decomponentes personalizables e intercambiables para el desarrollo deuna aplicación. En otras palabras, un framework se puede considerarcomo una aplicación genérica incompleta y configurable a la quepodemos añadirle las últimas piezas para construir una aplicacióncorrecta. (8)

Los objetivos principales que persigue un framework son: acelerar elproceso de desarrollo, reutilizar código ya existente y promoverbuenas prácticas de desarrollo como el uso de patrones.

Hoy en día la utilización de un framework es indispensable a la hora de realizar algún tipo de software o aplicación ya que este brinda al desarrollador ciertas ventajas que facilitan el trabajo a realizar.

Algunas de estas ventajas son:

- ✓ El programador no necesita plantearse una estructura global de la aplicación, sino que el framework le proporciona un esqueleto que hay que "rellenar".
- ✓ Permite seguir determinadas pautas que facilitan el trabajo de desarrollo y mantenimiento: separación de presentación y lógica, una sintaxis coherente, entre otras cosas.
- ✓ Facilita la colaboración. Cualquiera que haya tenido que "pelearse" con el código fuente de otro programador (¡o incluso con el propio, pasado algún tiempo!) sabrá lo difícil que es entenderlo y modificarlo; por tanto, todo lo que sea definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.
- ✓ Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al framework concreto para facilitar el desarrollo.

1.10.3.1 Symfony

Symfony es un framework PHP que facilita el desarrollo de aplicaciones Web, es considerado el mejor framework para crear estos tipos de aplicaciones y ha sido probado en alguno de los sitios Webmás grandes del mundo como por ejemplo Yahoo! Answers, Dailymotion, Delicious, además es el framework más documentado del mundo, ya que cuenta con miles de páginas de documentación distribuidas en varios libros gratuitos y decenas de tutoriales. (9)

Características:

- ✓ Fácil de instalar y configurar en sistemas Windows, Mac y Linux.
- ✓ Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server).
- ✓ Compatible solamente con PHP 5 desde hace años, para asegurar el mayor rendimiento y acceso a las características más avanzadas de PHP.
- ✓ Preparado para aplicaciones empresariales, ya que se puede adaptar con facilidad a las políticas y arquitecturas propias de cada empresa u organización.
- ✓ Flexible hasta cualquier límite y extensible mediante un completo mecanismo de plugins.
- ✓ Publicado bajo licencia MIT (Massachusetts Institute of Technology) de software libre y apoyado por una empresa comprometida con su desarrollo.
- ✓ Sigue la mayoría de mejores prácticas y patrones de diseño para la Web.
- ✓ Traducido a más de 40 idiomas y fácilmente traducible a cualquier otro idioma.

1.10.3.2 Ext.js

ExtJS es una biblioteca de JavaScript para el desarrollo de aplicaciones Web interactivas usando tecnologías como AJAX, DHTML y DOM. Originalmente fue construida como una extensión de la biblioteca YUI, en la actualidad puede usarse como extensión para las bibliotecas jQuery y Prototype.

ExtJS es una librería construida con JavaScript que proporciona una interfaz cuya potencia radica en la rica colección de componentes para el diseño de interfaces del lado del cliente. Tiene incluidos la mayoría de los controles de los formularios Web incluyendo tablas para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramientas, menús y muchos otros. Dentro de su librería de componentes incluye componentes para el manejo de datos, lectura de XML, lectura de datos JSON e implementaciones basadas en AJAX. Presenta el uso de JavaScript con una programación orientada a objetos. (10)

1.10.4 Servidor Web

Un servidor Web es un programa que se puede estar ejecutando continuamente en una computadora, que sirve para atender y responder peticiones de los clientes (navegadores de internet) proporcionando los recursos que soliciten, usando el protocolo HTTP o el protocolo HTTPS (la versióncifrada y autenticada).

1.10.4.1 Apache

El servidor Web Apache es desarrollado por el ApacheServer Project (Proyecto Servidor Apache) cuyo objetivo es la creación de un servidorweb fiable, eficiente y fácilmente extensible con código fuente abierto gratuito. Corre en plataformas Unix (BSD, GNU/LINUX, otras), Windows, Macintosh, entre otras que implementen el protocolo HTTP/1.1. Presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido. (4)

Ventajas:

- ✓ Arquitectura modular. Los usuarios de Apache pueden adicionar fácilmente funcionalidad a sus ambientes específicos.
- ✓ Portabilidad. Apache trabaja sobre todas las versiones recientes de UNIX y Linux, Windows, BeOs, mainframes.
- ✓ Es robusto y seguro.
- ✓ Es una tecnología gratuita de código fuente abierto.
- ✓ Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- ✓ Permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en tu servidor.

Además el servidor Web Apache es uno de los mejores y el más utilizado entre los servidores Web que existen.

1.11 Sistema Gestor de Base de Datos (SGDB)

Un Sistema Gestor de Base de Datos, es una aplicación que permite a los usuarios definir, crear, mantener y relacionar las bases de datos y proporciona un acceso controlado a las mismas. Es la aplicación que interactúa con los usuarios de los programas de aplicación y la base de datos. 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. (11)

1.11.1 Oracle

Oracle es un Sistema de Gestión de Base de Datos Relacional o RDBMS por el acrónimo en inglés de Ralational Data Base Management System desarrollado por Oracle Corporation.

Podemos destacar como características que:

- ✓ Se basa en la tecnología cliente/servidor.
- ✓ Tiene soporte de transacción.
- ✓ Presenta estabilidad.
- ✓ Escalabilidad.
- ✓ Soporte multiplataforma.

Oracle es el mayor y más usado Sistema Manejador de Base de Dato Relacional (RDBMS) en el mundo.

1.12 Patrones de Diseño

Algo importante a tener en cuenta a la hora de desarrollar una aplicación, es la importancia de la reutilización, ya que nos ahorra tiempo a la hora de programar, y una de las cosas que nos permite aplicar esto, son los patrones de diseño, que son soluciones reutilizables a los problemas de programación que nos encontramos diariamente.

Un patrón de diseño no es una clase o una biblioteca que puede ser conectado a nuestro sistema, sino algo más, es una plantilla que tiene que aplicarse a la situación correcta, por lo que si se aplica en el lugar equivocado puede ser un arma de doble filo que podría ser desastroso para nuestra aplicación y crearía muchos problemas. Sin embargo, implementándolo en el lugar correcto y en el momento adecuado, puede ser nuestro salvador.

Existen básicamente, 3 tipos de patrones de diseño:

Estructural: Los patrones estructurales, generalmente crean relación entre las entidades, lo que facilita que estas entidades puedan trabajar en conjunto.

Ejemplos:

- ✓ AdapterPattern (patrón adaptador).
- ✓ DecoratorPattern (patrón decorador).

Creacional: Los patrones creacionales indican mecanismos de instanciación (Crea un objeto, instancia de una clase); por lo que, facilitan la creación de objetos de una manera que se adapte a la situación.

Ejemplos:

- ✓ Factory MethodPattern (patrón método de fábrica).
- ✓ SingletonPattern (patrón instancia única).

De comportamiento: Los patrones de comportamiento son usados en lacomunicación entre entidades y hacen más fácil y flexible que estas entidades puedan comunicarse.

Ejemplos:

- ✓ StrategyPattern (patrón de estrategia).
- ✓ ObserverPattern (patrón observador).
- ✓ Chain-of-CommandPattern.

Ahora existen algunos patrones comunes en PHP, que son los más destacados y más utilizados en la programación como son:

Factory MethodPattern: El patrón método de fábrica es una clase que actúa como una fábrica de instancias de objetos cuyo objetivo principal es encapsular el procedimiento creacional que diferentes clases pueden tener, en una sola función. Al proporcionar el contexto adecuado al método de fábrica, éste será capaz de devolver el objeto correcto.

SingletonPattern: El patrón de diseño Singleton se asegura de tener una sola instancia de una clase particular durante su tiempo de ejecución, y proporciona un punto de acceso global a ella.

DecoratorPattern: El patrón decorador nos permite añadir comportamientos nuevos, o adicionales, a un objeto en tiempo de ejecución, dependiendo de la situación.

StrategyPattern: El patrón de estrategia nos permite decidir qué curso de acción debería tener un programa, basado en un contexto específico en tiempo de ejecución. El programa encapsula dos algoritmos diferentes dentro de dos clases y decide, en tiempo de ejecución, que estrategia debe seguir.

También existen los Patrones de Software para la Asignación General de Responsabilidades (GRASP). Estos patrones describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades constituyendo un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.

En GRASP se pueden destacar 5 patrones principales que son:

- ✓ Experto: La responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo.
- ✓ Creador: Ayuda a identificar quien debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.
- ✓ Alta cohesión: La información que almacena una clase debe ser coherente y debe estar (en la medida de lo posible) relacionada con la clase.
- ✓ Bajo acoplamiento: Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.
- ✓ Controlador: Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

1.12.1 Patrón MVC

Modelo Vista Controlador es un patrón de arquitectura de software dividido en tres capas que separa la lógica de negocio de la interfaz de usuario, facilitando la evolución por separado de ambos aspectos e incrementala reutilización y flexibilidad. (16)

El *modelo* es el encargado de guardar los datos en un medio persistente y de la lógica de negocio, la *vista*se encarga de presentar la interfaz al usuario, en sistemas Web, esto es típicamente HTML y el

código que provee datos dinámicos a la página y el *controlador* maneja y responde las solicitudes del usuario, procesando la información necesaria y modificando el modelo en caso necesario.

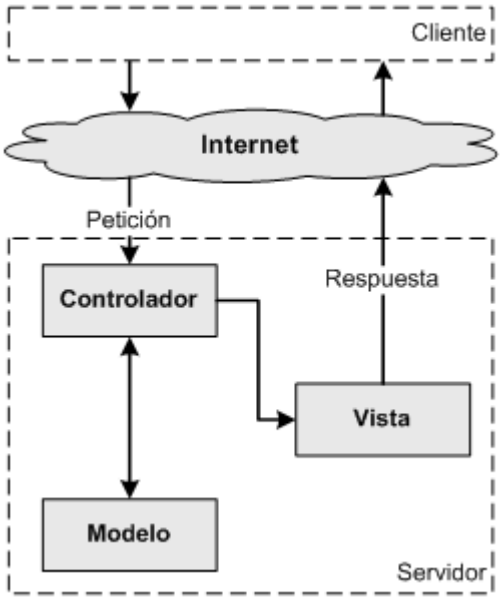


Figura 2 Patrón arquitectónico MVC

1.13 Conclusiones Parciales

Como resultado de la investigación a lo largo del capítulo han sido expuestos los principales puntos de interés abordados en la investigación, donde se analizaron las mejores prácticas de programación y se abordó sobre la metodología a utilizar en este caso RUP y como ésta se puede adaptar para que sea utilizada para desarrollar aplicaciones Web, donde se obtuviera una buena calidad y una buena documentación del mismo. Se mencionan las herramientas más importantes a emplear, entre ellas, el Visual Paradigm para el modelado de la solución, la implementación se realizará mediante los lenguajes de programación Web PHP y JavaScript, el servidor será Apache, el Sistema Gestor de Base de Datos Oracle y el frameworkSymfony para facilitar el desarrollo, así como el estudio de algunos patrones de diseño.

CAPÍTULO 2: Diseño de la Solución

2.1 Introducción

En el presente capítulo se describe el diseño del módulo Embarcaciones de Recreo para el Sistema de Gestión Integral de la Aduana. 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 cómo se adapta el módulo ER a la arquitectura definida para el GINA.

2.2 Patrones de diseño utilizados

Para el desarrollo del sistema como se había planteado en el capítulo anterior se enmarca dentro del marco arquitectónico definido para los sistemas integrados al Departamento Productivo Sistemas Tributarios y de Aduanas, donde se trata de utilizar aquellas tecnologías, métodos y herramientas que proporcionen las mejores prácticas de programación, como es el caso del frameworkSymfony que 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.

2.2.1Patrones GRASP implementados.

Creador

La clase actions.class.php contiene las acciones definidas para el módulo Embarcaciones de Recreo 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 (mti_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. Además cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, la implementación de ciertas responsabilidades en las clases *Peer que puedan ser reutilizadas es un ejemplo de alta cohesión. La incorporación de funciones reutilizables por los distintos subsistemas de la aplicación a las clase *Acces*también es una evidencia de alta cohesión en la solución propuesta.

2.2.2 Patrón GOF utilizado.

Singleton: Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

El subsistema para el despacho de las embarcaciones de recreo necesita consultar de forma reiterada cientos de datos que son gestionados por el plugin*sfUtilPlugin*, con el objetivo de agilizar estas consultas se realizó la implementación de la clase *Acces*mediante el patrón *Singlenton*, esta clase carga en memoria un grupo de arreglos con los datos más usados que son útiles para el despacho (y para el resto de los subsistemas de sfMtiPlugin), de manera que sean recuperados una única vez y puedan ser consultados desde todos los subsistemas, disminuyendo el tiempo de ejecución de las distintas funcionalidades.

2.2.3Patrón 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.

Symfony toma lo mejor del patrón arquitectura MVC y lo implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo, quedando dividido el contenido de cada capa como se muestra a continuación. (21)

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

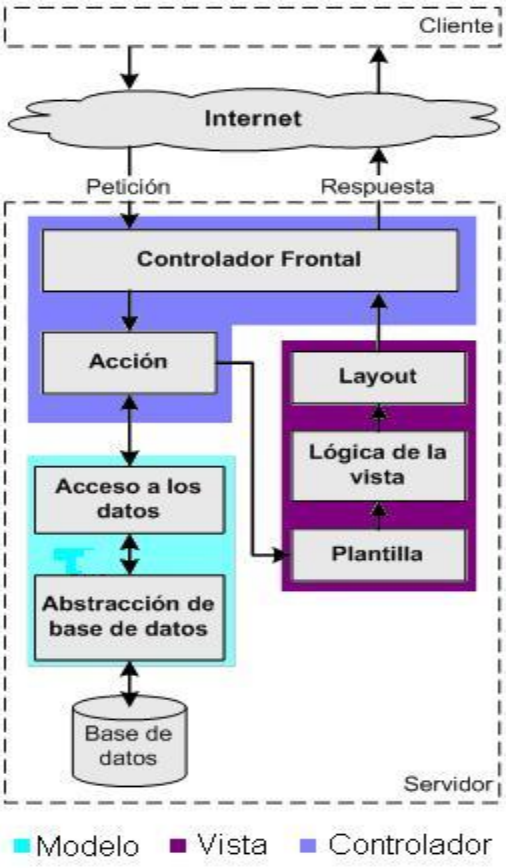


Figura 3 MVC en Symfony

La capa del Modelo

La capa del modelo se puede dividir en la capa de acceso a los datos y en la capa de abstracción de la base de datos. De esta forma, las funciones que acceden a los datos no utilizan sentencias ni consultas que dependen de una base de datos, sino que utilizan otras funciones para realizar las consultas. Así, si se cambia de sistema gestor de bases de datos, solamente es necesario actualizar la capa de abstracción de la base de datos.

La capa de la Vista

La capa de la vista también puede aprovechar la separación de código este caso en un layout y en una plantilla. Normalmente, el layout es global en toda la aplicación o al menos en un grupo de páginas. La plantilla sólo se encarga de visualizar las variables definidas en el controlador.

La capa del Controlador

Entre las tareas comunes de un controlador se encuentran el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares. Por este motivo, el controlador normalmente se divide en un controlador frontal, que es único para cada aplicación, y las acciones, que incluyen el código específico del controlador de cada página.

Una de las principales ventajas de utilizar un controlador frontal es que ofrece un punto de entrada único para toda la aplicación. Así, en caso de que sea necesario impedir el acceso a la aplicación, solamente es necesario editar el script correspondiente al controlador frontal. Si la aplicación no dispone de controlador frontal, se debería modificar cada uno de los controladores.

2.3 Clases del Diseño con estereotipos Web

Una clase de diseño debe ser la encapsulación completa de todos los atributos y métodos que se puedan esperar, en forma razonable, que existan para la clase, además debe ser lo más completa posible, o sea, aquella suficientemente detallada que sirva como base para generar código fuente.

En el caso de las aplicaciones Web es más importante la modelación de la lógica, que los detalles de presentación, por tanto, para darle solución al diseño como se plantea en el capítulo 1 se utilizarán los diagramas de clase con estereotipos Web.

La utilización del frameworkSymfony 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 éstas 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 5

Requisito Funcional: Gestionar Despacho ER

Este requisito funcional engloba los procesos de Registrar los Despachos de Entrada y Salida de las ER así como permitir modificar la información de los mismos.

En el caso del requisito Registrar Despacho de Entrada se tiene una relación de agregación entra la página cliente “CP_RegistrarDespachoEntradaER” y el formulario “frm_registrarDespachoEntrada” que contiene los componentes necesarios para la solución. Luego los datos entrados son enviados al servidor y recibidos por las funciones que se encuentran dentro de la clase “actions.class.php”, las cuales mediante las clases del paquete de Acceso a Datos son capaces de registrar y obtener información. (Ver Figura 4)

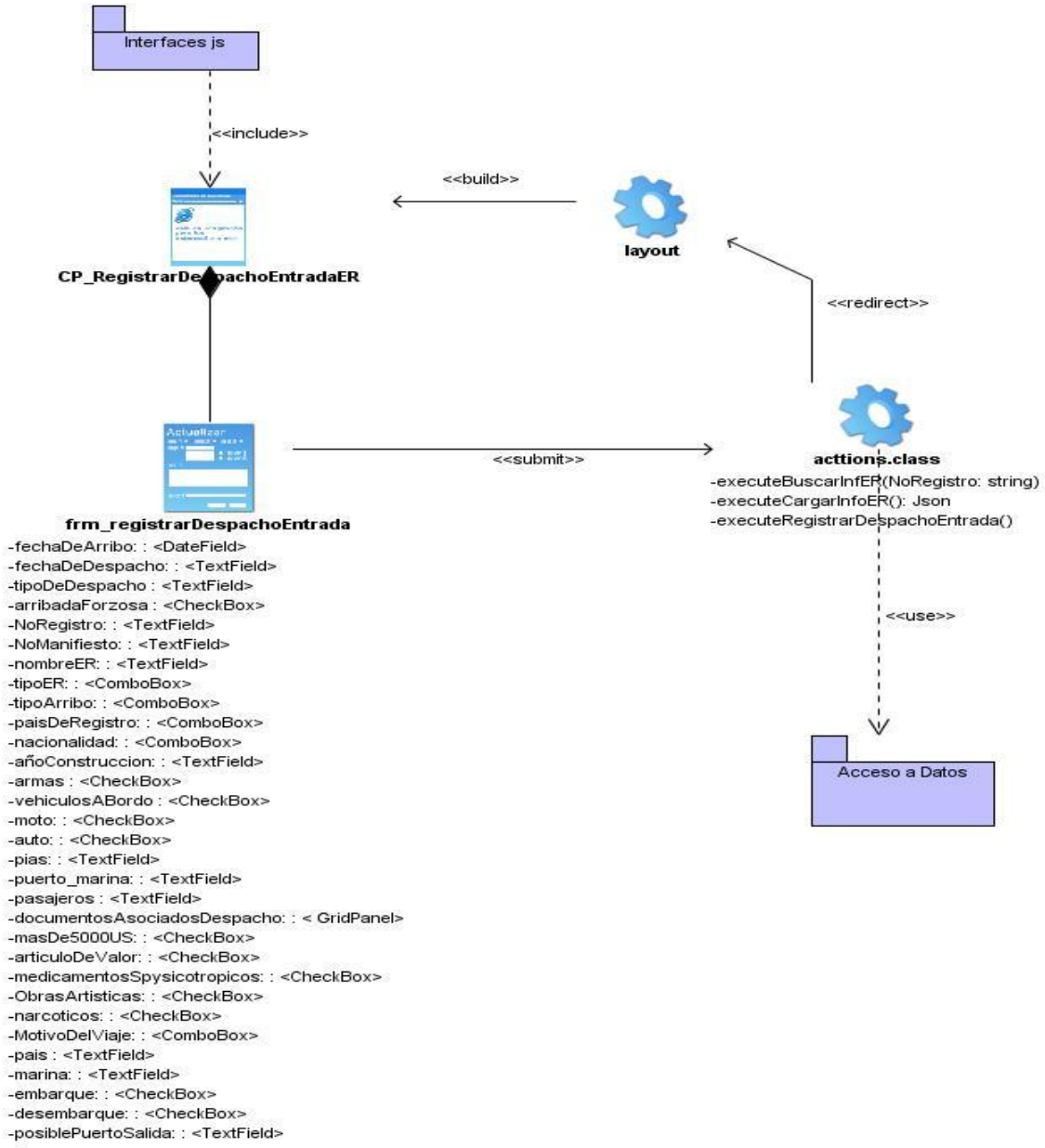


Figura 4 Diagrama de clases para el requisito Registrar Despacho Entrada ER

Los demás diagramas de clases del diseño se encuentran incluidos en los anexos.

2.3.1Paquetes de interfaces JS y acceso a datos.

El paquete “Interfaces JS” contiene los códigos JavaScript que son ensamblados por el navegador una vez que la página cliente se está interpretando y son los script que constituyen las interfaces del módulo ER. (Ver Figura 5)

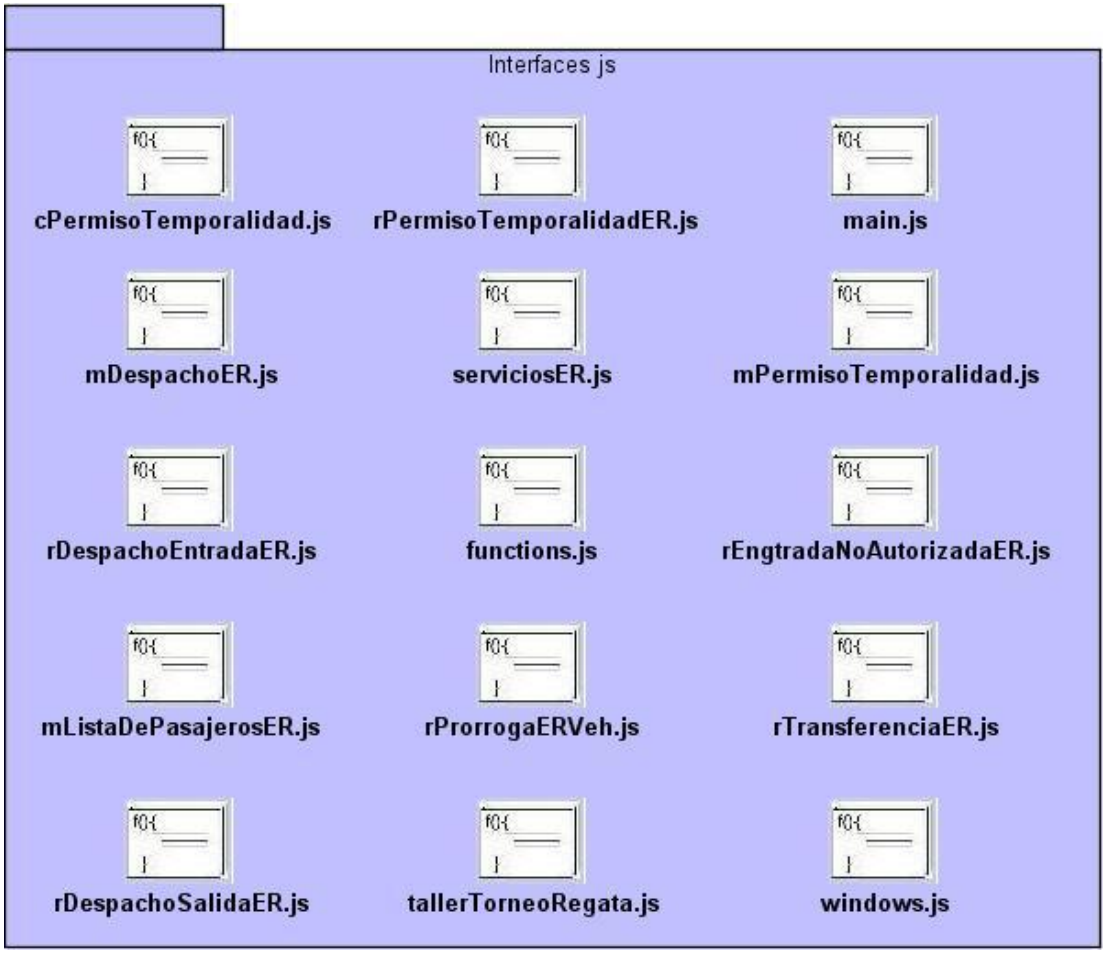


Figura 5 Paquete de Interfaces JS

El paquete “Acceso a Datos” contiene tres paquetes en su interior Objetos, Objetos Peer y Propel. Ver Figura 6. 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.

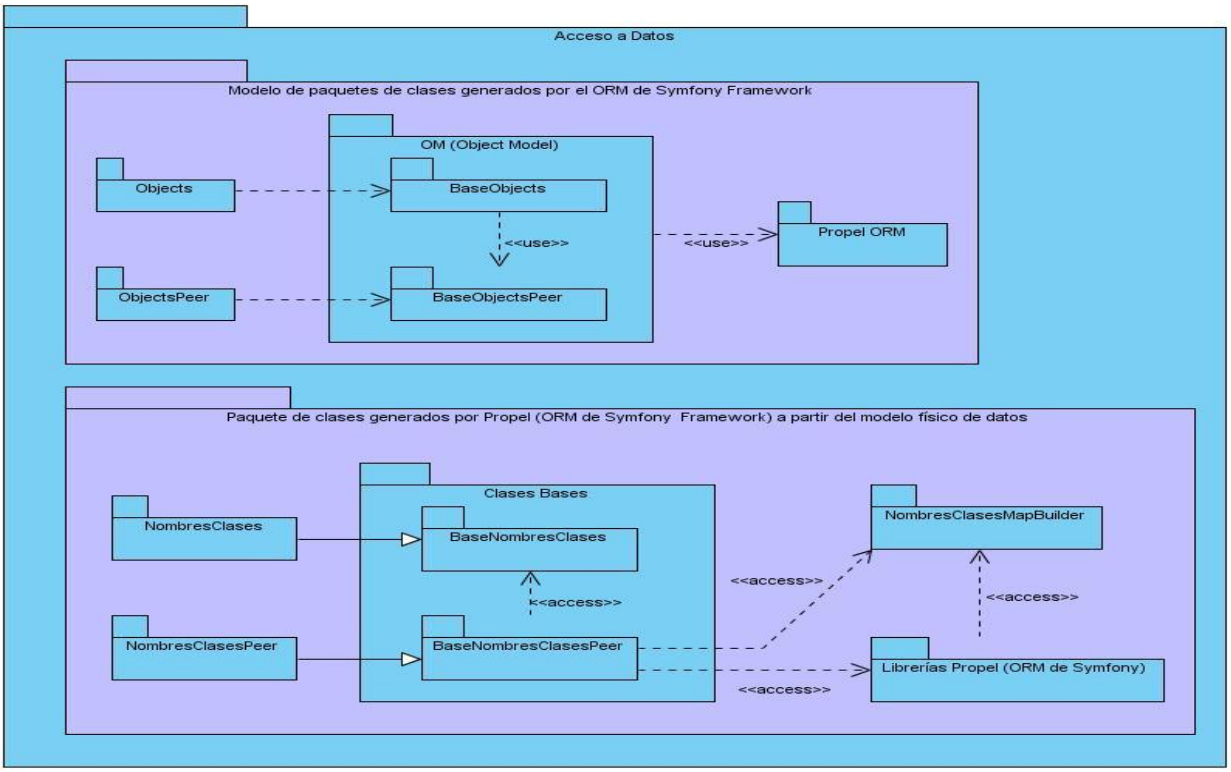


Figura 6 Paquete de acceso a datos

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 7)

Por cada tabla de la base de datos, existen 2 clases que pertenecen al paquete Objetos, por ejemplo: la clase MtiViaje hereda de BaseMtiViaje, 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 BaseMtiViajePeer. Las clases del paquete de Objetos Peer contienen los métodos y atributos para efectuar consultas a la base de datos, al igual que en el 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: MtiViajePeer y

BaseMtiViajePeer. 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 8)

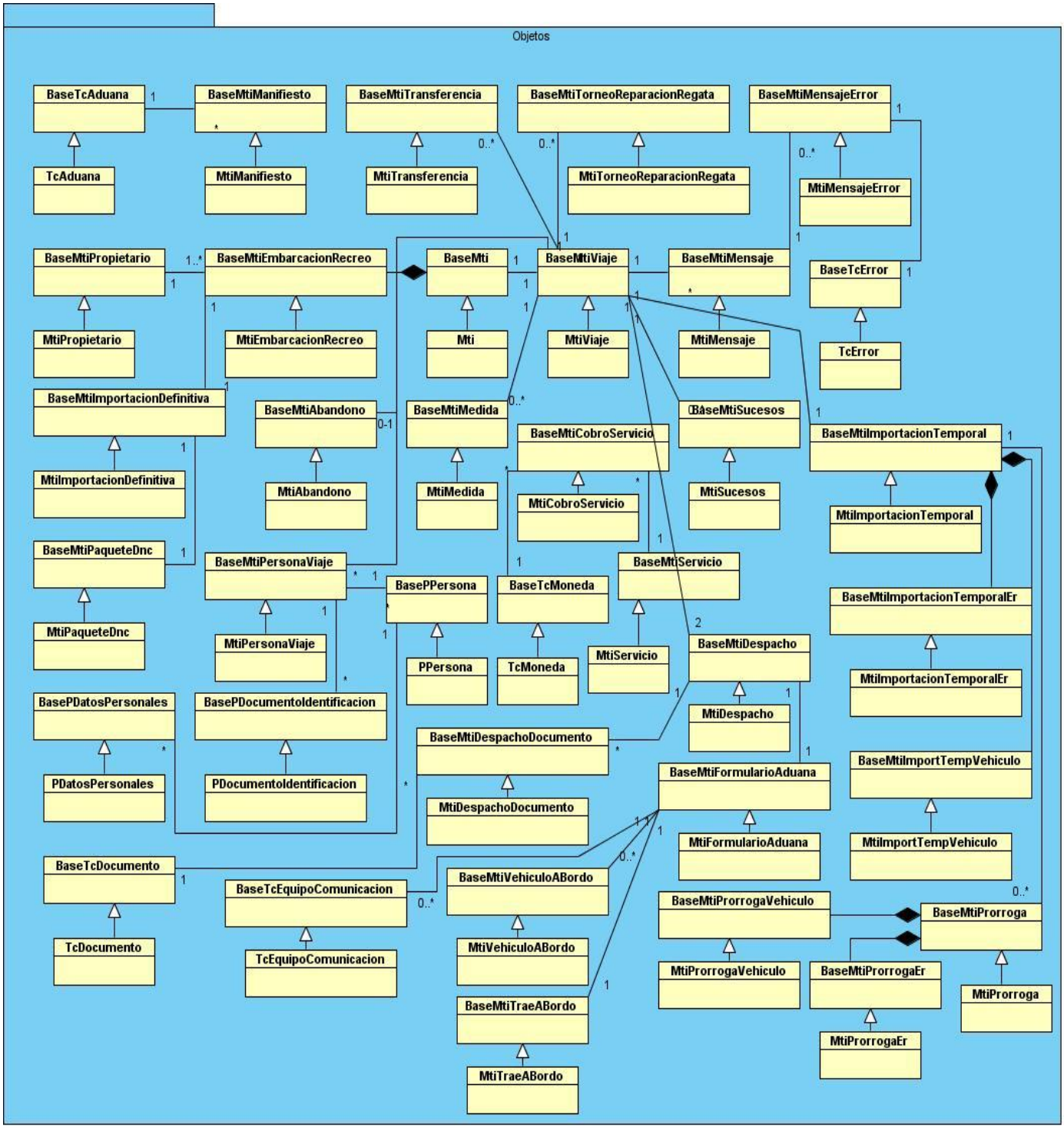


Figura 7 Paquete de Objetos

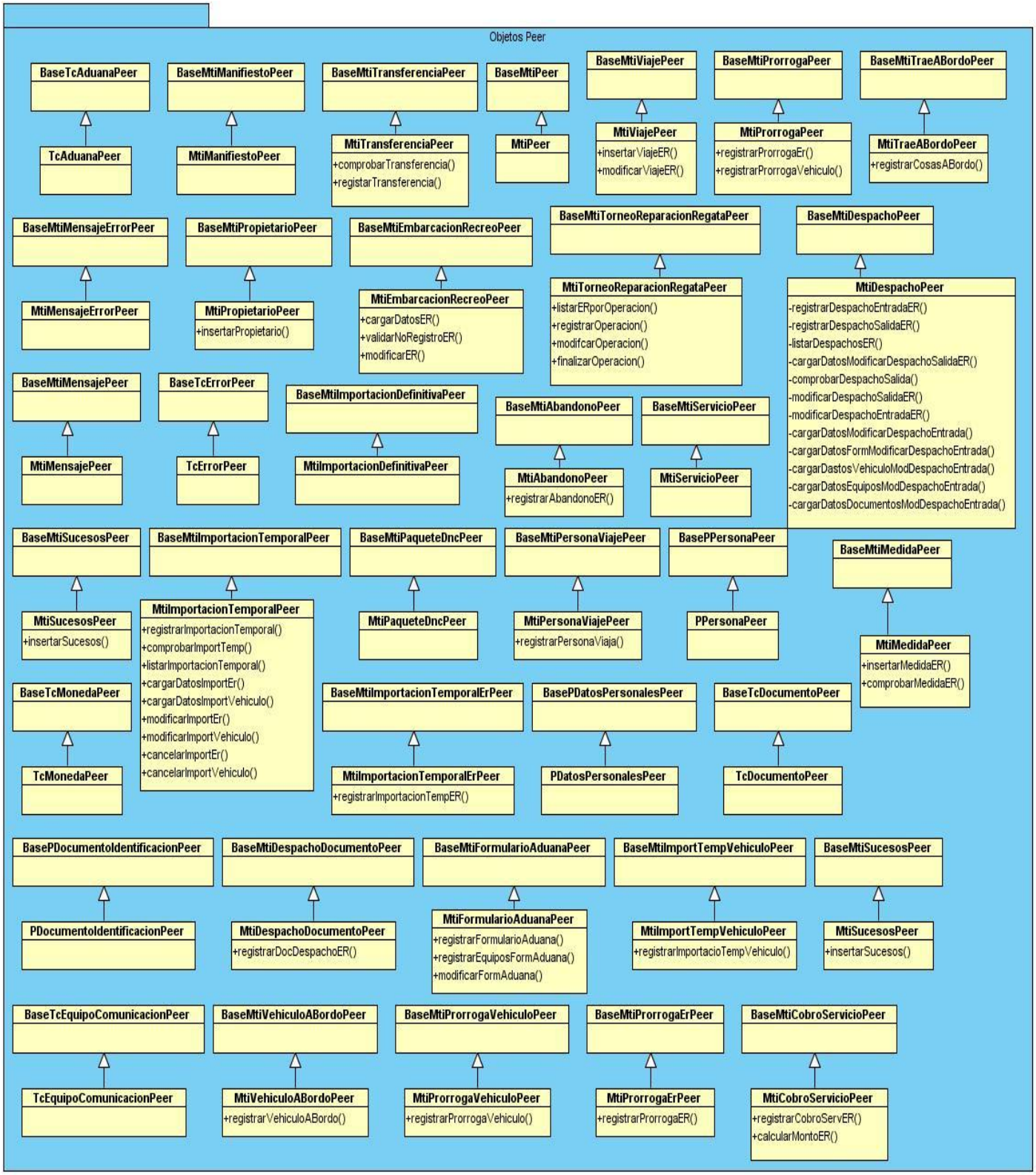


Figura 8 Paquete de Objetos Peer

2.4 Diagramas de Secuencia

Los diagramas de secuencia muestran gráficamente las interacciones del actor y de las operaciones a que dan origen, mostrando un determinado escenario de un caso de uso, los eventos generados por actores externos, su orden y los eventos internos del sistema.

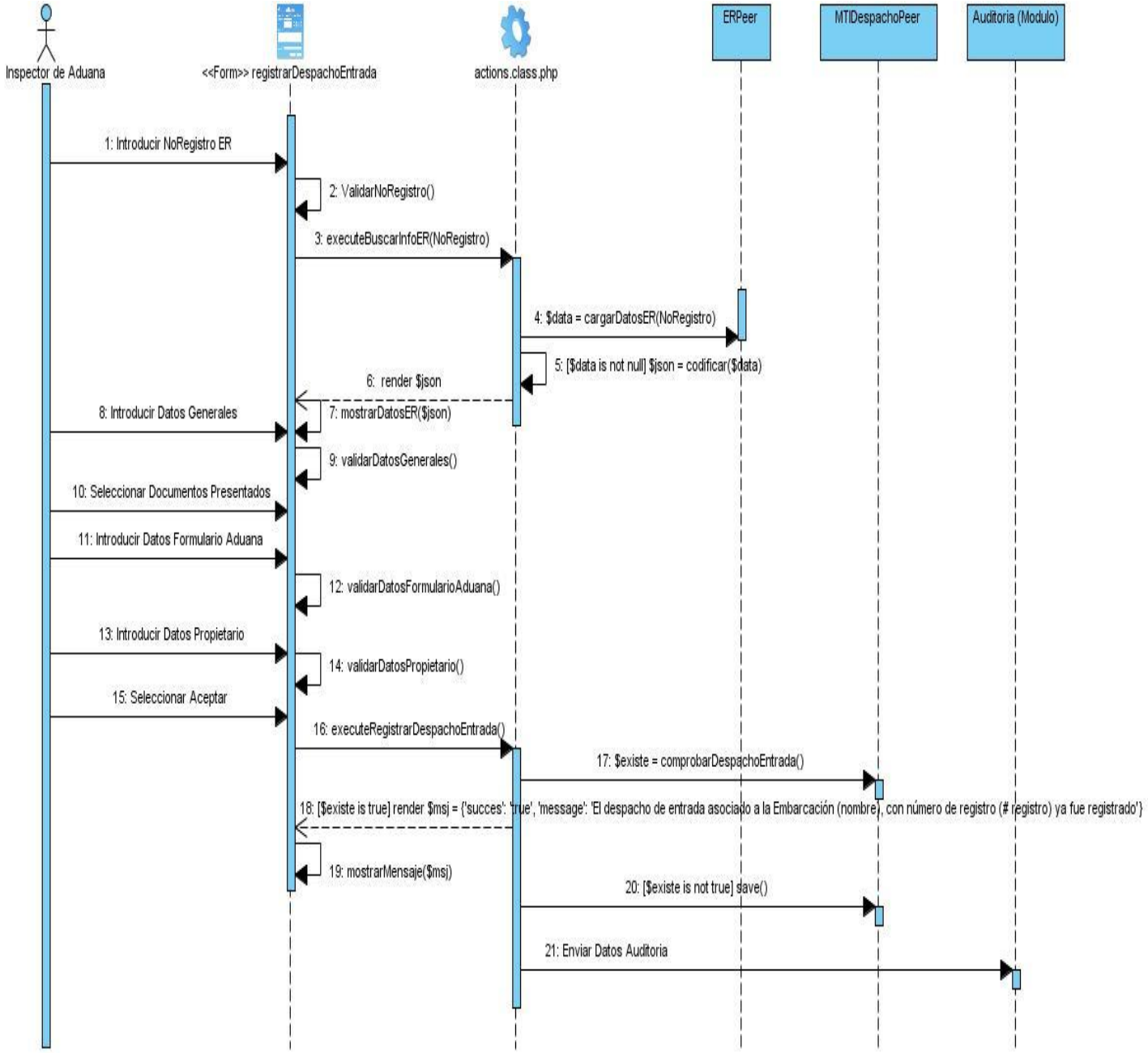


Figura 9 Diagrama de secuencia registrar despacho entrada embarcación de recreo

2.5 Diseño de la base de datos

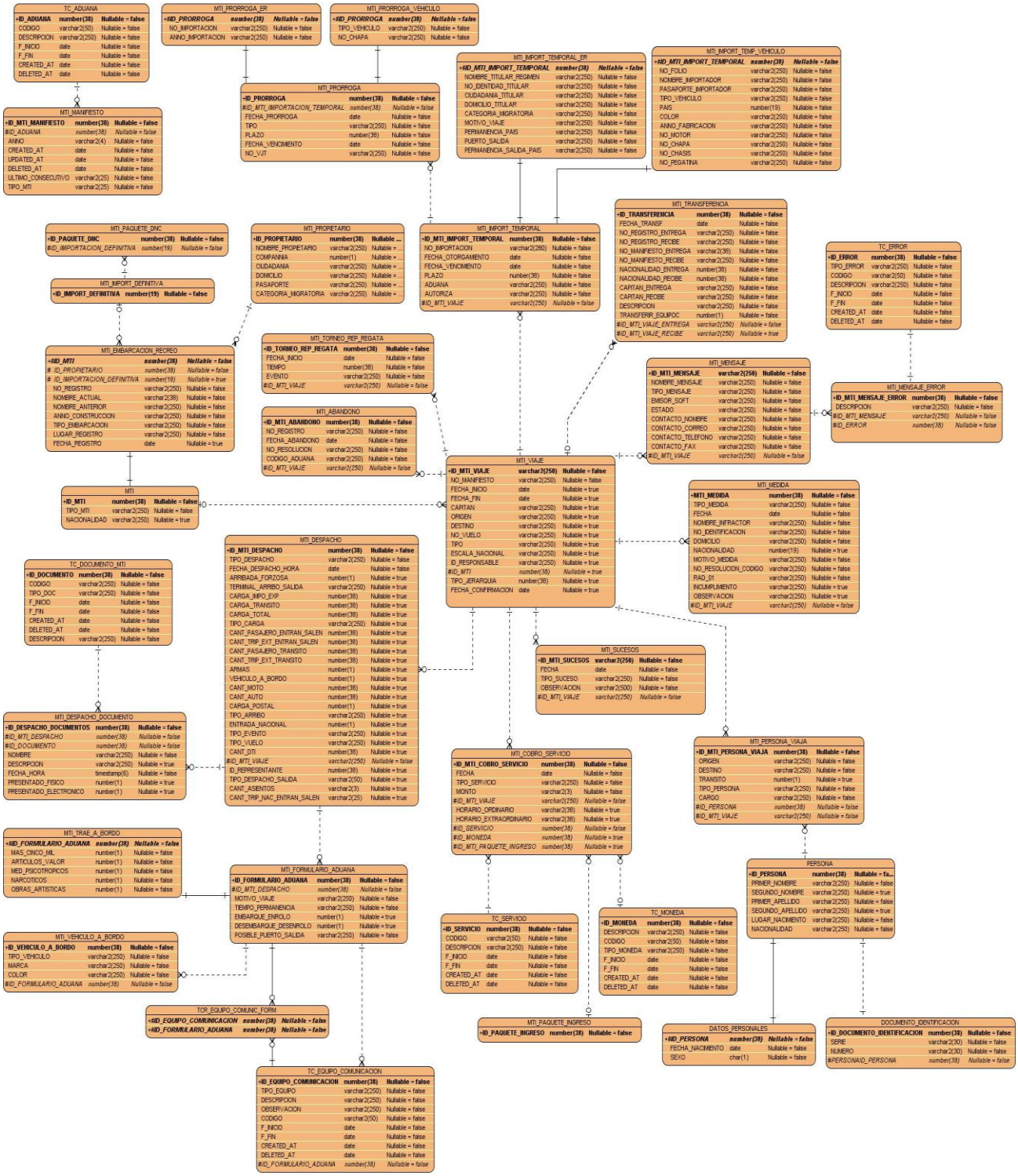


Figura 10 Modelo de datos generado para el módulo Embarcaciones de Recreo.

2.6 Validación del diseño

La medición es esencial para cualquier disciplina, y la ingeniería del software no es una excepción. Medir permite tener una visión más clara y profunda, y proporciona un mecanismo para la evaluación objetiva. (22)

A continuación se aplican un conjunto de métricas de diseño orientado a objeto y se analizan los resultados para determinar la calidad del diseño planteado.

Métricas orientadas a clases

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

Métricas propuestas por Lorenz y Kidd

Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones para una clase individual, y promedian los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones a lo largo y ancho de la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y asuntos relacionados con el código, y las métricas orientadas a valores externos examinan el acoplamiento y la reutilización. (23)

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

- ✓ **Tamaño operacional de clase (TOC).**
- ✓ **Relaciones entre clases (RC).**

Tamaño operacional de clase (TOC): esta métrica es muy usada por diseñadores de software, con una amplia documentación y literatura, fácil de calcular y bastante efectiva. Se basa en el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Atributo de Calidad	Modo en que lo afecta
Responsabilidad	El aumento del TOC implica el aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	El aumento del TOC implica el aumento de la complejidad de implementación de la clase.
Reutilización	El aumento del TOC implica la disminución del grado de reutilización de la clase.

Tabla *2*: Evaluación de los atributos de calidad, métrica tamaño operacional de clase.

Para un total de 26 clases que pertenecen al diseño, se obtuvieron tras aplicar la métrica, valores promedio de 3.5operaciones por clase.(Ver Tabla 3)

Total de Clases	Promedio de operaciones
26	3.5

Tabla 3. Cantidad de clases y promedio de operaciones.

Para evaluar las métricas son necesarios los valores de los umbrales para los parámetros de calidad. Algunos especialistas plantean umbrales para esta métrica basándose en el promedio de operaciones por clases obtenidos, estos valores fueron los aplicados en el diseño de este sistema. (Ver Tabla 4)

	Categoría	Criterio	
Responsabilidad	Baja	<=Promedio	<= 3.5
	Media	>Promedioy <= 2*Promedio	>3.5 y <=7
	Alta	>2*Promedio	>7
Complejidad	Baja	<=Promedio	<= 3.5
	Media	>Promedio y <= 2*Promedio	>3.5 y <=7
	Alta	>2*Promedio	>7
Reutilización	Baja	>2*Promedio	>7
	Media	>Promedio y <= 2*Promedio	>3.5 y <=7
	Alta	<=Promedio	<= 3.5

Tabla 4. Valores de los umbrales para la métrica: tamaño operacional de clase.

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

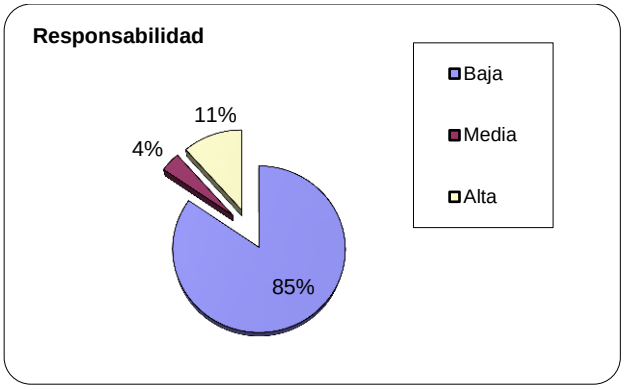


Figura 8. Representación de las clases según la responsabilidad.

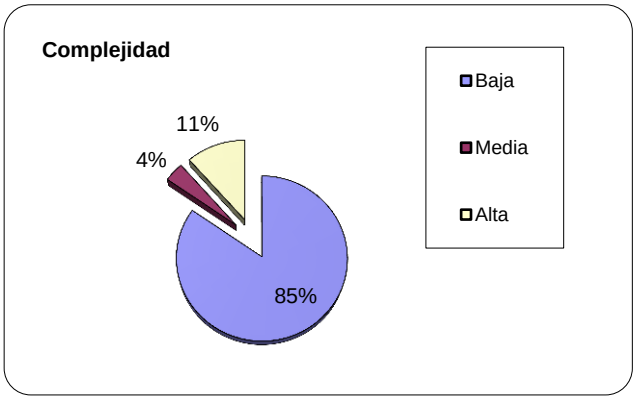


Figura 9. Representación de las clases según la complejidad.

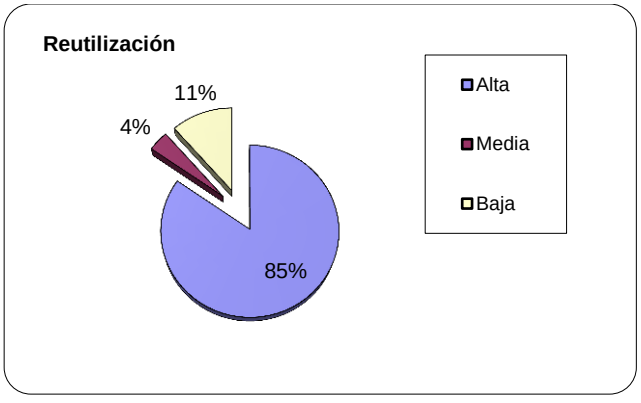


Figura 10. Representación de las clases según la reutilización

Tras un análisis de los resultados arrojados por la evaluación bajo los instrumentos de medición de la métrica TOC, se demuestra que se alcanzaron buenos valores para cada uno de los atributos de calidad evaluados, puesto que, como se puede observar, el diseño presenta un 85% de baja responsabilidad de las clases evitando darle demasiada responsabilidad a la mayoría de las clases para poder aumentar la reutilización de las mismas beneficiadas por 85%. Esta proporcionalidad logra evitar un alto porciento de complejidad, uno de los factores que impide el correcto funcionamiento de un diseño y de todas las medidas que se ejecutan para hacerlo mucho más manejable. Estos valores demuestran que los indicadores de reutilización, complejidad y responsabilidad no se ven afectados.

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

Atributo de Calidad	Modo en que lo afecta
Acoplamiento	El aumento del RC implica el aumento del acoplamiento de la clase.
Complejidad de mantenimiento	El aumento del RC implica el aumento de la complejidad del mantenimiento de la clase.
Reutilización	El aumento del RC implica la disminución en el grado de reutilización de la clase.

Tabla 5: Evaluación de los atributos de calidad, métrica relaciones entre clases.

Para un total de 26 clases que pertenecen al diseño, se obtuvieron tras aplicar la métrica, valores promedio de 2,35 asociaciones de uso por clase. (Ver Tabla 6)

Total de Clases	Promedio asociaciones de uso
26	2.35

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

Los valores de los umbrales para esta métrica que se basan en el promedio de asociaciones de uso por clases, estos valores fueron los aplicados en el diseño de este sistema.

	Categoría	Criterio	
Acoplamiento	Ninguno	0	
	Bajo	1	

	Medio	2	
	Alto	>2	
Complejidad de mantenimiento	Bajo	<=Promedio	<=2.35
	Medio	> Promedio y <= 2*Promedio	>2.35 y <=4.7
	Alto	> 2*Promedio	>4.7
Reutilización	Bajo	>2*Promedio	>4.7
	Medio	> Promedio <= 2*Promedio	>2.35 y <=4.7
	Alto	<=Promedio	<=2.35

Tabla 7. Valores de los umbrales para la métrica relaciones entre clases.

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

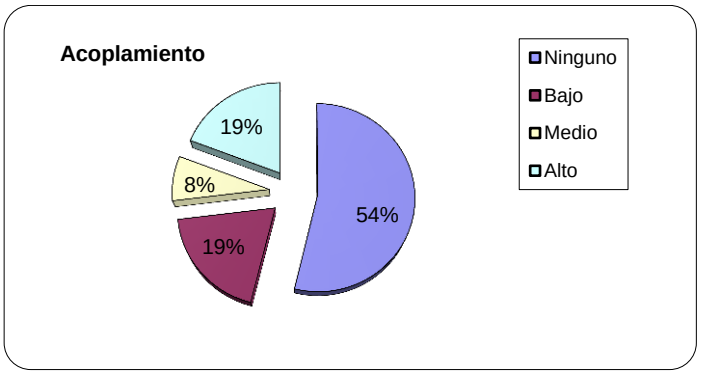


Figura 11. Representación de las clases según el acoplamiento.

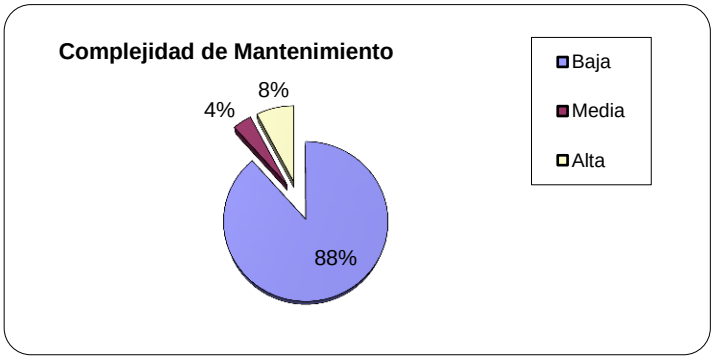


Figura 12. Representación de las clases según complejidad de mantenimiento.

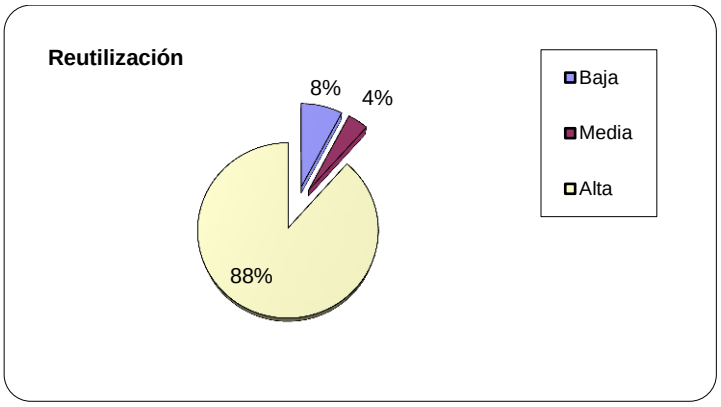


Figura 13. Representación de las clases según la reutilización.

La evaluación de los resultados obtenidos durante la aplicación de la métrica RC, demuestra que el diseño asumido tiene una calidad aceptable, puesto que, el nivel de acoplamiento entre las clases arroja que para un 54% no existe niveles de dependencia, lo cual es un factor aceptable desde el punto de vista de la implementación del software, mientras que la capacidad de reutilización del software alcanza niveles de 88% de forma positiva.

Estos resultados de los atributos de calidad se suman a los obtenidos por las pruebas de TOC y demuestran el uso de un buen diseño de software.

2.7 El módulo “Embarcaciones de Recreo” orientado a la arquitectura del GINA

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 11 muestra cómo se integra el módulo Embarcaciones de Recreo al Sistema de Gestión Integral de la Aduana, especificando que el mismo es parte del subsistema Despacho de Medios de Transporte Internacional este subsistema se encuentra definido dentro de las aplicaciones como “sfMtiPlugin” y dentro se encuentra el módulo llamado “ER”.

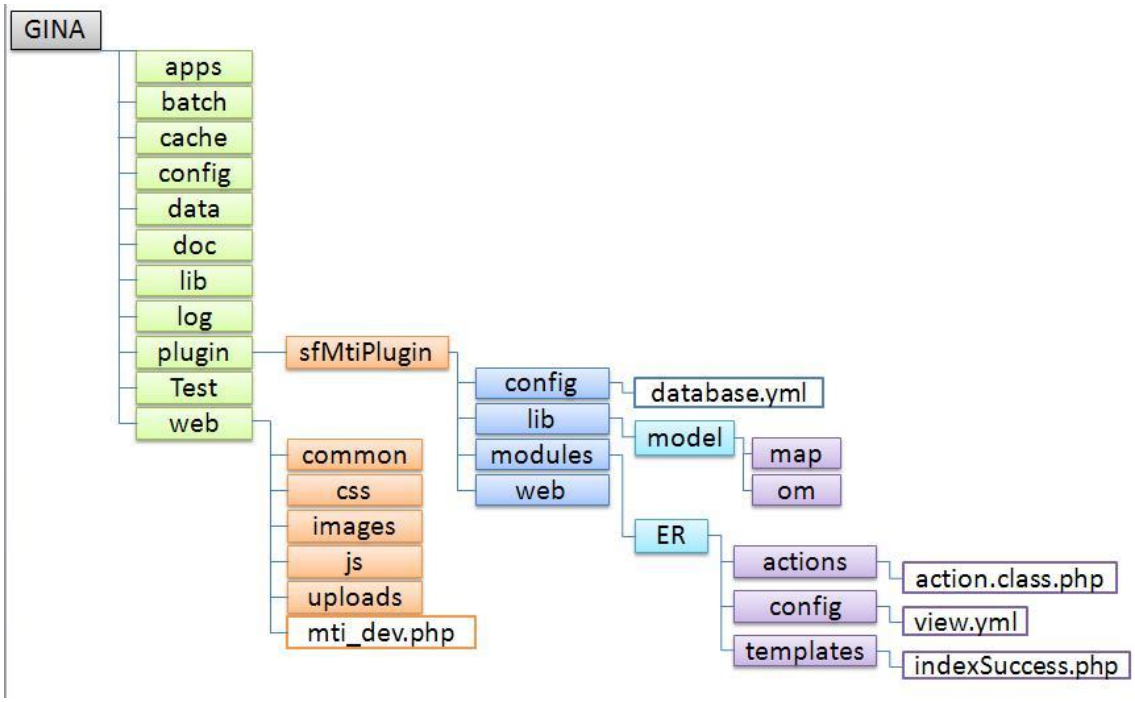


Figura 11 Módulo Embarcaciones de Recreo integrado al GINA

El módulo Embarcaciones de Recreo (ER) consta de tres carpetas, estas son: *actions*, *config* y *templates.* 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 "*sfMtiPlugin*" se encuentran cuatro carpetas, en una de ellas, específicamente en "*config*" está archivo el *databases.yml*, utilizado para la conexión a la base de datos, en la carpeta "*lib*" se encuentran las clases definidas para el modelo de la aplicación, y en la carpeta "web" se encuentran los ficheros javascript que se utilizarán para las pantallas del módulo. En la dirección GINA/web/ se encuentra el fichero "*mti_dev.php*" que representa el controlador frontal de la aplicación en desarrollo.

2.8 Conclusiones Parciales

A lo largo de este capítulo se obtienen los artefactos del diseño de la aplicación. Se realizaron losdiagramas de clases del diseño con estereotipos Web, diagrama de secuencia y diseño de la base de datos, los cuales servirán de entrada para el flujo de implementación en el siguiente capítulo. Así como la validación del diseño utilizando las métricas TOC y RC. También se muestra como se encuentra integrado el módulo ER al sistema GINA.

CAPÍTULO 3: Implementación y Prueba de la Solución

3.1 Introducción

En este capítulo se aborda los aspectos más importantes relacionados con la implementación del sistema y la validación del mismo, se realiza el diagrama de despliegue y de componte, se enuncian los estándares de codificación que se siguieron para generar el código. Además de algunos fragmentos de código de la aplicación así como las interfaces correspondientes. También se realiza la validación del sistema a través de los casos de pruebas que se le realizaron a los casos de uso del sistema.

3.2 Estándar de codificación

Actualmente existen 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.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armónico, la legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento y mantenimiento del software.

La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta el final del proyecto de software. No es práctico adoptar un estándar de codificación una vez iniciado el trabajo. En el caso del Departamento de Soluciones para la Aduana, el estándar de codificación fue definido por la dirección del proyecto en sus inicios y se encuentra registrado en el documento “Propuesta de un estándar de codificación”. (14)

Todos los nombres de las acciones deben de estar en la nomenclatura "CamelCase[3]" que inicia por la palabra *execute*, el nombre de las clases expresando en notación "UpperCamelCase[4]", además del uso "Peer" como sufijo. Los nombres de las variables deben expresar claramente el contenido de la misma.

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.

3.3 Modelo de Implementación

El resultado de la etapa de diseño y los artefactos que en esta se generan dan comienzo a la implementación del sistema cuyo propósito es desarrollar la arquitectura y el sistema como un todo, en esta etapa es donde se implementan las clases encontradas durante el diseño.

"El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc (...) describe cómo se organizan los componentes de acuerdo con los mecanismos de estructura disponibles en el entorno de implementación y en el lenguaje de programación utilizados, y como dependen los componente uno de los otros (...) define una jerarquía de subsistemas de implementación que contiene componentes de interfaces."(8)

3.4 Diagrama de Despliegue

El diagrama de despliegue permite apreciar de forma visual cómo se encuentran relacionados físicamente los componentes de la aplicación. Describe la distribución física del sistema, muestra cómo están distribuidos los componentes de software entre los distintos nodos de cómputo. Este diagrama permite comprender la correspondencia entre ambas arquitecturas software y hardware. Los nodos se utilizan para modelar la topología del hardware sobre el cual se ejecuta el sistema. Representa típicamente un procesador o un dispositivo sobre el que se pueden desplegar los componentes.

---

[3]**CamelCase**: Consiste en escribir frases o palabras compuestas eliminando los espacios intermedios y poniendo en minúscula la primera letra y en mayúscula las demás primeras letras de cada palabra contigua.

[4]**UpperCamelCase**: Consiste en escribir frases o palabras compuestas eliminando los espacios intermedios y poniendo en mayúscula la primera letra de cada palabra incluyendo la primera letra de la frase.

Durante el desarrollo de la solución fueron tomadas en cuenta las especificaciones generales del modelo de despliegue utilizado para el Sistema GINA, el cual está implementado siguiendo la arquitectura cliente-servidor.

El diagrama de despliegue que se muestra a continuación en la Figura 12 representa de forma general la ubicación del módulo ER, el cual está integrado al Sistema GINA dentro del servidor de aplicaciones.

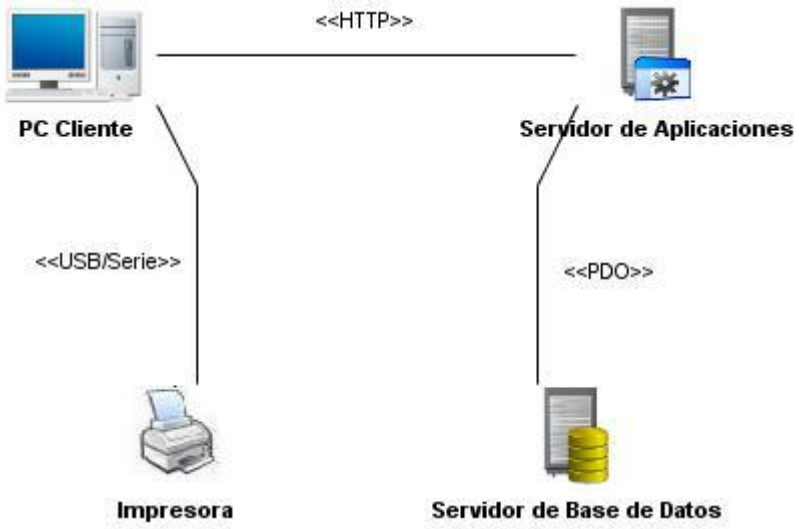


Figura 12 Diagrama de despliegue general del Sistema GINA

3.5 Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Además muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas, pueden ser simples archivos, paquetes, bibliotecas cargadas dinámicamente, etc.

En el siguiente diagrama se representan los componentes del módulo Embarcación de Recreo (Ver Figura 13), donde se evidencian las relaciones de los componentes, desde el acceso a través del controlador frontal (mti_dev.php), la clase actions.class.php, cada una de las interfaces de la aplicación, las clases del negocio y su interacción con otros subsistemas, así como los elementos de configuración.

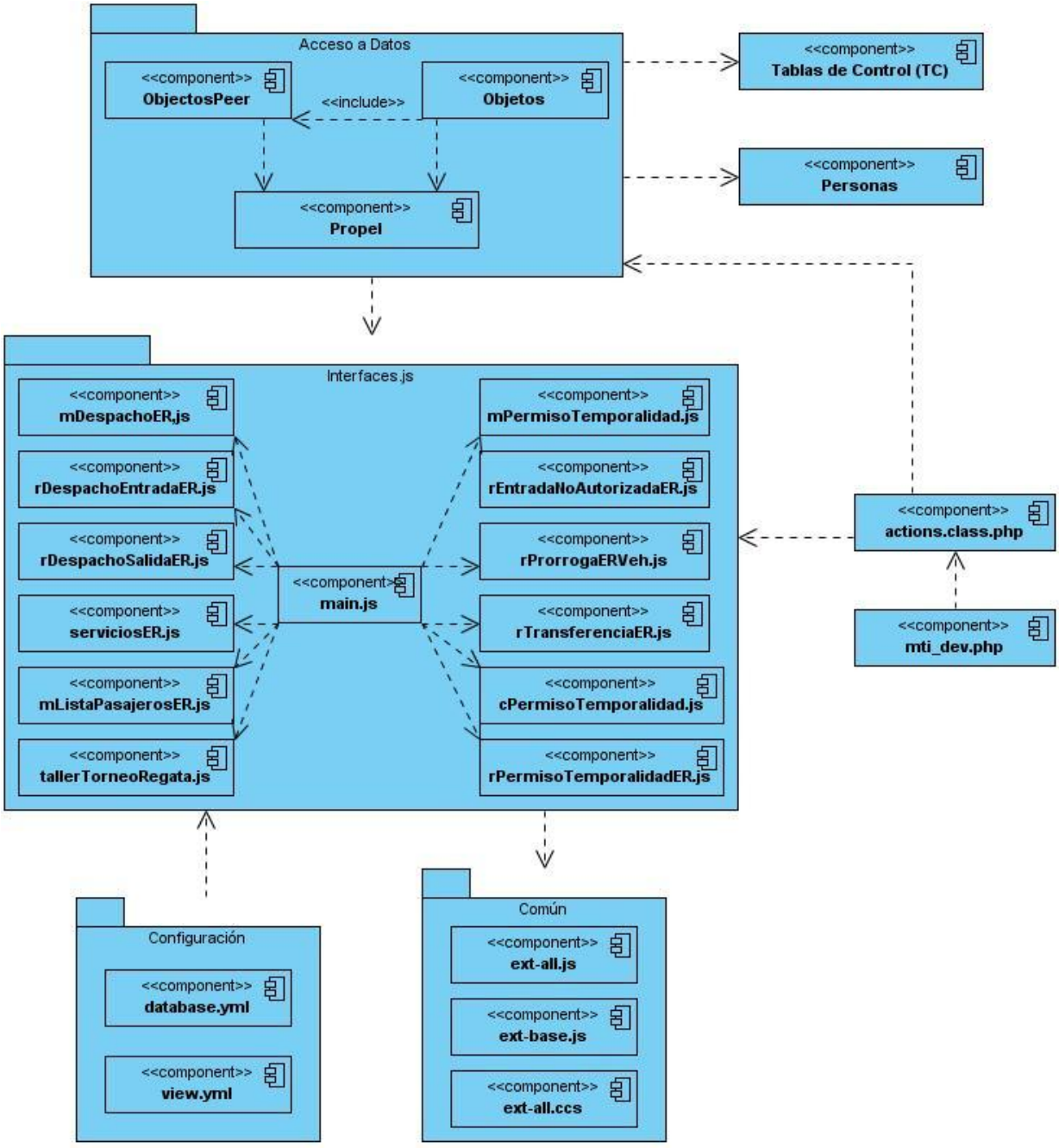


Figura 13 Diagrama de componentes Módulo Embarcación de Recreo

3.6 Generando 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".

Dentro de un proyecto, las operaciones se agrupan de forma lógica en aplicaciones. Normalmente, una aplicación se ejecuta de forma independiente respecto de otras aplicaciones del mismo proyecto. Cada aplicación está formada por uno o más módulos. Un módulo normalmente representa a una página Web o a un grupo de páginas con un propósito relacionado.

Para crear el proyecto, la aplicación y el módulo, al tratarse de una plataforma como Windows XP se hace de la siguiente forma:

Se accede a través de la consola hacia la raíz del directorio donde se quiere crear, ejemplo (C:\www mkdir GINA) luego de creada la carpeta se ejecutan los comandos (C:\www\GINA symfonyinit-project GINA), (C:\www\GINA symfonyinit-appmti) y (C:\www\GINAsymfonyinit-module er). Pero en este trabajo lo que se desarrollo es un módulo de la aplicación GINA por lo que ya no es necesario realizar todos estos pasos, solamente la creación del módulo dentro del *sfMtiPlugin,* sería de la siguiente forma: (C:\www\GINAsymfonygenerate-module sfMtiPluginer*)* de esta forma obtendremos una estructura de directorios para el módulo como la que sigue:

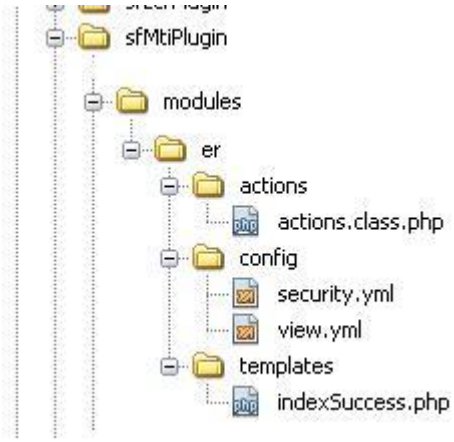


Figura 14 Estructura del módulo ER

3.7 Generando el esquema y el modelo

En orden de crear las clases que representen las tablas y relaciones de la base de es necesario crear una representación XML de la misma y generar el modelo de dicha representación. Symfony propone una ventaja a la hora de manejar este XML, para ello es necesario editar los ficheros databases.yml y propel.ini. Ya editados correctamente dichos archivos se puede ejecutar el comando (symfonypropel:build-schema) 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 se procede a generar el modelo, con el comando (symfonypropel:build-model), arrojando como resultado las clases de los Paquetes de Objetos y ObjetosPeer.

3.8 Interfaces del sistema

Para la realización de la aplicación se desarrollaron interfaces amigables y seguras, acorde con el personal de la Aduana General de la República que trabajará con la misma. A continuación se presenta la interfaz principal del módulo (Ver Figura 15). En dicha interfaz se implementó un menú que da paso a la ejecución de cada uno de los Requisitos Funcionales del sistema (Ver Figura 16), ejemplo parael escenario Registrar Despacho Entrada. (Ver Figura 17)

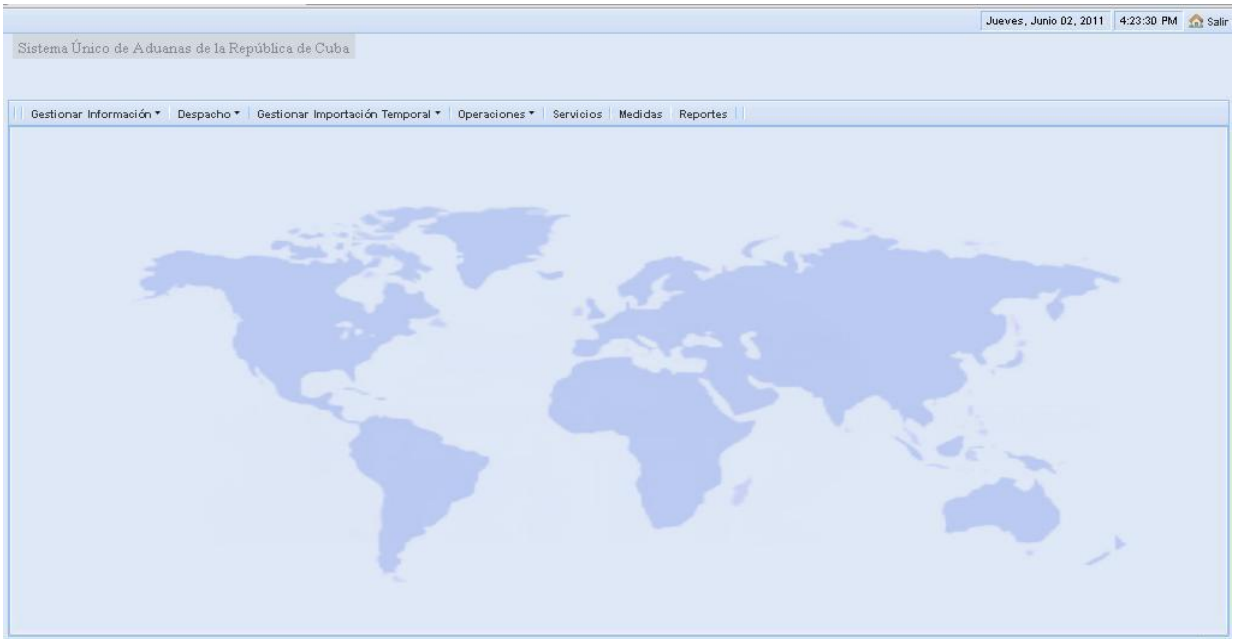


Figura 15 Interfaz principal del Módulo Embarcación de Recreo

Figura 16 Menú Principal

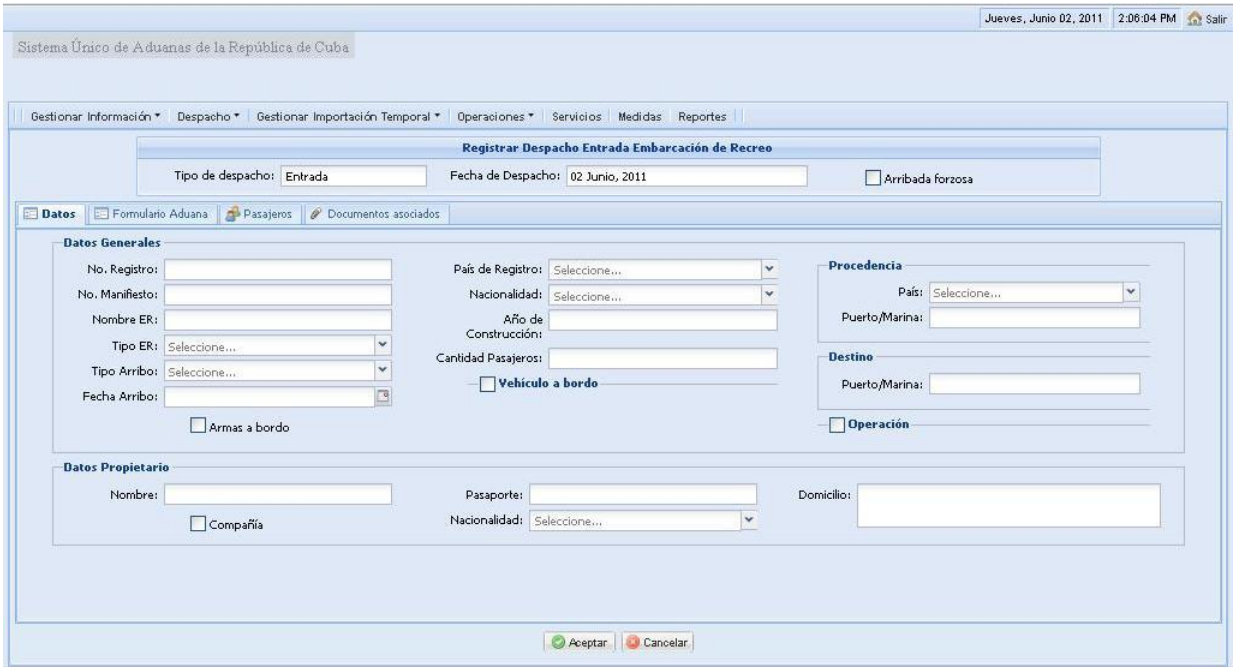


Figura 17 Interfaz Registrar Despacho Entrada

Posteriormente al introducir el No. Registro, la aplicación debe ser capaz de cargar los datos de la embarcación en caso de que esta se haya registrado anteriormente. Ver (Figura 18 y 19)

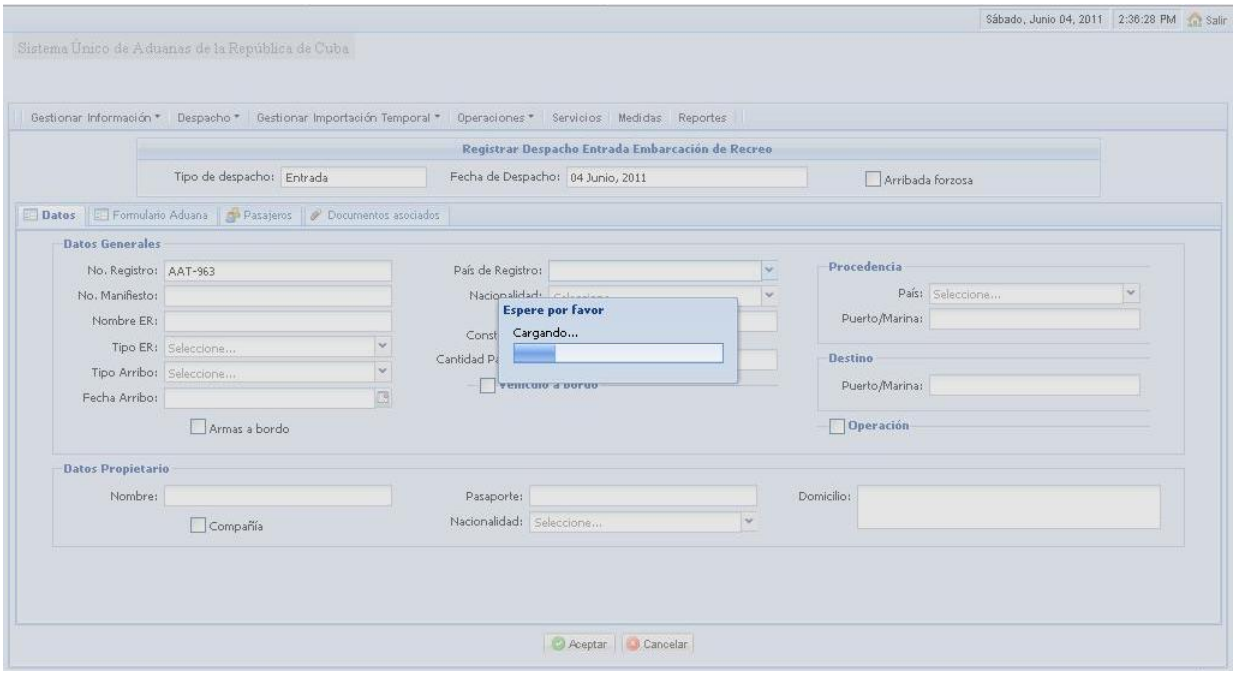


Figura 18 Cargando los datos de la embarcación

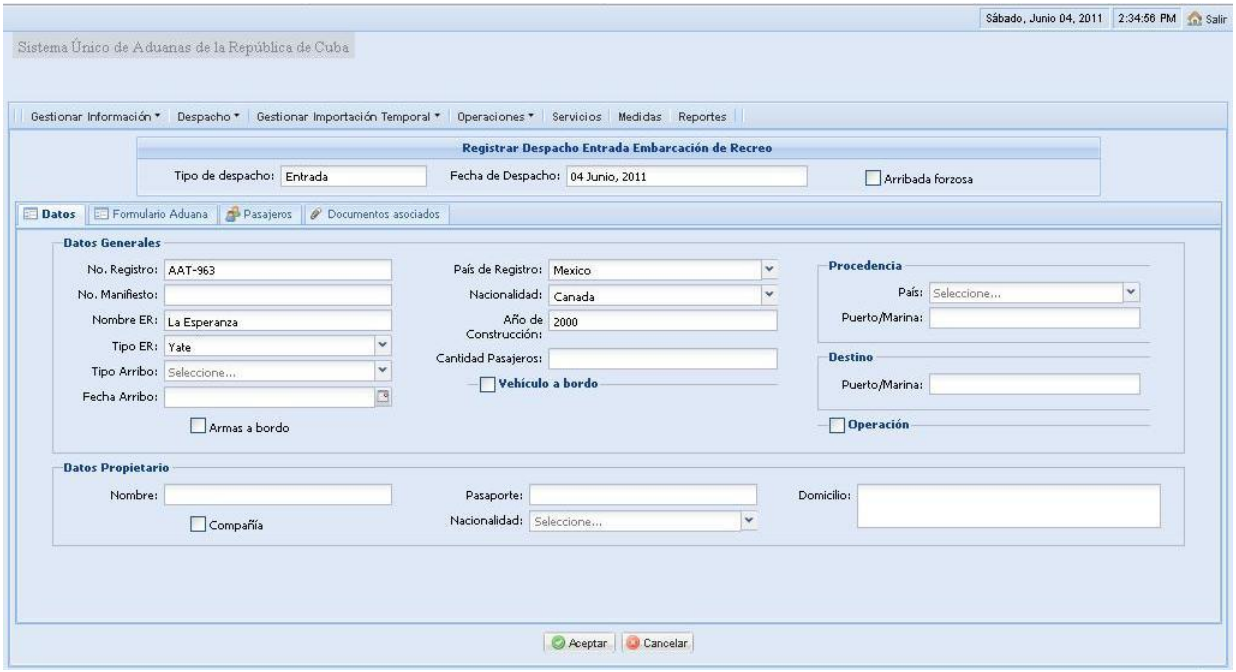


Figura 19 Datos de la embarcación

A continuación se muestra el código que es ejecutado en el actions.class.php que permite realizar esta acción. (Ver Figura 20)

```
public function executeBuscarInfER()
{
    $noRegistro = $this->getRequestParameter('noRegistro');
    $datos = MtiEmbarcacionRecreoPeer::cargarDatosER($noRegistro);
    return $this->renderText($datos);
}
```

Figura 20 Código fuente método executeBuscarInfER()

Luego el método *executeBuscarInfER()* hace una llamado al método *cargarDatosER*() (Ver Figura 21) que se encuentra dentro de la clase *MtiEmbarcacionRecreoPeer,* al cual se le pasa como parámetro el no. Registro introducido a través de la interfaz y posteriormente el método retorna el resultado de la búsqueda, en caso positivo los datos son cargados en la pantalla como se puede apreciar en la Figura 19.

```
public static function cargarDatosER($noRegistro)
{
    $criteria = new Criteria();
    $criteria->add(MtiEmbarcacionRecreoPeer::NO_REGISTRO, $noRegistro);
    $criteria->addJoin(MtiEmbarcacionRecreoPeer::ID_MTI, MtiPeer::ID_MTI);
    $criteria->addSelectColumn(self::NOMBRE_ACTUAL);
    $criteria->addSelectColumn(self::TIPO_EMBARCACION);
    $criteria->addSelectColumn(self::LUGAR_REGISTRO);
    $criteria->addSelectColumn(self::ANNO_CONSTRUCCION);
    $criteria->addSelectColumn(MtiPeer::NACIONALIDAD);
    $datos = self::doSelectStmt($criteria);

    $arrayDatos = array();
    while($result = $datos->fetch())
    {
        $arrayDatos = array(
            'nombreER' => $result[0],
            'tipoER'   => $result[1],
            'lugarRegistro' => $result[2],
            'annoConstruccion' => $result[3],
            'nacionalidad' => $result[4]);
     }

     $json = json_encode($arrayDatos);
     return $json;
}
```

Figura 21 Código fuente método cargarDatosER ()

Al introducir todos los datos correspondientes al despacho de entrada de la embarcación, estos son enviados y recibidos en la clase actions donde se ejecuta el método *executeRegistrarDespachoEntrada()*. (Ver Figura 22)

```
$criteria = new Criteria();
$criteria->add(MtiEmbarcacionRecreoPeer::NO_REGISTRO, $noReg);
$criteria->addAnd(MtiDespachoPeer::TIPO_DESPACHO, 'E');
$criteria->addJoin(MtiViajePeer::ID_MTI, MtiEmbarcacionRecreoPeer::ID_MTI);
$criteria->addJoin(MtiViajePeer::ID_MTI_VIAJE, MtiDespachoPeer::ID_MTI_VIAJE);
$dato = MtiViajePeer::doSelectOne($criteria);
$nombAnt = $nombEr;
if(is_null($dato)){
   try{
           $idPropietario = MtiPropietarioPeer::insertarPropietario($nombre, $compannia, $nacionalidadP, $d
           $c = new Criteria();
           $c->add(MtiEmbarcacionRecreoPeer::NO_REGISTRO,$noReg);
           $er = MtiEmbarcacionRecreoPeer::doSelectOne($c);
           if(is_null($er)){
                   $er = new MtiEmbarcacionRecreo($var = 'ENTRADA_AUTORIZADA');
           }else{
               if($er->getNombreActual() != $nombEr)
               {
                   $nombAnt = $er->getNombreActual();

               }
               $er->setModo('EXISTE');
               }
           $er = new MtiEmbarcacionRecreoForm($er);
       $datosER = array (
            'idPropietario'            => $idPropietario,
            'noRegistro'               => $noReg,
            'nombreActual'             => $nombEr,
            'nombreAnterior'           => $nombAnt,
            'annoConstruccion'         => $annoConst,
            'tipoEmbarcacion'          => $tipoEr,
            'lugarRegistro'            => $lugarReg,
            'mti' =>  array(
                          'tipoMti' => 'ER',
                          'nacionalidad' => $nacionalidad),
       );
    $er->bindAndSave($datosER);

    $c1 = new Criteria();
    $c1->add(MtiEmbarcacionRecreoPeer::NO_REGISTRO, $datosER['noRegistro']);
    $c1->addSelectColumn(MtiEmbarcacionRecreoPeer::ID_MTI);
    $idMti = MtiEmbarcacionRecreoPeer::doSelectOne($c1);
    $idMtiViaje = MtiViajePeer::insertarViajeER($noManif, $fechaArribo, $origen, $destinoPM, 'E', $idMti[0]
    $idMtiDespacho = MtiDespachoPeer::registrarDespachoEntradaER($tipoDespacho, $fechaDespacho, $arribadaF
    $idFormAduana = MtiFormularioAduanaPeer::registrarFormularioAduana($idMtiDespacho, $traeABordo, $motiv
    MtiTraeABordoPeer::registrarCosasABordo($idFormAduana, $cincomilUsd, $articulosValor, $medicamentosP,

   $request = $this->request();
   if($request->hasParameter('jsonVehiculoABordo'))
           {
                   $datosVehiculos = json_decode($this->getRequestParameter('jsonVehiculoABordo'));
```

Figura 22 Segmento de código fuente método executeRegistrarDespachoEntrada().

3.9 Prueba

La fase de pruebas en el desarrollo de software es muy importante ya que esta consiste en probar las aplicaciones construidas, además las pruebas permiten verificar, validar y revelar la calidad de un producto software. En esta fase es donde se encuentran y se corrigen los errores de las fases anteriores (5).

3.9.1 Pruebas de Caja Negra.

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software (14).

A continuación se derivan conjuntos de condiciones de entrada que utilicentodos los requisitos funcionales de un programa.
Las pruebas de caja negra pretenden encontrar estos tipos de errores:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores en la interfaz.
- ✓ Errores en estructuras de datos o en accesos a bases de datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y de terminación.

3.9.1.1 Casos de prueba.

Un caso de prueba permite detallar la forma en que se va a probar el sistema, incluyendo los datos de entrada con las que se realizará la prueba correspondiente, las condiciones de ejecución y resultados obtenidos (5).

Deben verificar:

- ✓ Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificación de los requerimientos.

- ✓ Si el producto se comporta tal y como se describe en las especificaciones funcionales del diseño.

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

Durante la realización de las pruebas al sistema se realizaron 9 Casos de Pruebas (uno para cada requisito funcional) y para cada uno de sus requisitos secundarios dividiéndolos en secciones. Luego de una detallada revisión se encontraron 10 no conformidades, en la primera iteración: 9 de validación y 1 de interfaz. Las no conformidades fueron corregidas, comprobándose la eliminación de las mismas en una segunda iteración, obteniéndose como resultado de la misma 0 no conformidades.(Ver Figura 23)

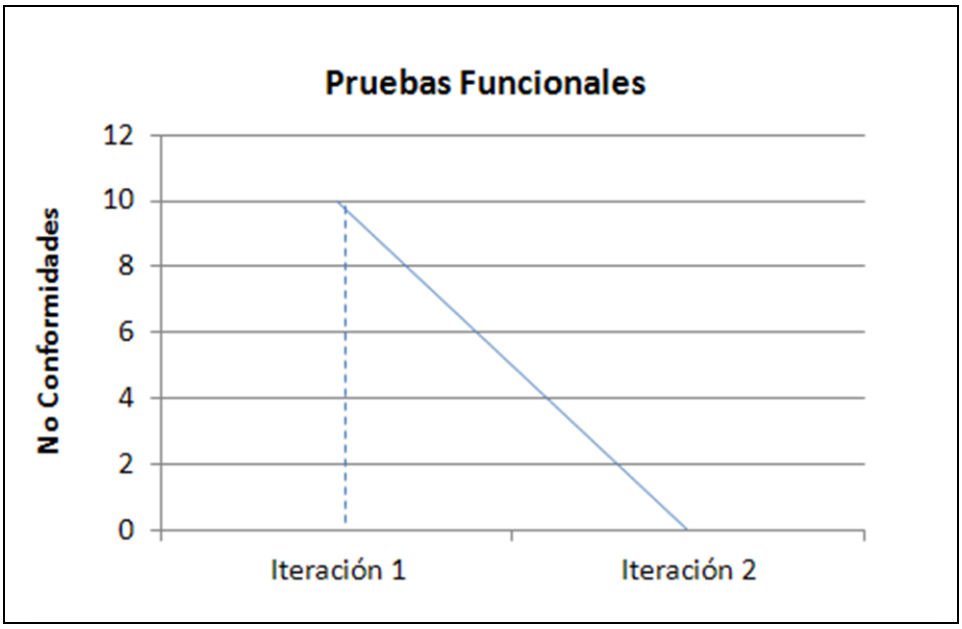


Figura 23 Iteraciones de las pruebas funcionales realizadas

A continuación se muestra el diseño de dos caso de prueba correspondiente el requisito funcional: Gestionar despacho de Embarcaciones de Recreo, dividido en tres secciones: 1: Registrar Despacho Entrada, 2: Registrar Despacho Salida, 3: Modificar Despacho (Entrada/Salida), y el segundo caso de prueba correspondiente al requisitosfuncional: Registrar Medida.

3.9.1.1.1 CPR 1 Gestionar despacho Embarcaciones de Recreo

Descripción General:

Requisitos: “Registrar Despacho Entrada”, “Registrar Despacho Salida”, “Modificar Despacho (Entrada/Salida)”. El sistema permite registrar los despachos de entrada/salida de las Embarcaciones de Recreo así como su modificación.

Condiciones de Ejecución:

- ✓ El usuario debe estar autenticado en el sistema.
- ✓ El usuario debe seleccionar el menú Despacho.

Sección 1: Registrar Despacho Entrada.

Nombre de la sección	Escenario de la sección	Descripción de la funcionalidad	Flujo Central
1: Registrar Despacho Entrada.	EC 1.1 Introduce no.Registro.	El sistema carga en el formulario los datos de la embarcación asociada al no.Registro dado (si tiene).	-Introduce no.Registro y da un click fuera. -El sistema carga los datos.
	EC 1.2 Registrar sin llenar correctamente los datos obligatorios del despacho.	El sistema no permite registrar un despacho.	-Se presiona el botón “Aceptar”. -El sistema muestra los campos obligatorios en rojo y no permite registrar un despacho.
	EC 1.3 Registrar llenando correctamente los datos obligatorios del despacho.	El sistema permite registrar un despacho.	-Se presiona el botón “Aceptar”. -El sistema muestra un cartel informando al usuario que el despacho ha sido registrado con éxito.

Sección 2: Registrar Despacho Salida

Nombre de la sección	Escenario de la sección	Descripción de la funcionalidad	Flujo Central
2: Registrar Despacho Salida.	EC 2.1 Buscar despachos sin especificar criterio de búsqueda.	El sistema lista los despachos de entrada filtrados por la fecha actual.	-El sistema muestra un listado con los despacho de entrada realizados.
	EC 2.2 Buscar despacho especificando un rango de fecha.	El sistema lista los despachos de entrada filtrados por el rango de fecha.	-El sistema muestra un listado con los despachos de entrada realizados.
	EC 2.2 Realizar despacho salida sin especificar despacho.	El sistema no permite realizar un despacho de salida sin que se haya seleccionado un despacho.	-Se presiona el botón “Realizar despacho salida” -El sistema muestra un mensaje informativo al usuario, aclarando que debe seleccionar un despacho.
	EC 2.3 Realizar despacho especificando un despacho.	El sistema permite realizar el despacho de salida del despacho seleccionado.	-Se selecciona el despacho. -Se presiona el botón “Realizar despacho salida”. -El sistema muestra una pantalla con los datos de la embarcación.
	EC 2.3.1 Registrar despacho sin llenarcorrectamente los datos obligatorios.	El sistema no permite registrar el despacho de salida.	-Se presiona el botón “Aceptar”. -El sistema muestra los campos obligatorios en rojo y no permite registrar el despacho.
	EC 2.3.2 Registrar despacho llenando correctamente los datos obligatorios.	El sistema permite registrar el despacho de salida.	-Se presiona el botón “Aceptar”. -El sistema muestra un cartel informando al usuario que el despacho ha sido registrado con éxito.

Sección 3: Modificar Despacho (Entrada/Salida)

Nombre de la sección	Escenario de la sección	Descripción de la funcionalidad	Flujo Central
3: Modificar Despacho (Entrada/Salida)	EC 3.1 Buscar despachos sin especificar el tipo de despacho (entrada/salida).	El sistema no permite buscar los despachos sin especificar el criterio de búsqueda.	-Se presiona el botón “Buscar”. -El sistema muestra los campos obligatorios para el criterio de búsqueda y un mensaje informativo.
	EC 3.2 Buscar despachos especificando el tipo de despacho.	El sistema lista los despachos por el criterio de búsqueda especificado filtrados por la fecha actual.	-El sistema muestra un listado con los despachos.
	EC 3.3 Buscar despachos especificando el tipo de despacho y un rango de fecha.	El sistema lista los despachos por el criterio de búsqueda especificado y filtrado por el rango de fecha dado.	-El sistema muestra un listado con los despachos.
	EC 3.4 Modificar despacho sin especificar uno.	El sistema no permite realizar una modificación de un despacho sin que se haya seleccionado anteriormente.	-Se presiona el botón “Modificar despacho” -El sistema muestra un mensaje informativo al usuario, aclarando que debe seleccionar un despacho.
	EC 3.5 Modificar despacho especificando uno.	El sistema permite realizar la modificación del despacho que se haya seleccionado anteriormente.	-Se selecciona el despacho. -Se presiona el botón “Modificar despacho”. -El sistema muestra una pantalla con los datos del despacho.
	EC 3.5.1 Guardar los datos del despacho modificado con los datos obligatorios incorrectos.	El sistema no permite registrar la modificación de los datos del despacho seleccionado.	-Se presiona el botón “Aceptar”. -El sistema muestra los campos obligatorios en rojo y no permite registrar la modificación del despacho.
	EC 3.5.2 Guardar los datos del despacho	El sistema permite registrar la modificación de los datos del	-Se presiona el botón

	modificado con los datos obligatorios correctos.	despacho seleccionado.	“Aceptar”. -El sistema muestra un cartel informando al usuario que lamodificación del despacho se realizó con éxito.

3.9.1.1.2 CPR 1 Registrar Medida

Descripción General: Requisitos Registrar Medida. El sistema permite registrar medidas correspondientes a una Embarcación de Recreo determinada.

Condiciones de Ejecución:

- ✓ El usuario debe estar autenticado en el sistema.
- ✓ El usuario debe seleccionar el menú Medidas.

Nombre de la sección	Escenario de la sección	Descripción de la funcionalidad	Flujo Central
1: Registrar Medida	EC 1.1 Registrar medida sin llenarcorrectamente los datos obligatorios.	El sistema no permite registrar la medida	-Se presiona el botón “Aceptar”. -El sistema muestra los campos obligatorios que deben ser llenados mostrando un mensaje informativo.
	EC 1.2 Registrar medida llenando correctamente los datos obligatorios (ya la medida fue registrada anteriormente)	El sistema valida que la medida no haya sido registrada anteriormente, en caso positivo no permite registrarla.	-Se presiona el botón “Aceptar”. -El sistema muestra un mensaje informando que la medida ya fue registrada con anterioridad.
	EC 1.3 Registrar medida llenando correctamente	El sistema valida que la medida no haya sido registrada	-Se presiona el botón

	los datos obligatorios (medida nueva)	anteriormente, en caso negativo permite registrarla.	“Aceptar”. -El sistema muestra un mensaje informando al usuario que la medida se registró satisfactoriamente.

3.10 Conclusiones Parciales

En este capítulo se describió como se llevó a cabo la construcción de la aplicación. Así como la creación del diagrama de componentes donde se evidencia la descripción de los elementos físicos del sistema y sus relaciones; y la construcción del diagrama de despliegue. También se mostró fragmentos de código de uno de los requisitos funcionales de la aplicación, junto con su interfaz de usuario. Además se valida el sistema desarrollado mediante pruebas de caja negra, realizando casos de prueba a todas las funcionalidades del mismo en dos iteraciones. Este proceso permite detectar la mayor cantidad de no conformidades que este presentaba para darle solución a las mismas. Luego de una segunda iteración no se encontraron no conformidades, obteniendo una aplicación que satisfaga los requisitos definidos.

Conclusiones Generales

Al concluir la realización de este trabajo se cumplieron los objetivos propuestos con el mismo, se elaboró la fundamentación teórica de la investigación donde se realizó un breve análisis de la metodología de desarrollo a seguir así como las mejores prácticas de programación Web, se hizo referencia a las herramientas, framework y leguajes de programación utilizados y el estudio de los patrones de diseño. También se realizaron los diagramas de diseño de la aplicación, los de secuencia y clases del diseño con estereotipos Web. Posteriormente se hizo una validación del diseño realizado utilizando las métricas Tamaño Operacional de Clases (TOC) y Relaciones entre Clases (RC),arrojado resultados satisfactorios, para el TOC se evidencia una baja responsabilidad entra las clases y un aumento de su reutilización, evitando así una alta complejidad en las clases y para la RC demostró que para un 54% no existen niveles dependencia entra las clases, lo que posibilita una alta capacidad de reutilización. Se realizó la implantaciónde la aplicación con los artefactos generados en la etapa de diseño, y se confeccionaron los casos de prueba para validar el correcto funcionamiento de la aplicación, obteniendo como resultado 10 no conformidades en la primera iteración, las cuales fueron corregidas en su totalidad en la segunda iteración. Lográndose así la construcción de una versión funcional de un sistema informático para llevar a cabo el despacho de la entrada y salida de las Embarcaciones de Recreo, logrando un mejor control de estas durante su estancia en una marina cubana, que responde al 100% de los requisitos funcionales exigidos por el cliente, ajustándose de esa forma a las necesidades primarias que propiciaron la realización de este trabajo:

• Bajo costo de producción.
• Ajuste a las normativas aduaneras de la AGR.
• Automatización de los procesos de negocio relativos al despacho de las Embarcaciones de Recreo.

Por la importancia que tiene para la AGR el buen funcionamiento de esta aplicación se recomienda:

- ✓ Seguir trabajando en el desarrollo del sistema, para mejoras futuras.
- ✓ Refactorizar con el objetivo de optimizar la aplicación en términos de rendimiento y tiempo de respuesta a las peticiones, así como encontrar posibles fallas no previstas.

Referencias

1. **República**, A.G.d.l. Glosario de términos aduaneros.**Volume**. [Citado: 2010].

2. **Seamax Project** 2009 2010 [cited 2009 diciembre]; Available from: http://www.seamaxproject.com/es/index.php/Portada.

3. **Lluch Jordi**. A.F.P.,**PecciJulia**, **Gordillo Manuel**, **Benítez José**, **Navarro Juan** DIANA – SISTEMA DE GESTIÓN Y AYUDA A LA NAVEGACIÓN EN PUERTOS DEPORTIVOS. 2009 [cited; Available from: http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=1044

4. **Márquez Díaz*José, Sampedro**Leonardo, Vargas***Félix.** redalyc. *Instalación y configuración de Apache, un servidor Web gratis.* [En línea] 2002. redalyc.uaemex.mx/pdf/852/85201202.pdf.

5. **Benitez, Rolando Javier.** benitez-progra.blogspot.com. [En línea] 1 de 12 de 2010. http://benitez-progra.blogspot.com/2010/12/programacion-orientada-objetos.html.

6. **Tuesta, Martín.** unapiquitos.edu.pe. [En línea]. [Citado: 2010]. http://www.google.com/url?sa=t&source=web&cd=2&ved=0CCAQFjAB&url=http%3A%2F%2Fwww.unapiquitos.edu.pe%2Fintranet%2Fpagsphp%2Fdocentes%2Farchivos%2FCapitulo%25205%2520-%2520Herramientas%2520CASE.doc&rct=j&q=Herramientas%20CASE%20%28Computer%20Aided%20Softwa.

7. **Netbeans.** netbeans.org. [En línea][Citado: 2010]. www.netbeans.org/community/releases/68/index_es.html.

8. **Scribd.** [En línea][Citado: 2011]. http://es.scribd.com/doc/55136485/47/Ventajas-de-utilizar-un-Framework-de-trabajo.

9. **Symfony.es.** [En línea][Citado: 2010 - 2011]. http://www.symfony.es/que-es-symfony/.

10. **Cobas Díaz,Walberto.** Revista Tino. [En línea] 2007. http://revista.jovenclub.cu/index.php?option=com_content&task=view&id=634&Itemid=100.

11. **GilFidel, Albrigo Javier, Do Rosario Javier.** SISTEMAS DE GESTIÓN DE BASE DE DATOS SGBD / DBMS. Valencia : Universidad de Carabobo. Facultad Experimental de Ciencias y Tecnología, 14 de Febrero de 2005.

12. **Sierra, María.***Modelado de Implementación.* [PDF].[Citado: 2011]. s.l. : Univ. Cantabria – Fac. de Ciencias, Univ. Cantabria – Fac. de Ciencias. Práctica 7, Sesión 1.

13. **dsi.uclm.es.** dsi.uclm.es. [En línea] [Citado el: 17 de enero de 2011.] www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

14. **GomezPiñeiro,Yordanys.** Enterno Virtual de Aprendizaje. *04. Metodologías_de_desarrollo_de_software.* [En línea] 2009. http://eva.uci.cu/mod/resource/view.php?id=33748.

15. **Enterno Virtual de Aprendizaje.***Introduccion a Rup y UML.* [En línea] 2009. http://eva.uci.cu/mod/resource/view.php?id=34099.

16. **fdi.ucm.es**. *Estructura de las Aplicaciones Orientadas a Objetos. El patrón Modelo-Vista-Controlador (MVC).* [En línea] 2011. www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.

17**. Visual Paradigm**. *Visual Paradigm.* [En línea] 2004. [Citado el: 20 de enero de 2010.] http://www.visual-paradigm.com/product/vpuml/.

18. **CEIGE, Dpto Aduana.***Propuesta de un Estándar de Codificación*. [Citado*:* 2011].

19. **Entorno Visual de Aprendiza**. *Material_de_caja_b_y_caja_n.* [En línea] 2010. [Citado el: 20 de mayo de 2011.] http://eva.uci.cu/mod/resource/view.php?id=14104.

20**. Ávila Gonzalez, Rolando.***Componente para visualizar curvas de registro de pozos.* Ciudad de la Habana : s.n., 2010. Tesis.

21. **Capítulo 2**. **Explorando el interior de Symfony, 2.1**. El patrón MVC. [aut. libro] FrancoisZaninotto y FabienPotencier. *Symfony 1.2, la guía definitiva.* 2008.

22. **Herrera, L.H Tamayo y R.P.***Información Adelantada de Pasajeros: Análisis y Diseño.* Habana : s.n., 2009.

23. **Lorenz, M. y Kidd, J.***Object-Oriented Software Metrics.* 1994.

Bibliografía

1. **Vélez, Antonio.***Rup: Etapa de Analisis y Diseño.* [PDF] [Citado: 2010]. Palmira : Universidad del Valle.

2. **Maikel.** cristalab.com. *Crear e implementar el patrón de diseño Singleton en PHP.* [En línea] 8 de nov de 2008. http://www.cristalab.com/tutoriales/crear-e-implementar-el-patron-de-diseno-singleton-en-php-c256l/.

3. **Morales., Sanz Daniel Mínguez. García Emilio José.** www.eici.ucm.cl. *Metodologías para el Desarrollo de AplicacionesWeb: UWE.* [En línea][Citado: 2010] www.eici.ucm.cl/.../DASBD-Metodolog-ADasParaElDesarrolloDeaplicacionesWeb_UWE.pdf.

4. **Valle, Universidad del.** eisc.univalle.edu.co. *DIAGRAMAS DE CLASES DEL DISEÑO.* [En línea][Citado: 2011]. htttp://www.eisc.univalle.edu.co/cursos/.../DS2-CLASE-DIAGRAMAS_2.pdf.

5. **UCI.** Tema No. 3: El diseño metodológico de la investigación científica.[Citado: 2010].

6. **Herrington, Jack D.** digg.com. *Five common PHP design patterns.* [En línea] 18 de julio de 2006. http://www.digg.com/news/story/Five_common_PHP_design_patterns.

7. **Escalona, M.J., Koch, N..***Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo.* [En línea]. [Citado: 2010]. http://www.sistemas.edu.bo/.../Requisitos/Ingenier_a%20de%20Requisitos%20en%20Aplicaciones%20Web%20-....

8. **O'REILLY, TIM.** www.sociedadinformacion.fundacion.telefonica.com. *QUÉ ES WEB 2.0. PATRONES DEL DISEÑO Y MODELOS DEL NEGOCIO PARA LA SIGUIENTE GENERACIÓN DEL SOFTWARE.* [En línea] 23 de feb de 2006. http://www.sociedadinformacion.fundacion.telefonica.com/.../seccion=1188&idioma=es_ES&id=2009100116300061&activo=4.do?... .

9. **www.baluart.net**. *Introducción a los patrones de diseño con PHP.* [En línea] 19 de julio de 2010. http://www.baluart.net/.../introduccion-a-los-patrones-de-diseno-con-php.

10. **Silva Darío Andrés, Mercerat Bárbara.** www.lifia.info.unlp.edu.ar. *Construyendo aplicaciones web con una metodología de diseño orientada a objetos.* [En línea] 1 de feb de 2002. http://www.lifia.info.unlp.edu.ar/papers/2001/Silva2001.pdf.

11. **Tudela, Calvo José Carlos.** www.intelligenia.com. *Reutilizació, Parametrización y Patrones de Diseño.* [En línea][Citado: 2011]. http://www.intelligenia.com.

12. **Mi Respuesta.com**. *Qué es un servidor web ?* [En línea] 2005-2011. http://www.misrespuestas.com/que-es-un-servidor-web.html.

13. **Insfrán Emilio, Tejadillos Elena, Martí Sofía, Burbano Margarita.***Transformación de Especificación de Requisitos en Esquemas Conceptuales usando Diagramas de Interacción.* Valencia : Universidad Politécnica de Valencia.

14. **República**, A.G.d.l. Glosario de términos aduaneros.**Volume**. [Citado: 2010].

15. **Seamax Project** 2009 2010 [cited 2009 diciembre]; Available from:http://www.seamaxproject.com/es/index.php/Portada.

16. **Lluch Jordi**. A.F.P.,**Pecci Julia**, **Gordillo Manuel**, **Benítez José**, **Navarro Juan** DIANA – SISTEMA DE GESTIÓN Y AYUDA A LA NAVEGACIÓN EN PUERTOS DEPORTIVOS. 2009 [cited; Available from: http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=1044

17. **Márquez Díaz*José, Sampedro**Leonardo, Vargas***Félix.** redalyc. *Instalación y configuración de Apache, un servidor Web gratis.* [En línea] 2002. redalyc.uaemex.mx/pdf/852/85201202.pdf.

18. **Benitez, Rolando Javier.** benitez-progra.blogspot.com. [En línea] 1 de 12 de 2010. http://benitez-progra.blogspot.com/2010/12/programacion-orientada-objetos.html.

19. **Tuesta, Martín.** unapiquitos.edu.pe. [En línea].[Citado:2010]. http://www.google.com/url?sa=t&source=web&cd=2&ved=0CCAQFjAB&url=http%3A%2F%2Fwww.unapiquitos.edu.pe%2Fintranet%2Fpagsphp%2Fdocentes%2Farchivos%2FCapitulo%25205%2520-%2520Herramientas%2520CASE.doc&rct=j&q=Herramientas%20CASE%20%28Computer%20Aided%20Softwa.

20. **Netbeans.** netbeans.org. [En línea].[Citado: 2010]. www.netbeans.org/community/releases/68/index_es.html.

21. **Scribd.** [En línea].[Citado: 2011]. http://es.scribd.com/doc/55136485/47/Ventajas-de-utilizar-un-Framework-de-trabajo.

22. **Symfony.es.** [En línea]. [Citado: 2010 - 2011]. http://www.symfony.es/que-es-symfony/.

23. **Cobas Díaz,Walberto.** Revista Tino. [En línea] 2007. http://revista.jovenclub.cu/index.php?option=com_content&task=view&id=634&Itemid=100.

24. **GilFidel, Albrigo Javier, Do Rosario Javier.** SISTEMAS DE GESTIÓN DE BASE DE DATOS SGBD / DBMS. Valencia : Universidad de Carabobo. Facultad Experimental de Ciencias y Tecnología, 14 de Febrero de 2005.

25. **Sierra, María.***Modelado de Implementación.* [PDF]. [Citado: 2011]. s.l. : Univ. Cantabria – Fac. de Ciencias, Univ. Cantabria – Fac. de Ciencias. Práctica 7, Sesión 1.

26. **dsi.uclm.es.** dsi.uclm.es. [En línea] [Citado el: 17 de enero de 2011.] www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

27. **Gomez Piñeiro, Yordanys.** Enterno Virtual de Aprendizaje. *04. Metodologías_de_desarrollo_de_software.* [En línea] 2009. http://eva.uci.cu/mod/resource/view.php?id=33748.

28. **Enterno Virtual de Aprendizaje.***Introduccion a Rup y UML.* [En línea] 2009. http://eva.uci.cu/mod/resource/view.php?id=34099.

29. **fdi.ucm.es**. *Estructura de las Aplicaciones Orientadas a Objetos. El patrón Modelo-Vista-Controlador (MVC).* [En línea] 2011. www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf.

30**. Visual Paradigm**. *Visual Paradigm.* [En línea] 2004. [Citado el: 20 de enero de 2010.] http://www.visual-paradigm.com/product/vpuml/.

31. **CEIGE, Dpto Aduana.***Propuesta de un Estándar de Codificación.* [Citado:2011].

32. **Entorno Visual de Aprendiza**. *Material_de_caja_b_y_caja_n.* [En línea] 2010. [Citado el: 20 de mayo de 2011.] http://eva.uci.cu/mod/resource/view.php?id=14104.

33**. Ávila Gonzalez, Rolando.***Componente para visualizar curvas de registro de pozos.* Ciudad de la Habana : s.n., 2010. Tesis.

34. **Capítulo 2**. **Explorando el interior de Symfony, 2.1**. El patrón MVC. [aut. libro] FrancoisZaninotto y FabienPotencier. *Symfony 1.2, la guía definitiva.* 2008.

35. **Herrera, L.H Tamayo y R.P.***Información Adelantada de Pasajeros: Análisis y Diseño.* Habana : s.n., 2009.

36. **Lorenz, M. y Kidd, J.***Object-Oriented Software Metrics.* 1994.

CVS: Siglas en inglés de *ConcurrentVersionsSystem* (Sistema de Control de Versiones), es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto (de programa) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren.

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 ObjectNotation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

DOM: (DocumentObjectModel - Modelo de Objetos del Documento o Modelo en Objetos para la representación de Documentos) es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

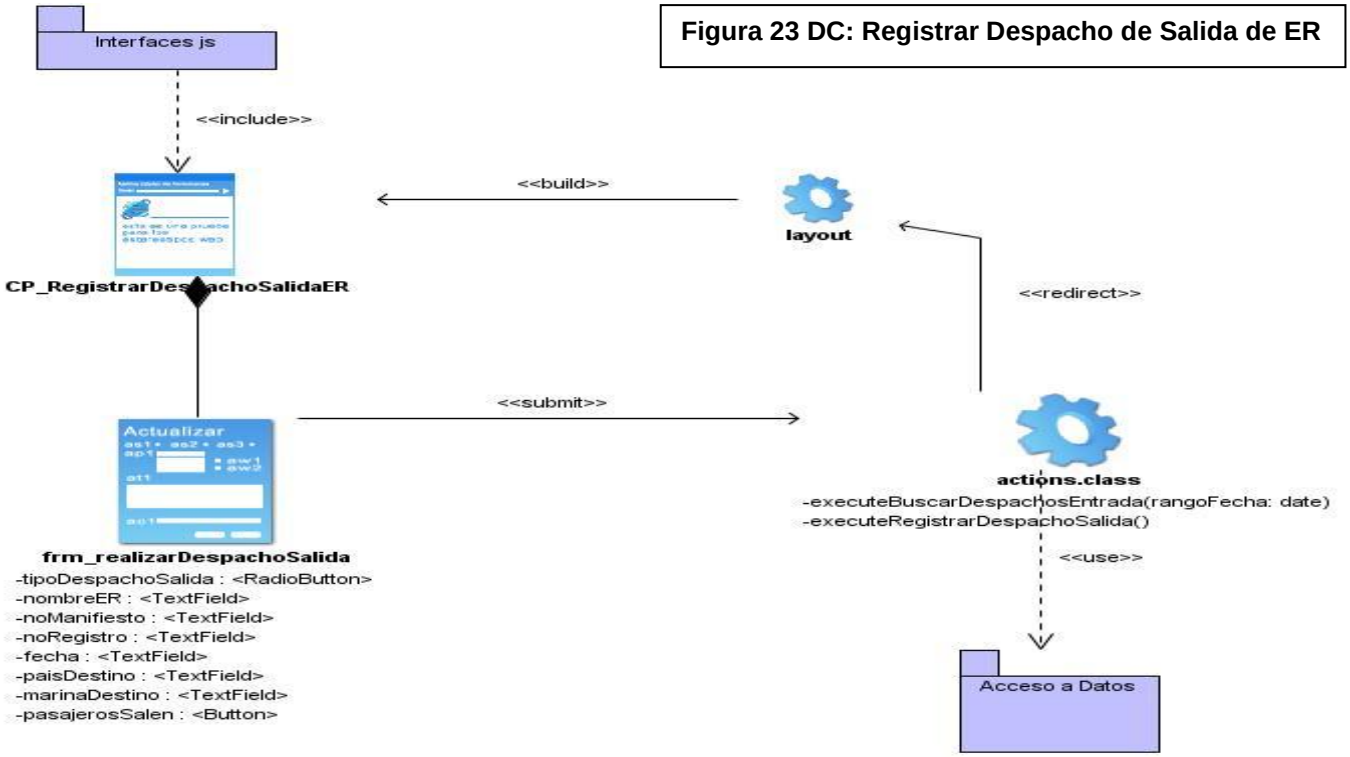


Figura 23 DC: Registrar Despacho de Salida de ER

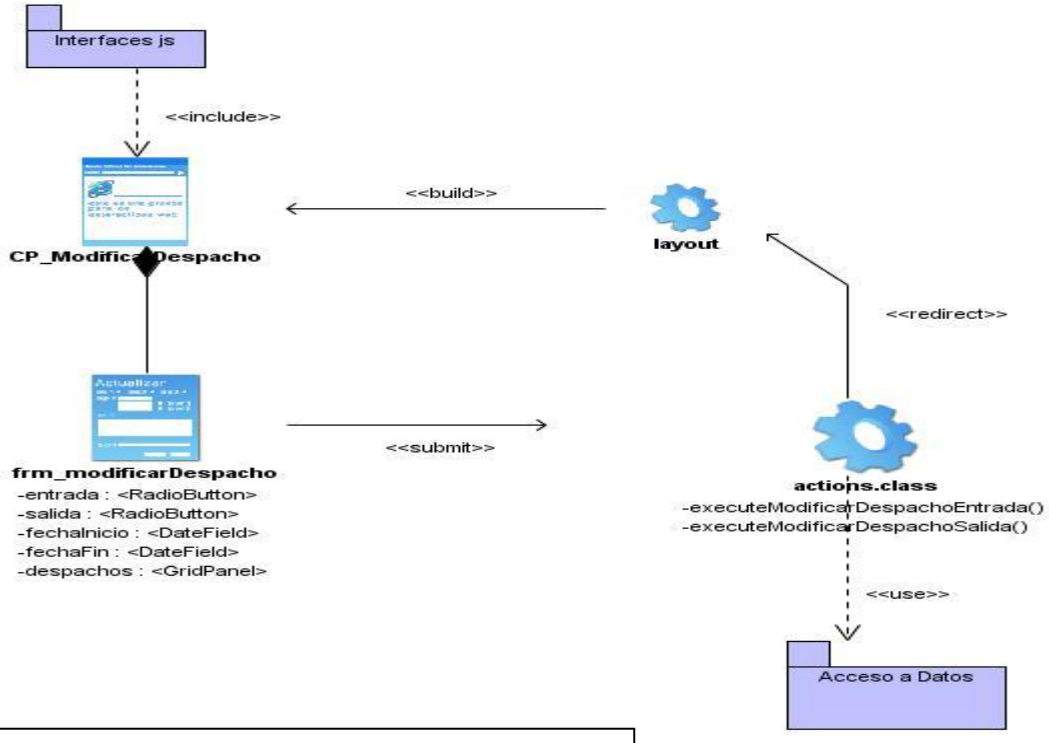


Figura 24 DC: Modificar Despacho Entrada/Salida ER

Figura 25 DC: Registrar Permiso Temporalidad ER

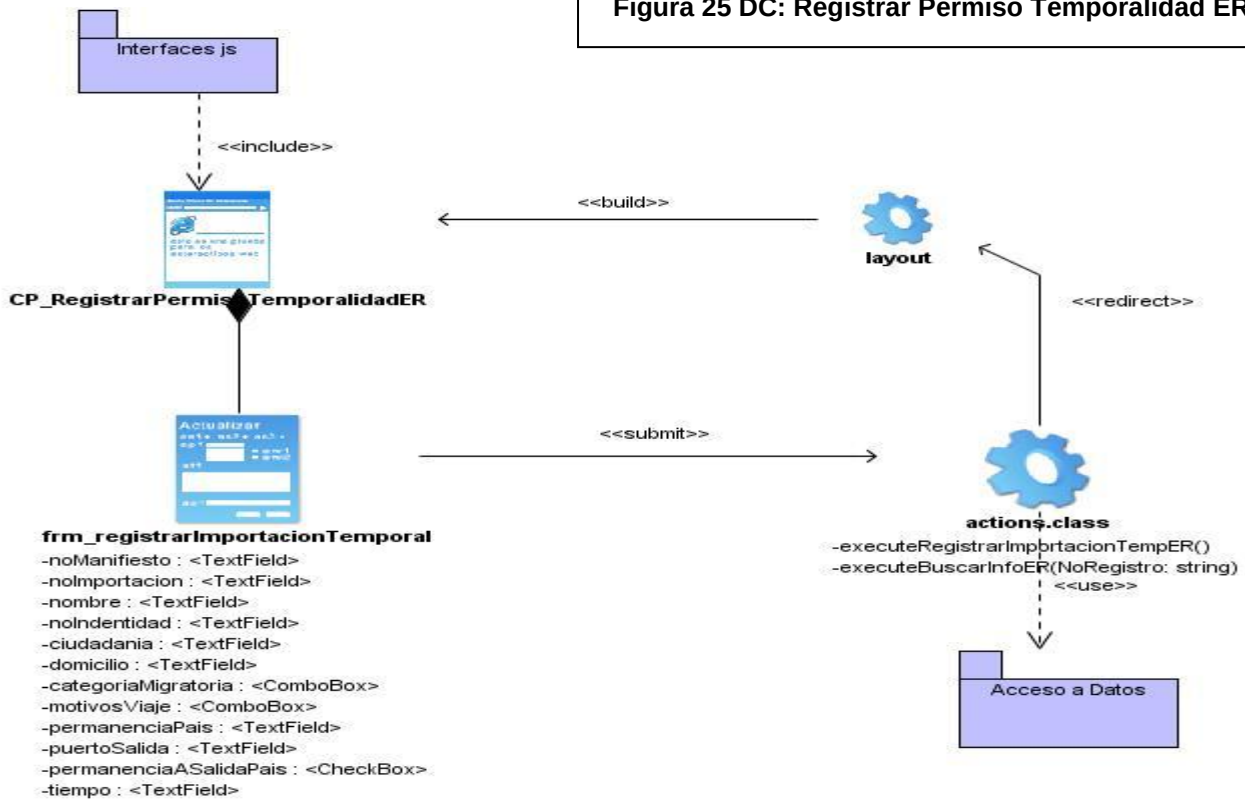


Figura 26 DC: Registrar Permiso Temporalidad Vehículo

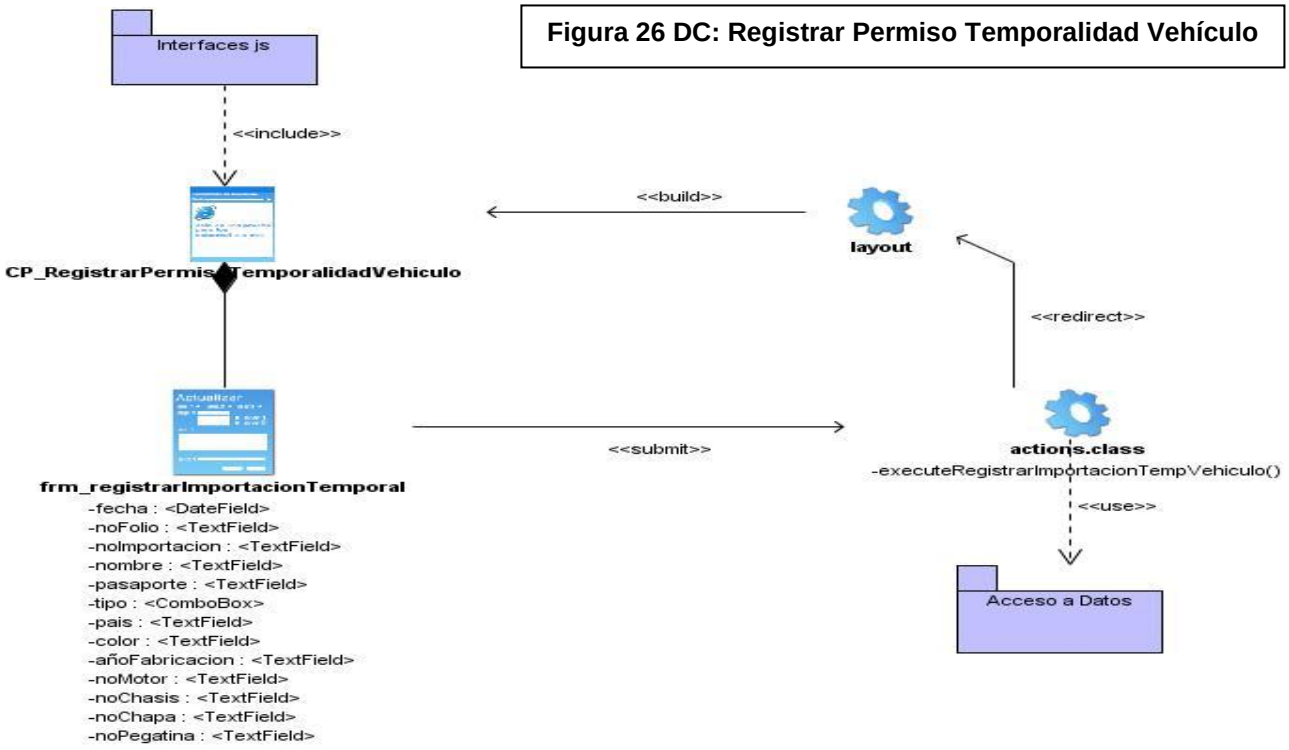


Figura 27 DC: Modificar Permiso de Temporalidad

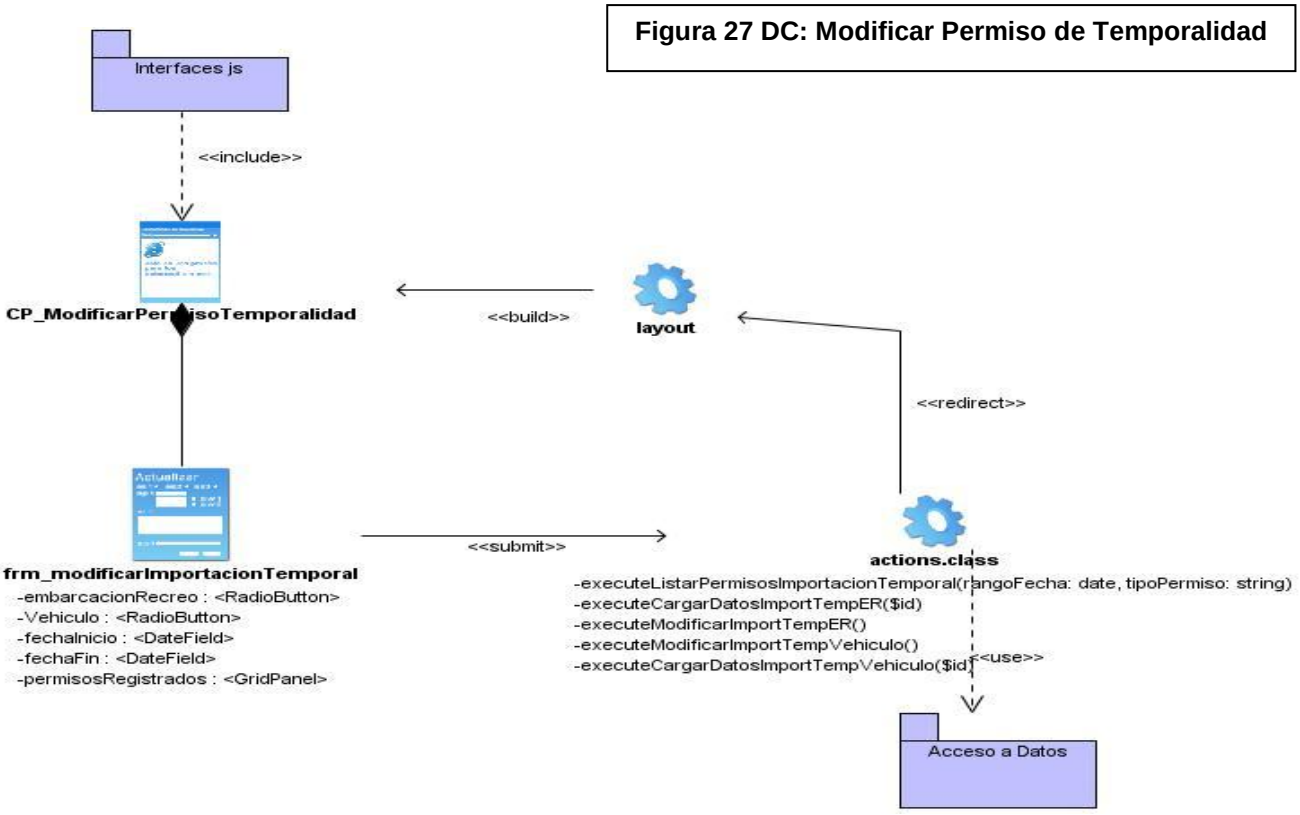


Figura 28 DC: Cancelar Permiso de Temporalidad

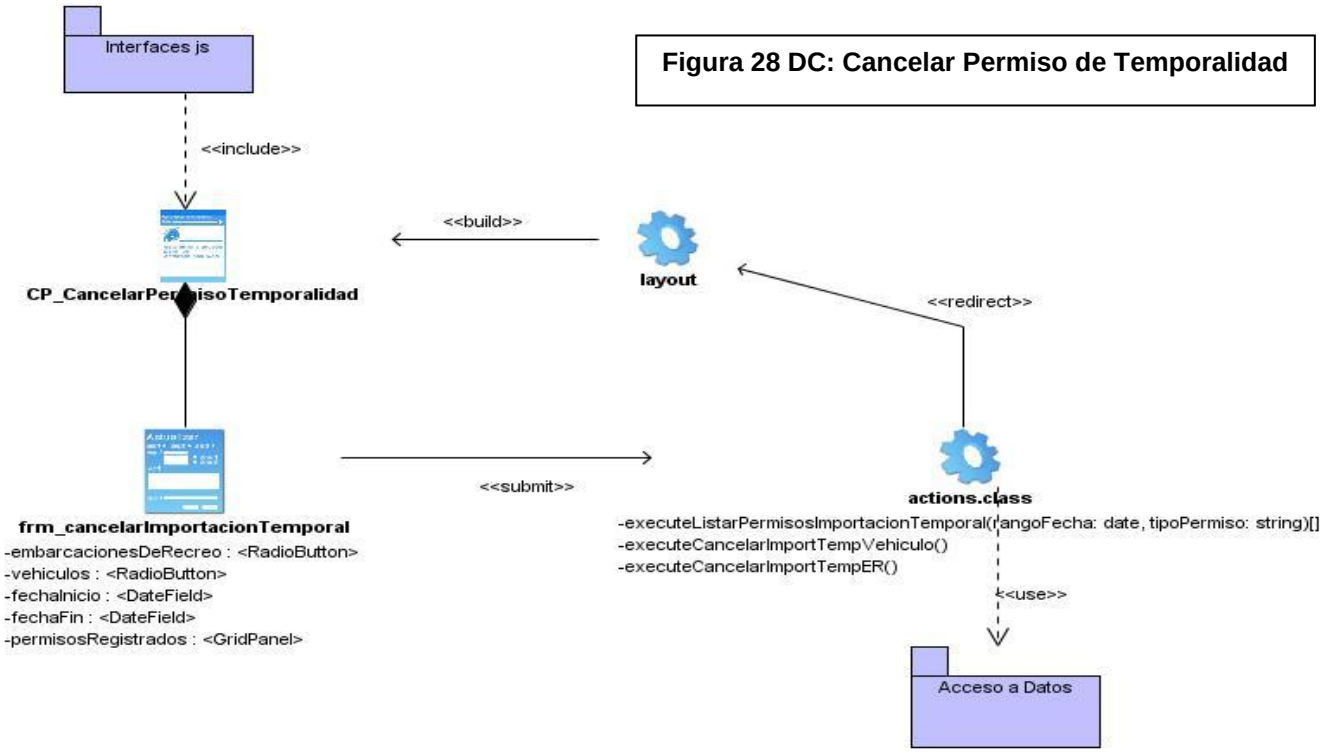


Figura 29 DC: Registrar Medida

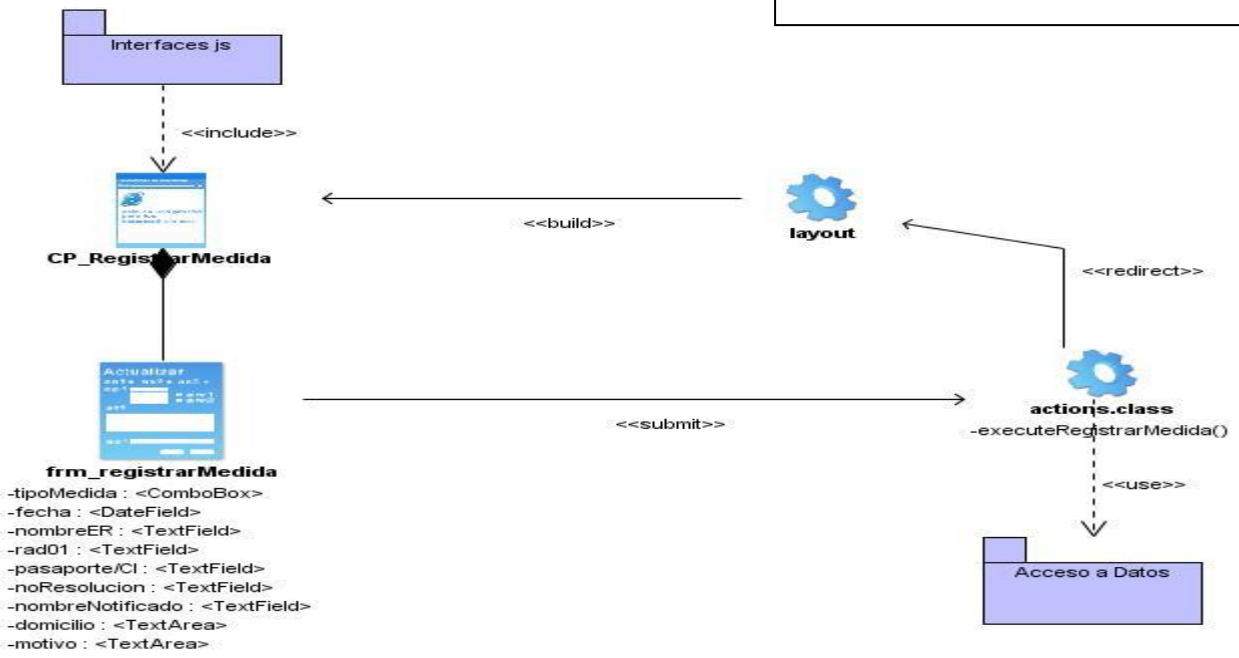
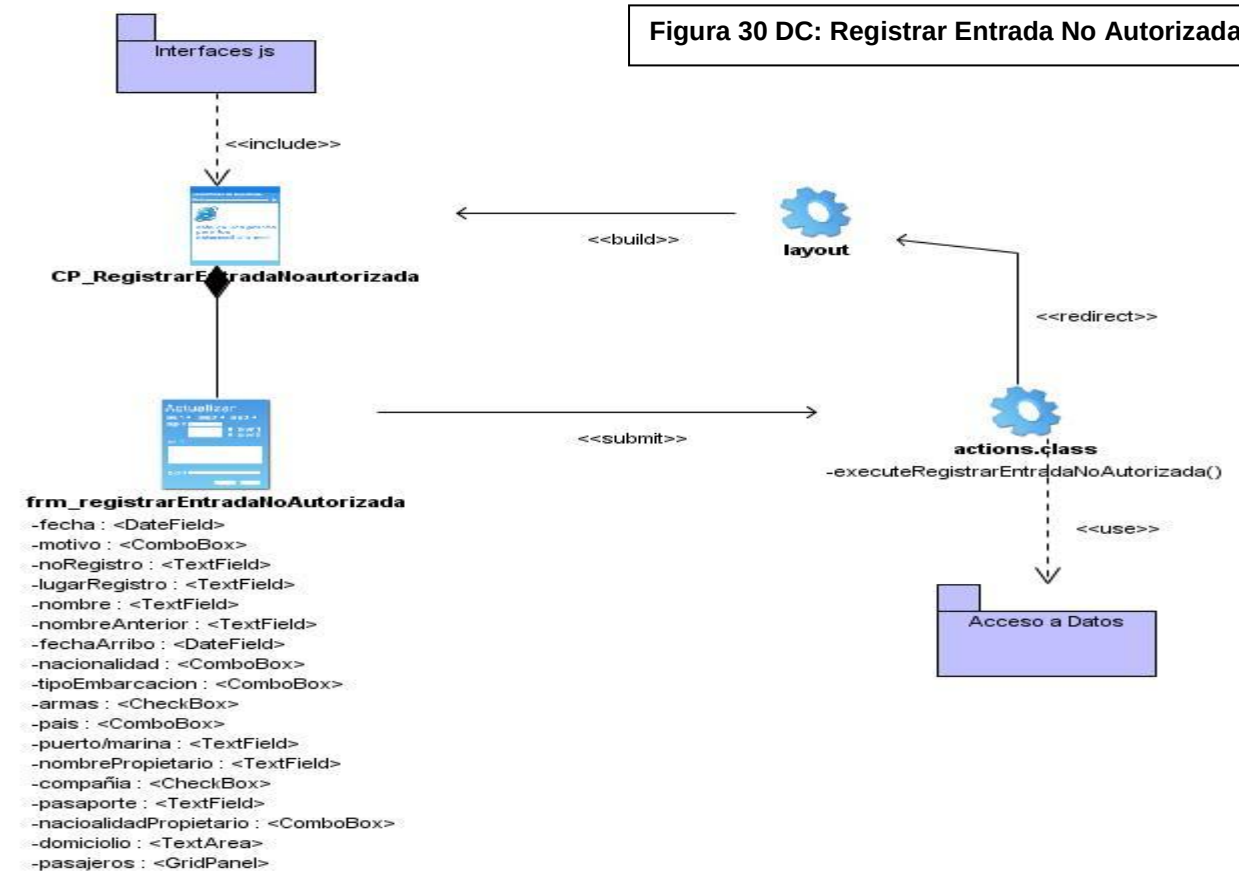


Figura 30 DC: Registrar Entrada No Autorizada



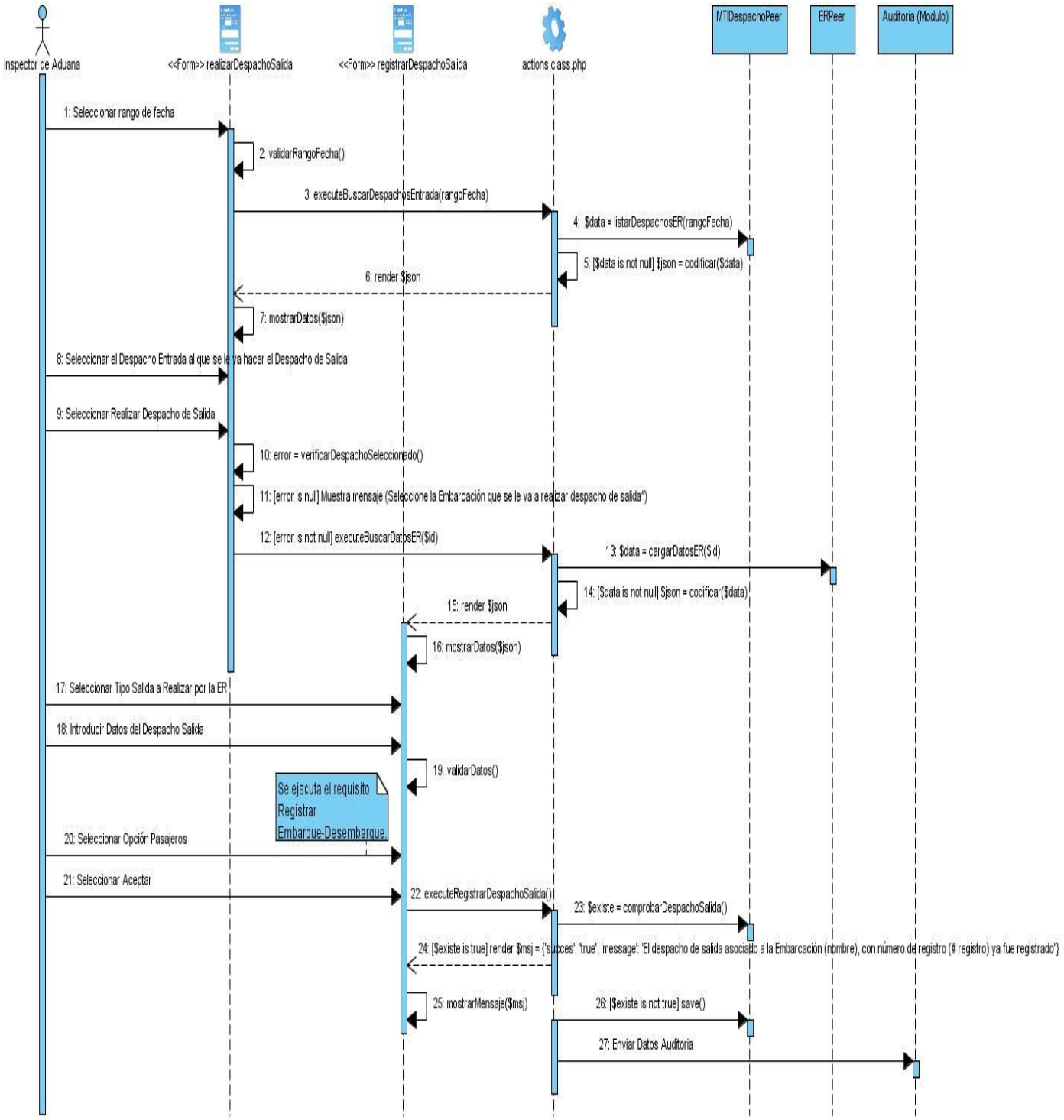


Figura 31 DS: Registrar Despacho Salida

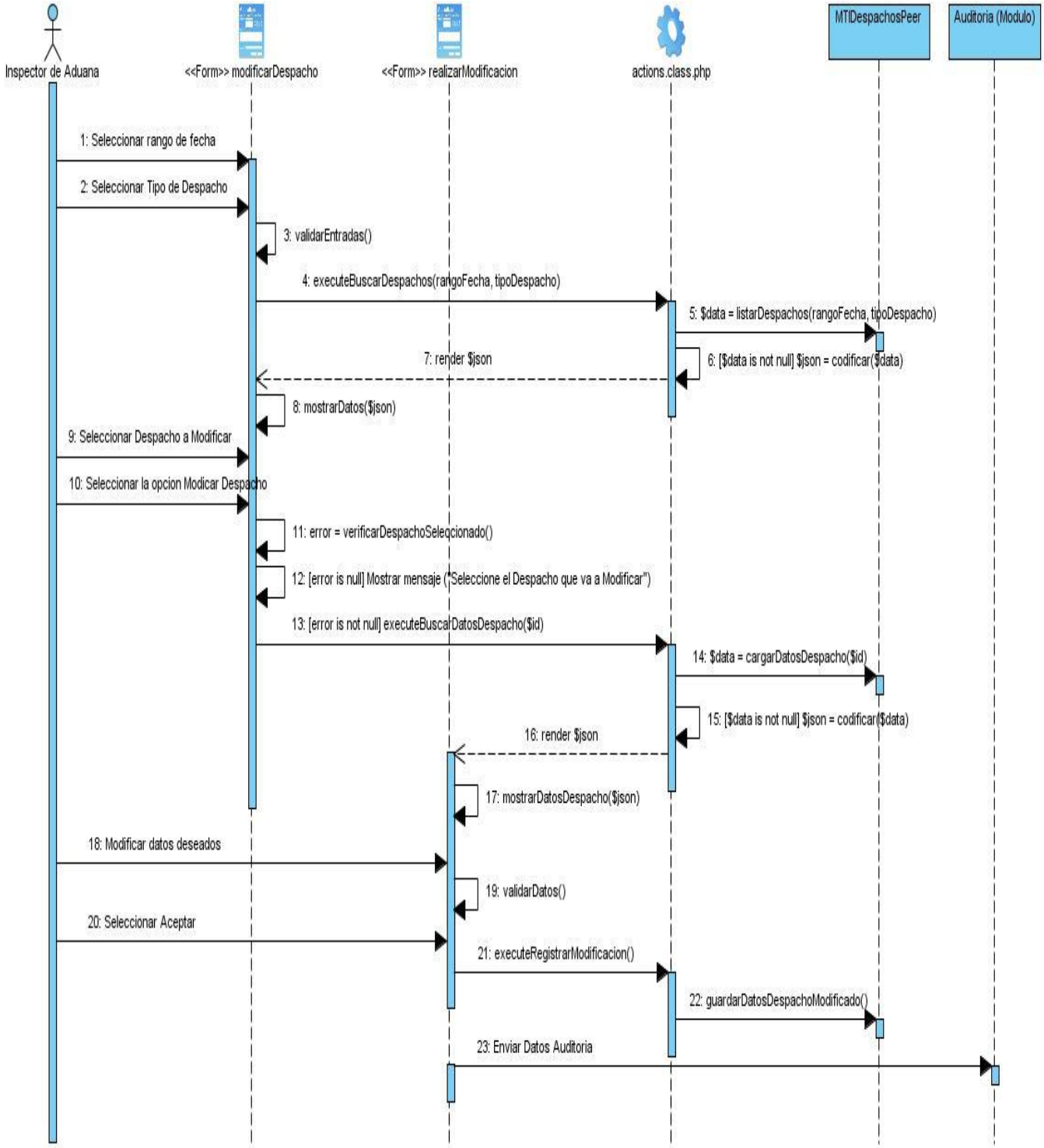


Figura 32 DS: Modificar Despacho

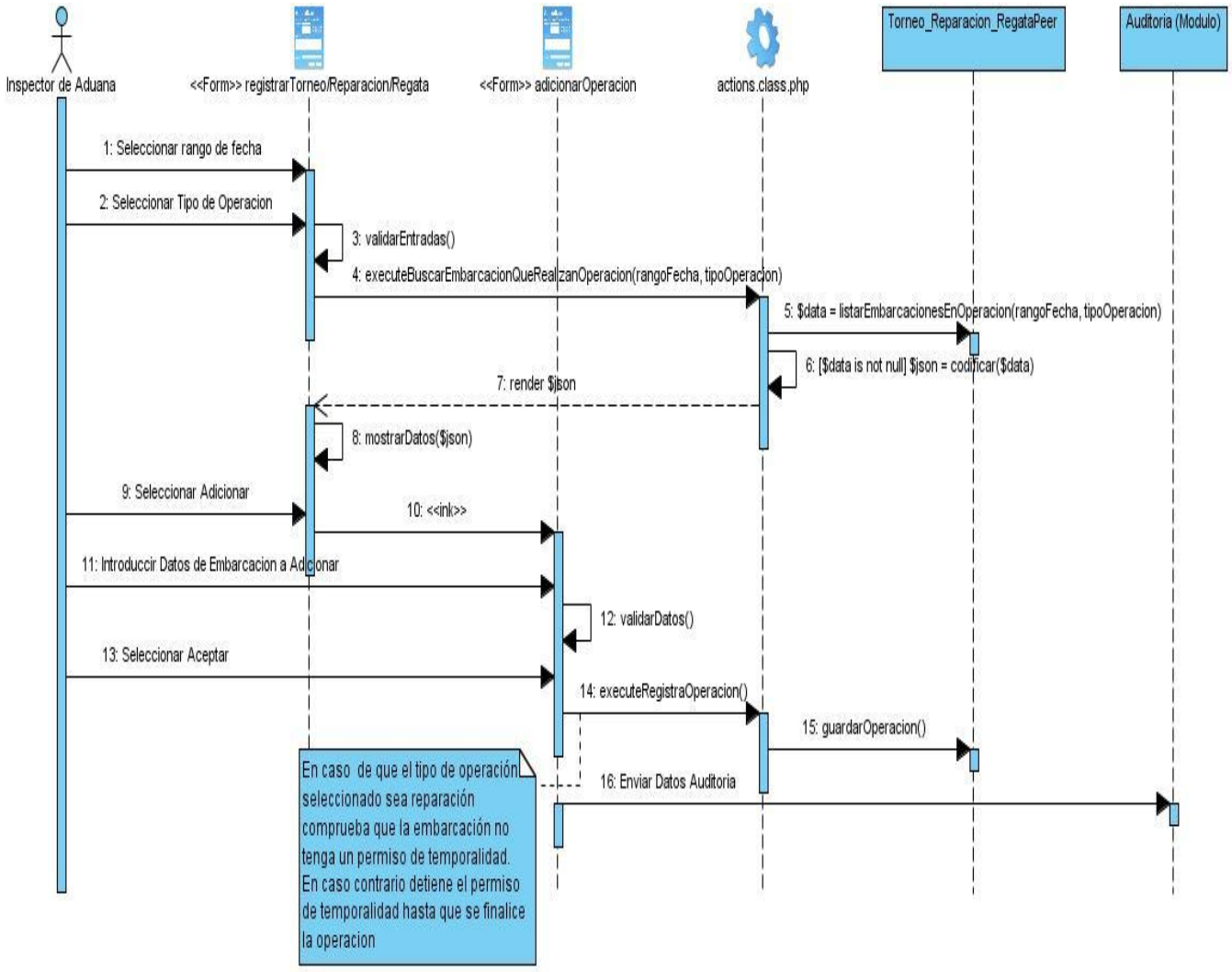
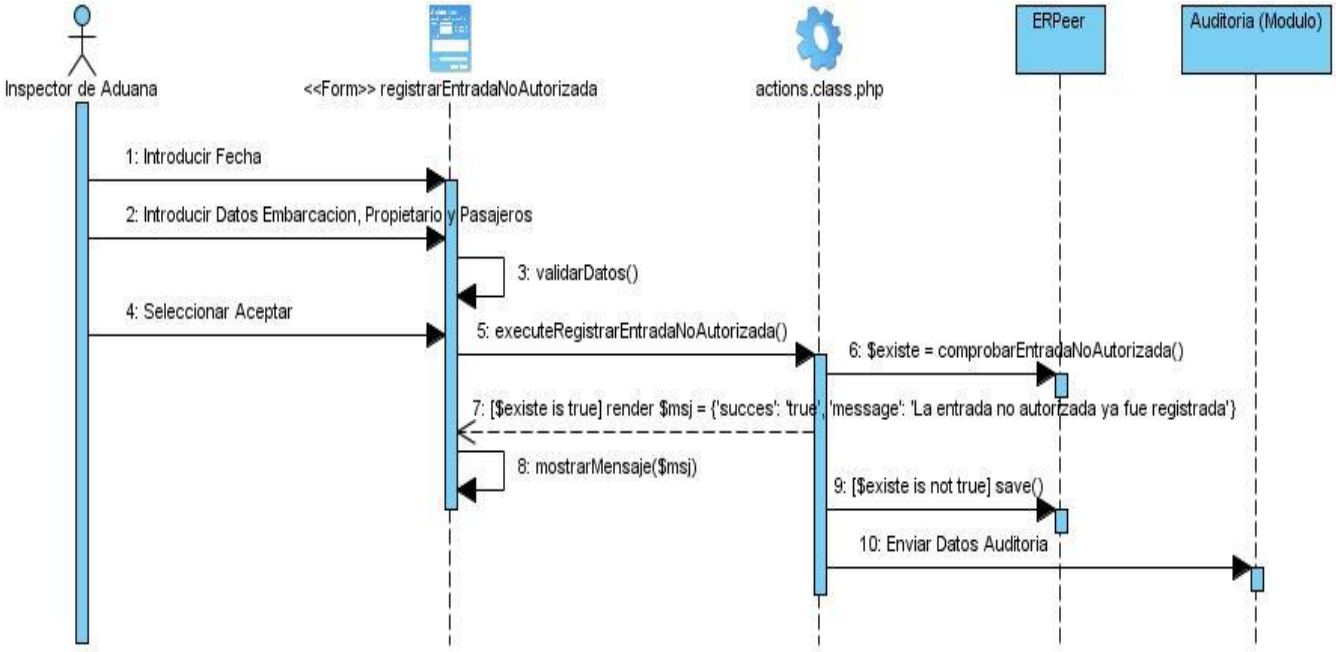


Figura 33 DS: Registrar Torneo/Regata/Reparación

Figura 34 DS: Registrar Entrada No Autorizada



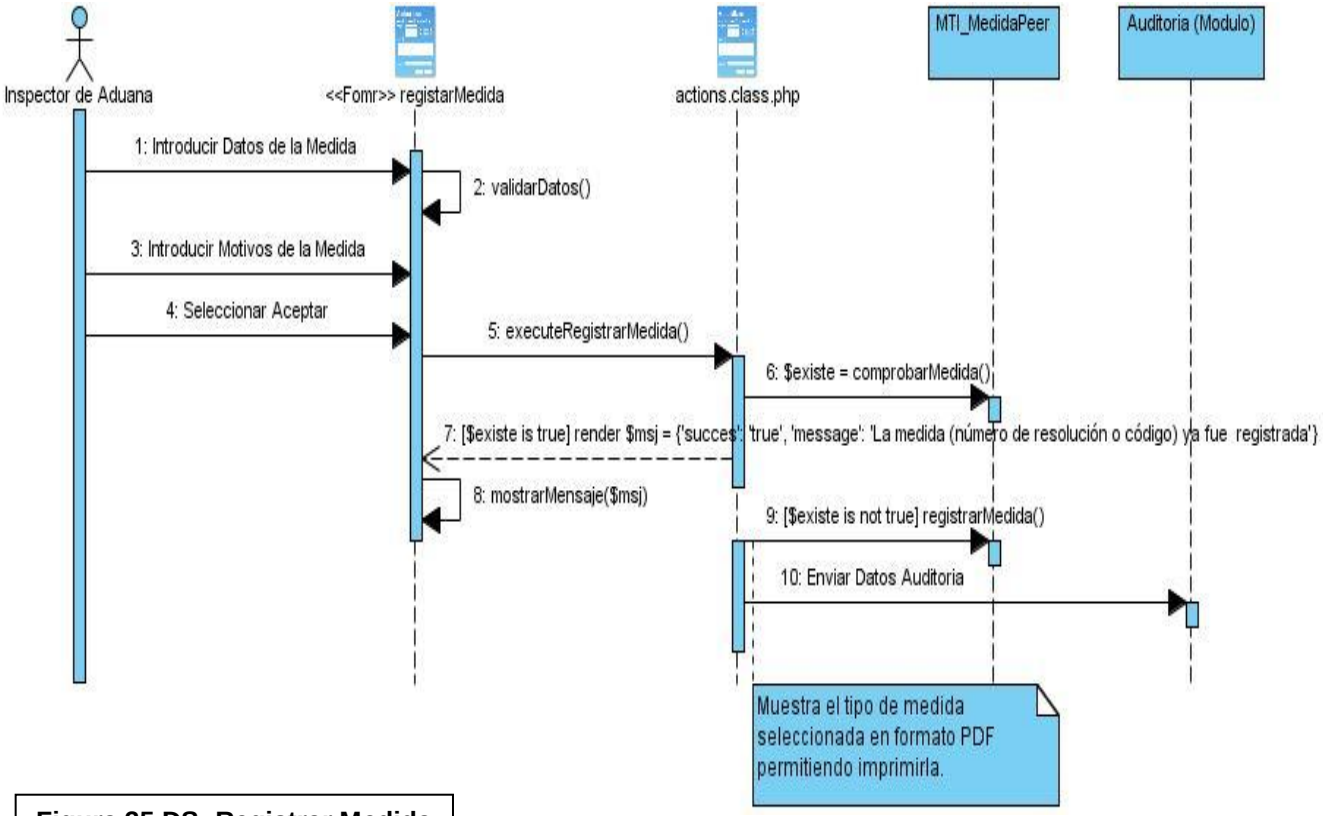


Figura 35 DS: Registrar Medida