Universidad de las Ciencias Informáticas

Facultad 3

Título: Diseño e Implementación del Módulo Despacho de los Medios de Transporte Marítimos para el Sistema de Gestión Integral de la Aduana.

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas

Autor: Ismail Vega Mena

Tutor: Ing. Manuel Ramón Almaguer Ochoa.

Ciudad de la Habana, Junio del 2011.

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas para que haga el uso que estime pertinente con este trabajo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año 2011.

_____ _____

Ismail Vega Mena Ing. Manuel Ramón Almaguer Ochoa

A mis padres Mario Jesús Vega Rodríguez del Rey y Gilda Beatriz Mena Trimiño, sin su apoyo no hubiera llegado tan lejos.

A mi mamá por haber sido mi guía durante estos 5 años, gracias a ti me he convertido en la persona que soy hoy, gracias por haber tenido siempre una respuesta llena de amor para todas mis preocupaciones, mami eres lo mejor que me ha dado la vida.

A mi papá muchas gracias por haber estado ahí cada vez que te necesité y por haberme dejado seguir mis sueños, tú has sido mi ejemplo papi. Gracias por depositar toda tu confianza en mí, sencillamente eres el mejor padre del mundo.

A mi tutor al cual le debo mucho tiempo y sacrificio, gracias por aguantarme desde tercer año. A mi jefe de tribunal, de proyecto y profesor al cual le debo mis conocimientos básicos de programación Alain quien es uno de los mejores pedagogos de esta universidad.

A mis dos cotutores Gretter por parte de la programación siempre con una respuesta a mis inquietudes y Leo por parte del PowerPoint y el documento.

A mis amigos de la infancia:
Javier, Adrián, Yansey y Yunior.
A mis amigos aquí en la UCI:
A Wilber (por su tranquilidad al realizar las cosas), Bosch(por ser junto a Wilber mis metas de superación constante en casi todos los sentidos), Perdomo (por sus consejos, siempre escuchando lo que tengo que decir), Asdrúbal(por molestarse al verme haciendo lo que no tenía que hacer), Gilberto(por estar siempre ahí cuando todos se iban de fiesta y yo me quedaba), Richard(por su incondicionalidad ante todo), Carlos Abel (por sus críticas constructivas, constantemente, a cualquier hora y en todos los sentidos), Dayron (por su manera tan eficaz de evadir al chucho), Leo (por ser tan preocupado por sus amistades ya sean recientes de verdad me lamento por no haberte conocido antes), Alexander (por ser a veces más crítico que Carlos Abel), Janio (por su caminao), Charles (por ser el rival a cual no he podido

superar en ajedrez desde primer año), a Gustavo (por ser un gran compañero tanto en el aula como en el apto), al Yasma (por su risa y buen humor). Sin ustedes la vida aquí en la UCI no hubiese sido la misma...

A mis compañeros de laboratorio, profesores y estudiantes por dedicarme un minuto cada vez que lo necesitaba, a Ricardo y a Céspedes, a Adrián por la base de Datos y todo en lo que me ayudo, a Álvaro por ayudarme con la interfaz visual ya que sin él no hubiera podido probar nunca mi aplicación, te debo mucho hermano, a JJ, eres el mejor explicando cualquier cosa te lo agradezco de corazón y al tribunal por toda su ayuda y consejos.

A Mayde, Mariem, Lili, Elizabet, Adneris, Beba, Natacha, a todas muchas gracias, un beso.

Este trabajo va dedicado a mi madre, la mejor que puede existir, quien ha sido el modelo de excelencia, tolerancia y fuerza, quien con sus enseñanzas y amor ha luchado siempre por darme todo para poder lograr mis fines. A mi padre, por haberme estimulado a seguir adelante, y por ser un ejemplo de disciplina y superación constante.

A todos los que siempre creyeron en mí.

En la Aduana General de la República de Cuba se realiza el proceso de Despacho de los Medios de Transporte Internacional que arriban y salen del país, ellos pueden ser aeronaves, buques o embarcaciones de recreo. El objetivo de este proceso se centra en garantizar el despacho de la documentación requerida por la Aduana, pretendiendo sea ágil, dinámico y simplificado el despacho, con el fin de reducir los trámites de carga y descarga cumpliendo con los requisitos establecidos por la aduana. Este proceso consiste en el cumplimiento de un conjunto de formalidades necesarias para que las mercancías sean sometidas a un régimen aduanero. Para su realización es necesario manejar gran cantidad de información, la cual en la actualidad se realiza de forma manual, lo que provoca que se prolongue considerablemente el procesamiento de las mismas.

Para el desarrollo de este trabajo se propone como solución el diseño y la implementación del proceso de Despacho de Buques como Medio de Transporte Internacional.

Para desarrollar el sistema fue necesario dirigir el proceso a través de una metodología de desarrollo de software, y hacer uso de herramientas ya definidas para generar los artefactos que propone dicha metodología. Se realiza una valoración crítica del diseño propuesto por los analistas del sistema, se valida la propuesta de solución a través del desarrollo de un conjunto de pruebas realizadas para garantizar la calidad del software y se exponen algunas recomendaciones.

PALABRAS CLAVE: Aduana General de la República de Cuba (AGR), Medios de Transporte Internacional (MTI).

INTRODUCCIÓN **1**

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA **4**

1.1. INTRODUCCIÓN 4
1.2. DEFINICIONES 4
1.2.1. MTI *4*
1.2.2. GINA *4*
1.2.3. Proceso de Despacho de los Medios de Transporte Internacional *4*
1.2.4. Buque: *5*
1.3. APLICACIONES DE SISTEMAS INFORMÁTICOS EN EL MUNDO PARA EL CONTROL ADUANERO 5
1.3.1. SOFIA *5*
1.4. DISEÑO DE SISTEMAS DE SOFTWARE 8
1.5. PATRONES QUE SE UTILIZAN EN EL DISEÑO DE APLICACIONES WEB 11
1.6. BUENAS PRÁCTICAS EN LA PROGRAMACIÓN WEB 14
1.7. METODOLOGÍA DE DESARROLLO 15
1.8. LENGUAJE DE PROGRAMACIÓN 16
1.9. FRAMEWORK DE TRABAJO 16
1.10. INTERFAZ DE USUARIO 17
1.11. SERVIDOR WEB 17
1.11.1. Apache 2.2.x *17*
1.12. SISTEMA GESTOR DE BASE DE DATOS 19
1.13. HERRAMIENTAS DE DESARROLLO 19
1.13.1. Herramienta CASE *19*
1.14. ENTORNO DE DESARROLLO INTEGRADO (IDE) 20
1.14.1. NetBeans IDE 6.9 ..... *20*
1.15. MOZILLA FIREFOX VERSIÓN 3.5 21
1.16. CONCLUSIONES PARCIALES 22

CAPÍTULO 2: DISEÑO DEL SISTEMA **23**

2.1. INTRODUCCIÓN 23
2.2. REQUISITOS FUNCIONALES 23
2.3. VALIDACIÓN DE LOS REQUISITOS DE SOFTWARE 24
2.4. PATRONES DE DISEÑOS UTILIZADOS 24
2.4.1. Patrones GRASP *24*
2.4.2. Patrones GOF *26*
2.5. PATRÓN DE ARQUITECTURA 26
2.5.1. Modelo Vista Controlador (MVC) *26*
2.6. DIAGRAMAS DE CLASE DEL DISEÑO 29
2.6.1. Requisito Funcional: Confirmar Arribo/Salida *29*
2.6.2. Extensiones para el diseño Web *31*
2.6.3. Clases del diseño con estereotipos web *33*
2.7. DIAGRAMAS DE SECUENCIA 33
2.8. DIAGRAMA DE CLASES 36
2.9. DIAGRAMA DE DESPLIEGUE 37
2.10. MODELO DE DATOS 38

2.11. MÉTRICAS PARA EL MODELO DE DISEÑO 39
2.11.1. Métricas orientadas a clases 40
2.11.2. Métricas orientadas a Operaciones 46
2.12. CONCLUSIONES PARCIALES 47

CAPÍTULO 3: IMPLEMENTACIÓN DE LA SOLUCIÓN 48

3.1. INTRODUCCIÓN 48
3.2. ESTÁNDAR DE CODIFICACIÓN 48
3.3. DIAGRAMA DE COMPONENTES FÍSICOS 51
3.4. DIAGRAMA DE COMPONENTES LÓGICOS 52
3.5. IMPLEMENTACIÓN DE SYMFONY 52
3.6. IMPLEMENTACIÓN DEL PROYECTO, LA APLICACIÓN Y EL MÓDULO 53
3.7. IMPLEMENTACIÓN DEL ESQUEMA Y EL MODELO 53
3.8. INTERFAZ DE USUARIO 54
3.9. PRUEBAS 58
3.9.1. Pruebas de Caja Negra 59
3.10. CONCLUSIONES PARCIALES 62

CONCLUSIONES 63

RECOMENDACIONES 64

REFERENCIAS BIBLIOGRÁFICAS 65

BIBLIOGRAFÍA 67

GLOSARIO DE TÉRMINOS 69

TABLA 1 CRONOGRAMA DE TAREAS. 3
TABLA 2. ETAPAS DEL DESARROLLO DE SOFTWARE. 8
TABLA 3 ESTEREOTIPOS PARA LAS CLASES. 32
TABLA 4 ESTEREOTIPOS PARA LAS RELACIONES ENTRE CLASES. 32
TABLA 5 RELACIONES QUE SE ESTABLECEN ENTRE LAS CLASES QUE CONFORMAN LA EXTENSIÓN UML PARA WEB. 33
TABLA 6 PROMEDIO DE PROCEDIMIENTOS POR CLASE. 41
TABLA 7 TÉCNICAS PARA RESPONSABILIDAD 41
TABLA 8 TÉCNICAS PARA LA COMPLEJIDAD DE LA IMPLEMENTACIÓN. 41
TABLA 9 TÉCNICAS PARA REUTILIZACIÓN. 41
TABLA 10 CP1: ADICIONAR ARRIBO/SALIDA. 59
TABLA 11 CP2: CONFIRMAR ARRIBO/SALIDA. 60
TABLA 12 CP3: REGISTRAR DESPACHO ENTRADA/SALIDA. 61
TABLA 13 CP4: REGISTRAR NOTIFICACIÓN DE INFRACCIÓN. 61
TABLA 14 CP5: OTORGAR NÚMERO DE MANIFIESTO. 61

FIGURA 1 CAPAS DEL MVC 28
FIGURA 2 COMPONENTES DEL MVC 28
FIGURA 3 REQUISITO: CONFIRMAR ARRIBO/SALIDA. 30
FIGURA 4 REQUISITO: ADICIONAR ARRIBO. 31
FIGURA 5 PAQUETE DE ACCESO A DATOS. 33
FIGURA 6 DIAGRAMA DE SECUENCIA CONFIRMAR ARRIBO/SALIDA DE BUQUES. 34
FIGURA 7 DIAGRAMA DE SECUENCIA ADICIONAR ARRIBO 35
FIGURA 8 DIAGRAMA DE CLASES 36
FIGURA 9 DIAGRAMA DE DESPLIEGUE 37
FIGURA 10 MODELO DE DATOS. 38
FIGURA 11 POR CIENTO DE RESPONSABILIDAD. 42
FIGURA 12 POR CIENTO DE COMPLEJIDAD. 42
FIGURA 13 POR CIENTO DE REUTILIZACIÓN. 43
FIGURA 14 NIVELES DE HERENCIA POR CLASE. 44
FIGURA 15 NÚMERO DE DESCENDIENTES POR CLASE. 44
FIGURA 16 TÉCNICAS PARA REUTILIZACIÓN 45
FIGURA 17 TÉCNICAS PARA ABSTRACCIÓN. 45
FIGURA 18 TÉCNICAS PARA COHESIÓN 45
FIGURA 19 TÉCNICAS PARA CANTIDAD DE PRUEBAS. 45
FIGURA 20 DIAGRAMA DE COMPONENTES. 51
FIGURA 21 INTERFAZ PRINCIPAL 54
FIGURA 22 ADICIONAR ARRIBO EXTJS 55
FIGURA 23 CARGAR FICHERO EXTJS 57

Introducción

En la Aduana General de la República de Cuba se realiza el proceso de Despacho de los Medios de Transporte Internacional, tanto en su entrada como salida a las aduanas correspondientes. En este proceso se abarcan todas las acciones referentes al despacho de los buques como MTI que arriban y salen de las aduanas marítimas del país y a la prestación de servicios a los mismos, el encargado de realizar este proceso es el inspector de aduana.

En la actualidad se cuenta con problemas que impiden la realización del despacho de estos medios con calidad y confiabilidad, pues este proceso se lleva a cabo de forma manual, lo cual provoca que la búsqueda de información histórica, el procesamiento de los documentos y el control de las operaciones en las que intervienen los MTI sean más tediosos. Para dar solución a este problema un equipo de desarrollo del proyecto Gestión Integral de Aduanas (GINA) realizó un análisis profundo de las necesidades actuales de la Aduana General de la República para el despacho de los MTI, análisis que al ser validado se obtuvo como resultado los artefactos necesarios para alimentar el proceso de diseño, como fueron: especificación y descripción de requisitos funcionales, modelo de procesos.

A partir de esta situación problemática surge como **problema a resolver:** ¿Cómo materializar los requisitos explícitos en el análisis de los procesos de negocio involucrados en el despacho de Medios de Transporte Marítimos, siguiendo las pautas de arquitectura establecidas? Basado en esto, se define como **objeto de estudio:** Los sistemas informáticos para el control aduanero de los Medios de Transporte Marítimos y como **campo de acción:** Diseño e implementación del Módulo Despacho de Buques como Medio de Transporte Marítimo.

Para dar solución al problema se propone como **objetivo general**: Diseñar e Implementar el Módulo Despacho de los Medios de Transporte Marítimos para el sistema Gestión Integral de Aduanas siguiendo la arquitectura definida.

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

- Realizar un estudio sobre las tecnologías y herramientas a utilizar.
- Revisar la bibliográfica de la documentación relacionada con el Módulo Medios de Transporte Internacional.
- Revisar los requisitos funcionales.

- Diseñar e implementar el módulo para la gestión del proceso de despacho de buques.

Posibles resultados:

- Establecer las pautas a seguir en el diseño de la aplicación a implementar.
- Realizar Diagramas de Interacción entre clases.
- Realizar Diagrama de Secuencias para los diferentes escenarios.
- Realizar Modelo de Datos.
- Obtener el Código fuente de la aplicación.
- Realizar Diagrama de Despliegue.

El presente trabajo de diploma está conformado por tres capítulos:

Capítulo 1:

- Incluye un estado del arte del tema tratado a nivel internacional y nacional de los sistemas usados en la actualidad para dar solución al problema que se enfrenta.
- Se enuncian los principales conceptos relacionados con el sistema a diseñar.
- Se hace referencia a la metodología de desarrollo de software así como de las herramientas y el gestor de base de datos seleccionados para realizar este trabajo.

Capítulo 2:

- En este capítulo se realiza el diseño de la solución.
- Se muestran los diagramas de clases del diseño con estereotipos Web., así como los diagramas de secuencia.
- También se construye el modelo físico de datos y el diagrama de despliegue.
- Se especifica la solución propuesta en el marco de la arquitectura definida para el sistema Gestión Integral de Aduanas (GINA).

Capítulo 3:

- Se hace referencia al estándar de codificación utilizado.
- Se presenta el diagrama de componentes.
- Se procede a implementar la solución propuesta.
- Se realizan pruebas de caja negra para demostrar que las funciones del software son operativas.

Fecha	Tarea
1/12/2010 al 8/12/2010	1. Establecer las pautas a seguir en el diseño de la aplicación a implementar.
7/12/2010 al 10/12/2010	2. Realizar los diagramas de Interacción entre clases.
4/2/2011 al 11/2/2011	3. Realizar diagrama de secuencias para los diferentes escenarios.
11/2/2011 al 19/2/2011	4. Realizar Modelo de Datos.
20/2/2011 al 21/5/2011	5. Obtener Código fuente de la aplicación.
21/5/2011 al 1/6/2011	6. Realizar Diagrama de Despliegue.

Tabla 1 Cronograma de Tareas.

Capítulo 1: Fundamentación Teórica

1.1. Introducción

En el presente capítulo se describen los elementos principales que fundamentan el contenido de este trabajo. Se realizará un estudio acerca de la aplicación de sistemas informáticos en el mundo para el control aduanero, los mejores patrones que se siguen en el diseño de aplicaciones web y las mejores prácticas para realizar el diseño de la aplicación. También se detallarán aspectos importantes de la metodología, lenguaje y programas que serán utilizados para el desarrollo del producto.

1.2. Definiciones

1.2.1. MTI

Medio de Transporte Internacional (MTI), es uno de los subsistemas con que cuenta el sistema Gestión Integral de Aduanas (GINA), en el cual se realiza, específicamente, el control y despacho de los medios de transporte internacional, que son los buques, yates y las aeronaves.

1.2.2. GINA

La Aduana General de la República (AGR) trabaja en estos momentos en conjunto con la Universidad de las Ciencias Informáticas (UCI) en el sistema Gestión Integral de Aduanas (GINA) conformado por diferentes procesos. El GINA tiene como objetivo automatizar el procesamiento informativo referente a todas las operaciones que conforman los diferentes procesos, ya sea de Medios de Transporte Internacional, Importaciones y Exportaciones con y sin carácter comercial, Bultos Postales, 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.

1.2.3. Proceso de Despacho de los Medios de Transporte Internacional

El Despacho de los Medios de Transporte Internacional, es el conjunto de acciones mediante las cuales se tramita ante la Aduana, por el mando del buque, la aeronave o su consignatario, la documentación que declara la carga, provisiones, pasajeros y tripulantes que transportan los mismos.

1.2.4. Buque:

Toda embarcación que navegue, fondee, se mueva, atraque y desatraque en territorio aduanero, con independencia de su tipo, calado, tamaño o cualquier otra característica.(FIDEL CASTRO RUZ 1996)

1.3. Aplicaciones de sistemas informáticos en el mundo para el control aduanero

Actualmente en el mundo se requiere que las aduanas apliquen las Tecnologías de la Información y las Comunicaciones (TIC) a sus operaciones con el fin de lograr la informatización de las aduanas. Para ello se requiere el uso de sistemas que provean la transmisión electrónica previa de la información a las aduanas y el intercambio electrónico de información a efectos de exportación y de importación.

1.3.1. SOFIA

Es un sistema informático de despacho aduanero que interactúa en forma directa con sus usuarios: Despachantes de Aduana, Empresas de Transporte, Depositarios, Funcionarios de Aduana y con los Organismos vinculados al Comercio Exterior.

El Sistema SOFIA, implementado por la Dirección General de Aduanas del Paraguay tiene sus orígenes en Francia. Por el año 1976 se puso en marcha un Sistema Informático llamado Sistema de Computación del Flete Internacional Aéreo (SOFIA), que como su nombre lo indica, era una aplicación que cubría solamente las importaciones realizadas con flete aéreo.
Más tarde este sistema se perfecciona y pasa, de la importación con flete aéreo, a cubrir las importaciones con todo tipo de fletes. Este sistema se llamó SOFI (Sistema de Computación para el Flete Internacional).

El Sistema de Información se trata de la construcción de un conjunto operacional que toma como base y punto de partida el Sistema SOFI francés, que va incorporando nuevas funcionalidades para satisfacer las necesidades de gestión de la Aduana Paraguaya, conforme a la evolución impuesta por el comercio globalizado.(Dirección Nacional de Aduanas Paraguay 2010)

Objetivos:

- Simplificación y agilización de los procedimientos de des-aduanamiento de las mercaderías.
- Aplicación en forma rápida y uniforme de la legislación en vigencia.

- Mejoramiento de la eficacia administrativa y capacidad de gestión de la Dirección Nacional de Aduanas.
- Obtención de estadísticas de comercio exterior.
- Información para la lucha contra el fraude.

Módulos del Sistema

Declaración Sumaria

Este módulo tiene a su cargo:

- La gestión del transporte de mercaderías desde su ingreso hasta su salida.
- El registro del Manifiesto de Carga (consolidados y no consolidados).
- El ingreso de las Mercaderías en los Depósitos.
- El control de las diferencias detectadas en el mencionado ingreso (faltante o sobrante).
- El registro de todos los eventos realizados por los distintos usuarios (Agentes de Transporte, Depositarios, Despachantes y Funcionarios de Aduana).

Declaración Detallada

Este módulo tiene a su cargo:

- El registro y control de los datos declarados por el despachante para la formulación de sus despachos de importación o exportación. Estos datos se ingresan a nivel de carátula y por ítem[1]. Los últimos están determinados por partida arancelaria utilizada conforme a la Nomenclatura Común del Mercosur (NCM) que respeta el sistema armonizado de codificación y designación de mercancías.

- El control del acceso del despachante de Aduana, el cual debe estar debidamente inscripto en los registros de la Dirección Nacional de Aduanas.

- La interacción con el Módulo Arancel para la ejecución automática de las reglas referentes a: prohibiciones, restricciones, ventajas, etc. Conforme a las disposiciones legales vigentes a los efectos

[1] *Se utiliza para hacer distinción de artículos o capítulos en un escrito.*

de requerir al despachante de Aduana los distintos documentos que necesitará para la oficialización de su Declaración y determinar el monto exacto que deberá abonar por cada tasa o impuesto existente.

Control de Valor

Es una herramienta que permite el seguimiento del comportamiento de los valores declarados, determinando aquellas operaciones cuyos valores señalados se encuentran fuera de márgenes previamente definidos y en base a ello ejecutar alguna acción. Su dinámica consiste en la generación de una Base de Datos con valores de comparación asociados a la mercadería (Familias de Valor ó Familias de Mercaderías), la cual es consultada por el Sistema al momento de la validación del ítem o sub ítem para ejecutar las acciones definidas por cada caso.(Dirección Nacional de Aduanas Paraguay 2010)

Sistema Automatizado Aduanero Integral (SAAI)

Sistema automatizado que permite el control de las operaciones aduaneras en las cuarenta y ocho Aduanas de México desde 1993. Su control empieza con la autodeclaración electrónica de Pedimientos[2] por parte de los agentes y apoderados aduanales y continúa hasta los procesos de entrada y/o salida de mercancías. La información es concentrada en un sistema centralizado y se usa por diferentes entidades gubernamentales para generar las estadísticas nacionales de Comercio Exterior.(Administración General de Aduanas 2005)

Beneficios y funcionalidades del SAAI

- Utiliza un formato flexible denominado Pedimento.
- Fácil mantenimiento del sistema.
- Es actualizado en forma constante en atención a las nuevas disposiciones que regulan el comercio exterior, así como respecto de nuevas funcionalidades.
- Ventanilla única para el despacho aduanero.
- Simplificación de trámites.
- Eliminación del papel (digitalización).
- Opera con base en la normatividad establecida.

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

- Permite transmitir ágil y eficientemente las operaciones de comercio exterior, de manera electrónica al ciento por ciento.
- El ciento por ciento de las operaciones son validadas por el SAAI.
- No hay procedimientos de captura manual.
- Utiliza tecnología de punta.
- Incorpora estándares internacionales.

1.4. Diseño de sistemas de software

Historia

El contexto en el que se ha desarrollado el software está fuertemente ligado a la evolución de los sistemas informáticos. Un mejor rendimiento del hardware, una reducción del tamaño y un costo más bajo, han dado lugar a sistemas informáticos más sofisticados. La evolución del software dentro del contexto de las áreas de aplicación de los sistemas basados en computadoras, puede verse de la siguiente manera:

Los primeros años	La segunda era	La tercera era	La cuarta era
1950 - 1965	1965 - 1975	1975 - 1985	1985 - 2011
• Orientación por lotes • Distribución limitada • Software "a medida"	• Multiusuario • Tiempo real • Bases de Datos • Software como producto	• Sistemas distribuidos • Incorporación de "inteligencia" • Hardware de bajo costo • Impacto en el consumo	• Potentes sistemas de escritorio • Tecnología orientada a objetos • Sistemas expertos • Redes neuronales artificiales • Computación paralela

Tabla 2. Etapas del desarrollo de software.

En la **cuarta era** las tecnologías orientadas a objetos están comenzando a ser utilizadas en muchas áreas. Las técnicas de cuarta generación para el desarrollo de software están cambiando la forma en que algunos segmentos de la comunidad informática construyen los programas. Los sistemas expertos y el software de inteligencia artificial se han trasladado del laboratorio a las aplicaciones prácticas. El software de redes neuronales artificiales ha abierto excitantes posibilidades para el reconocimiento de formas y habilidades de procesamiento de información al estilo de como lo hacen los humanos.

Conforme se transita por la cuarta era, continúan intensificándose los problemas asociados con el software de computadoras:

- La sofisticación del hardware ha dejado desfasada la capacidad de construir software que pueda explotar el potencial del hardware.
- La capacidad de construir nuevos programas no puede dar abasto a la demanda de nuevos programas.
- La capacidad de mantener los programas existentes está amenazada por el mal diseño y el uso de recursos inadecuados.

En el desarrollo de las aplicaciones Web uno de los modelos de arquitecturas más usadas, es el modelo de arquitectura en capas, el cual se ha convertido actualmente en el modelo por excelencia en el desarrollo de software para la gestión empresarial. Cada capa de este modelo compone un subsistema en el que se ubican clases con responsabilidades propias, donde sus objetivos son facilitar la reutilización de código y componentes gracias a la herencia y encapsulamiento que brindan los lenguajes de POO. Esto brinda grandes beneficios a las aplicaciones que utilizan esta arquitectura, como los siguientes:

- Aislamiento de la lógica de aplicaciones en componentes independientes, susceptibles de ser reutilizados después en otros sistemas.

- Asignación de recursos a cada una de las capas por separado, brindando la posibilidad de desarrollarlas en paralelo.

- Independencia del acceso a los datos en relación con la lógica del negocio.

La principal ventaja que tiene esta arquitectura es que puede cambiarse la capa de presentación sin que cambie la capa del negocio. Además el acceso a datos también puede ser trasladado a una base de datos distinta a otro modelo de base de datos, o a otro servidor sin influir en la capa de negocio. La propia capa de negocio también puede a su vez ser reemplazada sin influir estos cambios significativamente en las capas de datos y presentación. Dentro de esta arquitectura en capas, la capa de acceso a datos es fundamental ya que es la encargada de hacer persistir la información que se maneja en el negocio y realizar todas las operaciones demandadas con ella.

La evolución del diseño de software, como parte del proceso de desarrollo de software, es un proceso continuo que se ha ido produciendo durante las últimas tres décadas. Los primeros trabajos sobre diseño se centraron sobre los criterios para el desarrollo de programas modulares y los métodos para mejorar la arquitectura del software de una manera descendente. Los aspectos procedimentales de la definición del diseño evolucionaron hacia una filosofía denominada programación estructurada. Posteriores trabajos propusieron métodos para la traducción del flujo de datos o de la estructura de los datos, en una definición de diseño. Nuevos enfoques para el diseño proponen un método orientado a objetos para la obtención del diseño.

Cada metodología de diseño de software introduce heurísticas y notaciones propias, así como una visión algo particular de lo que caracteriza a la calidad del diseño. Sin embargo, todas las metodologías tienen varias características comunes:

- Un mecanismo para la traducción de la representación del campo de información en una representación de diseño.
- Una notación para representar los componentes funcionales y sus interfaces.
- Heurísticas para el refinamiento y la partición.
- Criterios para la valoración de la calidad.

Objetivos del Diseño

El objetivo más importante es:

- Entregar las funciones requeridas por el usuario (satisfaga una especificación funcional dada).

Pero además para lograr esto deben considerarse los aspectos:

- Rendimiento: cuán rápido permitirá el diseño realizar el trabajo dado un recurso particular de hardware. Es decir que contemple las limitaciones del medio donde será implementado el sistema, y alcance los requerimientos de performance y uso de recursos.

- Control: protección contra errores humanos, máquinas defectuosas, o daños intencionales.

- Confiabilidad: facilidad con la cual el diseño permite modificar el sistema.

Además deberá:

- Satisfacer criterios de diseño sobre la forma interna y externa del producto obtenido.
- Satisfacer restricciones sobre el proceso de diseño en sí mismo tales como el tiempo, costo y las herramientas disponibles para hacer el diseño.

Una vez establecidos los requisitos del sistema, el diseño es la primera de tres actividades técnicas (diseño, codificación y prueba). Cada actividad transforma la información de forma que finalmente se obtiene un software para computadora validado.

1.5. Patrones que se utilizan en el diseño de aplicaciones web

Breve historia de los patrones de diseño

Un patrón de diseño es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo, desde el análisis hasta el diseño y desde la arquitectura hasta la implementación.

Los diseñadores de software extendieron la idea de patrones de diseño al proceso de desarrollo de software. Debido a las características que proporcionaron los lenguajes orientados a objetos (como herencia, abstracción y encapsulamiento) les permitieron relacionar entidades de los lenguajes de programación a entidades del mundo real fácilmente, los diseñadores empezaron a aplicar esas características para crear soluciones comunes y reutilizables para problemas frecuentes que exhibían patrones similares.

Fue por los años 1994, que apareció el libro "Design Patterns: Elements of Reusable Object Oriented Software" escrito por los ahora famosos Gang of Four (GoF, que en español es la pandilla de los cuatro) formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Ellos recopilaron y documentaron 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos. Desde luego que ellos no son los inventores ni los únicos involucrados, pero ese fue luego de la publicación de ese libro que empezó a difundirse con más fuerza la idea de patrones de diseño.

Muchos diseñadores y arquitectos de software han definido el término de patrón de diseño de varias formas que corresponden al ámbito a la cual se aplican los patrones. Luego, se dividió los patrones en diferentes categorías de acuerdo a su uso. (1999)

GRASP

Los patrones GRASP acrónimo de General Responsibility Assignment Software Patterns (patrones generales de software para asignar responsabilidades), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresadas en forma de patrones. Craig Larman en su libro "UML y Patrones" define: "Los patrones GRASP constituyen un apoyo para la enseñanza que ayuda a uno a entender el diseño de objetos esencial, y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable."(1999)

GOF

Los patrones GOF (acrónimo de Gang of Four en español banda de los cuatro), se dividen en tres grupos fundamentales, de creación, estructura y comportamiento. Los primeros se encargan de mostrar una guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones. Estas decisiones serán normalmente resueltas dinámicamente, decidiendo que clases instanciar o sobre que objetos se delegará responsabilidades. Los de estructura se encargan de describir la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Y los de comportamiento se utilizan para manejar, organizar y combinar comportamientos.

El grupo de GoF clasificaron los patrones en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

- **Creacionales**: Patrones creacionales tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
- **Estructurales**: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso simples u compuestos.
- **Comportamiento**: Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

En el segundo nivel, ellos clasificaron los patrones en 2 ámbitos: Clases y objetos. Es así que, se tienen 6 tipos de patrones:

Creacionales

- **Creacional de la Clase**
  Los patrones creacionales de Clases usan la herencia como un mecanismo para lograr la instanciación de la Clase. Por ejemplo el método Factoría.
- **Creacional del objeto**
  Los patrones creacionales de objetos son más escalables y dinámicos comparados con los creacionales de Clases. Por ejemplo la Factoría abstracta y el patrón Singleton.

Estructurales

- **Estructural de la Clase**
  Los patrones estructurales de Clases usan la herencia para proporcionar interfaces más útiles combinando la funcionalidad de múltiples Clases. Por ejemplo el patrón Adaptador (Clase).
- **Estructural de Objetos**
  Los patrones estructurales de objetos crean objetos complejos agregando objetos individuales para construir grandes estructuras. La composición del patrón estructural del objeto puede ser cambiado en tiempo de ejecución, el cual da flexibilidad adicional sobre los patrones estructurales de Clases. Por ejemplo el Adaptador (Objeto), Facade, Bridge, Composite.

Comportamiento

- **Comportamiento de Clase**
  Los patrones de comportamiento de Clases usan la herencia para distribuir el comportamiento entre Clases. Por ejemplo Interpreter.
- **Comportamiento de Objeto**
  Los patrones de comportamiento de objetos permite analizar los patrones de comunicación entre objetos interconectados, como objetos incluidos en un objeto complejo. Ejemplo Iterator, Observer, Visitor.

Por qué son útiles los patrones

- Contribuyen a reutilizar diseño, identificando aspectos claves de la estructura de un diseño que puede ser aplicado en una gran cantidad de situaciones. La importancia de la reutilización del diseño no es despreciable, ya que ésta provee de numerosas ventajas: reduce los esfuerzos de desarrollo y mantenimiento, mejora la seguridad, eficiencia y consistencia del diseño, y proporciona un considerable ahorro en la inversión.
- Mejoran (aumentan, elevan) la flexibilidad, modularidad y extensibilidad, factores internos e íntimamente relacionados con la calidad percibida por el usuario.
- Incrementan el vocabulario de diseño, ayudando a diseñar desde un mayor nivel de abstracción. (1994)

1.6. Buenas prácticas en la programación Web

Las buenas prácticas de programación sirven para mejorar la comprensión y claridad del código, además permite unificar criterios entre los distintos programadores de la comunidad. Estas parten de la base de que el código es la mejor documentación que puede tener un sistema, por esto también es la mejor herramienta que tiene un programador de comunicar su trabajo con el resto de los programadores. A la hora de desarrollar un software los programadores experimentados y los que no lo son tanto siguen numerosas prácticas de programación 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, la configurabilidad y la seguridad del sistema.

Algunas 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.
- Manejar datos con precaución
- Prácticas para el manejo de datos
- Realizar mantenimiento del código

Prácticas de mantenimiento del código:

- Utilizar las normas.
- Eliminar el código obsoleto.
- Análisis de todos los cambios en el código.

Al utilizar las buenas prácticas de programación el código obtiene valor agregado ya que adquiere:

- Fácil integración y reutilización.
- Fácil comprensión por parte del programador.
- Unificación de criterios.
- Eliminación de zonas oscuras de código.
- Fácil comunicación entre programadores.
- Claridad y correctitud en el código.
- Incremento significable en la mantención del Software.

Todas estas prácticas serán tomadas en cuenta para la realización de la aplicación y estarán vinculadas con las necesidades del problema a resolver y la elección de la metodología a seguir.(Mark G. Graff, Kenneth R. van Wyk. 2003)

1.7. Metodología de desarrollo

La metodología de desarrollo de software está compuesta por procedimientos, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. Para el desarrollo de la solución se utiliza como metodología un modelo de desarrollo definido por el Centro de Informatización para la Gestión de Entidades (CEIGE) tomando en consideración las adecuaciones realizadas en el Departamento de Desarrollo de Soluciones Aduaneras, específicamente para la fase de diseño. Con la aplicación dicho modelo se generarán un conjunto de artefactos para el diseño de la solución, los cuales serían: Diagrama de Clase, Diagrama de Modelo de Datos, Diagrama de Secuencia con un

grupo de adecuaciones definidas en el departamento, Diagrama de Componente y Diagrama de Despliegue.

1.8. Lenguaje de programación

Como lenguaje de programación se utiliza el PHP en su versión 5.2. Conocido como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones Web dirigidas a bases de datos. Este es un lenguaje de programación que se interpreta y ejecuta directamente en el servidor en el que está albergada la página Web, con lo que el visitante a la misma, únicamente recibe el resultado buscado por el código en el que está escrito. Es orientado a objetos y está ampliamente difundido en todo el mundo, presentando grandes volúmenes de documentación que son de gran ayuda para los desarrolladores.

1.9. Framework de trabajo

Symfony 1.2.8

Es un framework de PHP totalmente libre y de los más usados en la actualidad. Está diseñado para optimizar el desarrollo de aplicaciones Web. Entre sus características se presenta que separa la lógica del negocio, la lógica de servidor y la presentación de la aplicación. Facilita herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. Además presenta abstracción a la base de datos, esto tiene como ventaja que se puede cambiar de gestor de base de datos realizando pequeños cambios en el archivo de configuración.(Fabien Potencier, 2008) Es compatible con la mayoría de gestores de bases de datos, MySQL, PostgreSQL, Oracle y SQL Server de Microsoft entre otros.

Este Framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además proporciona una estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

¿Por qué Symfony?

En estos momentos la meta fundamental a alcanzar por el Polo de Sistemas Tributarios y de Aduanas es la de poder realizar la mayor cantidad de aplicaciones en el menor tiempo posible, de manera

profesional y competitiva, dando respuesta a las necesidades de los clientes, garantizando su satisfacción y al mismo tiempo manteniendo la calidad, usabilidad y eficiencia de los productos. Realizar esto en aplicaciones implementadas desde cero es algo prácticamente imposible, por lo que es inminente la utilización de un Framework de entre los que existen en el ámbito de la programación en PHP. (José Antonio Cobo Rodríguez 2008)

1.10. Interfaz de Usuario

La interfaz de usuario se desarrolla con las librerías ExtJS[3] 3.0 para el lenguaje Java Script, estas permiten el desarrollo de aplicaciones Web interactivas usando tecnologías como AJAX, JSON, DHTML. ExtJS proporciona una serie de componentes para incluir dentro de una aplicación Web que facilitan mucho el trabajo de los desarrolladores y hacen más agradable la aplicación para los usuarios, algunos de estos componentes son: los cuadros, las áreas de texto, campos para fechas, campos numéricos, combos, botón de radio, las casillas de verificación, editores de HTML, árbol de datos, pestañas, menú al estilo de Windows, entre otras. Tiene como ventaja además que una vez que el navegador haya interpretado el código Java Script, el usuario se sentirá trabajando en el sistema como si fuera una aplicación de escritorio. Esto está dado porque la transacción de información se realiza a partir de comunicaciones asincrónicas.(Colin Ramsay 2008)

1.11. Servidor Web

El servidor Web es un programa que corre sobre el servidor que escucha las peticiones HTTP que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor Web buscará una página Web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición. Va a ser fundamental en el desarrollo de las aplicaciones del lado del servidor que vayamos a construir, ya que se ejecutarán en él.(2010a)

1.11.1. Apache 2.2.x

Servidor Web que más se utiliza en el mundo, lo que demuestra que es una solución dominante y ampliamente probada, funciona sobre cualquier plataforma, permite que otros ordenadores vean la Web mediante un navegador. Es una solución altamente configurable y extensible a través de

[3] *ExtJS es una librería de Java Script para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM.*

módulos, se integra perfectamente con varias tecnologías, lenguajes, plataformas, bases de datos, etc. Es considerado el servidor Web con la mejor funcionalidad/velocidad. Tiene capacidad para servir páginas tanto de contenido estático, como de contenido dinámico a través de otras herramientas soportadas que facilitan la actualización de los contenidos mediante bases de datos, ficheros u otras fuentes de información. La versión 2.2.x, incorpora grandes novedades y mejoras, combina las herramientas libres Apache, PHP, logrando centrar la atención en millones de sitios Web dinámicos. Es muy sencillo ampliar las capacidades de este servidor pues actualmente existen muchos módulos que son adaptables a este. Permite personalizar la respuesta ante los posibles errores que se puedan presentar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto. Tiene una alta configurabilidad en la creación y gestión de logs. Permite la creación de ficheros de log a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en tu servidor. Esta característica unida a su, robustez y estabilidad hacen que millones de servidores reiteren su confianza en este programa.(Adriana Alfonso Luis 2010)

El servidor web Apache es un servidor web desarrollado por el Apache Server Project (Proyecto Servidor Apache) cuyo objetivo es la creación de un servidor web fiable, eficiente y fácilmente extensible con código fuente abierto gratuita. 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.
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, Linux y Windows.
- Es robusto y seguro.
- Es una tecnología gratuita de código fuente abierta.
- 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.(2010b)

1.12. Sistema gestor de base de datos

Oracle 11g es un sistema gestor de base de datos relacional desarrollado por la empresa “Oracle Corporation”, es considerado de los más completos y potentes, destacado por su soporte de transacciones, estabilidad, escalabilidad y ser multiplataforma. La aduana cubana ha adquirido experiencias positivas con Oracle usándolo desde el año 1997. Es un manejador de base de datos relacional que hace uso de los recursos del sistema informático en todas las arquitecturas de hardware, para garantizar su aprovechamiento al máximo en ambientes cargados de información. Es el conjunto de datos que proporciona la capacidad de almacenar y acudir a estos de forma recurrente con un modelo definido como relacional. Además es una suite de productos que ofrece una gran variedad de herramientas. Oracle es el mayor y más usado Sistema Manejador de Base de Dato Relacional (RDBMS) en el mundo.(2009a)

1.13. Herramientas de desarrollo

1.13.1. Herramienta CASE[4]

Como herramienta CASE se utiliza el Visual Paradigm for UML 6.4, resulta muy provechosa su utilización pues soporta UML 2.1 además de BPMN y generación de código para PHP. El trabajo con UML permite realizar 13 diagramas de los cuales se pueden mencionar el diagrama de casos de uso, el diagrama de clases, diagrama de secuencias y el de despliegue. Presenta además la posibilidad de generar la documentación del trabajo realizado, la administración de los requerimientos y la creación de esquemas a partir de una base de datos ya existente y viceversa.(Rafael Andrés Céspedes Basteiro 2009) Soporta el ciclo de vida completo del desarrollo de software, análisis y diseño orientados a objetos, construcción, pruebas y despliegue. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. A continuación se listan sus características:

- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio.
- Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse) a través de XMI (nueva característica).

[4] *Siglas en inglés que se utilizan para referirse a Ingeniería de Software Asistida por Computadora.*

- Ingeniería de ida y vuelta. Ingeniería inversa - Código a modelo, código a diagrama. Ingeniería inversa Java, C++, Esquemas XML, XML, .NET exe/dll, CORBA IDL.
- Generación de código - Modelo a código, diagrama a código.
- Diagramas de flujo de datos.
- Soporte para ORM y generación de objetos Java desde la base de datos.
- Generación de bases de datos y transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos, desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas, reorganización de las figuras y conectores de los diagramas UML. (Nallelys Rodríguez Santos 2010)

1.14. Entorno de Desarrollo Integrado (IDE)

1.14.1. NetBeans IDE 6.9

NetBeans es un IDE desarrollado por Sun Microsystems, de código abierto y multiplataforma. Permite diseñar aplicaciones de forma fácil con solo arrastrar objetos a la interfaz de un formulario. Es una plataforma pensada para escribir, compilar, depurar y ejecutar programas. NetBeans no solo permite el desarrollo de aplicaciones de escritorio, también permite el desarrollo de aplicaciones para la web y para dispositivos portátiles. La programación en este IDE se realiza a través de componentes modulares o módulos. Las aplicaciones construidas a partir de módulos pueden ser extendidas ya que estos permiten ser desarrollados independientemente por otros desarrolladores de software, de ahí que sea una aplicación flexible/extensible.

Entre las características de la plataforma están la administración de las interfaces de usuario, administración de las configuraciones del usuario, administración del almacenamiento, administración de ventanas, framework basado en asistentes, entre otras cosas.(Pedro Abigantús Pérez 2010)

¿Por qué NetBeans como IDE?

Eclipse es un entorno de desarrollo integrado que, aunque pretende ser un entorno versátil tolerando varios lenguajes de programación, con el que mejor se integra es con el lenguaje Java.
NetBeans es un entorno de desarrollo que tiene una interfaz amigable y fácil de comprender aún cuando los usuarios son inexpertos. Posee herramientas para crear aplicaciones profesionales ya sean de escritorio, empresariales, web con PHP 5 y móviles, no solo en Java sino también en C/C++ y Ruby. Incluye soporte para el trabajo con Symfony. Provee soporte para el trabajo con XML, AJAX y modelado empleando UML. Por las características que presenta NetBeans IDE y por ser además una plataforma flexible es que se determinó el uso de este IDE en el desarrollo de este sistema.

1.15. Mozilla Firefox versión 3.5

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

- Excelente navegación por pestañas.
- Marcadores dinámicos.
- Corrector ortográfico in time.
- Gestor de descargas integrado.
- Manejador de gran cantidad de extensiones.
- Herramientas de desarrollo web integradas.
- Bloqueo de pestañas emergentes.
- Personalización de temas.

Las extensiones de Firefox son pequeños programas que se le añaden y que expande sus capacidades, funcionalidad que casi ningún otro navegador posee. Su uso posibilita a Firefox debuguear(2009b) páginas web, manejar imágenes, música, videos y multimedia, entre otros. Firefox está construido sobre el motor de dibujado Mozilla Gecko, uno de los más rápidos disponibles hoy en día. Cumple perfectamente los estándares del W3C (del inglés, World Wide Web Consortium) tales

como: CSS (del inglés, Cascading Style Sheets), DOM (del inglés, Document Object Model), entre otros. Mientras que navegadores como Opera probablemente fallen en una página dinámica, Mozilla Firefox generalmente tiene éxito. Actualmente existen numerosos lenguajes de programación web. Los del lado del cliente que indican al navegador donde colocar cada texto, imagen o video y la forma que tendrán estos al ser colocados en la página; y los del lado del servidor ejecutados en el servidor web justo antes de que se envíe la página a través de Internet al cliente, permitiendo que se muestre en la misma los datos deseados por el servidor.(2009a)

1.16. Conclusiones Parciales

Como resultado del estudio realizado a lo largo del capítulo han sido expuestos los principales puntos de interés abordados en la investigación, demostrando la necesidad de implementar un sistema que se adecue a las necesidades y legislaciones de las aduanas cubanas. Además se realizó un estudio de los sistemas extranjeros que trabajan con un objetivo similar, de los cuales se tomaron ideas que se implantarán en la solución propuesta, de forma tal que el producto cuente con opciones similares a las de sus competidores en el mundo. También se analizaron los patrones de diseño a utilizar para el desarrollo del producto, las mejores prácticas de programación y se escogió la metodología de desarrollo de software establecida por el centro. Como herramienta de modelado se utilizará el Visual Paradigm for UML 6.4, la implementación se realizará mediante los leguajes de programación web PHP 5.2 y Java Script, el servidor web será Apache 2.2.17, el sistema gestor de base de datos Oracle 11g, el entorno de desarrollo NetBeans IDE 6.9 y el framework para facilitar el desarrollo Symfony 1.2.8.

Capítulo 2: Diseño del Sistema

2.1. Introducción

En el presente capítulo se describe la solución propuesta y el diseño del módulo Despacho de Buques como Medios de Transporte Marítimos (MTI) para el Sistema de Gestión Integral de la Aduana comenzando por una descripción de los procesos a automatizar y los problemas existentes que hacen necesario la informatización de estos. 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, los diagramas de secuencia y se presenta el diseño de la base de datos del sistema. Se utilizan métricas que ayudan a la detección de errores y a la mejora de la calidad en el diseño en fases tempranas o sea antes que dicho diseño se haya implementado en código.

2.2. Requisitos funcionales

Requisito Funcional 1

- El sistema permitirá confirmar el Arribo/Salida de los buques en un rango de fechas seleccionado.

Requisito Funcional 2

- El sistema permitirá gestionar el despacho de buques al inspector. Permitiendo registrar la entrada, registrar la salida, modificar y cancelar el despacho de buques.

Requisito Funcional 3

- El sistema deberá otorgar el número de manifiesto a un buque automáticamente consultando el último número de manifiesto otorgado a ese tipo de Medio de Transporte Internacional (MTI) y luego le asigna un número consecutivo al que se asignó anteriormente.

Requisito Funcional 4

- El sistema permitirá poner al cobro los servicios de buques.

Requisito Funcional 5

- El sistema deberá recepcionar documentos electrónicos automáticamente.

Requisito Funcional 6

- El sistema permitirá la modificación de la lista de tripulantes y en consecuencia la acción de enrolar o desenrolar a un tripulante.

Requisito Funcional 7

- El sistema permitirá registrar notificación de infracción.

2.3. Validación de los requisitos de software

La validación de requisitos tiene como misión demostrar que estos definen realmente el sistema que el usuario necesita. Además tiene el propósito de conocer errores y prevenirlos a tiempo antes de comprometer recursos en las fases futuras del desarrollo de software. Existen un conjunto de técnicas que se utilizan para llevar a cabo la validación de los requisitos.(Rafael Andrés Céspedes Basteiro 2009)

- Revisiones: Esta técnica consiste en la lectura y corrección de la documentación o modelado de la definición de requisitos para validar la correcta interpretación de la información transmitida.
- Prototipos: La técnica de prototipos es una ayuda en todas las etapas del ciclo de desarrollo, porque permite aclarar un problema o visualizar una solución.

Algunas propuestas sobre esta técnica señalan que se basa en obtener de la definición de requisitos, prototipos, que a pesar de no tener la totalidad de las funcionalidades del sistema, permitan al usuario tener una idea sobre la interfaz visual del sistema y su interoperabilidad. Con anterioridad se desarrolló la validación de los requisitos del módulo Despacho de los Medios de Transporte Marítimos, donde se revisaron por parte de la dirección del proyecto y fueron aceptados por los clientes, obteniendo gran aceptación.

2.4. Patrones de diseños utilizados

2.4.1. Patrones GRASP

Creador

La clase actions.class.php contiene las acciones definidas para el módulo Medios de Transporte Internacional (Buques) 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. El patrón Creador ayuda a identificar quien debe ser el responsable de la instanciación o creación de nuevas clases de objetos.

Controlador

Todas las peticiones Web son manejadas por un solo controlador frontal (ejemplo: 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.(2010b)

Experto

Este patrón plantea asignar una responsabilidad al experto en información, es decir, la clase que contiene toda la información necesaria para realizar la labor que tiene encomendada.

Beneficios:
La encapsulación es mantenida, desde que los objetos usan sus propias informaciones para realizar tareas. Esto permite poco acoplamiento, lo cual conduce a sistemas más robustos y de mantenimiento mucho más fácil.

Bajo Acoplamiento

Plantea que debe haber pocas dependencias entre las clases. Se utiliza para mantener las clases lo menos relacionadas posibles para en caso de que ocurran cambios, estos repercutan lo menos posible en las otras. Uno de los principales síntomas de un mal diseño y alto acoplamiento es una herencia muy profunda.

Alta Cohesión

Symfony permite la asignación de responsabilidades con alta cohesión, por ejemplo la clase actions.class.php tiene la responsabilidad de 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.(2008)

2.4.2. Patrones GOF

Solitario

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

Decorador

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

Fábrica abstracta

Se utiliza este patrón al trabajar con objetos de distintas familias de manera que no se mezclen entre sí, haciendo transparente el tipo de familia concreta que se esté usando. Cuando el framework necesita, por ejemplo crear un nuevo objeto, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea. (Grupo UML 2010)

2.5. Patrón de Arquitectura

2.5.1. Modelo Vista Controlador (MVC)

El patrón MVC es un patrón arquitectónico que ayuda a darle cierta estructura lógica a las aplicaciones. Su principal objetivo es separar la lógica del negocio de la lógica de presentación ó interfaz.

Para realizar las más sencillas de las páginas en Symfony se necesitan al menos siete scripts diferentes:

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

Por lo que aparecen muchos archivos para abrir y modificar cada vez que se crea una página. Pero Symfony simplifica este proceso. Toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo.(José Antonio Cobo Rodríguez 2008)

El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador.

- **Modelo**: Es la representación de la información que maneja la aplicación. El modelo en sí son los datos puros que puestos en contexto del sistema proveen de información al usuario y a la aplicación.
- **Vista**: Es la representación del modelo en forma gráfica, disponible para la interacción con el usuario. En el caso de una aplicación Web, la "Vista" es una página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.
- **Controlador**: Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el modelo en caso de ser necesario.

Figura 1 Capas del MVC

MVC proviene de Model View Controller (Modelo Vista Controlador), y como bien indica su nombre en este patrón se identifican tres componentes fundamentales que se relacionan entre sí.

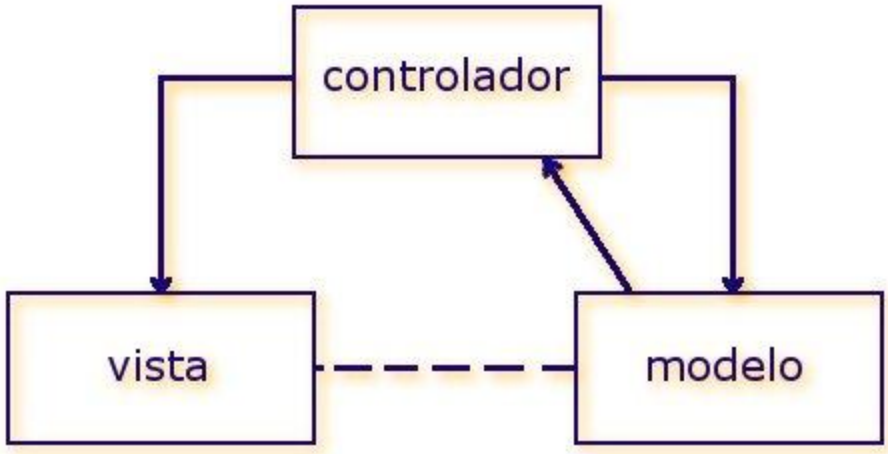


Figura 2 Componentes del MVC

En el patrón MVC el controlador recibe los eventos de la interfaz, se encarga de llamar en el modelo al experto del negocio que sabe que es lo que hay que hacer con la petición del usuario. Una vez que el modelo ha realizado su tarea se lo comunica al controlador pudiendo además transferirle algunos datos a este. El controlador entonces le dice a la vista o interfaz que se actualice con los cambios hechos en el modelo. Nótese que en la imagen anterior existe una línea discontinua entre el modelo y la vista. Esto se debe a que en algunas implementaciones del patrón MVC cuando ocurre un cambio

en el modelo él mismo envía una notificación a la vista para que se actualice. Pero esta variante no es muy frecuente. (Debug_mode_on 2010)

Ventajas y desventajas del MVC

Entre las principales ventajas se encuentran:

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

Como desventajas:

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

2.6. Diagramas de clase del diseño

2.6.1. Requisito Funcional: Confirmar Arribo/Salida

En el requisito funcional Confirmar Arribo/Salida se tiene una relación de agregación entre la página cliente “CP_Confirmar_Arribo/Salida” y el formulario “form_Confirmar_Arribo/Salida”, que contiene los componentes necesarios para la solución. Luego los datos introducidos son enviados al servidor y recibidos por las funciones de la clase “MTI_Actions.class.php”, las cuales mediante las clases del paquete de Acceso a Datos son capaces de registrar y obtener información.

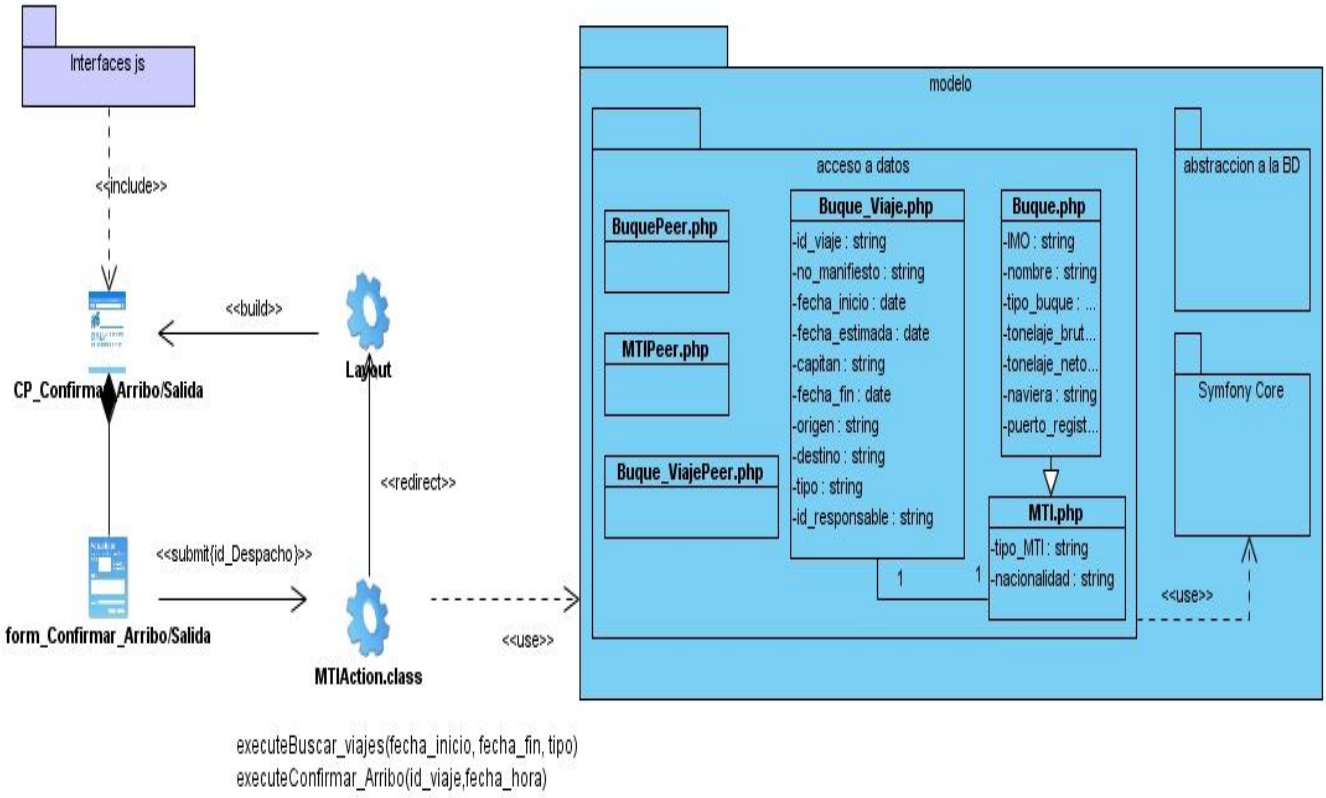


Figura 3 Requisito: Confirmar Arribo/Salida.

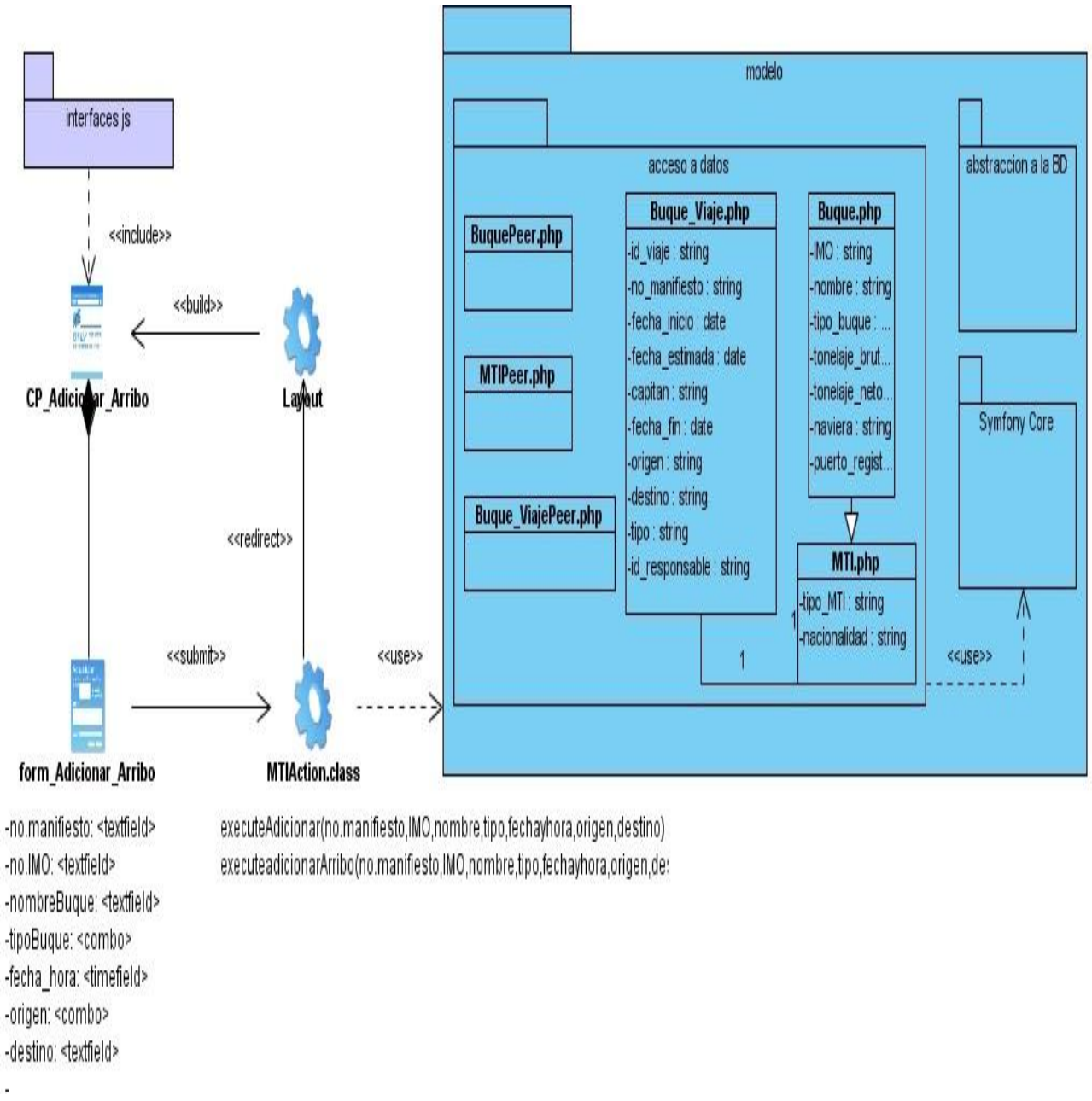


Figura 4 Requisito: Adicionar Arribo.

2.6.2. Extensiones para el diseño Web

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

Estereotipos para las clases		
Estereotipos	**Imagen**	**Descripción**
Página Servidora		Representa una página Web dinámica que contiene el código ensamblado por el servidor cada vez que se solicita. Típicamente, una página servidora contiene scripts que se ejecutan en el servidor y que actúan recíprocamente con los recursos del lado del servidor estos son: las bases de datos, los componentes de la lógica del negocio, sistemas externos, y así sucesivamente.
Página Cliente		Representa una instancia de una página cliente. Es una página Web con formato HTML. Las páginas cliente son interpretadas y mostradas por los navegadores del cliente y además pueden contener scripts que se interpretan en el navegador.
Formulario		Representa un formulario (elemento encargado de realizar envíos a las páginas servidoras)

Tabla 3 Estereotipos para las clases.

Estereotipos para las relaciones entre clases	
<<link>>	Representa un apuntador desde una página cliente hacia una página cliente o página servidora. Corresponde directamente con una etiqueta <a> (ancla) de HTML.
<<submit>>	Esta relación siempre se da entre un "formulario" y una "página servidora", por supuesto, la "página servidora" procesa los datos que el "formulario" le envía.
<<build>>	Sirve para identificar cual(es) "página(s) servidora(s)" son responsables de la creación de una "página cliente". Una "página servidora" puede crear varias "páginas cliente", pero una "página cliente" sólo puede ser creada por una sola "página servidora". Esta relación siempre es unidireccional.
<<redirect>>	Esta es también una relación unidireccional que indica que una página Web redirige hacia otra.
<<include>>	Asociación direccional desde una "página servidora" a otra "página servidora" o "página cliente". Esta asociación indica que las páginas incluidas son procesadas mientras que la página se ensambla.

Tabla 4 Estereotipos para las relaciones entre clases.

2.6.3. Clases del diseño con estereotipos web

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

Tabla 5 Relaciones que se establecen entre las clases que conforman la extensión UML para Web.

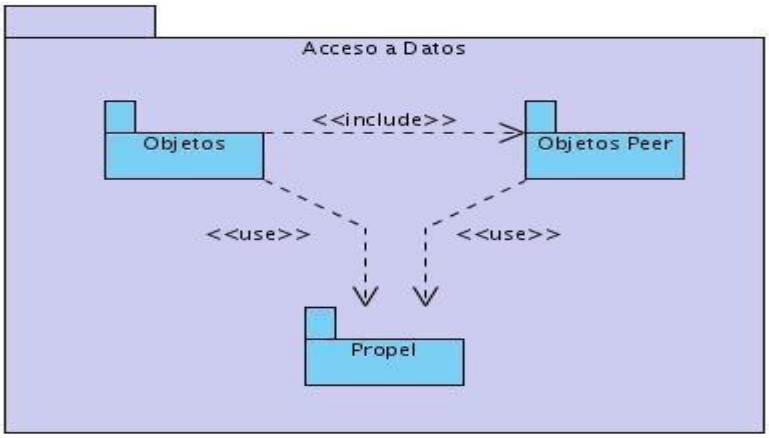


Figura 5 Paquete de acceso a datos.

El paquete “Acceso a Datos” contiene tres paquetes en su interior Objetos, Objetos Peer y Propel. Ver Figura 5. 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.

2.7. Diagramas de secuencia

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

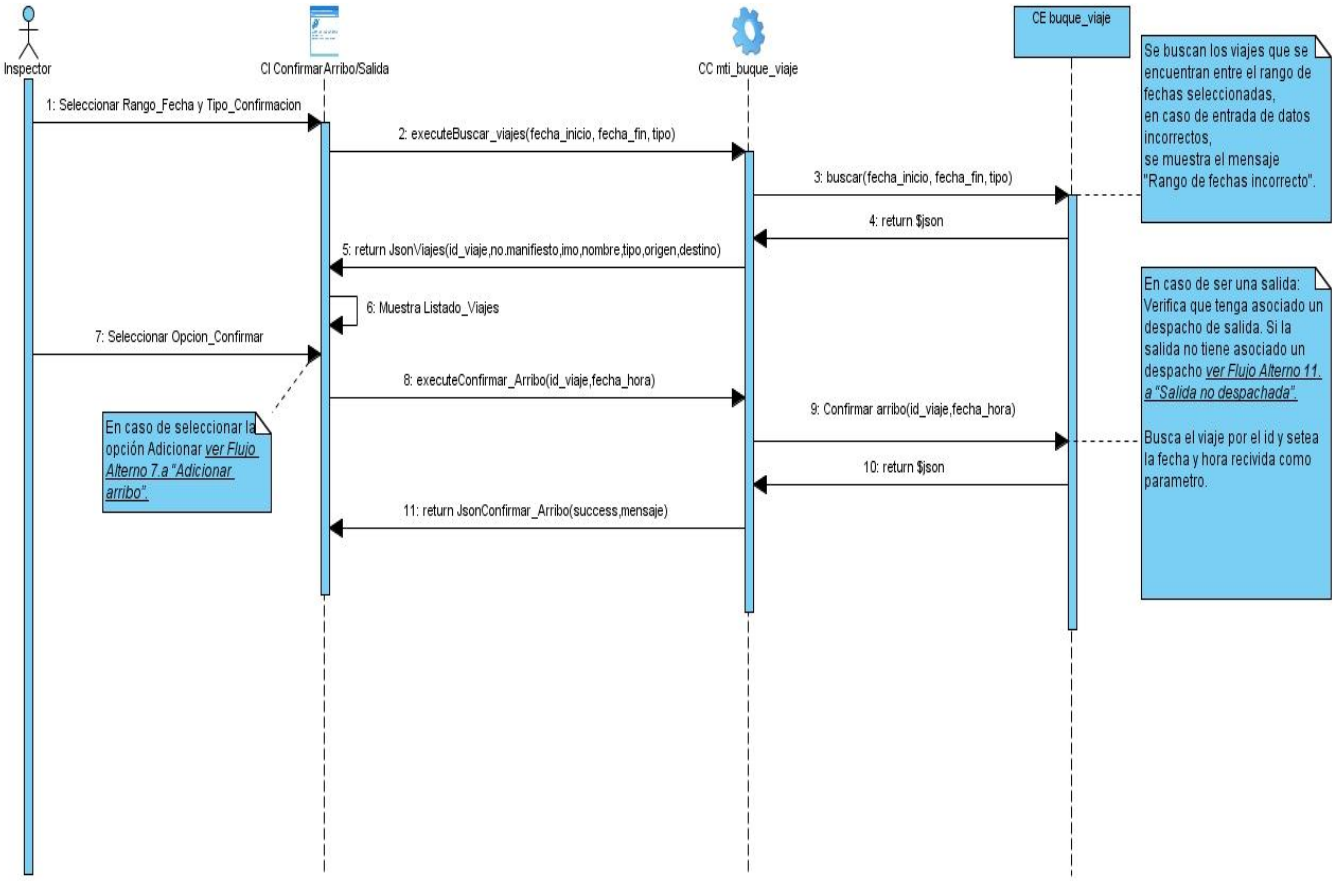


Figura 6 Diagrama de secuencia confirmar arribo/salida de buques.

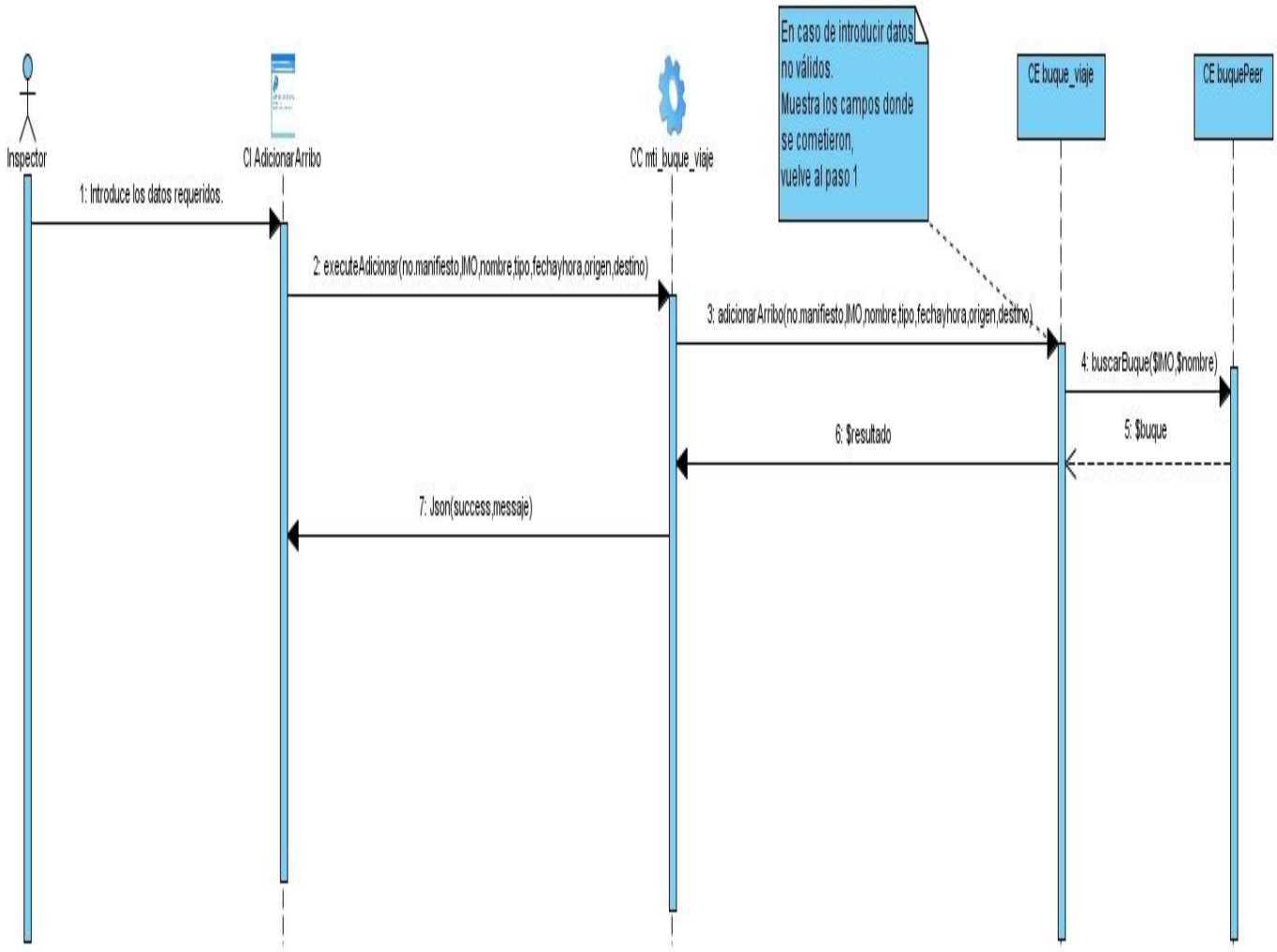


Figura 7 Diagrama de secuencia adicionar arribo.

2.8. Diagrama de Clases

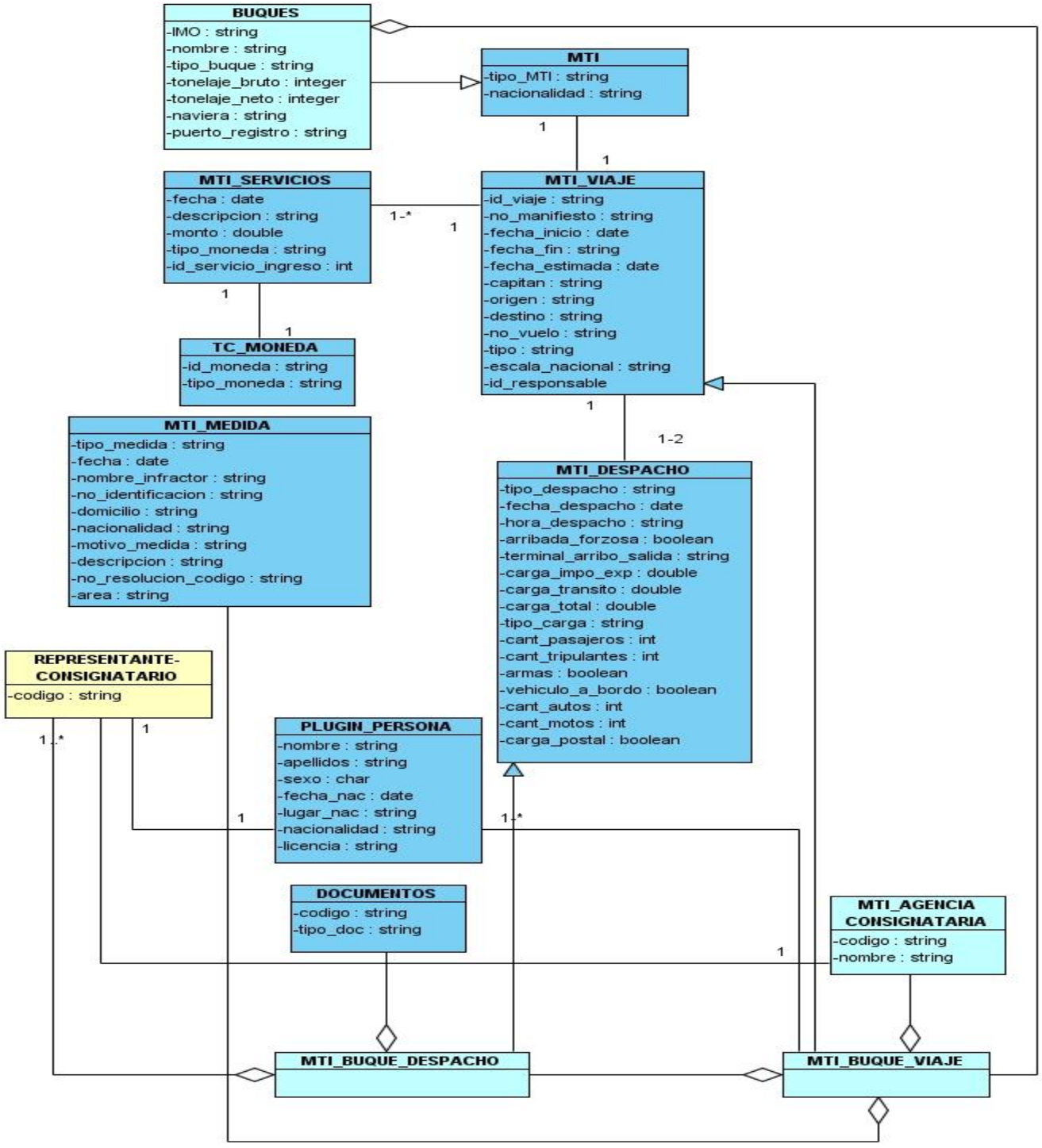


Figura 8 Diagrama de Clases

2.9. Diagrama de despliegue

En el diagrama de despliegue se muestra cómo y dónde se desplegará el sistema. Las máquinas físicas y los procesadores se representan como nodos, y la construcción interna puede ser representada por nodos o artefactos embebidos. Los estereotipos permiten precisar la naturaleza del equipo: dispositivos, procesadores y memoria.

A continuación se muestra el diagrama de despliegue correspondiente a la aplicación, donde se puede observar que la misma se encontrará desplegada en una unidad de procesamiento, contenida dentro del Servidor de Aplicaciones Web (Apache). La base de datos estará desplegada en otra unidad de procesamiento corriendo el servicio de Oracle y la asociación entre los nodos representa la ruta de comunicación entre ellos.

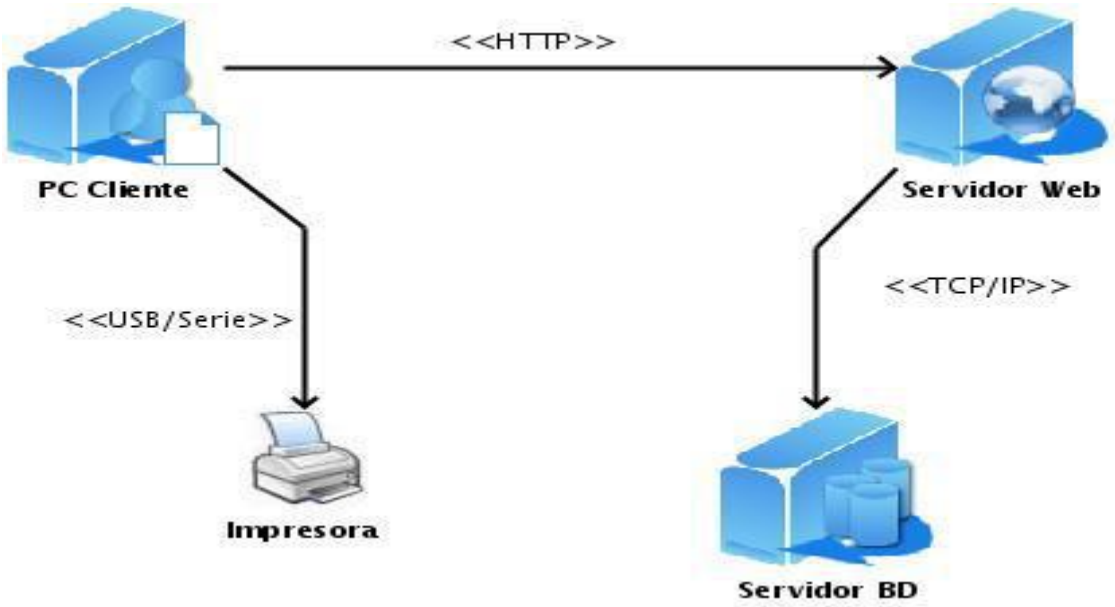


Figura 9 Diagrama de despliegue.

2.10.Modelo de Datos

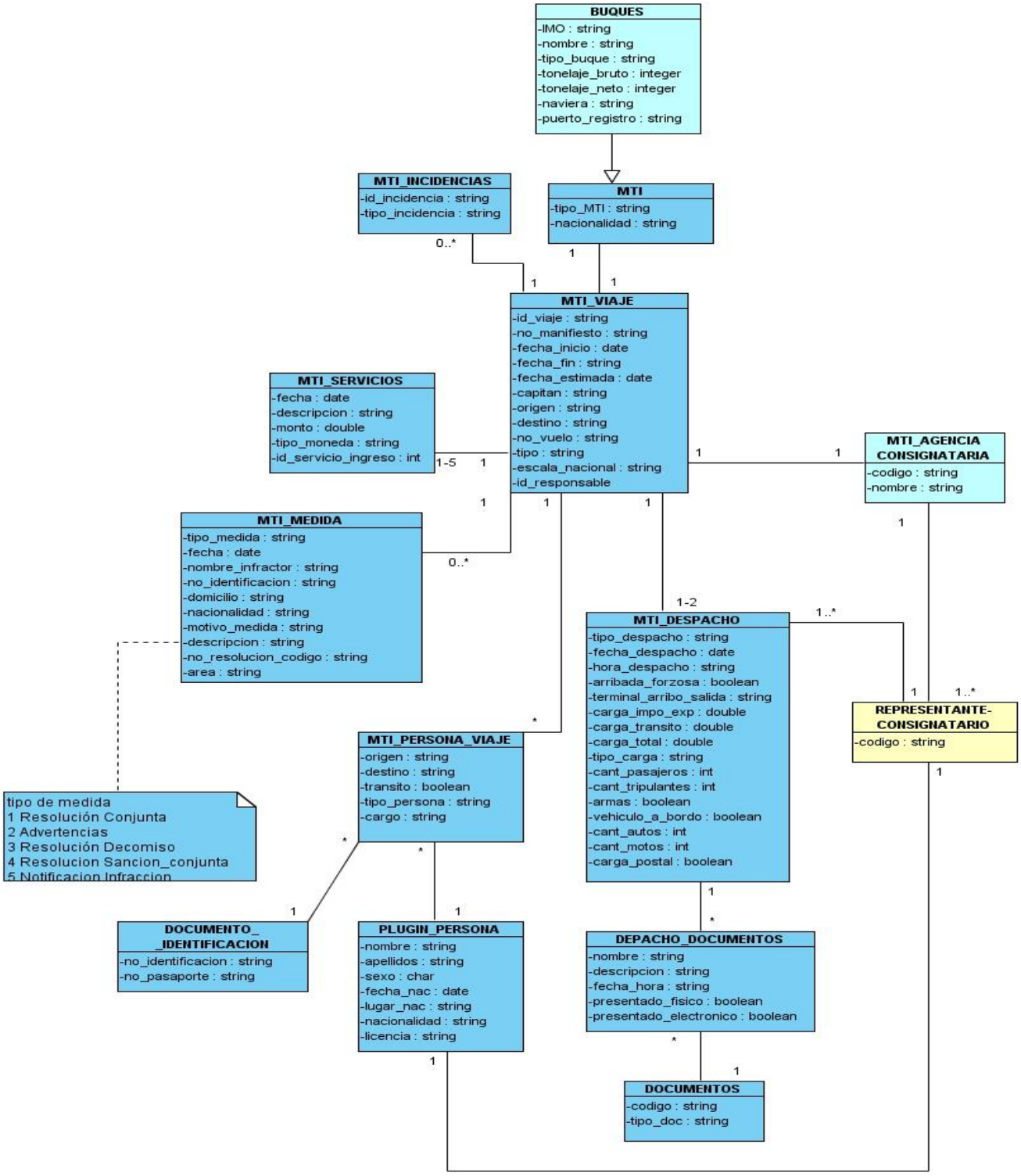


Figura 10 Modelo de datos.

2.11. Métricas para el Modelo de Diseño

Las métricas para diseño proporcionan al diseñador una mejor visión interna y ayudan a que el diseño evolucione a un nivel superior de calidad.

Para la aplicación de estas métricas se definen cinco características fundamentales:

Localización: Es una característica del software que indica la forma en que se concentra la información dentro de un programa.

Las métricas de software se han centrado en la estructura interna o complejidad de las funciones (longitud del módulo, cohesión, o complejidad ciclomática) o bien en la forma en que las funciones se conectan entre sí (acoplamiento de módulos).

Encapsulamiento: Se define el encapsulamiento como “el empaquetamiento de una colección de elementos”. El encapsulamiento influye en las métricas cambiando el objetivo de la medida, que pasa de ser un único módulo a ser un paquete de datos (atributos) y módulos de procesamiento (operaciones). Además, el encapsulamiento impulsa a la medida hasta un nivel de abstracción más elevado.

Ocultamiento de la información: Suprime los detalles operativos de un componente de un programa. Tan sólo se proporciona la información necesaria para acceder a ese componente o a aquellos otros componentes que deseen acceder a él.

Herencia: Es un mecanismo que hace posible que los compromisos de un objeto se difundan a otros objetos. Dado que la herencia es una característica fundamental de muchos sistemas, hay muchas métricas que se centran en ella. Entre los ejemplos que se tratarán más adelante: se cuentan el número de descendientes (número de instancias inmediatas de una clase), número de predecesores (número de generalizaciones inmediatas), y grado de anidamiento de la jerarquía de clases (profundidad de una clase dentro de una jerarquía de herencia).

Abstracción: Es un mecanismo que permite al diseñador centrarse en los detalles esenciales de algún componente de un programa (tanto si es un dato como si es un proceso) sin preocuparse por los detalles de nivel inferior. Cuando los niveles de abstracción van elevándose, se ignoran más y más detalles, por lo tanto, se proporciona una visión más general de un concepto u objeto. A medida que se

pasa a niveles más reducidos de abstracción, se muestran más detalles, lo cual proporciona una visión más específica de un concepto.

2.11.1. Métricas orientadas a clases

Las técnicas que se usaron para evaluar estas métricas en el diseño propuesto fueron planteadas por los especialistas del proyecto de Planificación de Recursos Empresariales (ERP) por sus siglas en inglés, perteneciente al Centro de Informatización para la Gestión de Entidades (CEIGE). Son necesarios los valores arrojados por estas técnicas ya que estos han sido para los parámetros de calidad, una polémica a nivel mundial en el diseño de sistemas.

2.11.1.1. Métricas propuestas por Lorenz y Kidd

En el libro de métricas realizado por 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.

Tamaño de clase (TC)

Esta métrica plantea que si existen valores grandes de TC estos mostrarán que una clase puede tener demasiada responsabilidad, lo cual reducirá la reutilizabilidad de la clase y complicará la implementación y la comprobación, por otra parte cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente.

Para evaluar esta métrica se tuvieron en cuenta la cantidad de procedimientos por clase y se obtuvo el siguiente resultado: para un total de 15 clases que definen operaciones, existe un promedio aproximado de operaciones de un 7.47 como se muestra en la tabla 6.

Total de clases	15
Promedio de procedimientos	7.46666667

Tabla 6 Promedio de Procedimientos por Clase.

Teniendo en cuenta la tabla anterior se aplicaron las técnicas de métricas para dar una categoría a las clases según la cantidad de procedimientos presentes en las mismas.

	Categoría	Criterio
Responsabilidad	Baja	<= Prom
	Media	Entre Prom. y 2*Prom
	Alta	>2*Prom.

Tabla 7 Técnicas para Responsabilidad.

Complejidad Implementación	Baja	<= Prom
	Media	Entre Prom. y 2*Prom
	Alta	>2*Prom.

Tabla 8 Técnicas para la Complejidad de la Implementación.

Reutilización	Baja	>2*Prom
	Media	Entre Prom. y 2*Prom
	Alta	<= Prom

Tabla 9 Técnicas para Reutilización.

La aplicación de la métrica TC al diseño propuesto para el subsistema arrojó los resultados que se observan a continuación:

De un total de 15 clases que definen operaciones existe un 27% de clases con alta responsabilidad, un 6% de clases con un nivel medio de responsabilidad y un 67% de clases con responsabilidad baja.

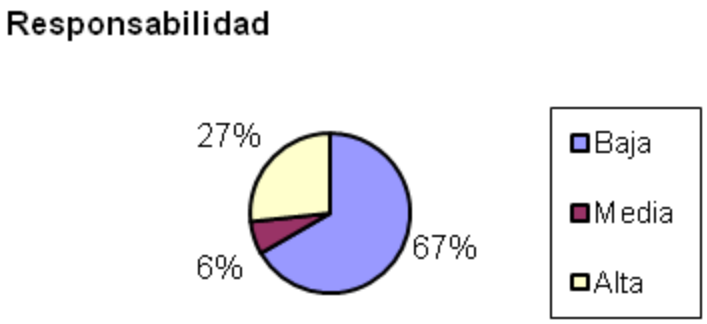


Figura 11 Por ciento de Responsabilidad.

Del total de clases que definen operaciones el 67% tiene baja complejidad, existe un 6% de clases de media complejidad, así como un 27% de clases con alta complejidad.

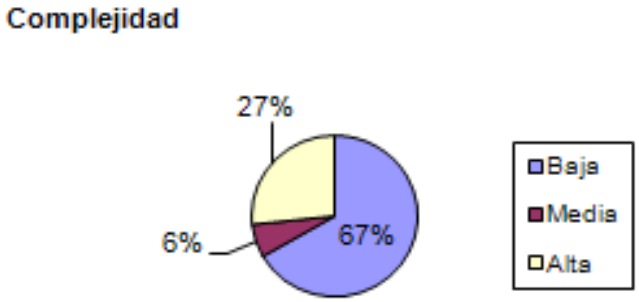


Figura 12 Por ciento de Complejidad.

Del total de las 15 clases que definen operaciones, los resultados arrojados por las técnicas fueron: un 27% de clases con baja reutilización, un 6% de las mismas presentan un porciento de reutilización medio, así como el 67% restante que representa las clases de alta reutilización.

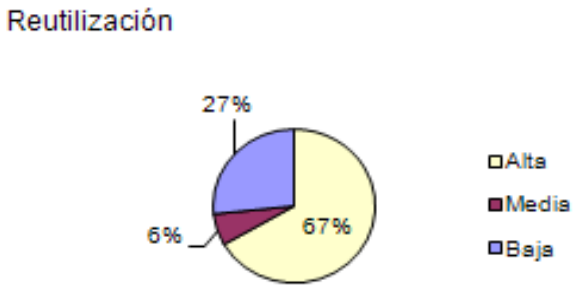


Figura 13 Por ciento de Reutilización.

Los valores obtenidos al concluir la evaluación del diseño del subsistema aplicando la métrica TC demuestran que existen bajos niveles de responsabilidad y de complejidad así como un alto nivel en el indicador de reutilización, lo cual garantiza la calidad del diseño realizado.

2.11.1.2. Métricas propuestas por Chidamber & Kemener

Uno de los conjuntos de métricas de software a los que se hace más ampliamente referencia es el propuesto por Chidamber y Kemener. Estas métricas de diseño propuestas se basan en clases, a las cuales suele aludirse con el nombre de conjunto de métricas CK para sistemas. Estas están enfocadas fundamentalmente a la herencia y se aplicaron dos durante la evaluación del diseño.

Árbol de Profundidad de Herencia (APH)

Esta métrica se define como la longitud máxima desde el nodo hasta la raíz del árbol. A medida que crece el APH, es más probable que las clases de niveles inferiores hereden muchos métodos. Esto da lugar a posibles dificultades cuando se intenta predecir el comportamiento de una clase. Una jerarquía de clases profunda (con un valor grande de APH) lleva también a una mayor complejidad de diseño. Por el lado positivo, los valores grandes de APH implican que se pueden reutilizar muchos métodos.

Para evaluar esta métrica se tuvieron en cuenta la cantidad de descendientes por clase como se muestra en la tabla 15:

Clase	Clase Padre	Niveles del árbol de herencia.
MtiBuque	Mti	1
MtiBuqueViaje	MtiViaje	1
MtiBuqueDespacho	MtiDespacho	1

Figura 14 Niveles de herencia por clase.

Para un total de 3 clases que son herederas y no poseen descendientes, resulta en el nivel 1 del árbol de herencia, por esto se concluye que el APH se encuentra en un nivel bajo de profundidad de herencia lo que a su vez garantiza el bajo acoplamiento entre las clases.

Número de Descendientes (NDD)

Se denominan descendientes a las subclases que son inmediatamente subordinadas a otras. A medida que crece el número de descendientes, se incrementa la reutilización, pero también es cierto, que a medida que crece el NDD, la abstracción representada por la clase predecesora puede verse diluida. Lo cual trae consigo la posibilidad de que algunos de los descendientes no sean realmente miembros propios de la clase predecesora. A medida que el NDD va creciendo, la cantidad de pruebas crecerá también.

Clase	Clase Padre	Número de descendientes.
MtiBuque	Mti	0
MtiBuqueViaje	MtiViaje	0
MtiBuqueDespacho	MtiDespacho	0

Figura 15 Número de Descendientes por Clase.

A continuación se plantean las técnicas definidas para esta métrica:

	Categoría	Criterio
Reutilización	Baja	<= Prom
	Media	Entre Prom. y 2*Prom
	Alta	>2*Prom.

Figura 16 Técnicas para Reutilización.

Abstracción	Indefinida	>5
	Afectada	Entre 2 y 5
	Definida	<=2

Figura 17 Técnicas para Abstracción.

	Categoría	**Criterio**
Cohesión	Baja	>5
	Media	Entre 2 y 5
	Alta	<=2

Figura 18 Técnicas para Cohesión.

Cantidad de pruebas	Baja	<=2
	Media	Entre 2 y 5
	Alta	>5

Figura 19 Técnicas para cantidad de pruebas.

Las clases que heredan no tienen descendientes, lo cual trae consigo que haya una baja reutilización, una muy definida abstracción de clase, la cohesión de la jerarquía de clases es alta y la cantidad de pruebas es baja.

En el modelo de datos elaborado la herencia que existe mantiene bajos niveles, lo cual garantiza el bajo acoplamiento entre clases ya que la colaboración se mantiene en un mínimo aceptable estableciendo solo las relaciones necesarias entre clases. Esto también facilita que no haya carencia de cohesión en los métodos.

2.11.2. Métricas orientadas a Operaciones

Dado que la clase es la unidad dominante en los sistemas, se han planteado menos métricas para las operaciones de clases. Algunos especialistas describen esto cuando plantean que los resultados de los últimos estudios indican que los métodos tienden a ser pequeños, tanto en términos del número de sentencias como en términos de su complejidad lógica, lo cual sugiere que la estructura de conectividad de un sistema pueda resultar más importante que el contenido de los módulos individuales.

Métricas propuestas por Lorenz y Kidd

Lorenz y Kidd proponen en su libro algunas métricas sencillas para el conocimiento cualitativo de la efectividad de las funcionalidades del subsistema partiendo del diseño realizado. A continuación se aplican dos de ellas.

Número Medio de Parámetros por Operación (NPavg)

Esta métrica plantea que en cuanto sea más grande el número de parámetros de la operación, será más compleja la colaboración entre objetos. En general, el NPavg debería de mantenerse tan bajo como sea posible.

En el diseño del subsistema se diseñaron las funcionalidades con la cantidad de atributos mínima y suficiente lo cual permite obtener los resultados requeridos por el requisito y evita que aumente la complejidad entre objetos.

Tamaño medio de operación (TOavg)

Esta métrica plantea que el número de mensajes enviados por la operación proporciona una alternativa para el tamaño de la operación. A medida que asciende el número de mensajes enviados por una única operación, es posible que las responsabilidades no hayan sido bien estipuladas dentro de la clase.

En las funcionalidades diseñadas para el subsistema son muy pocas las que emiten mensajes, lo cual evidencia una correcta distribución de responsabilidades dentro de las clases.

2.12.Conclusiones Parciales

A lo largo de este capítulo se obtienen los artefactos del diseño de la aplicación. Se realizaron las clases del diseño con estereotipos web, diagrama de secuencia, diseño de la base de datos y el diagrama de despliegue que muestra las relaciones físicas entre los componentes hardware y software en el sistema final, los cuales servirán de entrada para el flujo de implementación en el siguiente capítulo. Para validar el diseño de la solución se definieron un conjunto de métricas que contribuyeron a la detección de errores y a la mejora de la calidad en el diseño.

Capítulo 3: Implementación de la Solución

3.1. Introducción

Este capítulo describe cómo los elementos del modelo de diseño son implementados en términos de componentes y cómo los mismos se organizan de acuerdo con los nodos referidos en el modelo de despliegue. Se exponen las distintas pruebas realizadas a cada caso de uso, siguiendo particularmente, el método de pruebas de Caja Negra.

3.2. Estándar de Codificación

Actualmente se definen 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.

Se utilizó el estándar establecido por el Dpto. Aduana, CEIGE.

Como regla general se tiene:

1. Cuando se incluyen abreviaturas en mayúsculas no se debe incluir su nombre completo, sino que se utiliza el primer nombre en mayúscula y el resto en minúsculas.

   Ejemplo:

   - GetHtmlStatistic//Correcto
   - GetHTMLStatistic//Incorrecto

Aplicaciones

1. Las aplicaciones deben tener nombres que dejen reflejado bien claramente cuál es el propósito de la misma, ya en una palabra o siglas.

En caso de ser mediante siglas se pondrán todas en mayúsculas

2. Se debe evitar mientras sea posible la utilización de palabras compuestas o la utilización de varias palabras, en caso de que sea palabras compuestas se utilizará la notación UpperCamelCase[5].

Módulos

1. Deben referirse a los nombres de tablas en caso de que se trate de un módulo generado por el CRUD.
2. En caso de que sean módulos del negocio de la aplicación debe cumplir con las mismas reglas de codificación de los nombres de las aplicaciones.

Acciones

Symfony trae su propia nomenclatura para las clases de las acciones y sus funciones, pero no especifica claramente cuál es el nombre que se le debe poner a cada una de las funcionalidades del modelo que serán accedidas por el usuario [dígase acciones].

1. Dentro de las especificaciones del Framework está que cada una de estas acciones debe comenzar con la palabra execute.
2. Todos los nombres de acciones deben estar en la nomenclatura “CamelCase” comenzando por la palabra execute.
3. En caso de ser acciones referentes a un módulo de CRUD de una tabla deben ser nombres específicos como executeNuevo, executeEditar, etc.
4. Los nombres de las acciones deben especificar con la menor cantidad de palabras cual es el objetivo de la acción, de ser posible estar en infinitivo. Debe especificar bien claro cuál es la acción que se pretende ejecutar con la acción pero sin especificar los parámetros que recibe.

   Ejemplo:

   - Correcto: usuario/editar
   - Incorrecto: usuario/editar_dado_id

[5] *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.*

Plugins

Symfony especifica que la notación de los plugins siempre debe estar atada al sufijo Plugin. El nombre de un plugin debe indicarse con la menor cantidad de palabras, cual es el objetivo del plugin e inciarse con el prefijo sf.

Nombre de las clases

1. Los nombres de las clases deben estar expresados en notación UpperCamelCase.
2. No se deben utilizar guiones bajos en su nombre "_".
3. Deben expresar con claridad cuál es el alcance y la responsabilidad de la clase.
4. Los nombres de las clases no deben estar atados a las clases de las que se deriva, cada clase debe tener un significado por ella misma, no en dependencia de la clase de la que deriva.
5. En los nombres compuestos por más de tres palabras se debe revisar el diseño, no sea que se le estén dando a la clase más responsabilidades de las que realmente tiene.

Nombres de los archivos de las clases

Los nombres de los archivos de las clases deben estar compuestos por el nombre de la clase seguido de un punto y la palabra "class" y la extensión del archivo ".php".

Variables

1. Los nombres de las variables deben expresar claramente el contenido de la misma.
2. Pueden estar referidas en singular o plural.
3. Se definen al principio de las estructuras donde son utilizadas.
4. En caso de que no se le asigne un valor inicial se deben inicializar con un valor que indique el tipo de dato más general al que debe pertenecer.
    - Los tipos de datos cadena son definidos con comillas dobles (");
    - Los tipos de datos de caracteres se definen con comillas simples (');
    - En caso de que se espere almacenar tipos de datos diversos no se inicializa.

3.3. Diagrama de componentes físicos

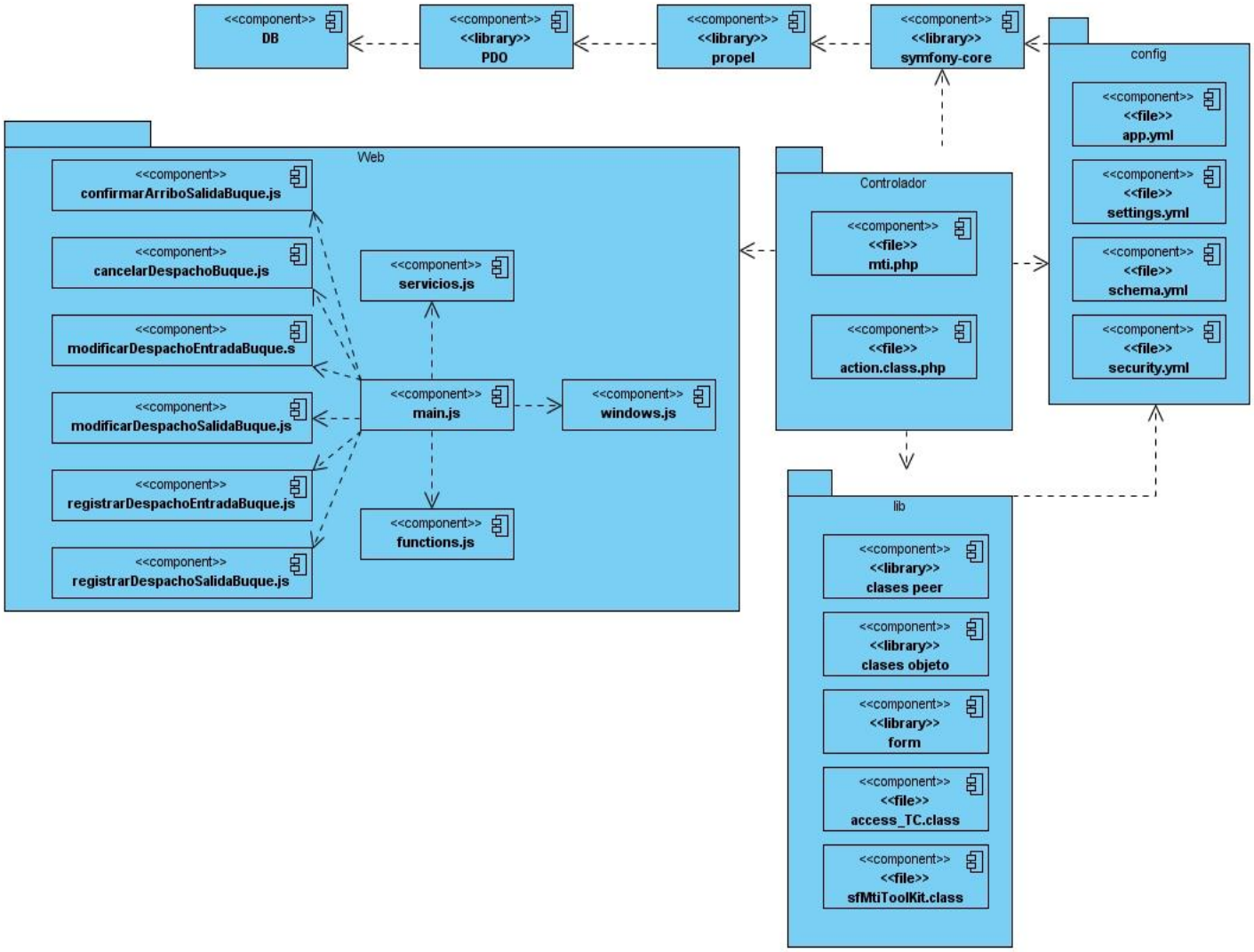
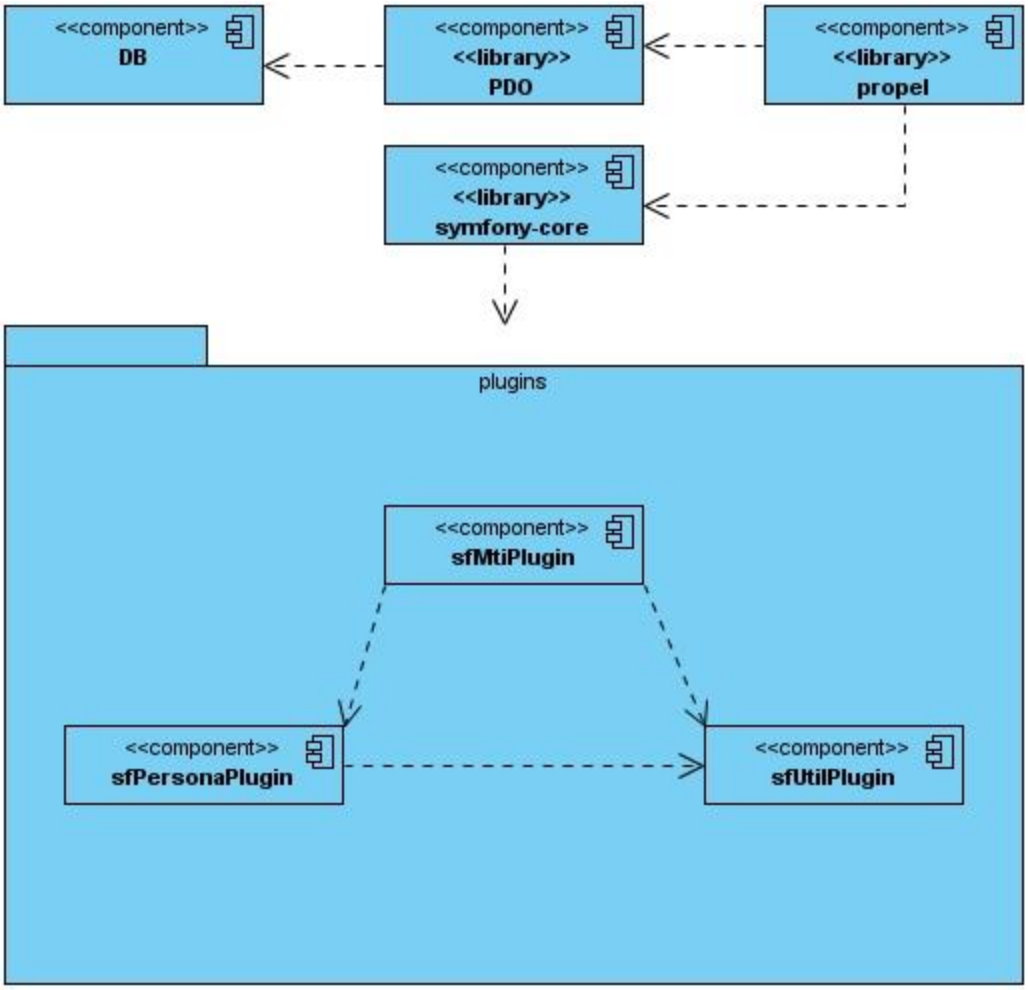


Figura 20 Diagrama de Componentes.

3.4. Diagrama de componentes lógicos



3.5. Implementación de Symfony

Antes de empezar a desarrollar aplicaciones Symfony, es necesario instalar los archivos y librerías que componen el framework. Como Symfony está programado con PHP y sus creadores han sido muy cuidadosos, Symfony funciona igual de bien en cualquier sistema operativo (Windows, Mac OS X y Linux).

Por este motivo, la instalación es idéntica en cualquier sistema operativo y tan sólo es necesario modificar las rutas de los directorios y algún otro detalle menor.

Symfony se puede instalar de tres formas diferentes:

- Para probar Symfony lo más rápido posible para ver sus posibilidades, es recomendable implementar el entorno de pruebas o sandbox.

- Si se conoce Symfony y se quiere instalar de la mejor forma posible para desarrollar proyectos con el mismo framework, se debe realizar la instalación mediante PEAR, que además es la instalación más común.

- Para usuarios muy avanzados, si se desea tener un control absoluto sobre la versión de Symfony, y hacer cambios en el código fuente del framework y además que cada proyecto disponga de su propia versión de Symfony, se debe realizar la instalación mediante el repositorio Subversion.

3.6. Implementación del proyecto, la aplicación y el módulo

Symfony considera un proyecto como "un conjunto de servicios y operaciones disponibles bajo un determinado nombre de dominio y que comparten el mismo modelo de objetos"(Potencier, 2008) Dentro de un proyecto Symfony, las operaciones se agrupan de forma lógica en aplicaciones las que están formadas por uno o varios módulos, a su vez, un módulo representa a una página web o grupo de ellas con un propósito estrechamente relacionado. Para crear el proyecto, la aplicación y el módulo, al tratarse de una plataforma como Windows XP se hace de la siguiente forma:

- Se crea la carpeta del proyecto (/wamp/www mkdir GINA)

- Se ejecutan los comandos (/wamp/www/GINA symfony init-project GINA), ((/wamp/www/GINA symfony init-app MTI) y ((/wamp/www/GINA symfony init-module MTI MTIBuques).

3.7. Implementación del 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*. Se editó correctamente dichos archivos se puede ejecutar el comando (symfony propel-build-schema xml) para generar el esquema que representará las tablas de la base de datos en XML y que será almacenado en el archivo *schema.xml*. Si se generó correctamente se procede a generar el modelo, no sin antes recordar que al usar Oracle se debe especificar los campos de las tablas que serán autoincrementables.

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

3.8. Interfaz de usuario

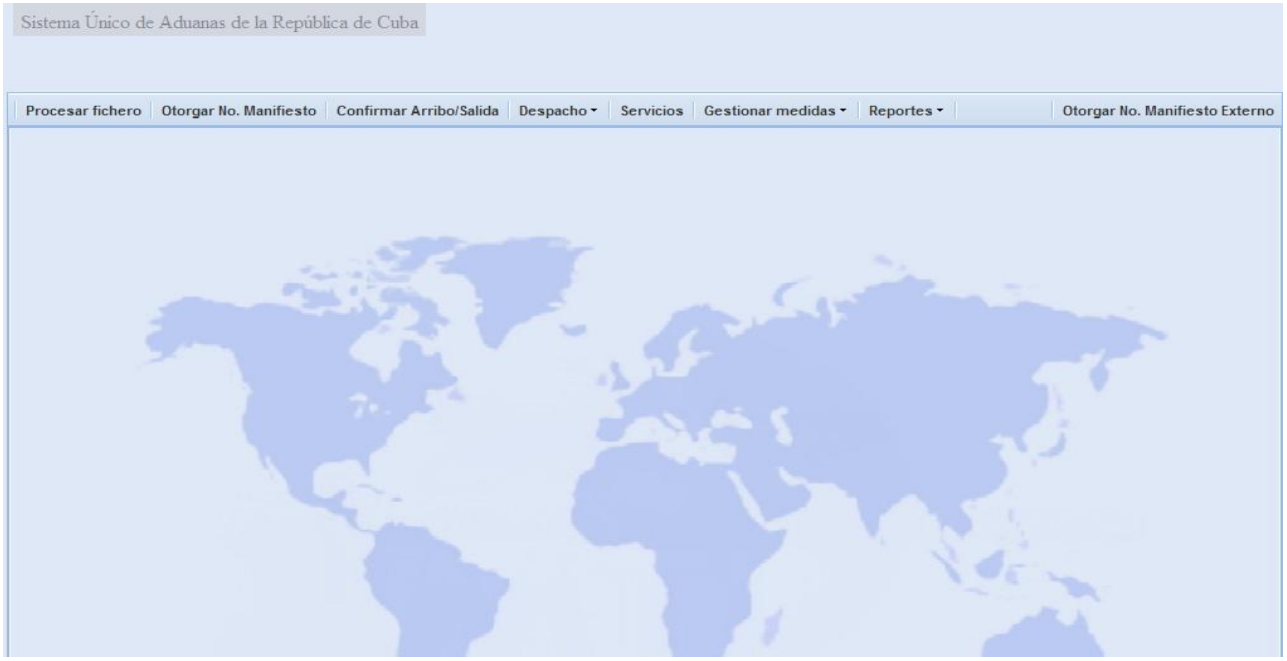


Figura 21 Interfaz principal.

Para el escenario Adicionar Arribo del Caso de Uso Adicionar Arribo se tiene en cuenta el principio de usabilidad de tener siempre informado al usuario sobre el estado del sistema, así como para el resto de los Casos de Uso, por ejemplo, dejando un mensaje de que los datos se guardaron.

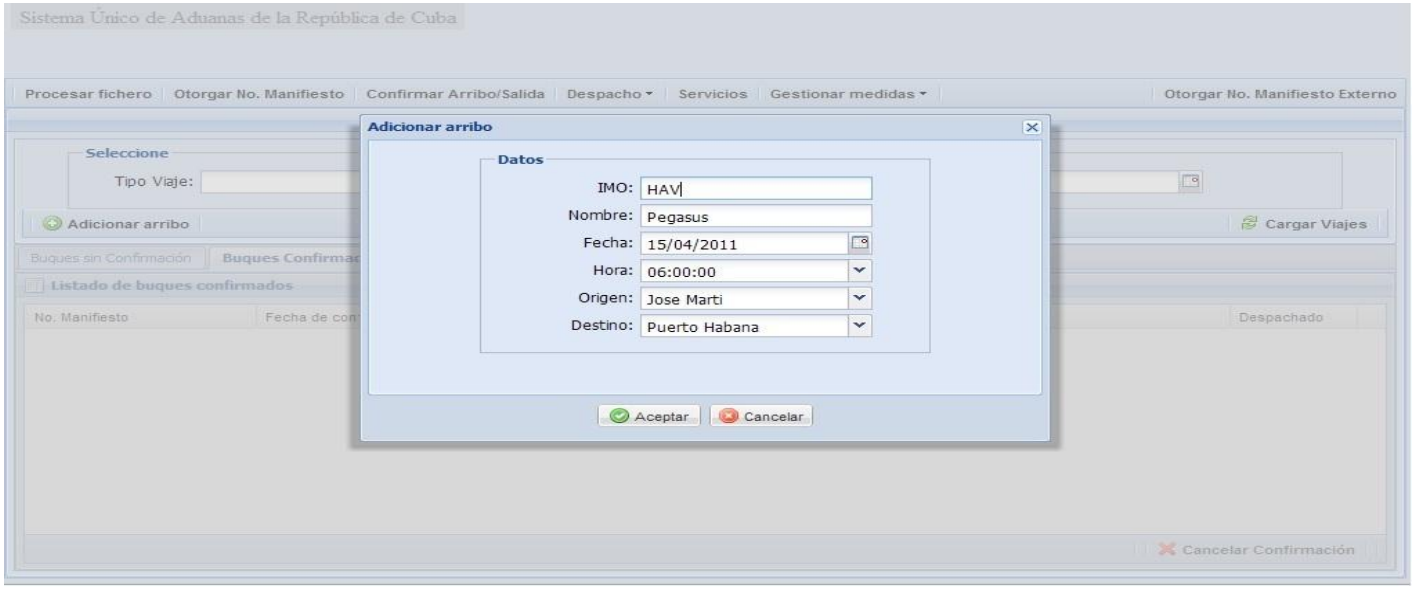


Figura 22 Adicionar arribo ExtJS.

Fragmento de código correspondiente a la pantalla:

```
PopAdicionarArribo = Ext.extend(Ext.Window, {
    resizable: false,
    modal: true,
    shadowOffset: 10,
    border: false,
    buttonAlign:'center',
    layout: 'fit',
    height: 310,
    width: 500,
    closeAction: 'destroy',
    plain: true,
    // ---Datos procedentes de la pantalla anterior --
    arribo: null,
    initComponent: function(){
        var obj = this;

        this.on('show', function(){
            obj.doLayout();
            obj.on('hide', function(){
                obj.destroy();
            });
        });
        this.items = [new Ext.form.FormPanel({
            bodyStyle:'padding:5px 5px 0',
```

```
            id:'formAdd',
            width: 350,
            frame:true,
            labelAlign:'right',
            labelWidth:100,
            defaults: {
                anchor:'70%',
                cls:'center'
            },
            items:[{
                xtype:'fieldset',
                title:'Datos',
                defaultType: 'textfield',
                defaults:{
                    anchor:'90%'
                },
                items:[
                {
                    fieldLabel: 'IMO',
                    id:'idImo',
                    allowBlank:false
                },{
                    fieldLabel: 'Nombre',
                    id:'idNombre',
                    allowBlank:false
                },{
                    fieldLabel: 'Fecha',
                    id:'idFecha',
                    xtype:'datefield',
                    allowBlank:false
                },{
                    xtype: 'timefield',
                    fieldLabel: 'Hora',
                    allowBlank: false,
                    id: 'idHora',
                    // name: 'horaArribo',
                    format: 'H:i:s',
                    readOnly: true
                },{
                    xtype: 'combo',
                    id:'idOrigen',
                    fieldLabel: 'Origen',
                    typeAhead: true,
                    anchor:'90%',
                    triggerAction: 'all',
                    lazyRender:true,
                    mode: 'local',
                    valueField: 'origen',
                    displayField: 'displayText',
                    readOnly: true,
                    allowBlank: false,
store: new Ext.data.ArrayStore({id: 0,fields: ['origen','displayText']
})
 }
```

Además se probaron otras alternativas, como es la animación de cuando se está cargando un fichero.

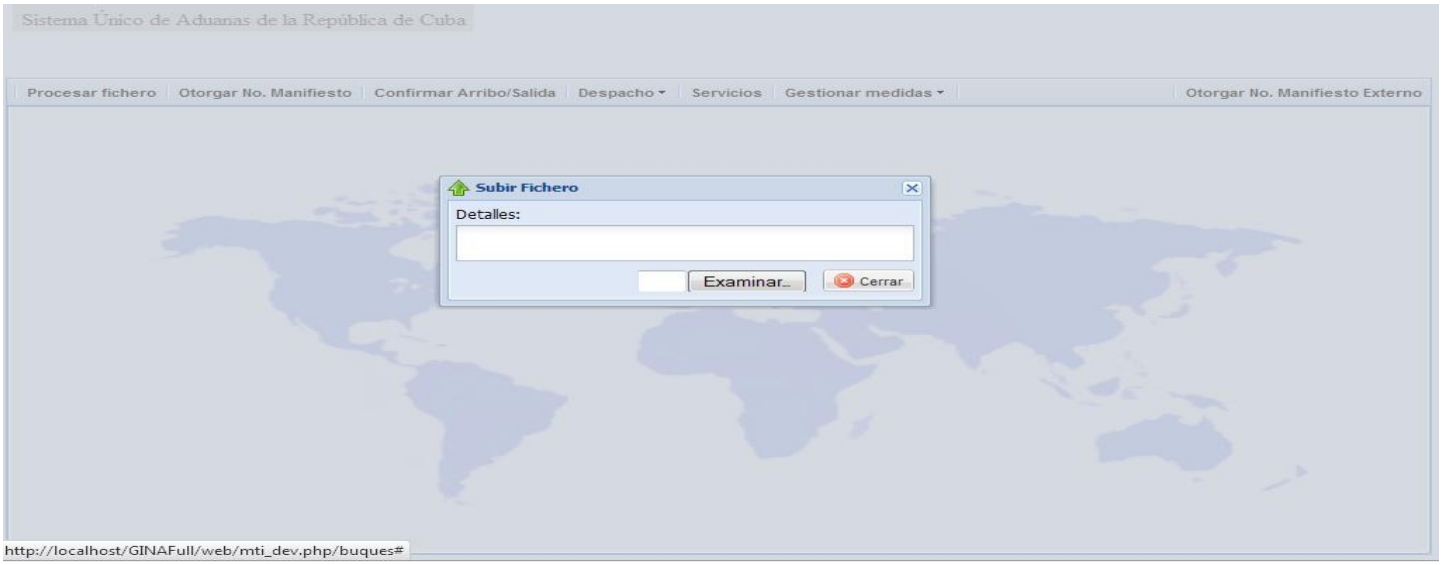


Figura 23 Cargar fichero ExtJS.

Fragmento de código de requisito Adicionar Arribo/Salida:

```
static function addArriboBuques($datos) {

        $errores = array();
        $criteria = new Criteria();
        $entrada = $datos['tipo'] == self::TIPO_VUELO_ENTRADA ? true : false;
        $criteria->addAnd(MtiViajePeer::ORIGEN, $datos['origen']);
        $criteria->addAnd(MtiViajePeer::DESTINO, $datos['destino']);
        $criteria->addAnd(MtiViajePeer::TIPO, $datos['tipo']);
        $criteria->addJoin(MtiViajePeer::ID_MTI, MtiBuquePeer::ID_MTI);
        $criteria->add(MtiBuquePeer::IMO, $datos['mtiBuque']['imo']);
        $criteria->add(MtiBuquePeer::NOMBRE, $datos['mtiBuque']['nombre']);

        $result = MtiViajePeer::doSelectOne($criteria);

        if ($result == null) {

            if ($datos['tipo'] == self::TIPO_VUELO_ENTRADA) {
                $result = new MtiViaje($datos);
                $result->setModo(MtiViaje::ADD_VIAJE_ENTRADA);
            } else
            if ($datos['tipo'] == self::TIPO_VUELO_SALIDA) {
                $result = new MtiViaje($datos);
                $result->setModo(MtiViaje::ADD_VIAJE_SALIDA);
            }

            $mtiViaje = new MtiBuqueViajeForm($result);
            if (!$mtiViaje->bindAndSave($datos)) {
```

```
                $result = sfMtiToolkit::errorSchemaToArray($mtiViaje-
>getErrorSchema());
                return $result;
            }
            // registrar el suceso ocurrido con el viaje
            $v = $mtiViaje->getObject();
            $datos = array(
                'fecha' => date('Y-m-d H:i:s'),
                'tipoSuceso' => self::SUCESO_REGISTRADO,
                'observacion' => sfConfig::get('app_obs_suceso_viaje_adicionado'),
                'idMtiViaje' => $v->getIdMtiViaje()
            );
            try {
                $aux = self::registrarSucesoViaje($datos);
            } catch (Exception $e) {

                return self::SUCESO_NO_REGISTRADO . self::VIAJE_ADICIONADO;
            }
            if ($aux) {

                return $v->getIdMtiViaje();
            } else {

                return self::SUCESO_NO_REGISTRADO;
            }
        } else {// caso en que el viaje a adicionar está en la base de datos
            $datos['idMtiViaje'] = $result->getIdMtiViaje();
            //registrar suceso con el vuelo en cuestión
            $datos = array(
                'fecha' => date('Y-m-d H:i:s'),
                'tipoSuceso' => self::SUCESO_INTENTO_ADICION_VIAJE_EXISTENTE,
                'observacion' =>
sfConfig::get('app_obs_obs_suceso_viaje_existente'),
                'idMtiViaje' => $result->getIdMtiViaje()
            );
            try {
                $aux = self::registrarSucesoViaje($datos);
            } catch (Exception $e) {
                return self::SUCESO_NO_REGISTRADO;
            }
            if ($aux) {
                return self::SUCESO_INTENTO_ADICION_VIAJE_EXISTENTE;
            } else {
                return self::SUCESO_NO_REGISTRADO;
            }
        }
    }
```

3.9. Pruebas

Las pruebas verifican que el producto funcione como se diseñó y que los requerimientos son satisfechos cabalmente, además de brindar soporte para encontrar y documentar defectos del sistema.

El principal objetivo de las pruebas es evaluar la calidad del producto que se está desarrollando, pues está presente durante todo el ciclo de desarrollo del sistema lo que posibilita que se vaya refinando constantemente y no al final. El papel de las pruebas no es asegurar la calidad, pero sí evaluarla, y proporcionar una realimentación a tiempo, de forma que los aspectos de calidad puedan resolverse de manera efectiva en tiempo y costo.

3.9.1. Pruebas de Caja Negra

Prueba de Especificación o de Caja Negra: Tiene como propósito verificar las relaciones de entrada y salida de una unidad. Su objetivo es verificar qué hace la unidad, pero sin averiguar cómo lo hace. Se centra y basa en la entrada/salida de datos de las operaciones de la unidad.

Cuando se considera el software de computadora, la prueba de Caja Negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de pruebas pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto. Permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes; errores de interfaz, en estructuras de datos o en acceso a bases de datos externas; errores de rendimiento, de inicialización y de terminación.

CP1: Adicionar Arribo/Salida		
Entrada	**Resultados**	**Condiciones**
El usuario introduce los datos del buque, la fecha y hora de llegada y el origen y destino del viaje y presiona “Aceptar”.	El sistema valida los datos y adiciona el arribo o la salida.	Los datos introducidos son correctos.
El usuario introduce los datos del buque, la fecha y hora de llegada, el origen y destino del viaje y presiona “Aceptar”.	El sistema muestra el mensaje: • “No se pudo adicionar el Arribo, contacte con el Administrador del Sistema”	El formato de los datos introducidos es incorrecto.
El usuario deja vacío un campo y, presiona “Aceptar”.	El sistema muestra los siguientes mensajes respectivamente: • “El campo (campo) es necesario”	Campo vacío.

Tabla 10 CP1: Adicionar Arribo/Salida.

CP2: Confirmar Arribo/Salida		
Entrada	**Resultados**	**Condiciones**
El usuario selecciona el viaje a confirmar e introduce la fecha de confirmación y presiona "Aceptar".	El sistema valida los datos y confirma el arribo o la salida.	Los datos introducidos son correctos.
El usuario selecciona el viaje a confirmar e introduce la fecha de confirmación y presiona "Aceptar".	El sistema muestra el mensaje: • "No se pudo confirmar el Arribo/Salida, contacte con el Administrador del Sistema"	El formato de los datos introducidos es incorrecto.
El usuario deja vacío el campo de la fecha de confirmación y, presiona "Aceptar".	El sistema muestra los siguientes mensajes respectivamente: • "El campo fecha de confirmación es obligatorio"	Campo de fecha vacío.
El usuario selecciona el viaje a confirmar e introduce la fecha de confirmación y presiona "Aceptar".	El sistema muestra los siguientes mensajes respectivamente: • "No se pudo realizar la petición requerida, el viaje ya fue confirmado"	Viaje confirmado.
El usuario selecciona un viaje de tipo salida a confirmar e introduce la fecha de confirmación y presiona "Aceptar".	El sistema muestra los siguientes mensajes respectivamente: • "No se pudo realizar la petición requerida, el viaje de salida no tiene un despacho asociado"	Viaje salida no despachado.

Tabla 11 CP2: Confirmar Arribo/Salida.

CP3: Registrar Despacho Entrada/Salida		
Entrada	**Resultados**	**Condiciones**
El usuario selecciona el viaje a despachar y presiona "Aceptar".	El sistema valida los datos introducidos, que el viaje no se le haya hecho un despacho y registra el despacho de viaje seleccionado.	Los datos introducidos son correctos.
El usuario selecciona el viaje a despachar y presiona "Aceptar".	El sistema muestra el mensaje: • "No se pudo realizar el despacho , contacte con el Administrador del Sistema"	El formato de los datos introducidos es incorrecto.

El usuario deja vacío un campo y, presiona "Aceptar".	El sistema muestra los siguientes mensajes respectivamente: • "El campo (campo) es necesario"	Campo vacío.
El usuario selecciona el viaje a despachar y presiona "Aceptar".	El sistema muestra los siguientes mensajes respectivamente: • "No se pudo realizar la petición requerida, el viaje ya fue despachado"	Viaje despachado.

Tabla 12 CP3: Registrar Despacho Entrada/Salida.

CP4: Registrar notificación de infracción		
Entrada	**Resultados**	**Condiciones**
El usuario introduce los datos requeridos y presiona "Aceptar".	El sistema valida los datos introducidos y registra la notificación de infracción.	Los datos introducidos son correctos.
El usuario introduce los datos requeridos y presiona "Aceptar".	El sistema muestra el mensaje: • "No se pudo realizar el despacho , contacte con el Administrador del Sistema"	El formato de los datos introducidos es incorrecto.
El usuario deja vacío un campo y, presiona "Aceptar".	El sistema muestra los siguientes mensajes respectivamente: • "El campo (campo) es necesario"	Campo vacío.

Tabla 13 CP4: Registrar notificación de infracción.

CP5: Otorgar número de manifiesto		
Entrada	**Resultados**	**Condiciones**
El usuario introduce los datos requeridos y presiona "Aceptar".	El sistema valida los datos introducidos y registra la notificación de infracción.	Los datos introducidos son correctos.
El usuario introduce los datos requeridos y presiona "Aceptar".	El sistema muestra el mensaje: • "No se pudo realizar el despacho , contacte con el Administrador del Sistema"	El formato de los datos introducidos es incorrecto.
El usuario deja vacío un campo y, presiona "Aceptar".	El sistema muestra los siguientes mensajes respectivamente: • "El campo (campo) es necesario"	Campo vacío.

Tabla 14 CP5: Otorgar número de manifiesto.

3.10.Conclusiones Parciales

En este capítulo se expuso la fase de Implementación verificando que se cumplieran todos los requisitos establecidos previamente. A través de dos formas de representación como la del Diagrama de Componentes que muestra la organización y la dependencia entre un conjunto de componentes y el Diagrama de Despliegue, se provee la vista de implementación del sistema.

Se abordó como está estructurado el sistema en cuanto a su arquitectura.

Además, se realizaron un conjunto de pruebas al sistema con el fin de obtener un producto con la mayor calidad posible.

Conclusiones

Con el desarrollo de este trabajo se puede afirmar que se cumplieron los objetivos trazados:

- Se realizó un estudio del arte de las tecnologías actuales para el desarrollo de sistemas web.
- Se seleccionaron las herramientas, lenguajes y metodologías utilizadas, priorizando fundamentalmente las tecnologías libres y multiplataforma.
- Se realizó el Modelo de Datos.
- Se realizó el Diagrama de Despliegue.
- Se realizó el diseño de la solución y en él los Diagramas de Interacción entre Clases y los Diagramas de Secuencias para los diferentes escenarios.
- Se validó el diseño utilizando métricas que ayudaron a la detección de errores y a la mejora de la calidad en el diseño.
- Se realizó la implementación del Módulo Despacho de Buques como Medio de Transporte Marítimo para el Sistema Integral de Gestión de la Aduana siguiendo la arquitectura definida.
- Se llevaron a cabo pruebas de Caja Negra sobre la interfaz del software proporcionando unas entradas y estudiando las salidas con las que se pudo comprobar que los resultados eran los esperados.
- Se obtuvo el código fuente de la aplicación.

Recomendaciones

Los resultados de este trabajo han sido los esperados y de acuerdo con los objetivos que fueron definidos se puede afirmar que se cumplieron todos los requisitos capturados. No obstante, para futuras investigaciones y proyectos que guarden relación con este trabajo, se recomienda:

- Continuar el desarrollo de este sistema, adicionándole nuevas funcionalidades.
- Que se realice el despliegue del sistema.
- Que se cree un componente que se encargue de realizar la interacción entre el módulo de ingreso y la solución propuesta.

Referencias Bibliográficas

Administración General de Aduanas, Administración General de Aduanas. Despacho Aduanero. Julio 2005. Available from world wide web:

<http://www.sedi.oas.org/dctc/AdmAcuerdos/Administracion%20de%20las%20Reglas%20de%20Origen/09%20-%20Presentacion%20despacho%20Aduanero.pdf>.

Adriana Alfonso Luis, Luis Alberto Rosado Hidalgo. Diseño e implementación de la sección Revistas para D'TIC, Centro Virtual de Recursos. Junio 2010.

CRAIG LARMAN. *UML y Patrones Introducción al análisis y diseño orientado a objetos.* Primera edición [México]: Prentice Hall, 1999.

Colin Ramsay, Shea Frederick. *Learning Ext JS*. 2008.
Dirección Nacional de Aduanas Paraguay. Sistema SOFIA. 2010. Available from world wide web: <http://zbmail.aduana.gov.py/49-4-sistema-sofia.html>.

Fabien Potencier,, Fabien Potencier. *Symfony la guía definitiva*. Diciembre 2008.

FIDEL CASTRO RUZ, FIDEL CASTRO RUZ. *Decreto Ley 162*. [La Habana, Cuba], 1996.

Gang of Four. *Design Patterns: Elements of Reusable Object Oriented Software*. 1994.

Grupo UML, Jaime Sánchez Ortiz. Grupo UML. 2010. Available from world wide web:
<http://umlpanamericana.blogspot.com/>.

José Antonio Cobo Rodríguez, José Antonio Cobo Rodríguez. Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas. Junio 2008.

Letelier. *"Pruebas de Aceptación como conductor del Proceso Software. 2007. "*. Febrero 2010a Available from world wide web:
<http://zweb.iti.upv.es/groups/squac/events/JTS2007/slides/3demayo16.45-PatricioLetelier.pdf/attach/3de mayo16.45-PatricioLetelier.pdf>.

librosweb. El Controlador Frontal | Symfony 1.0, la guía definitiva | LibrosWeb.es. 2010b. Available from world wide web: <http://www.librosweb.es/symfony/capitulo6/el_controlador_frontal.html>.

Mark G. Graff, Kenneth R. van Wyk. *Secure Coding: Principles & Practices*. 2003.

Nallelys Rodríguez Santos, Ivelisse Mercedes Díaz Rodríguez. Diseño e implementación del proceso de inscripción de documentos que se realizan en las Notarías Públicas de la República Bolivariana de Venezuela. Junio 2010.

Pedro Abigantús Pérez, Aymelis González Almora. IMPLEMENTACIÓN DE SERVICIOS PARA LAGESTIÓN DE INFORMACIÓN EN EL POLOPETROSOFT. 2010.

Rafael Andrés Céspedes Basteiro, Yisel Rodríguez Pérez. Diseño del Módulo Control de Personas del Sistema Único de Aduanas. 2009.

Roberto Carlos García Andino. Diseño e Implementación del Módulo Medios de Transporte Internacional. 2009a.

Six Revisions. How to Debug PHP Using Firefox with FirePHP. 2009b. Available from world wide web: <http://sixrevisions.com/web-development/how-to-debug-php-using-firefox-with-firephp/>
.
symfony-users. Symfony-users. Mayo 2008. Available from world wide web: <http://www.mail-archive.com/symfony-users@googlegroups.com/msg07161.html>.

Universidad Distrital Francisco José de Caldas Facultad TecnológicaRedes de DatosElmer Andrés Cotrino Ángel. SERVIDOR WEB APACHE. 2010c. Available from world wide web: <http://es.scribd.com/doc/49826447/SERVIDOR-WEB-APACHE>.

Bibliografía

1. ANGEL ALVAREZ, M. *Qué es* Java script. [Consultado el: 13 enero de 2011]. Disponible en: http://www.desarrolloWeb.com/articulos/25.php.

2. Potencier, Fabien and Zaninotto, François. 2008. Guía definitiva de Symfony. 2008.

3. José Antonio Cobo Rodríguez. Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas. Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba: junio de 2008.

4. Yisel Rodríguez Pérez, Rafael Andrés Céspedes Basteiro. Diseño del Módulo Control de Personas del Sistema Único de Aduanas. Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba: abril de 2009.

5. Roberto Carlos García Andino. Diseño e Implementación del Módulo Medios de Transporte Internacional. Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba: junio 2009.

6. APACHE. Comunidad de desarrolladores y usuarios de apache [Consultado el: 15 febrero de 2011]. Disponible en: http://www.apache.org/.

7. NETBEANS. Comunidad de usuarios de NetBeans [Consultado el: 14 enero de 2011].

   Disponible en: http://www.netbeans.org/.

8. PRESSMAN. R. S. Ingeniera de software, un enfoque práctico. McGraw-Hill, 2001. [Consultado el: 24 marzo de 2011].

9. SÁNCHEZ, J. I. P. Metodología para el Desarrollo de Software. 2005, [Consultado el: 6 febrero de 2011] Disponible en: www.lcc.uma.es/~jignacio/index_archivos/TEMA4.pdf.

10. VISUAL PARADIGM INTERNATIONAL. Visual Paradigm for UML [Consultado el: 15 enero de 2011].

    Disponible en: http://www.visual-paradigm.com/product/vpuml/.

11. Patrones de Diseño GRASP [Consultado el: 24 febrero de 2011].

    Disponible en: http://es.scribd.com/doc/53315954/Patrones-de-Diseno-GRASP

12. Patrones de Diseño GoF [Consultado el: 24 febrero de 2011].

    Disponible en: http://es.scribd.com/doc/55128700/6/Los-Patrones-GoF-Originales

13. DAEDALUS. 2006. Diseño de Sistemas. [Online] 2006.

    Disponible en: http://www.daedalus.es/AreasISDiseno-E.php.

14. "Understanding and Using Patterns in Software Development." Riehle, D. y Züllighoven, H. 1996,

    Theory and Practice of Object Systems.

15. Potencier, Fabien y Zaninotto, François. "Symfony | Open-Source PHP Web Framework". [En Línea] Sitio Oficial del proyecto Symfony. http://www.symfony-project.org.

16. Rumbaugh, James. El Lenguaje Unificado de Modelado. Manual de Referencia.

17. ADUANA GENERAL DE LA REPÚBLICA. RESOLUCIÓN No. 187-2008. 2008.

18. ADUANA GENERAL DE LA REPUBLICA. Metodología Interna para el Despacho

19. Larman, Craig. Introducción al análisis y diseño orientado a objeto. UML y Patrones.

Glosario de términos

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

MTI: Medios de Transporte Internacional.

UML: Unified Modeling Language. Lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

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

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

XML: Siglas en inglés de *Extensible Markup Language* (lenguaje de marcas extensibles), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

Ajax: Acrónimo de *Asynchronous JavaScript And XML* (javascript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas.

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

URL: acrónimo de *Uniform Resource Locator*, es decir, localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos.

Cuello de botella: En ingeniería, un cuello de botella es un fenómeno en donde el rendimiento o capacidad de un sistema completo es severamente limitado por un único componente. El componente es generalmente llamado punto del cuello de botella. El término es una derivación metafórica que hace referencia al cuello de una botella, donde la velocidad del flujo de un líquido es limitado por este cuello angosto.

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

Artefacto: Pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar actividades.

DOM: (Document Object Model) Modelo en Objetos para la representación de Documentos. Es un modelo computacional a través de la cual los programas y scripts pueden acceder y modificar dinámicamente el contenido, estructura y estilo de los documentos HTML y XML.

DHTML: (Dynamic HTML) Designa el conjunto de técnicas que permiten crear sitios web interactivos utilizando una combinación de lenguaje HTML estático, un lenguaje interpretado en el lado del cliente (como JavaScript), el lenguaje de hojas de estilo en cascada (CSS) y la jerarquía de objetos de un DOM.

Layout: En Symfony se conoce como Layout el enmarcado que se le da a las páginas.

ORM: (Object Relational Mapping) Mapeo de Objetos Relacional es una técnica de programación para convertir tipos de datos incompatibles entre sistemas de bases de datos y lenguajes orientados a objetos.

CRUD: (Create, Retrieve, Update, Delete) Es usado para referirse a las funciones básicas en bases de datos o la capa de persistencia en un sistema de software.

Javascript: Es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

Ruby on Rails: También conocido como RoR o Rails es un Framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC).

PEAR: (PHP Extensión and Application Repository) Es un entorno de desarrollo y sistema de distribución para componentes de código PHP. El proyecto PEAR fue fundado por Williams G. Molina G. en 1999 para promover la reutilización de código que realizan tareas comunes.

Java: Lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++.

Ruby: Lenguaje de programación reflexivo y orientado a objetos (lenguaje interpretado), creado por el programador japonés Yukihiro "Matz" Matsumoto, quien comenzó a trabajar en Ruby en 1993, y lo presentó públicamente en 1995. Combina una sintaxis inspirada en Python, Perl con características de programación orientada a objetos similares a Smalltalk.

GoF: (Gang of Four) Grupo de los Cuatro, se refiere al grupo de los autores de este tipo de patrones de diseño.