Universidad de las Ciencias Informáticas
Facultad 3

Título:

Diseño e Implementación del Módulo Solicitudes del Sistema Gestión Integral de Aduana.

Trabajo de Diploma para optar por el título de Ingeniero Informático.

Autor:

Yordani Cruz Segura.

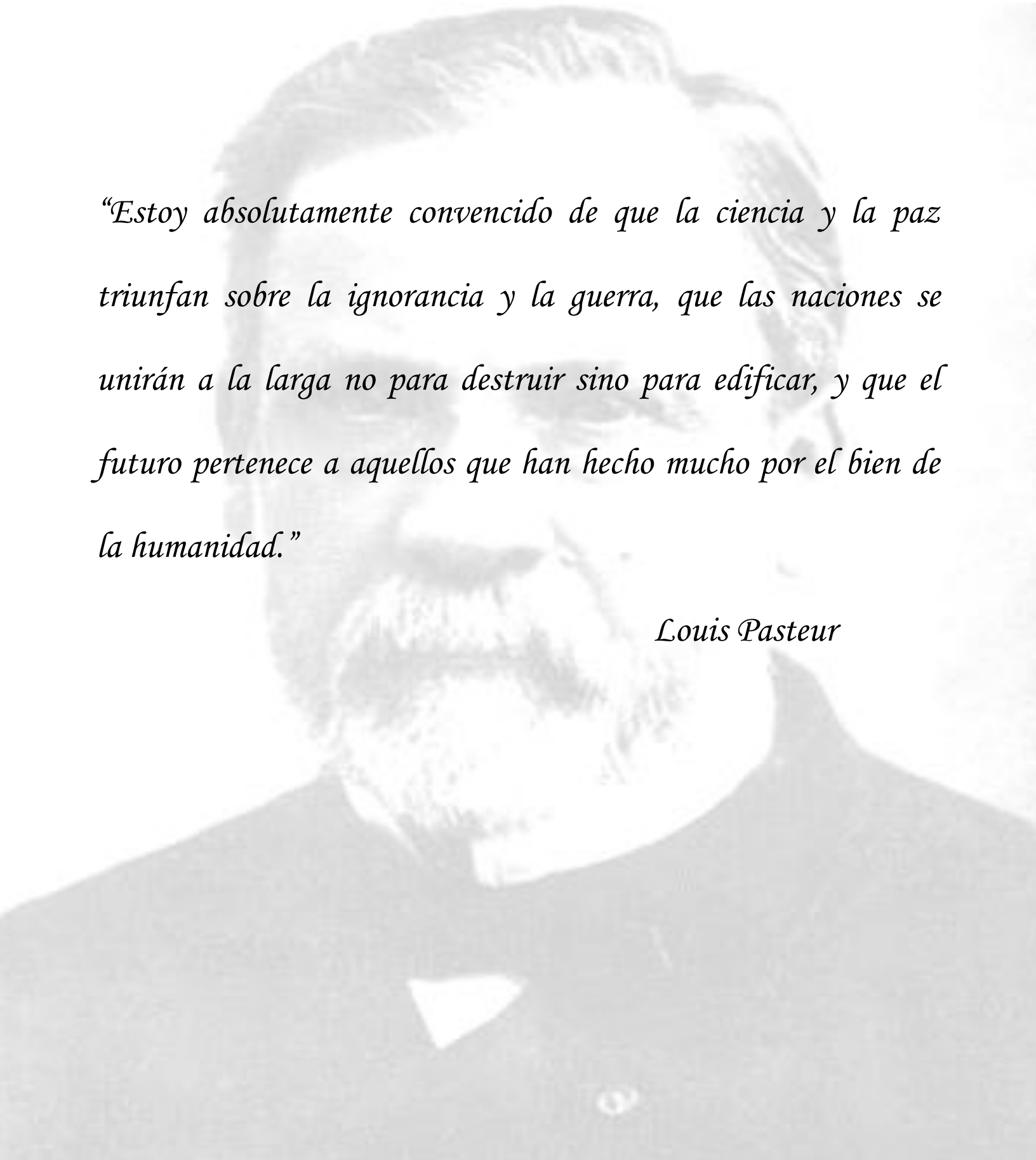
Tutores:

Ing. Julio Cesar Bravo Rodríguez.

Ing. Ernesto Daniel Vargas Allegue.

Ciudad de La Habana, junio 24 de 2011
"Año 53 de la Revolución"

Pensamiento



"Estoy absolutamente convencido de que la ciencia y la paz triunfan sobre la ignorancia y la guerra, que las naciones se unirán a la larga no para destruir sino para edificar, y que el futuro pertenece a aquellos que han hecho mucho por el bien de la humanidad."

Louis Pasteur

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Ing. Ernesto Daniel Vargas Allegue.

_____
Firma del Tutor

Ing. Julio Cesar Bravo Rodríguez

_____
Firma del Tutor

Yordani Cruz Segura

_____
Firma del Autor

Datos de Contacto

Ing. Julio Cesar Bravo Rodríguez**:**

Graduado de Ingeniero en Ciencias Informáticas por la UCI. Posee dos años de experiencia. Sin categoría docente. Jefe de proyecto del subsistema Despacho Comercial y diseñador principal del proyecto GINA.

Ing. Ernesto Daniel Vargas Allegue:

Graduado de Ingeniero en Ciencias Informáticas por la UCI. Posee dos años de experiencia. Cuenta con la categoría de instructor. Analista del subsistema Despacho Comercial.

Agradecimientos

En primer lugar a mis padres por todos los principios que me han inculcado desde pequeño y por confiar en mí y apoyar cada una de mis decisiones.

A mi hermana Odalis por ser una de las cosas más grandes en mi vida, por estar presente siempre que la he necesitado y ser como una madre para mí.

A mi hermano Pipo por toda su ayuda, su ejemplo y demostrarme su cariño a pesar de la lejanía.

A mis Tíos y Tías en especial a Esperanza por acogerme como un hijo más.

A mis primos Maité, Yasmani y Tito que han sido como mis hermanos.

A mi primo Edel por toda su ayuda, por sus consejos y confiar en mí.

A los vecinos y amigos del barrio por formar parte de mi familia.

A mis compañeros de aula que han sido mi familia más cercana en estos cinco años de carrera, por toda su confianza, su sencillez, su modestia y su ayuda.

A mis amistades de Venezuela: Lisbany, Deysi, Yaité, Belkis y Yadimir por acogerme como un miembro más de su familia.

A todos los amigos que conocí en la vocacional y a los que se han ido sumando en estos cinco años por todos los momentos agradables que hemos pasado.

Al todo el colectivo de profesores que han hecho posible mi formación.

Al piquete del proyecto que me ha ayudado muchísimo todo este tiempo.

A mi tutor Julio por ser un excelente líder y por su ayuda incondicional.

A Suly y a su madre por toda la ayuda que me han brindado desde que las conocí, por escuchar mis problemas y más que eso hacerlos suyos.

A todos muchas gracias

Dedicatoria

A mis padres por haber hecho hasta lo imposible para que yo obtuviera este resultado y me han demostrado que siempre se puede llegar un poquito más.

A mis hermanos, en especial a Pipo y a Odalis por aguantar mis malcriadeces desde pequeño y darme tanto cariño.

A mis sobrinos para que tomen como ejemplo todo el sacrifico depositado en este sueño y superen los resultados obtenidos.

Resumen

La Aduana General de la República tiene especial interés en incorporar al GINA (Sistema de Gestión Integral de Aduana) un módulo que sea capaz de gestionar las solicitudes que son realizadas por las diferentes entidades del país para importar y exportar mercancías; este proceso se ha venido desempeñando con mecanismos de gestión y control manual, trayendo como consecuencia que el proceso de análisis y aprobación de las solicitudes no se realice de manera oportuna y que exista un margen de errores en las mismas.

En el presente trabajo se abarcan los procesos de diseño e implementación del Módulo Solicitudes del Sistema GINA, lo cual constituye un paso importante para lograr una mayor eficiencia en la gestión de las solicitudes realizadas en el Proceso de Despacho Comercial de la Aduana General de la República. Cuenta con un estudio del arte de algunos sistemas que manejan este tipo de información. Se realiza una descripción de las herramientas y tecnologías utilizadas, dando paso a la propuesta de solución. En dicha propuesta se muestran cada uno de los artefactos generados durante el diseño y la implementación. Se logra medir la calidad del trabajo realizado aplicando métricas al diseño y pruebas funcionales a la aplicación obtenida.

Palabras Claves: Aduana, GINA, Gestión, Solicitudes.

Índice

Introducción **1**

Capítulo 1: Fundamentación Teórica **5**

1.1. Introducción 5
1.2. Sistemas Informáticos para gestionar solicitudes 5
1.3. Descripción de la Solución 7
1.4. Tendencias y Tecnologías actuales 8
1.4.1. Metodología de desarrollo 8
1.4.2. Metodologías y herramientas de diseño usadas en el mundo 12
1.4.2.1. Metodologías de Diseño 12
1.4.2.2. Patrones de Diseño 14
1.4.2.3. Herramientas de diseño 16
1.4.3. Patrones Arquitectónicos 17
1.5. Herramientas utilizadas para desarrollar el sistema 18
1.5.1. Lenguaje de Programación 18
1.5.2. Framework 20
1.5.2.1. Framework de PHP a utilizar 20
1.5.2.2. Framework de JavaScript a utilizar 21
1.5.3. Entorno de Desarrollo Integrado (IDE) 22
1.6. Sistema Gestor de Base de Datos a utilizar 23
1.7. Conclusiones Parciales 24

Capítulo 2: Diseño e Implementación del Sistema **25**

2.1. Introducción 25
2.2. Patrones de Diseño Utilizados 25
2.3. Modelo de Diseño 26
2.3.1. Diagrama de Paquetes 27
2.3.2. Diagrama de Clases del Diseño 28
2.3.3. Diagramas de Secuencia Orientado a Actividades del Negocio 30
2.3.4. Diagramas de Secuencia Orientado a Actividades de Componentes Visuales 31
2.4. Diseño de la Base de Datos 33

Índice

2.5. Métricas para la validación del diseño 35
2.5.1. Resultado de la evaluación de las métricas. 38
2.6. Modelo de Implementación....38
2.6.1. Estándar de Codificación 39
2.6.2. Tratamiento de Errores 40
2.6.3. Comunicación entre Capas....40
2.6.3.1. Ejemplo de Comunicación entre Capas 41
2.6.4. Diagrama de Componentes 48
2.7. Conclusiones Parciales 50

Capítulo 3: Validación de la Solución....51

3.1 Introducción....51
3.2 Pruebas de Software 51
3.2.1 Aplicación de Pruebas de Caja Negra 51
3.3 Impacto de la Solución Obtenida. 56
3.4 Conclusiones Parciales 58

Conclusiones 59

Recomendaciones 60

Bibliografía....61

ANEXOS 64

GLOSARIO....70

Introducción

Las tecnologías de la información actualmente son elementos fundamentales para la superación y desarrollo de un país. Por eso, los países desarrollados basan su crecimiento en la aplicación y la programación estratégica de las herramientas computacionales y han definido políticas que los inducirán a su permanencia en el dinamismo mundial de los próximos años.

En los últimos tiempos en Cuba se ha notado un incremento en cuanto a desarrollo de la informática. Con el surgimiento de varios programas de la Revolución y con ellos la Universidad de las Ciencias Informáticas, se han abierto las puertas al desarrollo de aplicaciones encaminadas a dar soluciones óptimas a muchos de los problemas existentes en diferentes empresas cubanas y del mundo. Potenciando así la industria de desarrollo de software y aprovechando el potencial de cada uno de los estudiantes y trabajadores que en ella laboran. Una vez surgida dicha universidad, una de las primeras entidades cubanas que solicita realizar un sistema en conjunto con la misma es la Aduana General de la República (AGR).

Esta entidad está implicada actualmente en un proceso de informatización que abarca la mayoría de sus sectores. Las aplicaciones informáticas que están siendo utilizadas hasta el momento forman parte en su conjunto, del Sistema de Gestión Integral de Aduana (GINA), que está caracterizado por manejar Información clasificada a diferentes niveles.

El Sistema GINA está integrado por varios subsistemas encargados de manejar los diferentes procesos que se desarrollan en la aduana. Aunque muchos de estos subsistemas ya están funcionando con un gran nivel de aceptación por parte de los trabajadores aduaneros, aún no existe ningún módulo que controle todas las solicitudes que son realizadas por las entidades a la AGR.

De acuerdo a la situación descrita anteriormente surge la siguiente **situación problemática**: En el Proceso de Despacho Comercial, perteneciente a la aduana se maneja la información referente a las solicitudes realizadas por las diferentes empresas tanto para realizar exportaciones como importaciones al país. El manejo de esta información representa una pérdida considerable de tiempo y evidencia un insuficiente control sobre la misma. La documentación no se encuentra almacenada en formato digital, lo que impide que el proceso de análisis y aprobación de estas se realice de forma oportuna ya que disminuye la velocidad de acceso a los datos y aumenta el riesgo de pérdida de la información. Después

de haber capturado los requisitos y hacer un análisis se identifica que aún no se satisfacen las necesidades de los clientes.

Ante esta situación se traza el siguiente **problema a resolver:** ¿Cómo satisfacer las necesidades de los clientes del Módulo Solicitudes del Sistema GINA a partir de los Requisitos Funcionales (RF) obtenidos durante la etapa de análisis?

En búsqueda de la solución al problema planteado se seleccionó como **objeto de estudio:** Los sistemas de gestión de solicitudes y como **campo de acción** se cuenta con el diseño y la implementación de los sistemas de gestión de solicitudes.

Teniendo como punto de partida los RF obtenidos durante el análisis el **objetivo general** de esta investigación es: Diseñar e implementar el Módulo Solicitudes del Sistema GINA de la Aduana General de la República para satisfacer las necesidades de los clientes.

Para darle cumplimiento al objetivo mencionado anteriormente se tendrán en cuenta las siguientes **tareas investigativas**:

- Estudiar el estado del arte de la presente investigación.
- Estudiar las herramientas para darle cumplimiento a los objetivos.
- Elaborar el modelo de datos del Módulo Solicitudes de Aduana.
- Elaborar el diagrama de clases del diseño del Módulo Solicitudes de Aduana.
- Elaborar los diagramas de secuencias orientados a actividades del Módulo Solicitudes de Aduana.
- Elaborar el diagrama de paquetes.
- Diseñar e implementar las interfaces del Módulo Solicitudes.
- Implementar las clases del negocio del Módulo Solicitudes.
- Elaborar el diagrama de componentes del Módulo Solicitudes.
- Realizar pruebas funcionales al sistema.
- Realizar la validación con los clientes de la solución obtenida.
- Documentar las pruebas realizadas.

Para facilitar el desarrollo de las tareas anteriores se utilizaron varios **métodos investigativos** que permiten el estudio de las características del objeto de investigación y la construcción de modelos para facilitar el desarrollo del trabajo. A continuación se exponen cada uno de los métodos utilizados en la presente investigación.

Introducción

Analítico: Consiste en un examen sistemático de un problema, por medio de un análisis de fuentes publicadas, llevado a cabo con un propósito específico en mente. (1) Este método es utilizado en la realización del estudio de los artefactos del análisis y las herramientas a utilizar en el Módulo Solicitudes de Aduana, además del estudio del arte del tema en cuestión.

El método histórico – lógico: Estudia los aspectos concretos en la trayectoria real de los fenómenos y acontecimientos en el decursar de su historia. (1) Para este caso es empleado en el estudio del arte del tema a investigar, pues de esta manera se puede conocer acerca de la existencia y características de sistemas de este tipo que hayan sido creados anteriormente.

Modelado: Es una reproducción simplificada de la realidad que cumple una función heurística, ya que permite descubrir y estudiar nuevas relaciones y cualidades del objeto de estudio. (1) La modelación es justamente el proceso mediante el cual creamos modelos con vistas a investigar la realidad y por su gran utilidad es el más empleado en esta investigación. Se utiliza a la hora de desarrollar los diferentes diagramas generados durante el diseño y la implementación del Módulo Solicitudes de Aduana.

Implementación: Se encarga de demostrar que la solución tiene ciertas propiedades o que se comporta de una manera en específico y para ello la implementación es comparada con la de otras aplicaciones existentes. (2) Este método unido al de modelado es uno de los más utilizados pues será empleado durante el proceso de implementación del modelo para obtener el sistema que controle las solicitudes realizadas a la aduana.

El presente trabajo se encuentra estructurado de la siguiente forma:

Capítulo 1. Fundamentación Teórica: En este capítulo se realizará un análisis del estado del arte de algunos sistemas que se utilizan en la gestión de las solicitudes tanto a nivel nacional como internacional. Se tienen en cuenta aspectos fundamentales para la comprensión del sistema que se desea modelar, así como los conceptos más importantes, las tendencias actuales, las técnicas y tecnologías, entre otros aspectos de vital importancia que sirven como base para el desarrollo de esta investigación.

Capítulo 2. Diseño e Implementación del sistema: En este capítulo se mostrarán los principales artefactos obtenidos tanto en la etapa del diseño como la implementación, entre los que se encuentran: los diagramas de clases del diseño, los diagramas de secuencia orientado a actividades, los diagramas de paquetes, así como el modelo de datos incluyendo la descripción de las tablas, estándares de codificación, diagramas de componentes y un ejemplo de comunicación entre las capas de la aplicación.

Introducción

Capítulo 3. Validación de la solución: En su contenido se publicará una descripción de cada una de las pruebas funcionales realizadas. Teniendo como objetivo de dichas pruebas, demostrar que el sistema se comporta de tal manera que da solución a la problemática planteada al iniciar la investigación. Se describe también el impacto que ha tenido la aplicación una vez que ha sido implantada.

Capítulo 1: Fundamentación Teórica

1.1. Introducción

Teniendo en cuenta las posibilidades existentes y el auge alcanzado en la industria de desarrollo del software, se hace necesario agilizar la gestión de las solicitudes en la aduana, pues cada día son más las empresas que se acercan a la misma para realizar este proceso.

En el siguiente capítulo se manejarán conceptos que son fundamentales para la creación del sistema que se desea desarrollar, además de las características de algunas aplicaciones de este tipo existentes en el mercado nacional e internacional y las principales herramientas que serán empleadas tanto en el proceso de diseño como durante la implementación.

1.2. Sistemas Informáticos para gestionar solicitudes

Actualmente existen varios sistemas informáticos que son usados por los diferentes organismos y empresas para gestionar varios tipos de solicitudes. Cada uno de estos sistemas se orienta específicamente a las características y necesidades de la entidad donde será utilizado. Entre estos se encuentran los siguientes:

Supportickets:

Es un programa de gestión de solicitudes de soporte y conocimientos desarrollado en el lenguaje PHP y como Sistema Gestor de Base de Datos cuenta con MySQL; el mismo facilita el proceso destinado a recibir solicitudes de soporte de los clientes. El sistema viene con tres interfaces fundamentales y presenta dos módulos de administración, uno para responder los pedidos, asignar permisos a los miembros del plantel y obtener estadísticas y el otro para solicitudes y correo electrónico. Por otra parte, proporciona métodos de gestión simples pero potentes para manejar varias clases de bases de conocimientos. El software también puede ser utilizado para categorizar y localizar documentos, artículos y datos con gran eficacia. Esto les permite a los usuarios mantener organizados los documentos, agregar adjuntos y notas, y descargar y guardar páginas web. Además puede construir fácilmente una base con los conocimientos de toda la organización. (3)

Entre las ventajas que brinda a los clientes se encuentra lo simple que resulta generar actualizaciones de solicitudes por medio de correo electrónico o formularios web en línea. De esta forma, sus técnicos podrán seguir fácilmente el progreso de las solicitudes de soporte de principio a fin, abarcando propiedad, transferencias y enrutamiento. (3)

Esta potente herramienta no resuelve el presente problema, la misma maneja un conjunto de información ajena a los objetivos de este trabajo como es el caso de la organización de documentos y las descargas de archivos. Además no presenta la seguridad suficiente que requieren las solicitudes en la aduana. Aunque el hecho de estar implementada sobre la tendencia de aplicaciones web, constituye una de las características que reafirman la decisión de seguir esta línea, pues los mejores sistemas de este tipo a nivel mundial se ejecutan sobre la web.

Gestor de Solicitudes de IKOM:

Es un software de implementación a la medida que integra toda la información necesaria para el manejo del servicio de solicitudes. Esta herramienta está desarrollada sobre Microsoft SharePoint Server 2007, utilizando de esta manera todos los beneficios de funcionalidad y escalabilidad de dicha plataforma. (4)

Ayuda además a la gestión y organización de las distintas tareas que aseguran una mejora continua de la calidad en la prestación del servicio solicitado. Se orienta a la gestión de solicitudes para áreas de mantenimiento de infraestructura, soporte informático y de abastecimiento. (4)

Esta solución simplifica y agiliza la gestión de la información de los procesos de soporte, mediante una administración centralizada de los documentos y la automatización de las tareas mediante el uso de *workflows*[1]. Adicionalmente, fortalece los canales de comunicación dentro de la organización, posibilitando que los propios usuarios sean los encargados de ingresar y consultar su información. Funciona sobre un ambiente Web y bajo una política de permisos de accesos definibles por la administración de red interna de cada compañía. (4)

Teniendo en cuenta las características de la anterior aplicación, se determina que la misma no puede formar parte de la solución. Esta presenta un carácter privativo lo que está en contra de las políticas de la Universidad de las Ciencias Informáticas (UCI) y del país de fomentar el software libre. Por otra parte no

[1] El workflow se define como un sistema informático que organiza y controla tareas, recursos y reglas, necesarias para completar el proceso de negocio.

está orientada a gestionar solicitudes como las que se manejan en la aduana, sino a las realizadas en las áreas de soporte, abastecimientos y otras.

Sistema de Certificación de Origen Digital de Cuba (SCOD-CUBA):

Es una aplicación web desarrollada en Cuba que se encarga de gestionar las solicitudes de Certificación de Origen Digital (COD) [2]y permite automatizar procesos para la confección de estas certificaciones. La misma está desarrollada en el lenguaje PHP 5 y como Sistema Gestor de Base de Datos utiliza MySQL. (5)

El SCOD-CUBA garantiza el correcto llenado de cada solicitud, mantiene un registro de solicitudes realizadas, posibilita la gestión digital entre la entidad comercializadora y la entidad certificadora, mantiene seguridad e integridad pertinente para cada entidad y proporciona autenticidad de la solicitud expedida. Otra de sus ventajas es que permite la integración de los COD a los sistemas informáticos de la aduana, contribuyendo así al proceso de digitalización de las operaciones aduaneras en su conjunto. (5)

Este sistema cubano, a pesar de tener cierta semejanza con la solución que se desea obtener no resuelve el problema que se plantea. El mismo está orientado específicamente a las solicitudes de COD y no responde a las funcionalidades de las solicitudes realizadas en el despacho comercial de la aduana.

1.3. Descripción de la Solución

El Departamento de Soluciones para la Aduana del Centro para la Informatización de la Gestión de Entidades (CEIGE), en conjunto con el equipo de informatización de la AGR se ha dado la tarea de desarrollar una solución informática que gestione todos los procesos que maneja la aduana cubana. El objetivo fundamental de la creación de la solución antes mencionada radica en eliminar la utilización de software privativo y ganar en la calidad de los servicios que presta esta importante entidad. Esta aplicación web se conoce por el nombre de GINA y se ha venido desarrollando desde el año 2004. La misma está compuesta por varios subsistemas orientados cada uno a un objetivo en específico, teniendo como característica principal la interacción existente entre cada uno de ellos. Una de las prioridades actuales que tiene este proyecto, es precisamente la creación de un módulo que se encargue de gestionar las solicitudes que son realizadas por las diferentes empresas a la aduana, con el propósito de importar o exportar mercancías al país. Este proceso se realiza en el departamento de despacho comercial de las

[2]El Certificado de Origen Digital (COD) es un formulario oficial que debe acompañar la mercancía en su exportación.

diferentes entidades aduaneras. El mismo les permitirá a los agentes de dicha entidad introducir los datos de las solicitudes mediante una interfaz web y posteriormente les dará la posibilidad de aceptar o rechazar la solicitud si esta cumple con las restricciones y leyes impuestas por la AGR, además de brindarle algunos reportes como es el caso del historial del declarante.

1.4. Tendencias y Tecnologías actuales

Con el crecimiento y emancipación del internet se incrementa la tendencia al desarrollo de aplicaciones web por las ventajas que trae una solución de este tipo; por ejemplo, simplifica el código, reduce el tamaño de los archivos y el tiempo de desarrollo. También elimina la necesidad de mantener un contacto físico con los clientes por parte de las empresas que utilizan este tipo de tecnología, los mismos pueden acceder a los servicios web desde cualquier parte del mundo siempre y cuando exista conectividad.

Teniendo en cuenta la revolución tecnológica existente en las últimas décadas y el sin número de herramientas de desarrollo y tecnologías que esta ha desencadenado, a continuación se exponen varios elementos de cada una de las herramientas y metodologías que serán utilizadas en el desarrollo del presente trabajo. La selección de las herramientas mostradas a continuación, no forma parte de los objetivos de la investigación; pues fueron escogidas por el Departamento de Soluciones para la Aduana del CEIGE para el desarrollo de sus aplicaciones.

1.4.1. Metodología de desarrollo

El Proceso Unificado de Rational (RUP) constituye una guía rectora que decide quién hace, qué, cuándo y cómo lo hace. RUP dirige las actividades que se realizan por rol, les da un orden, define qué artefactos deberían ser desarrollados y puede además monitorear y medir los productos y actividades de un proyecto, teniendo en cuenta la calidad del producto final. RUP se caracteriza por dividir el ciclo de vida de la producción del software en cuatro fases:

- **Inicio o Conceptualización:** es donde se determina la visión del proyecto, o sea se comprende el entorno y se determina el alcance del producto. (6)
- **Elaboración:** en esta etapa se determinan los cimientos de la arquitectura y se analiza el dominio del problema. (6)
- **Construcción:** en esta fase se obtiene la capacidad operacional inicial del producto. (6)
- **Transición:** Se obtiene liberación del producto y se pone en manos de los usuarios finales. (6)

Además de esto cuenta con nueve flujos de trabajo, seis de ingeniería y tres de soporte los cuales son: Modelamiento del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba, Despliegue, Gestión de Configuración y Cambios, Gestión de Proyectos y por último Entorno, donde los 3 últimos constituyen los flujos de soporte. La metodología mencionada anteriormente puede especializarse para una gran variedad de sistemas, para diferentes áreas de aplicaciones, diferentes tipos de organizaciones, diferentes niveles de amplitud y diferentes tamaños de proyectos.

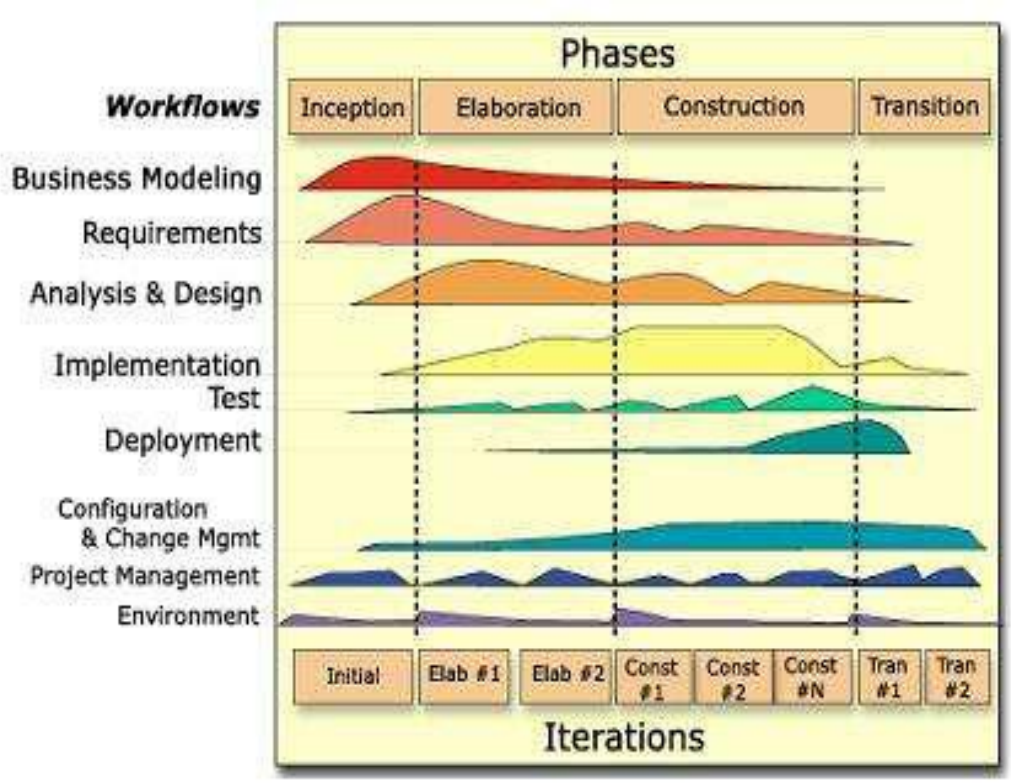


Figura 1.1. Fases e iteraciones de la Metodología RUP.

Conociendo que RUP es una metodología pesada y adaptable, en el Departamento de Soluciones para la Aduana, del cual el presente trabajo forma parte, se le realizaron algunas adaptaciones. Teniendo como objetivo la creación de un modelo que sea capaz de guiar todo el proceso de desarrollo en cada uno de los subsistemas del departamento. Entre las adaptaciones definidas se encuentran las siguientes:

- En este caso no es guiado por Casos de Uso sino por Requisitos Funcionales.

- En lugar de elaborar el clásico diagrama de secuencia se modela un diagrama de secuencia orientado a actividades.
- El modelado del negocio es orientado a procesos y como notación se utiliza BPMN.
- Sólo se generan los artefactos que se consideran necesarios en el proyecto.

Como se mencionaba anteriormente uno de los aspectos que caracteriza al modelo utilizado en el desarrollo de la aplicación es precisamente que se generan sólo los artefactos necesarios. A continuación se expone una breve explicación de cada uno de los artefactos definidos durante la etapa de diseño e implementación:

Diagrama de Paquetes:

Los diagramas se utilizan para representar diferentes perspectivas de un sistema de forma que un diagrama es una proyección del mismo. Existen varios tipos de diagramas que son generados en las distintas etapas por las que transita el desarrollo de un software y entre ellos se encuentra el diagrama de paquetes. (7)

El diagrama de paquetes permite dividir al sistema orientado a objetos organizándolo en subsistemas y detallando sus relaciones. Los usos más comunes para los diagramas de paquete radican en organizar diagramas de casos de uso y diagramas de clase, a pesar de que el uso de los diagramas de paquete no es limitado a estos elementos UML. (7)

Estos diagramas están compuestos por dos elementos fundamentales:

- Paquetes: es una agrupación de elementos que pueden ser clases, casos de uso o componentes y permite anidar paquetes entre sí. Se representa mediante un símbolo en forma de carpeta, el nombre se le coloca en la pestaña y el contenido dentro de la carpeta. (7)
- Dependencias: indica que un elemento de un paquete requiere a otro de un paquete distinto y se representa con una línea discontinua partiendo del paquete dependiente. (7)

Diagrama de Clases del Diseño:

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. (8)

Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema. Incluye modelar el vocabulario del sistema, las colaboraciones o esquemas. Los diagramas de clases también son la base para un par de diagramas relacionados como son los diagramas de componentes y los diagramas de despliegue. Los diagramas de clases son importantes no sólo para visualizar, especificar y documentar modelos estructurales, sino también para construir sistemas ejecutables aplicando ingeniería directa e inversa. (8)

Diagrama de Secuencia Orientado a Actividades:

Los Diagramas de Secuencia Orientado a Actividades son utilizados en el Departamento de Soluciones para la Aduana, con el propósito de orientar el diseño lo mejor posible a la etapa de implementación. Garantizando que este último proceso se realice con una mayor rapidez y calidad.

El mismo es considerado un híbrido entre los diagramas de actividades y los de secuencias propuestos por RUP en la etapa de diseño. Este diagrama cuenta con calles, actividades y sus relaciones, además permite agregar comentarios de manera que se esclarezcan los procesos.

En la presente solución se manejarán dos tipos de Diagramas de Secuencia Orientado a Actividades, uno dedicado al negocio y otro a la interfaz teniendo siempre presente que debe existir una correspondencia entre ambos.

Modelo de Datos:

Un modelo de datos es aquel que aporta la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos, así como la base formal para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información. Con respecto al diseño de bases de datos, el modelo de datos puede ser descrito mediante los requerimientos de información y proceso de una aplicación de uso intensivo de datos (por ejemplo, un sistema de información). Por otra parte se puede construir una representación de la aplicación que capture las propiedades estáticas y dinámicas requeridas para dar soporte a los procesos deseados (por ejemplo, transacciones y consultas). Además de capturar las necesidades dadas en el momento de la etapa de diseño, la representación debe ser capaz de dar cabida a eventuales futuros requerimientos. (9)

Diagrama de Componentes:

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. (7)

El diagrama de componentes es muy parecido al diagrama de casos de uso siendo utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. En él se situarán librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema. Uno de los usos principales es que puede servir para ver qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

1.4.2. Metodologías y herramientas de diseño usadas en el mundo

La evolución del diseño como parte del proceso de desarrollo de software es un proceso continuo que se ha ido produciendo durante las últimas 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 estos en una definición de diseño. Nuevos enfoques para el diseño proponen un método orientado a objetos para la obtención del mismo.

Como resultado de un largo proceso evolutivo por parte del diseño de software surgen un conjunto de metodologías que en dependencia de las características del producto que se desee desarrollar, se pueden aplicar de forma unitaria o de manera combinada.

1.4.2.1. Metodologías de Diseño

- Diseño Axiomático: Se pone en práctica cuando se emplean axiomas para tomar decisiones. Este enfoque se basa principalmente en las necesidades funcionales del cliente. (10)

- Diseño Orientado al Uso: En este caso el diseño se centra en las tareas asociadas al uso y sus objetivos. Este enfoque penaliza la usabilidad, ya que incrementa la curva de aprendizaje, pero logra solventar problemas complejos o de alta criticidad. (10)

- Diseño Centrado en el Usuario: Es la metodología de diseño más habitual en el mundo de desarrollo de software. Este enfoque coloca todas las necesidades, deseos y limitaciones del usuario como núcleo del proceso de diseño. Por lo cual esta metodología conlleva a una definición, investigación y análisis del usuario. Así mismo implica que el proyecto debe ser dividido por fases para garantizar los resultados. Dentro del diseño centrado en el usuario hay metodologías específicas, como son KES (*Kansei Engineering System*) o QFD (*Quality Function Deployment*). (10)

- Enfoque Ascendente: Consiste en partir de un elemento o de las características individuales de los elementos para desarrollar la totalidad del producto. (10)

- Enfoque Descendente: Consistente en generar el producto desconociendo los elementos que lo configuran, esta técnica se emplea para desarrollar por ejemplo plantillas web[3]. (10)

- Pensamiento de diseño: Es una metodología de resolución de problemas, y de detección y descubrimiento de nuevas oportunidades, se basa en un profundo conocimiento antropológico, en el intercambio de ideas entre equipos multidisciplinares y en los tests. (10)

- Desarrollo Modular: Su funcionamiento se basa en una descomposición de la programación en fracciones lógicas y manejables. Este tipo de programación se apega bien al diseño descendente porque enfatiza las interfaces entre los módulos, más que mantenerlas ignoradas hasta el final del desarrollo del sistema. Cada módulo debe ser funcionalmente cohesivo, de manera que satisfaga sólo una función. (10)

Algunos principios que pueden resultar de ayuda a la hora de tomar decisiones son:

- Navaja de Ockham: Consiste en eliminar los elementos innecesarios del diseño con el fin de obtener productos lo más sencillos posibles, de modo que se reduce la posibilidad de que existan incoherencias. (10)

- Principio KISS (*Keep It Short and Simple* o *Keep it simple, stupid*): Se emplea cuando la sencillez es un objetivo del diseño. El principio Kiss no es en sí mismo una metodología sino una estrategia de diseño y sigue el reduccionismo metodológico de Ockham. (10)

[3] Las plantillas web son utilizadas para crear páginas con un diseño general.

- Timtowtdi, (*There is more tan one way to do it*): Este principio hace referencia a que existen múltiples soluciones a un problema, es un principio muy popular en programación. (10)

Luego de realizar un profundo estudio de las metodologías y principios de diseños mencionados anteriormente y teniendo en cuenta las características de la aplicación que se desea desarrollar, se decidió usar en la presente investigación como metodologías de diseño: el Diseño Centrado en el Usuario, el Enfoque Ascendente y el Desarrollo Modular.

1.4.2.2. Patrones de Diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan. (11)

Los patrones de diseño no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables. Han revolucionado el diseño orientado a objetos y es muy importante su conocimiento por parte de todos los desarrolladores de software. (11)

Existen varios tipos de patrones de diseño, a continuación se muestran algunos de los empleados en el desarrollo de esta solución; que son implementados por el *framework*[4] utilizado, el que se explicará en epígrafes posteriores:

EXPERTO: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase contiene toda la información necesaria para realizar la labor que tiene encomendada. (12)

CREADOR: Este patrón, como su nombre lo indica, es el que crea, él guía la asignación de responsabilidades relacionadas con la creación de objetos, se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando:

- B contiene A
- B es una agregación (o composición) de A
- B almacena A

[4] Un framework es un conjunto de clases o estructuras que implementan los componentes de una aplicación genérica.

- B tiene los datos de inicialización de A (datos que requiere su constructor)
- B usa A. (12)

A la hora de crear objetos se deben tener en cuenta las características de la clase.

BAJO ACOPLAMIENTO: El acoplamiento es una medida de fuerza con que un elemento está a, tiene conocimiento de, confía en, otros elementos. Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. (12)

ALTA COHESION: La cohesión es una medida de la fuerza con la que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo. (12)

CONTROLADOR: Es un evento generado por actores externos. Se asocian con operaciones del sistema como respuestas a los eventos del mismo, tal como se relacionan los mensajes y los métodos. Normalmente un controlador delega en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad. No realiza mucho trabajo por sí mismo. (12)

DECORADOR: Responde a la necesidad de añadir dinámicamente funcionalidad a un objeto. Esto permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera. Este patrón puede ser utilizado con varios fines como los que se muestran a continuación:

- Añadir objetos individuales de forma dinámica y transparente.
- Responsabilidades de un objeto pueden ser retiradas.
- Cuando la extensión mediante la herencia no es viable.
- Hay una necesidad de extender la funcionalidad de una clase, pero no hay razones para extenderla a través de la herencia.
- Hay la necesidad de extender dinámicamente la funcionalidad de un objeto y quizás quitar la funcionalidad extendida. (12)

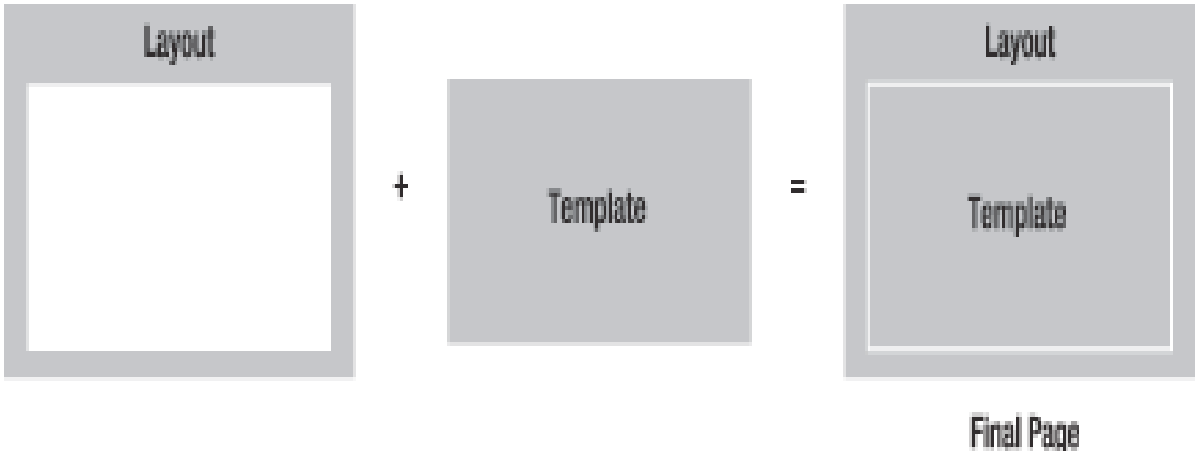


Figura 1.2 Ejemplo de Implementación del patrón Decorador en *Symfony*.

1.4.2.3. Herramientas de diseño

Las herramientas no son más que un recurso que se emplea para realizar una actividad o un trabajo. En este contexto vendría siendo una aplicación que se utiliza para la elaboración de un programa u otra aplicación.

Existen varios tipos de herramientas entre las que se encuentran las llamadas Herramientas CASE. Estas no son más que un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de Desarrollo de un Software. (13) A continuación se presentan algunas ventajas y características de la Herramientas CASE usada durante el diseño de la solución.

Visual Paradigm:

El Visual Paradigm para UML (Lenguaje de Modelado Unificado) es una Herramienta Case Cruzado de Ciclo de Vida. La misma soporta las últimas versiones del lenguaje UML y la Notación y Modelado de Procesos de Negocios. Además de proveer el modelado de negocios, cuenta con un generador de mapeo de objetos relacionales para los lenguajes de programación Java, .NET y PHP. (14)

Entre las principales características de esta herramienta se encuentran las siguientes:

- Proporciona valiosa ayuda a las personas que la utilizan visualizando, comunicando y aplicando sus diseños. (14)

- Genera un entorno de modelados visuales en el que se reúnen hoy todas las necesidades tanto de software y tecnología, como de comunicación. (14)
- Compatible hasta con 10 lenguajes. (14)
- Elevada interoperabilidad con otras aplicaciones. (14)
- Soporte a varios lenguajes en generación de código e ingeniería inversa a través de plataformas java. (14)
- Soporte de UML versión 2.1. (14)
- Sincronización entre diagramas de entidad-relación y diagramas de clases. (14)

1.4.3. Patrones Arquitectónicos

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular, el cual es recurrente dentro de un cierto contexto. Este esquema se especifica describiendo los componentes, con sus responsabilidades y relaciones. (15)

Modelo Vista Controlador (MVC): Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. *Symfony* está basado en un patrón clásico del diseño web conocido como arquitectura MVC, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el controlador es el Sistema de Gestión de Base de Datos. Este patrón divide una aplicación interactiva en tres áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. (16)
- La vista transforma el modelo en una página web que permite al usuario interactuar con ella. (16)
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. (16)

En la figura 1.2, se muestra la implementación del patrón MVC en *Symfony*, un *framework* de desarrollo que se estará explicando más adelante.

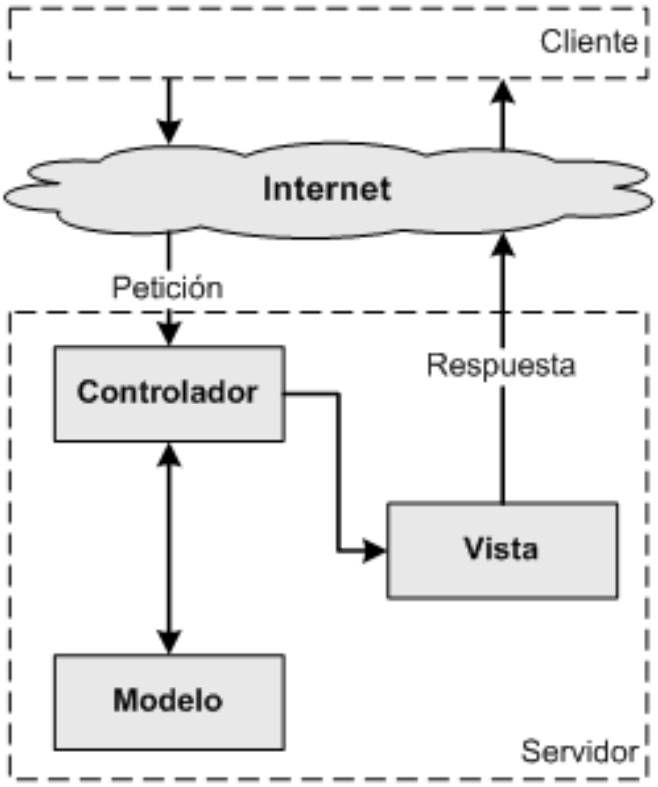


Figura 1.3. Implementación del Patrón MVC en *Symfony*.

1.5. Herramientas utilizadas para desarrollar el sistema

Con la tendencia al aumento del desarrollo de aplicaciones web son muchas las herramientas que han surgido encaminadas a resolver problemas determinados en este proceso. Cada una de ellas con sus características específicas y enmarcadas en un entorno determinado que proporciona una mayor variabilidad de ofertas. Las herramientas de desarrollo empleadas en este trabajo fueron las seleccionadas por el Departamento de Soluciones para la Aduana del CEIGE.

1.5.1. Lenguaje de Programación

En los últimos tiempos han surgido varios lenguajes de programación, que por sus características y adaptabilidad presentan un alto grado de aceptación por parte de los desarrolladores de sistemas a nivel mundial. Entre estos lenguajes se cuenta con algunos como Java; PHP 5 y .NET por mencionar algunos de los más utilizados en el desarrollo de aplicaciones de este tipo.

Otras de las tendencias actuales en cuanto a lenguajes de programación se refiere, es precisamente la programación orientada a objetos (POO). Siendo esta tendencia una técnica que se enfoca en los datos (objetos) y en la manera de llegar a ellos (interfaces), no en las herramientas que se utilizan para manejarlos, a fin de crear módulos que interactúen entre sí para lograr el objetivo del programa.

Como se mencionaba anteriormente uno de los lenguajes más utilizados para el desarrollo de aplicaciones web es el PHP.

PHP (acrónimo de *PHP: Hypertext Preprocessor*) es un lenguaje de código abierto y del lado del servidor lo que implica la necesidad de un servidor web para poder visualizar el contenido; el mismo es muy popular especialmente adecuado para desarrollo web y puede ser incrustado en HTML de manera transparente para los usuarios. (17)

Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas web dinámicas:

- Soporte para una gran cantidad de bases de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, SybasemSQL, Informix, entre otras. (18)
- Integración con varias bibliotecas externas, pues permite generar documentos en PDF (documentos de Acrobat Reader) y analizar código XML. (18)
- Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes. (18)
- Soportado por una gran comunidad de desarrolladores, como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente. (18)
- El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP. (18)
- Es un lenguaje versátil ya que puede usarse con la mayoría de sistemas operativos, ya sea basados en UNIX (Linux, Solares, FreeBSD...), como con Windows, el sistema operativo de Microsoft. (18)
- Con PHP se puede hacer cualquier cosa que se pueda realizar con un script CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas. (18)

- Posee una enorme eficiencia, pues con escaso mantenimiento y un servidor gratuito (en nuestro caso, Apache), puede soportar sin problema miles de conexiones simultáneas. (18)

1.5.2. Framework

Un *Framework* es un conjunto de clases o estructuras que implementan los componentes de una aplicación genérica, así como también componentes concretos que cumplen a cabalidad tareas concretas. Para desarrollar programas completos, los desarrolladores buscan e instancian los componentes apropiados. Sin lugar a dudas estos simplifican el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporcionan estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. También facilitan la programación de aplicaciones, ya que encapsulan operaciones complejas en instrucciones sencillas. (16)

1.5.2.1. Framework de PHP a utilizar

Actualmente existen varios *framework* de PHP que facilitan el desarrollo de las aplicaciones web que utilizan este potente lenguaje, entre ellos se encuentra *Symfony*.

Symfony es un completo *framework* desarrollado en PHP 5 y es compatible con la mayoría de los gestores de bases de datos existentes en el mundo, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft, el mismo está diseñado para optimizar el desarrollo de las aplicaciones web y se puede ejecutar tanto en plataformas *nix (Unix, Linux,etc.) como en plataformas Windows. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. A continuación se muestran algunas de las características de este *framework*. (16)

Características de *Symfony*:

- Es un *framework* muy amplio, e incluye un verdadero ORM, de nombre Propel, que es otro proyecto de código abierto y, probablemente, una de las mejores soluciones ORM para PHP. (16)
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo. (16)

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares). (16)
- Independiente del sistema gestor de bases de datos. (16)
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos. (16)
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador sólo debe configurar aquello que no es convencional. (16)
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web. (16)
- Preparado para aplicación empresarial y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo. (16)
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros. (16)

En el desarrollo de la solución se utilizará el *Framework* de PHP *Symfony* en su versión 1.2.8, teniendo en cuenta su superioridad ante los demás *framework* de PHP existentes en la actualidad para el desarrollo de grandes aplicaciones.

1.5.2.2. Framework de JavaScript a utilizar

Ext JS es un *Framework* de JavaScript que permite realizar aplicaciones web enriquecidas basándose en tecnologías AJAX, JSON, DHTML y DOM. Ext JS está patentado bajo licencia LGPL[5] lo que posibilita su uso para aplicaciones empresariales privativas de código cerrado. (19)

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

5 La licencia LGPL plantea que el software protegido por la misma puede ser libremente modificado, copiado, distribuido y utilizado libremente incluso junto a un software que no sea libre.

La utilización de un *Framework* de JavaScript como Ext JS facilita la separación de las capas de la vista con la del controlador desde el punto de vista productivo ya que el código utilizado en la primera es solamente JavaScript y no es necesario utilizar ningún tipo de código PHP, así los desarrolladores pueden centrarse más en el aprendizaje de un solo lenguaje. Además al soportar serialización de objetos mediante tecnología JSON, permite que los datos enviados desde el controlador como respuesta a la vista contengan sólo las propiedades de dichos objetos, pero no el comportamiento, minimizando los posibles errores de programación y los accidentes de que los objetos sean modificados erróneamente desde la vista. (19)

1.5.3. Entorno de Desarrollo Integrado (IDE)

Un Entorno de Desarrollo Integrado (en inglés *Integrated Development Environment* o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. (20)

Actualmente los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado (por ejemplo C, C++, C#, Java, Python y PHP entre otros). Brindando la posibilidad de que un mismo entorno de desarrollo tenga la posibilidad de utilizar varios lenguajes de programación, como es el caso del NetBeans.

NetBeans IDE

NetBeans IDE es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE, además es un producto libre y gratuito sin restricciones de uso. (22) Posee un amplio soporte para el lenguaje PHP así como para los *framework* de trabajo *Symfony* y Ext JS.

Para el desarrollo de la solución se ha elegido NetBeans IDE, en su versión 6.9, como Entorno de Desarrollo Integrado.

1.6. Sistema Gestor de Base de Datos a utilizar

Un Sistema Gestor de Base de Datos (SGBD) se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. Estos ayudan a realizar las siguientes acciones (21):

- Definición de los datos
- Mantenimiento de la integridad de los datos dentro de la base de datos
- Control de la seguridad y privacidad de los datos
- Manipulación de los datos

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

Los tipos de objetos de Oracle son tipos de datos definidos por el usuario que permiten modelar entidades complejas del mundo real en una estructura que trata cada entidad como una unidad atómica simple en la base de datos. (21)

Entre las principales ventajas de Oracle se destacan las siguientes:

- Las entidades complejas del mundo real y la lógica se pueden modelar fácilmente, lo que permite reutilizar objetos para el desarrollo de base de datos de una forma más rápida y con mayor eficiencia. (22)
- Los programadores de aplicaciones pueden acceder directamente a tipos de objetos Oracle, sin necesidad de ninguna capa adicional entre la base de datos y la capa cliente. (22)
- Las aplicaciones que utilizan objetos de Oracle son fáciles de entender y mantener porque soportan las características del paradigma orientado a objetos. (22)
- Tiene buen rendimiento y hace buen uso de los recursos. (22)
- Posee un rico diccionario de datos. (22)
- Brinda soporte a la mayoría de los lenguajes de programación. (22)
- Es un sistema multiplataforma, disponible en Windows, Linux y Unix. (22)

- Permite tener copias de la base de datos productiva en lugares lejanos a la ubicación principal. Las copias de la Base de Datos productiva pueden estar en modo de lectura solamente. (22)

Como desventaja este SGBD presenta las siguientes:

- Es un producto de elevado precio por lo que por lo general se utiliza en empresas muy grandes y multinacionales (22)
- Los costos de soporte técnico y mantenimiento son elevados. (22)
- Vulnerabilidades en la seguridad de la plataforma, se hace necesario aplicar parches de seguridad. (22)

1.7. Conclusiones Parciales

- La investigación realizada sobre los sistemas que gestionan solicitudes seleccionados en el presente trabajo ayudó a definir que ninguno responde a las necesidades de la AGR, determinándose que no pueden formar parte íntegramente de la solución.

- El estudio de las metodologías de diseño existente posibilitó escoger como guía durante esta etapa el enfoque ascendente, el desarrollo modular y el diseño centrado en el usuario. Teniendo en cuenta para la selección anterior las características de la aplicación que se desea desarrollar.

- Durante el diseño y la implementación las herramientas y tecnologías utilizadas serán las escogidas por el Departamento de Soluciones para la Aduana del CEIGE.

Capítulo 2: Diseño e Implementación del Sistema

2.1. Introducción

En el presente capítulo se desarrollan los flujos de trabajo de diseño e implementación. Teniendo como objetivos fundamentales la creación del modelo de diseño incluyendo el diagrama de clases del negocio, el diagrama de paquetes, el diagrama de secuencia de actividades de cada uno de los RF y las métricas aplicadas para evaluar el trabajo realizado, garantizando que el diseño esté lo más orientado posible a la programación y asegurando de esta forma una mayor rapidez y calidad en este proceso. También se mostrará el estándar de codificación empleado, el tratamiento de errores, el diagrama de componentes y un ejemplo de comunicación entre las diferentes capas con las que cuenta la solución.

2.2. Patrones de Diseño Utilizados

El desarrollo del sistema empleando el *framework Symfony* proporciona ventajas significativas para los desarrolladores de software. El marco de trabajo mencionado es capaz de fusionar buenas prácticas de trabajo por sí mismo, de forma que los desarrolladores no tengan que preocuparse por implementar varios de los patrones de diseño y arquitectónicos más utilizados en la actualidad, ya que el mismo *framework* los maneja. A continuación se exponen varios patrones de diseño que han sido utilizados durante el desarrollo de la solución:

Creador

La clase solicitudActions contiene las acciones definidas para el Módulo Solicitudes y es en ella misma donde se ejecutan las funciones que dan vida al sistema. 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 solicitudActions es el “creador” de las entidades.

Experto

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

Alta Cohesión

Symfony permite la asignación de responsabilidades con una alta cohesión, por ejemplo la clase SolicitudActions tiene la responsabilidad para definir las acciones sobre las plantillas y colabora con otras para realizar diferentes operaciones e instanciar objetos, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas, proporcionando que el sistema sea flexible frente a grandes cambios.

Controlador

Todas las peticiones Web son manejadas por un solo controlador frontal (dc_dev.php), siendo 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.

Bajo Acoplamiento

En el diseño de las clases se aplicó este patrón logrando un modelo con las relaciones necesarias que impulsan la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que lleve a resultados negativos. De esta forma se tiene un diseño de clases más independiente que reduce el impacto al cambio.

Decorador

Añade funcionalidad a una clase dinámicamente. El archivo layout.php, denominado también plantilla global, almacena el código HTML 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, o si se mira desde el otro punto de vista, el layout decora la plantilla.

2.3. Modelo de Diseño

El Modelo de Diseño es un modelo de objetos que describe la realización física de los casos de uso, especificando cómo los RF y no funcionales tienen impacto en el sistema. (23) En este modelo se utilizan muchos de los diagramas UML aplicados en el modelo de análisis. La diferencia radica en que estos diagramas están refinados y elaborados como parte del diseño proporcionando un mayor detalle para la implementación.

Capítulo 2: Diseño e Implementación del Sistema

Como punto de partida para la realización del presente modelo de diseño se cuenta con un conjunto de Requisitos Funcionales, los cuales se encuentran descritos en el documento: “Especificación de Requisitos Funcionales” del Módulo Solicitudes. A continuación se exponen cada uno de los RF que rigen la investigación:

- Solicitar Facilidad.
- Solicitar Liberación del Cotejo.
- Solicitar Régimen Temporal.
- Gestionar Solicitud de Facilidad.
- Gestionar Solicitud de Liberación del Cotejo.
- Gestionar Solicitud de Régimen Temporal.
- Consultar Historial del Declarante.

2.3.1. Diagrama de Paquetes

El diagrama de paquetes (figura 2.1) de la presente solución cuenta con un paquete fundamental llamado Solicitudes. El paquete mencionado anteriormente, está constituido por 4 paquetes internos; entre los que se encuentran: el paquete JS (hace referencia a las clases .js pertenecientes a las interfaces visuales de la aplicación), el paquete Controlador (contiene el controlador frontal, que para el caso de esta solución es *dc_dev.php)*, el paquete Lib (contiene todas las clases del negocio de la solución) y el paquete Config (constituido por los ficheros *propel.ini* y *database.yml*). Externamente el paquete Solicitudes está relacionado con cinco paquetes: el paquete TC (perteneciente al Subsistema Tablas de Control del Sistema GINA), el paquete Persona (perteneciente al Subsistema Persona del Sistema GINA), el paquete Contacto (representa al Subsistema Contacto del Sistema GINA), el paquete *Symfony* Lib (representa las librerías del *Framework Symfony*) y *Symfony* Core (representa el núcleo del *framewok* utilizado).

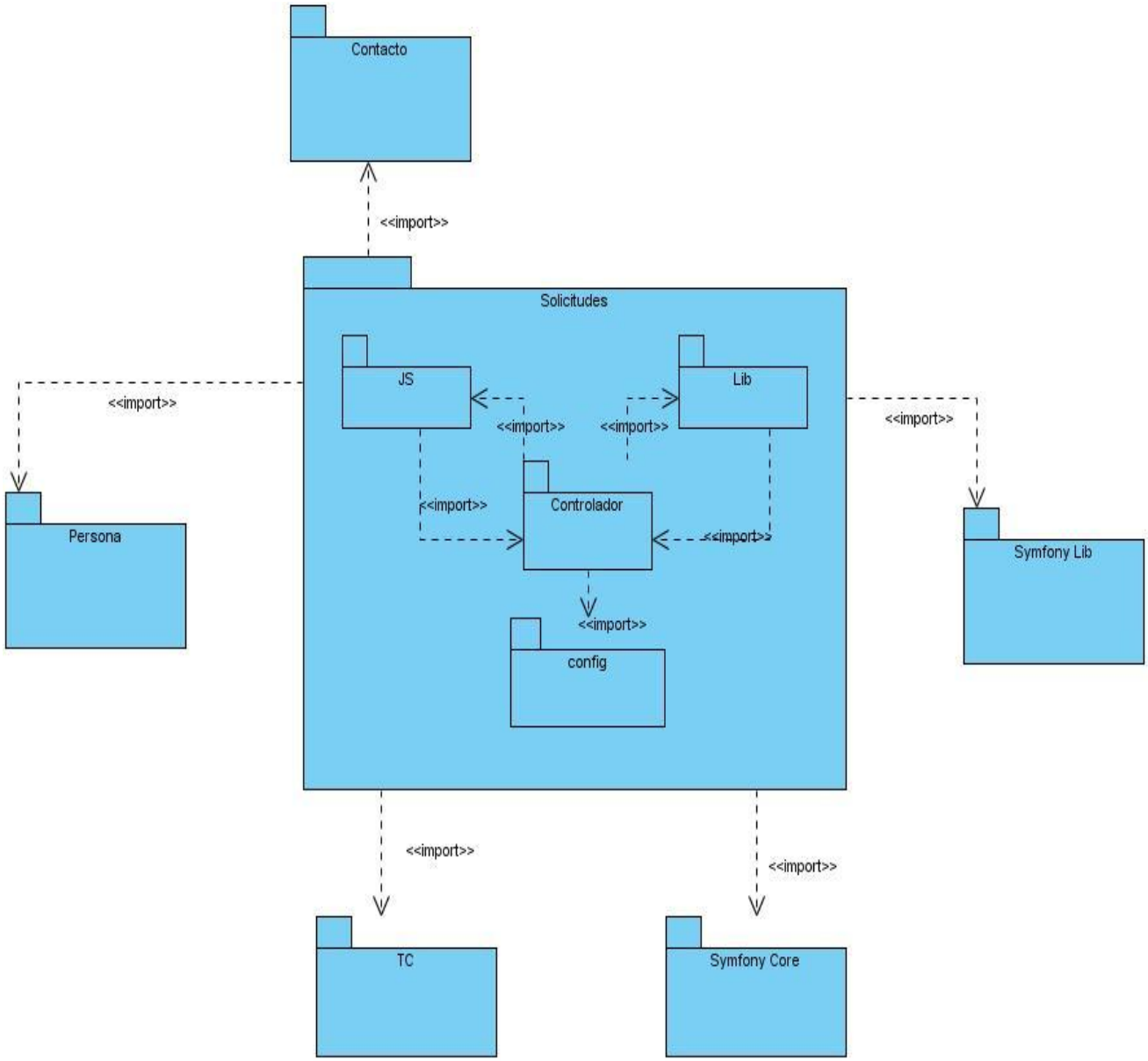


Figura 2.1. Diagrama de Paquetes del Módulo Solicitudes.

2.3.2. Diagrama de Clases del Diseño

Entre los artefactos generados durante el modelo del diseño se encuentra el diagrama de clases. Como parte de la presente investigación, en la figura 2.2 se muestra el diagrama de clases del diseño perteneciente al Módulo Solicitudes del Sistema GINA.

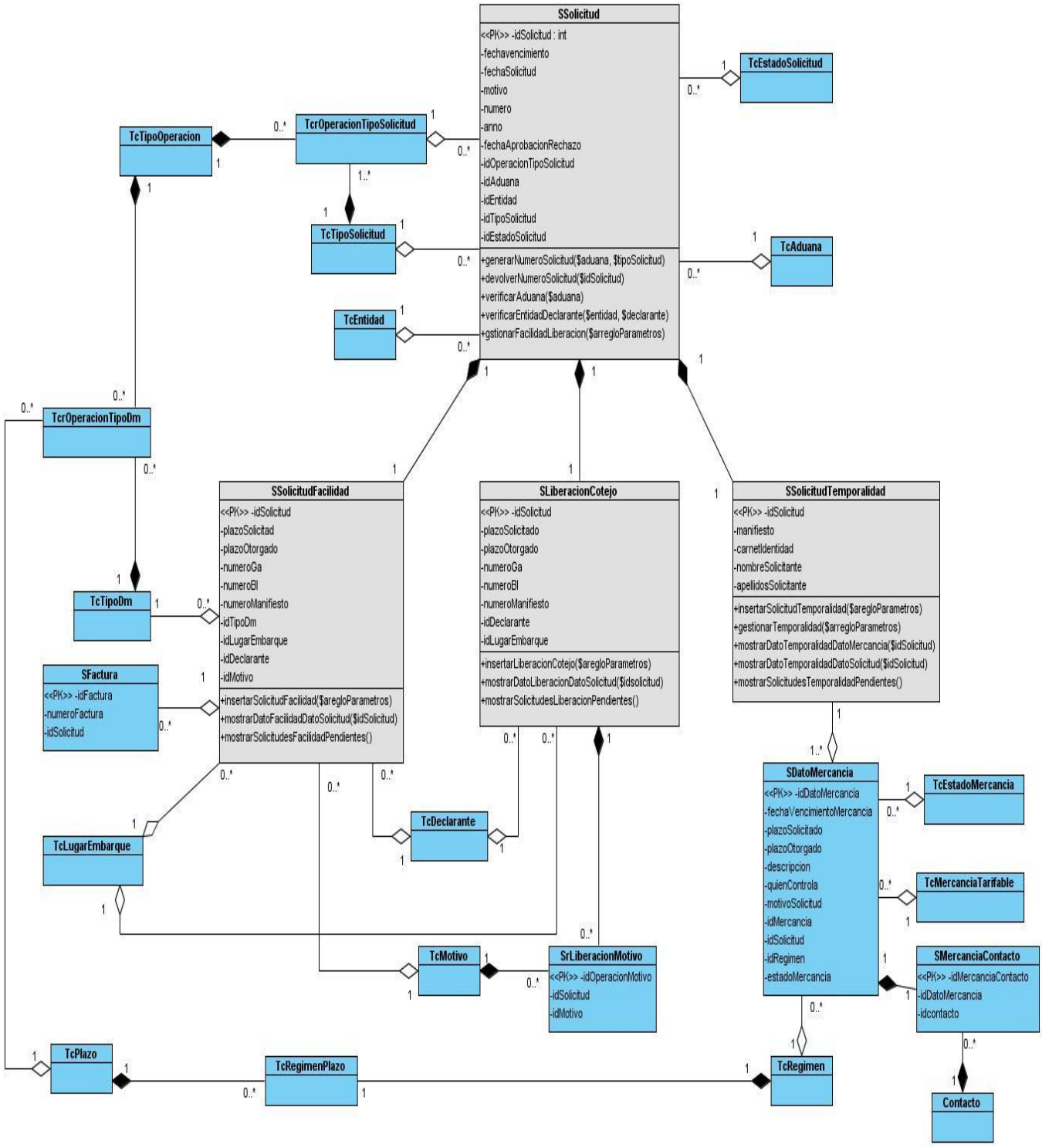


Figura 2.2. Diagrama de Clases del Diseño del Módulo Solicitudes.

El diagrama de Clases del Diseño del Módulo Solicitudes está constituido por 25 clases repartidas de la siguiente forma:

- Cuenta con 13 clases pertenecientes al esquema de la presente solución (de ellas sólo cuatro son consideradas arquitectónicamente significativas que son las representadas con el color gris).
- Presenta 11 clases que constituyen nomencladores pertenecientes al Subsistema Tablas de Control del Sistema GINA.
- Contiene una clase perteneciente al esquema del Subsistema Contacto del Sistema GINA.

2.3.3. Diagramas de Secuencia Orientado a Actividades del Negocio

El Diagrama de Secuencia Orientado a Actividades del Negocio es el encargado de describir el funcionamiento de cada uno de los RF. Puede existir más de un diagrama de este tipo asociado a un mismo RF, esto va en dependencia del nivel de complejidad que tengan estos requisitos y la cantidad de funcionalidades comunes que existan entre ellos. En la figura 2.3 se muestra un ejemplo de diagrama de este tipo, en este caso en particular se representa un segmento del perteneciente al RF Solicitar Facilidad. El mismo está constituido por 8 calles, cada una de ellas representando las áreas que abarca el negocio de esta funcionalidad. Comienza con el envío de los datos mediante la interfaz, estos son capturados en la clase controladora del Módulo Solicitudes y esta es la responsable de llamar a la funcionalidad *solicitarFacilidad(),* que se encuentra en la clase *SSolicitud.php*. En la funcionalidad *solicitarFacilidad(),* utilizando los datos entrados por parámetros, se realizan las validaciones pertinentes, consumiendo para ello los servicios brindados por los diferentes subsistema del GINA. En este caso se utiliza el servicio *obtenerDadoCriteria*[6] del Subsistema Tablas de Control para acceder a las tablas que pertenecen a este esquema, una vez validados los datos, estos son insertados a la base de datos mediante la utilización de los formularios generados por *Symfony*.

[6] obtenerDadoCriteria es un servicio que brinda el Componente Tablas de Control del Sistema GINA, al que se le pasa como parámetro un criterio determinado y devuelve un arreglo con los objetos que cumplan con el mismo.

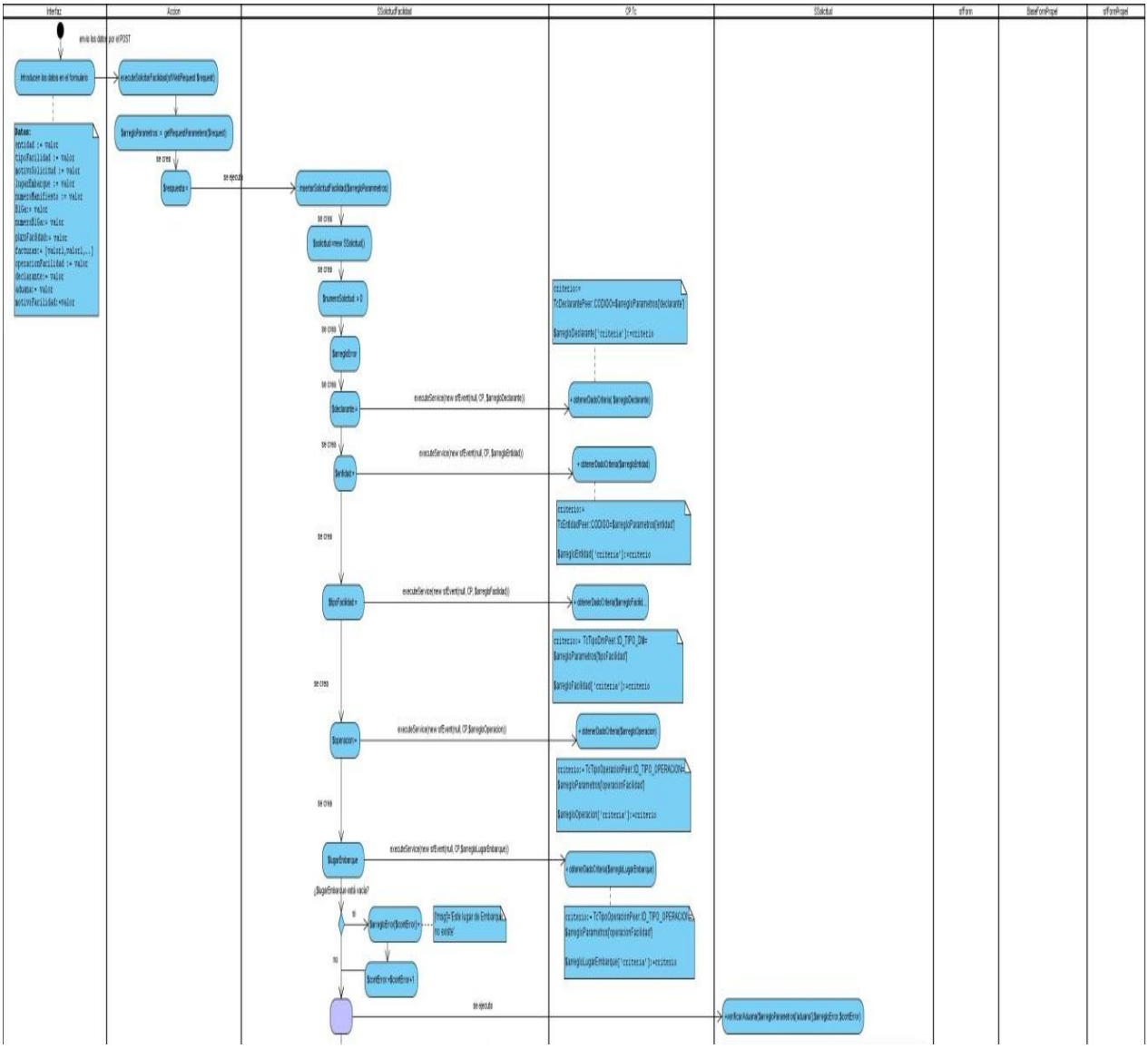


Figura 2.3. Segmento del Diagrama de Secuencia Orientado a Actividades del Negocio perteneciente al RF Solicitar Facilidad.

2.3.4. Diagramas de Secuencia Orientado a Actividades de Componentes Visuales

El Diagrama de Secuencia Orientado a Actividades de Componentes Visuales tiene como función fundamental guiar a los diseñadores de interfaz de usuarios; de manera que se les brinde la información necesaria para que el trabajo quede con calidad y en el menor tiempo posible. Se construye un diagrama para cada uno de los componentes visuales que necesita para su funcionamiento recibir o enviar algún

dato al negocio. En los comentarios de este diagrama se especifican varios parámetros que son útiles para el diseñador como es el caso del evento, nombre de la tabla o funcionalidad de la cual va a cargar o enviar la información, entre otros; de manera que se garantice una total correspondencia entre la información de este diagrama y su equivalente en el negocio. En la figura 2.4 se muestra uno de los diagramas de este tipo pertenecientes al RF Solicitar Facilidad. El mismo está formado por tres calles fundamentales, una de ellas representada por el actor que es el que comienza la funcionalidad. Una vez introducidos los datos se ejecuta la opción aceptar y esta desencadena una llamada a la acción solicitarFacilidad() enviando los datos que han sido introducidos en el formulario. Finalmente si no ocurre ningún tipo de error se devuelve el número de la solicitud insertada.

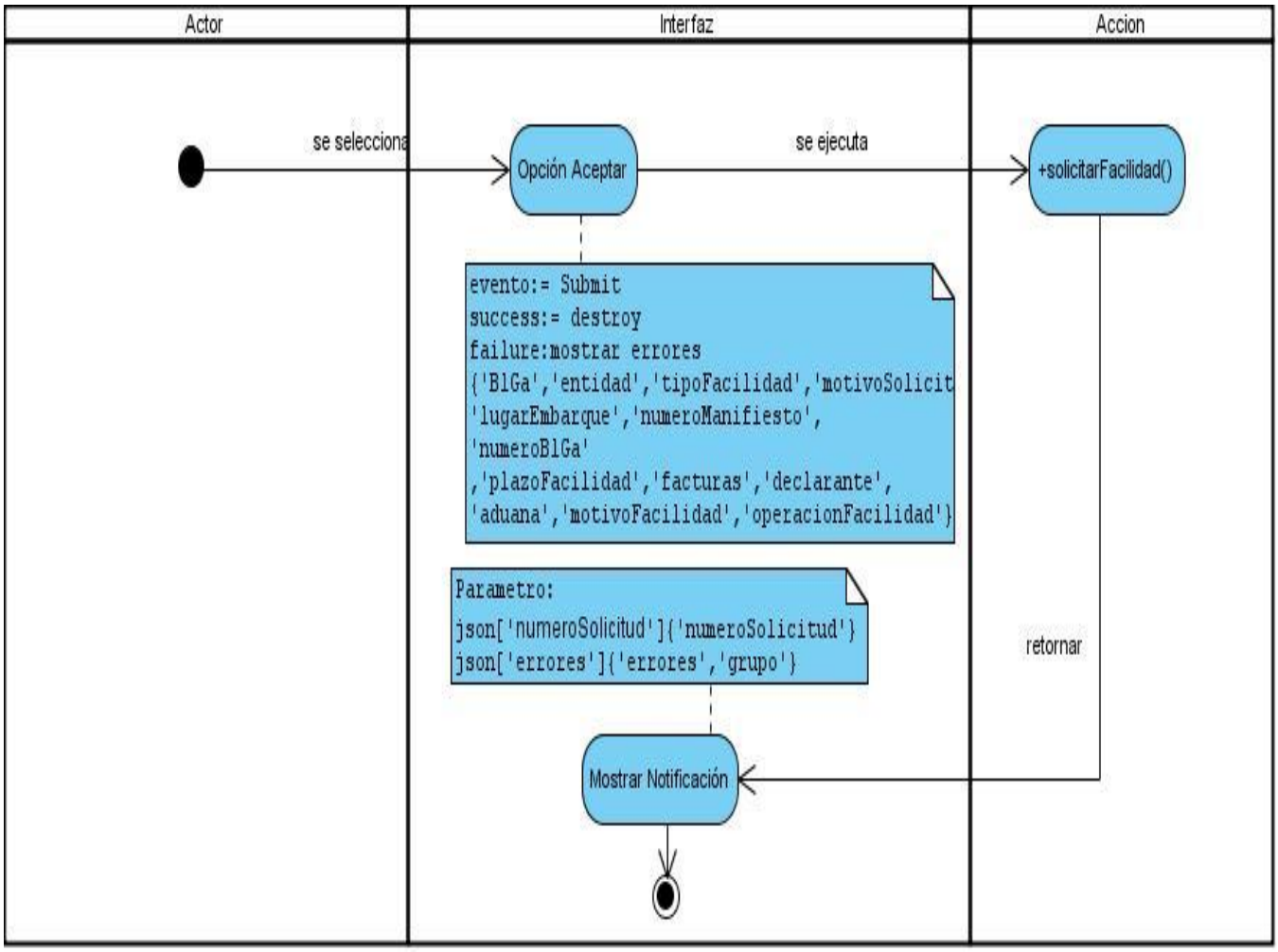


Figura 2.4. Diagrama de Secuencia Orientado a Actividades del Componente Visual Insertar Facilidad.

Capítulo 2: Diseño e Implementación del Sistema

2.4. Diseño de la Base de Datos

El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de ellos independientemente. (24)

Para realizar un buen diseño de una base de datos se deben tener en cuenta algunas consideraciones:

- La velocidad de acceso. (24)
- El tamaño de la información. (24)
- El tipo de la información. (24)
- Facilidad de acceso a la información. (24)
- Facilidad para extraer la información requerida. (24)
- El comportamiento del manejador de bases de datos con cada tipo de información. (24)

Durante el flujo de trabajo de diseño concebir e implementar la base de datos es uno de los hitos fundamentales de la elaboración de todo sistema que gestione información abundante. La base de datos en el desarrollo de este trabajo es de vital importancia porque con la misma se garantiza la persistencia de los datos de la aplicación que se desea obtener.

En la figura 2.5 se muestra el modelo de datos desarrollado en este trabajo, perteneciente al Módulo Solicitudes del Sistema GINA. El mismo está constituido por un total de 25 tablas, de ellas sólo 14 son propias del esquema del presente trabajo, representadas por los colores gris (nomencladores) y naranja. De las restantes tablas, 10 forman parte del Subsistema Tablas de Control, que es el encargado de llevar el control de los nomencladores del GINA, mostradas con el color verde. También se cuenta con una tabla representada con el color azul claro que pertenece al Subsistema Calendario. La base de datos se encuentra en 3ra Forma Normal pues todas sus tablas contienen una llave primaria, los atributos son atómicos y no contiene dependencias funcionales transitivas. De esta manera se evita la redundancia de los datos y los problemas que puedan ocurrir con las actualizaciones de estos en las tablas, protegiendo así la integridad de los mismos.

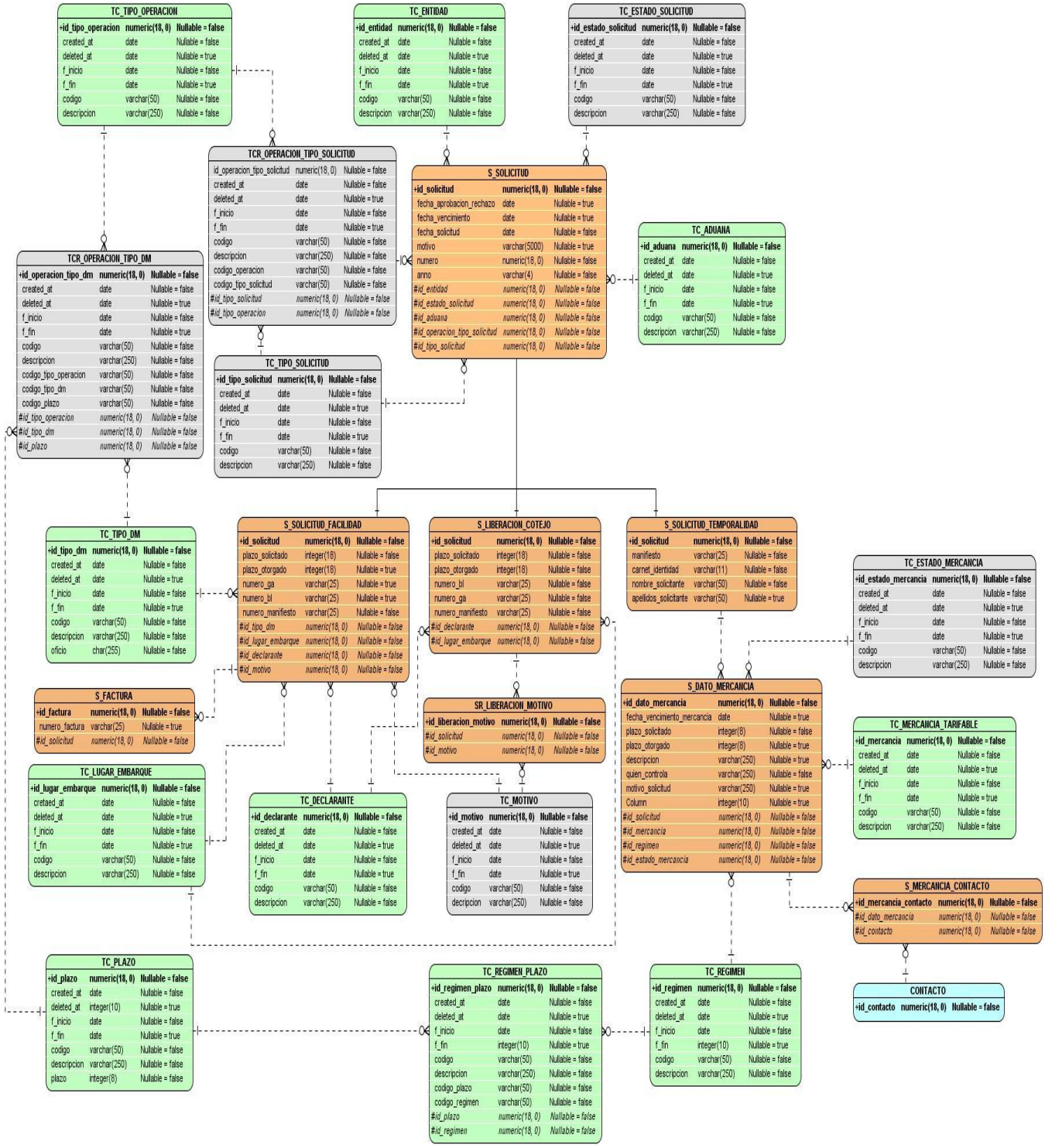


Figura 2.5. Modelo de Datos del Módulo Solicitudes.

Capítulo 2: Diseño e Implementación del Sistema

2.5. Métricas para la validación del diseño

El concepto de métrica es el término que describe muchos y muy variados casos de medición. Siendo una métrica una medida estadística que se aplica a todos los aspectos de calidad de software, los cuales deben ser medidos desde diferentes puntos de vista como el análisis, construcción, funcionalidad, documentación, métodos, proceso, usuario, entre otros. (25)

Las métricas para software como otras métricas no son perfectas, muchos expertos argumentan que se necesita más experimentación hasta que se puedan emplear bien las métricas de diseño. Sin embargo el diseño sin medición es una alternativa inaceptable. A continuación se mostrarán algunas de más comunes. Aunque ninguna es perfecta, pueden proporcionarle al diseñador una mejor visión interna y así el diseño evolucionará a un mejor nivel de calidad.
Existen numerosas métricas para le medición del diseño desarrollado, dentro de las que se encuentran:

- Tamaño operacional de clase (TOC): métrica que se basa esencialmente en la cantidad de funcionalidades que presenta la clase. (26)
- Profundidad de herencia (PH): mide el máximo nivel en la jerarquía de herencia. Es la cuenta directa de los niveles en la jerarquía de herencia. (26)
- Número de descendientes (ND): es el número de subclases subordinadas a una clase en la jerarquía, es decir, el número de subclases que pertenecen a una clase. (26)
- Número de operaciones redefinidas para una clase hija (NOR): muestra en qué medida las subclases redefinen el comportamiento de sus superclases. (26)

Luego de analizar todas las métricas mencionadas anteriormente, se decidió utilizar para validar el diseño de la solución la de Tamaño Operacional de Clase (TOC), propuesta por Lorenz y Kid[7]. La misma se encarga de medir la calidad de acuerdo a los atributos Responsabilidad, Complejidad de Implementación y Reutilización de las clases. Para la utilización de la Métrica Tamaño Operacional de Clase (TOC), se tuvo en cuenta los datos mostrados en la Tabla 2.1.

[7] Lorez y Kid: Especialistas autores del libro *"Object-Oriented Software Metrics". Prentice Hall. 1994.* Texto recomendado para introducirse en la medición de atributos de entidades desarrolladas con metodología orientada a objetos.

No	Módulo	Clase	Cantidad de Procedimientos
1	Solicitudes	SSolicitud	5
	Solicitudes	SSolicitudFacilidad	4
3	Solicitudes	SLiberacionCotejo	3
4	Solicitudes	STemporalidad	5

Tabla 2.1 Tabla que muestra las clases fundamentales del Módulo Solicitudes y sus procedimientos.

La aplicación de la métrica TOC define los siguientes atributos de calidad:

- Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta. (26)
- Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado. (26)
- Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software. (26)

En la figura 2.7 se muestra el comportamiento del nivel de procedimientos puesto en práctica en las clases del diseño realizado y a continuación en las figuras 2.8, 2.9 y 2.10 se representan los estados de los atributos de calidad: responsabilidad, complejidad de la implementación y reutilización respectivamente.

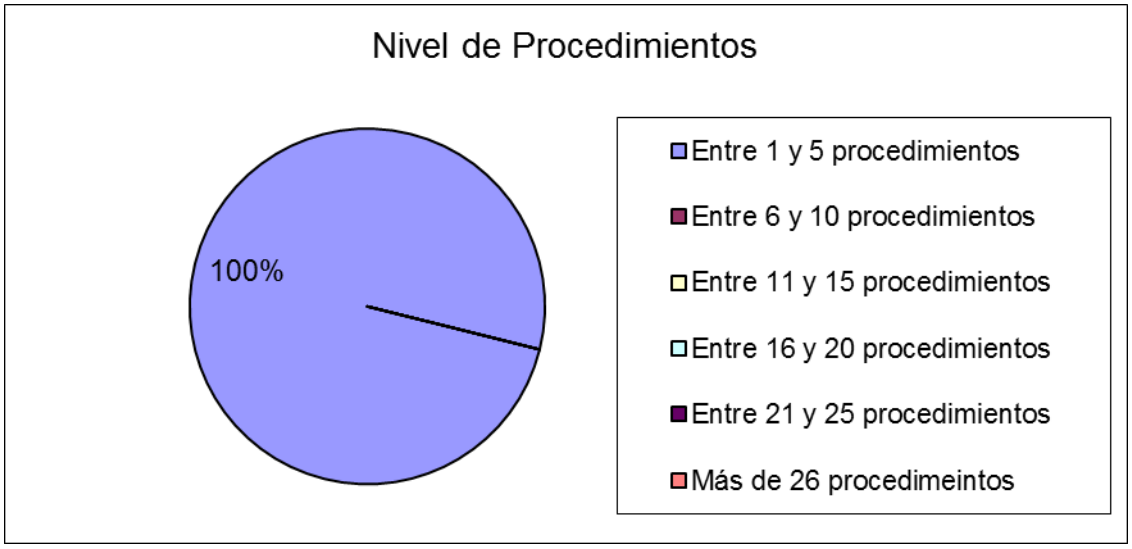


Figura 2.7. Representación del Nivel de Procedimientos de las Clases

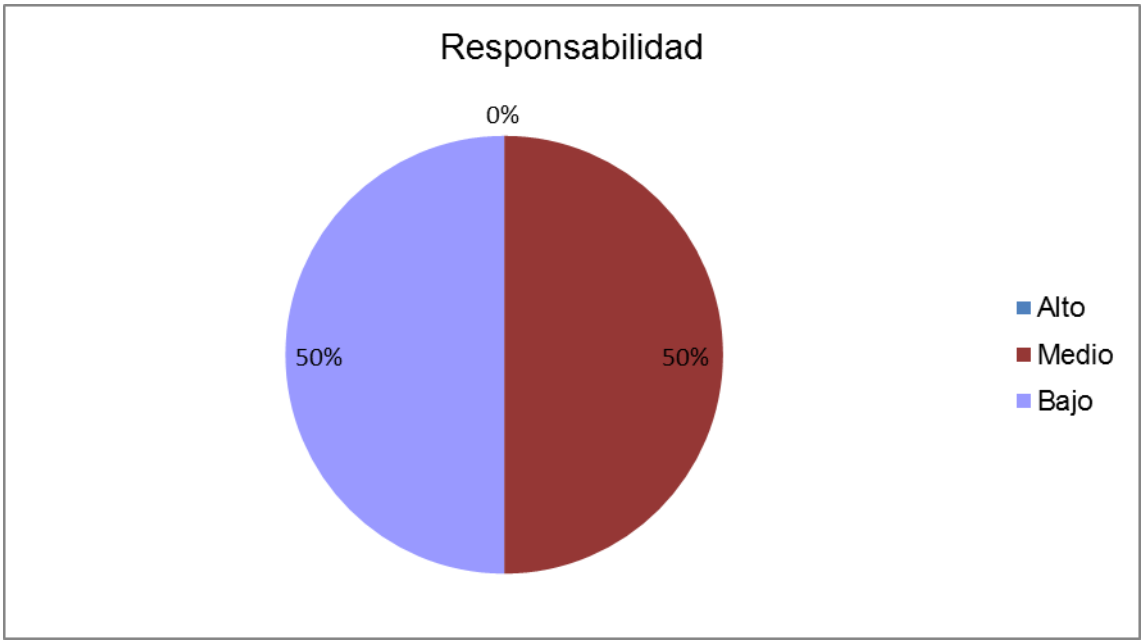


Figura 2.8. Representación de la Responsabilidad de las Clases

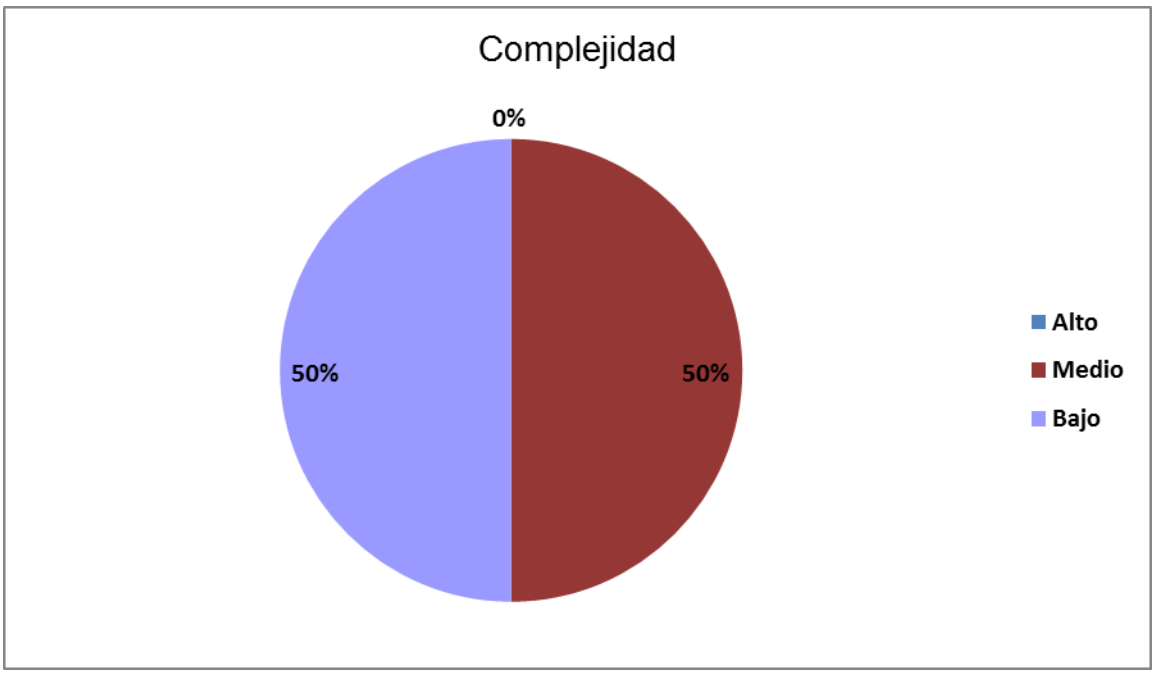


Figura 2.9. Representación de la Complejidad de las Clases

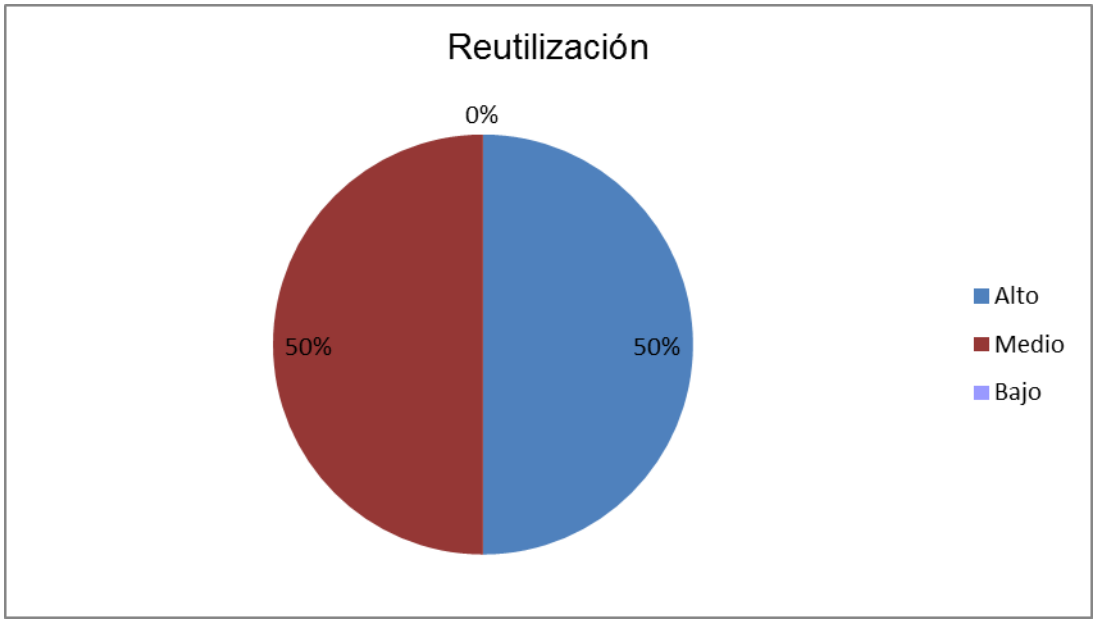


Figura 2.10. Representación de la Reutilización de las Clases

2.5.1. Resultado de la evaluación de las métricas.

Teniendo en cuenta la aplicación de las métricas, se determina que las clases del diseño en la presente solución contienen un promedio de 4.25 funcionalidades por clases. Encontrándose todas en el rango de 1 a 5 funcionalidades. Luego de analizar el promedio de funcionalidades por clases, se arriba a la conclusión de que el 50% de las mismas tiene un nivel bajo de responsabilidad y complejidad de implementación y el resto un nivel medio en ambos aspectos. Por otra parte, conociendo la inversa proporcionalidad que existe entre la responsabilidad y la complejidad de las clases contra la reutilización, se observa que la mitad de las clases presentan una alta probabilidad de reutilización y la otra mitad un nivel medio en este parámetro.

2.6. Modelo de Implementación

La implementación es sin duda una de las etapas fundamentales en el desarrollo de un software, durante la misma se generan varios artefactos entre los que se encuentra el modelo de implementación.

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes como ficheros de código fuente, ejecutables y similares. El modelo de

implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o los lenguajes de programación utilizados y cómo dependen los componentes unos de otros. (7)
En la presente solución este artefacto estará integrado por:

- El estándar de codificación utilizado durante la implementación del sistema.
- El tratamiento de errores en la presente solución.
- Un ejemplo de comunicación entre las diferentes capas.
- El Diagrama de Componentes.

2.6.1. Estándar de Codificación

Los estándares de codificación surgen como una necesidad de mantener una mejor comunicación entre los programadores de manera que se favorezca la reutilización y mantenimiento de los sistemas. Actualmente se han creado diferentes estándares para cada uno de los lenguajes existentes en el mundo.

El estándar de codificación del presente trabajo está regido por el documento: “Propuesta de un estándar de codificación”, perteneciente al Departamento de Soluciones para la Aduana del CEIGE.
Teniendo en cuenta que la implementación está basada en la utilización de un *framework* desarrollado en idioma inglés y, por otra parte la poca experiencia de los programadores en el trabajo con este idioma es válido el empleo de nomenclaturas en inglés y en español incluso en ambos idiomas a la vez. (27)

Uno de los requisitos que impone *Symfony* es que cada acción debe comenzar con la palabra *execute*, en este caso seguido del nombre de la acción en la nomenclatura *lowerCamelCase*[8]. (27)

Los nombres de las clases deben estar expresados en la notación *UpperCamelCase*[9], exponiendo con claridad cuál es el alcance y la responsabilidad de la clase, siempre teniendo en cuenta la estructura definida por el *framework* para cada clase en sus diferentes ámbitos. (27)

Las funciones definidas deben seguir la nomenclatura *UpperCamelCase* y su nombre debe dejar reflejado lo más claro posible la acción que realiza. (27)

8 El *lowerCamelCase* es un estilo de escritura donde la primera palabra comienza con minúscula y la primera letra de las demás con mayúscula.
9 El *UpperCamelCase* es un estilo de escritura donde la primera letra de cada una de las palabras es mayúscula.

Los nombres de las variables deben expresar claramente el contenido de la misma. Pueden ser expresadas lo mismo en singular que en plural. (27)

Para el caso de las llaves se colocará la primera en la misma línea del comienzo del bloque separada por un espacio y la llave de cierre del bloque en la última línea. (27)

Para comentar una línea se utilizará el o los caracteres que dicta el lenguaje de programación utilizado, al igual que para los comentarios de bloque, es decir, que requiera más de un línea. (27)

Para la documentación de las variables y las funciones se utilizará un comentario de bloque donde se especifique un comentario de la variable o función con el objetivo de la misma, el o los tipos de parámetros y el tipo de retorno si es necesario para la función. (27)

2.6.2. Tratamiento de Errores

Uno de los aspectos más importantes a tener en cuenta a la hora de desarrollar un software es el tratamiento de errores pues, para garantizar el correcto funcionamiento del sistema es fundamental identificar y controlar los posibles problemas que puedan presentarse a la hora de interactuar con el software. Entre los aspectos manejados en la presente investigación se encuentra la interacción con la base de datos (inserción, eliminación, modificación) tratando de que esta se realice de forma correcta. Para lograr el éxito en esta parte se realizan varias acciones entre las que se encuentran las validaciones conformadas a nivel de formulario mediante el uso de expresiones regulares. Garantizando estas últimas que el usuario introduzca cada uno de los valores obligatorios y con las características requeridas. Por otra parte en las diferentes funcionalidades cada uno de los datos manejados son verificados antes de ser enviados a la base de datos y en caso de encontrar algún problema, los mismos son mostrados a través de una alerta a los usuarios de manera que estos una vez identificados puedan ser corregidos y enviados nuevamente. Otro de los aspectos importantes utilizado en el tratamiento de errores del sistema desarrollado es la utilización de los formularios generados por *Symfony* para insertar, eliminar o modificar valores de la base de datos.

2.6.3. Comunicación entre Capas

La comunicación entre las capas en las que se divide la arquitectura está dada por mantener los paradigmas de la Programación Orientada a Objetos y la posibilidad de realizar sistemas con mayor

portabilidad. Se utiliza entre el cliente y el Controlador Frontal y viceversa tecnología JSON, con esta se serializan los datos pasados por el usuario en Objetos PHP. Permitiendo utilizar una interfaz de cualquier tipo para interactuar con el servidor de aplicación, además de la seguridad que brinda este tipo de comunicación en materia de desarrollo, ya que los implementadores de interfaz de usuario no tienen acceso al comportamiento de los objetos enviados desde la capa controladora. Siempre que sea posible la comunicación será utilizada en combinación con AJAX, para permitir mayor velocidad y dinamismo en la interacción de los sistemas por vía Web.

La comunicación entre las capas del Controlador y el Modelo, gracias a la implementación del ORM, está dada por el envío de objetos que contienen la información necesaria para manejar las peticiones del usuario o las acciones a realizar. (28)

2.6.3.1. Ejemplo de Comunicación entre Capas

Al entrar al sistema el usuario interactúa con la página principal (ver figura 2.11), donde se encuentra una barra de menú con las diferentes funcionalidades con las que cuenta la aplicación. Una vez que el usuario seleccione la opción deseada el sistema le mostrará la pantalla correspondiente, con el propósito de explicar mejor la comunicación entre las capas, se muestra como ejemplo el Requisito Funcional Solicitar Facilidad (ver figura 2.12).

Para solicitar una facilidad el usuario debe introducir todos los datos en el formulario. En el caso de los datos del declarante y del importador/exportador, una vez introducido el código se desencadena una petición AJAX, encargada de llenar los demás campos relacionados con esto. Por otra parte, para mostrar algunos de los datos que se encuentran nomenclados se utiliza el *tcCombo* que es una especialización de *Ext.form.ComboBox* y está destinado para cargar los datos de una determinada tabla de la base de datos. El componente mencionado anteriormente fue implementado en el Departamento de Soluciones para la Aduana.

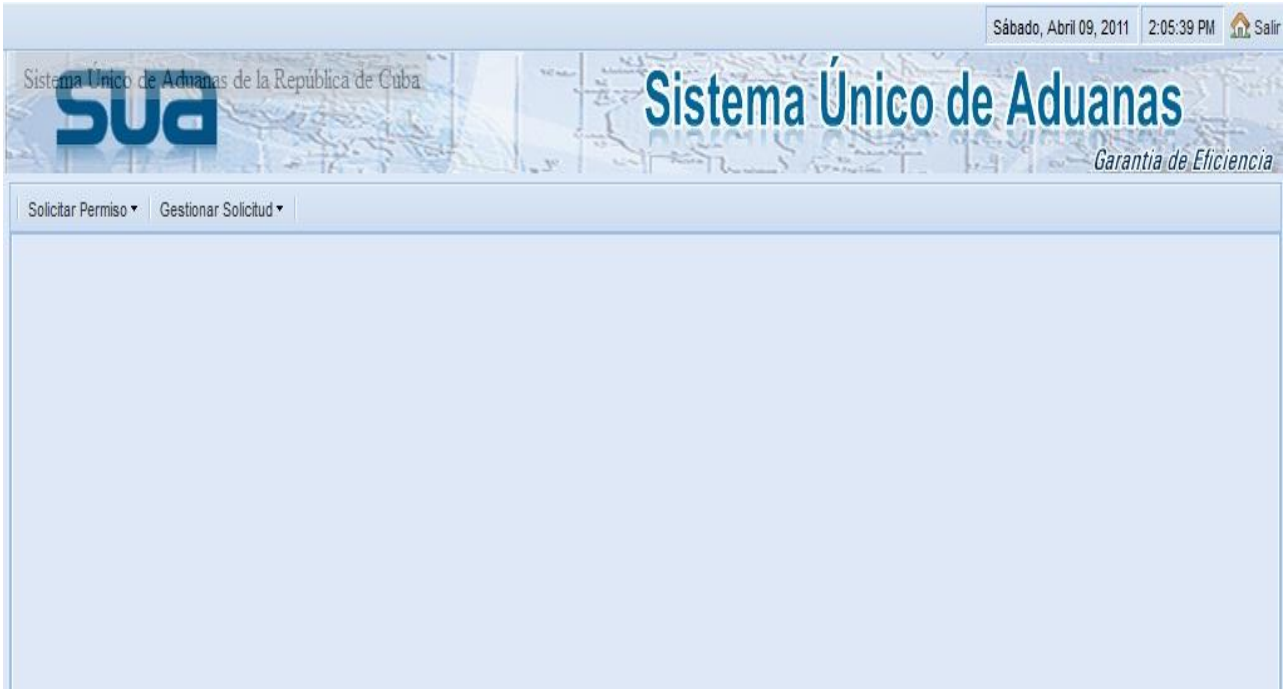


Figura 2.11. Pantalla Principal.

Figura 2.12. Pantalla Solicitar Facilidad.

Si el usuario no introduce los datos correspondientes en el formulario, el sistema le muestra un mensaje de error desde el JavaScript como se muestra en las figuras 2.13 a) y 2.13 b).

Figura 2.13 a). Pantalla de error que lanza cuando los datos no son correctos.

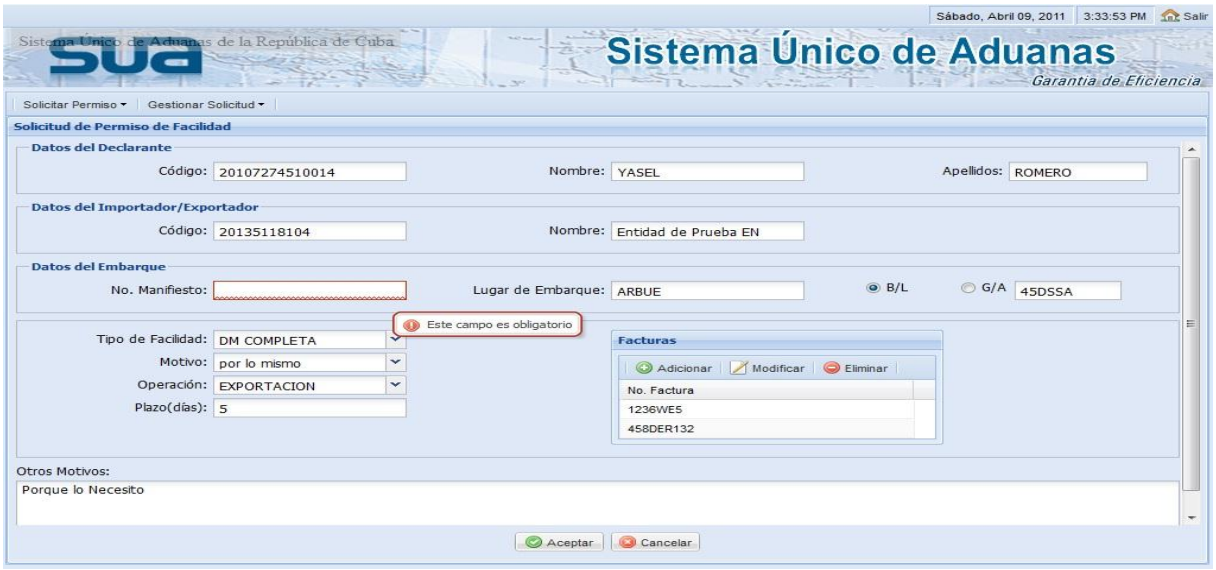


Figura 2.13 b). Pantalla que muestra error en los campos.

Al ocurrir algún error, luego de que el sistema notifique al usuario, se tendrá la posibilidad de corregir el problema hasta que los datos estén correctos como se muestra en la figura 2.14. Después que los datos estén en el formato requerido se selecciona la opción aceptar para enviar los mismos. Una vez seleccionada la opción mencionada anteriormente, mediante el evento *submit* por el método *POST*, son enviados los datos a la clase controladora *solicitudActions.* Para realizar el envío de los datos es necesario especificar la dirección URL de la acción que recibirá los mismos, como se muestra en las figuras 2.15 a) y 2.15 b) respectivamente.

Figura 2.14. Pantalla con los datos correctos.

```
SolicitarFacilidadCotejo = Ext.extend(Ext.form.FormPanel, {
title: 'Solicitud de Permiso de Facilidad',
buttonAlign: 'center',
defaultType: 'fieldset',
    layout: 'form',
   frame: true,
autoScroll: true,
facilidad: true,
   defaults: {
       anchor: '-20',
       layout: 'column',
cls: 'center top'
   },
loadURL: 'solicitud/solicitarFacilidad',
initComponent: function(){
varobj = this;
this.on('show', function(){
obj.on('hide', function(){
obj.destroy();
```

```
});
obj.doLayout();
        });
```

Figura 2.15 a). Código del encabezado del formulario.

```
this.buttons = [{
iconCls: 'iconOk',
            text: 'Aceptar',
            handler: function(){
Ext.Msg.wait("Cargando...", "Esperepor favor");
                if (obj.form.isValid()) {
                                    obj.form.submit({
                                    url: obj.loadURL,
                                    waitMsg: 'Enviando...',
                                    waitTitle: 'Esperepor favor',
                                    success: function(form, action){
                                                varmsg = Ext.decode(action.response.responseText);
                                                msg = "La solicitud se ha guardado con el n&uacute;mero " + msg.numeroSolicitud;
                                                            Ext.Msg.show({
                                                                        title: obj.title,
                                                                        msg: msg,
                                                                        icon: Ext.Msg.INFO,
                                                                        buttons: Ext.Msg.OK,
                                                                        width: 350
                                                            });
                                                            Ext.getCmp('mainPanelExt').getLayout().setActiveItem(0);
                                                },
                                                failure: function(form, action){
                                                            new ErroresTemporalidad({
                                                                        data: Ext.decode(action.response.responseText).errores ||
                                                                            []
                                                                        }).show();
                                                            }
                                                });
                                    }
                                    else {
                                                Ext.Msg.show({
                                                            title: 'Error',
                                                            msg: 'Existen campos con datos inv&aacute;lidos.',
                                                            icon: Ext.Msg.ERROR,
                                                            buttons: Ext.Msg.OK,
                                                            width: 300
                                                });
                                    }
            }
        }, {
iconCls: 'iconCancel',
            text: 'Cancelar',
            handler: function(){
Ext.getCmp('mainPanelExt').getLayout().setActiveItem(0);
}
        }];
```

Figura 2.15 - b). Código de los botones Aceptar y Cancelar.

Luego de ser enviados los datos a la clase controladora, son recibidos en la acción especificada como se muestra en la figura 2.16. Mediante una llamada a el método *getPostParameters()* se obtienen los parámetros. Posteriormente se hace una llamada a la funcionalidad *insertarSolicitud()* que se encuentra en la clase *SSolicitudFacilidad.php*, pasándole por parámetro los datos recibidos anteriormente.

```
public function executeSolicitarFacilidad(sfWebRequest $request) {
try {
        $arregloParametros = $request->getPostParameters();

        $con = Propel::getConnection('solicitud');
$con->beginTransaction();
        $respuesta = SSolicitudFacilidad::insertarSolicitud($arregloParametros);

$con->commit();
        return $this->renderText(json_encode($respuesta));
      } catch (Exception $e) {
        $con->rollBack();
        throw new Exception($e->getMessage());
}
   }
```

Figura 2.16. Segmento de código de la acción Solicitar Facilidad

Al realizar la llamada a la funcionalidad *insertarSolicitud(),* que se encuentra localizada en la clase *SSolicitudFacilidad.php* (figura 2.17), se obtienen los datos mediante las llaves con las que fueron enviados cada uno de ellos. También se realizan las validaciones para cada uno de los parámetros que la requieren y se insertan mediante la utilización de los formularios que genera *Symfony*. El resultado de esta funcionalidad depende, como se muestra en la figura 2.18 del comportamiento de la variable *$arregloError*. Si la variable mencionada anteriormente está vacía, quiere decir que no se detectó ningún tipo de error en la funcionalidad, por lo que se devuelve el número con el que fue almacenada la solicitud como se muestra en la figura 2.19. En caso contrario se retorna la misma con cada uno de los errores como se muestra en la figura 2.20. Para ambos casos se devuelve un arreglo, el cual es codificado posteriormente a formato JSON en la acción *executeSolicitarFacilidad(),* como se representa en la figura 2.16.

```
public static function insertarSolicitud($arregloParametros) {
      $solicitud = new SSolicitud();
      $numeroSolicitud = 0;
      $arregloError= array();
      $contError=0;
      $syscomponent = SysComponentsLocator::getInstance();
      $componente = $syscomponent->getComponent('tc');
```

```
//aquí busco el declarante por el código para obtener el id
$criterio = new Criteria();
$criterio->add(TcDeclarantePeer::CODIGO, $arregloParametros['declarante']);
$arregloDeclarante = array();
$arregloDeclarante['nombreTc'] = TcDeclarantePeer::TABLE_NAME;
$arregloDeclarante['estado'] = 'vigente';
$arregloDeclarante['criteria'] = $criterio;
$declarante = $componente->executeService(new sfEvent(null, 'tc.obtenerDadoCriteria', $arregloDeclarante));

//aquí busco la entidad por el código para obtener el id
$criterio = new Criteria();
$criterio->add(TcEntidadPeer::CODIGO, $arregloParametros['entidad']);
$arregloEntidad = array();
$arregloEntidad['nombreTc'] = TcEntidadPeer::TABLE_NAME;
$arregloEntidad['estado'] = 'vigente';
$arregloEntidad['criteria'] = $criterio;
$entidad = $componente->executeService(new sfEvent(null, 'tc.obtenerDadoCriteria', $arregloEntidad));
```

Figura 2.17. Segmento de código de la Funcionalidad *insertarSolicitud()*

```
if(empty ($arregloError)){
$respuesta['numeroSolicitud'] = $numeroSolicitud['numeroMostrar'];
$respuesta['success'] = true;
}
else{
$respuesta['errores']=$arregloError;
$respuesta['success'] = false;
}
return $respuesta;
```

Figura 2.18. Segmento de código que representa el comportamiento de la respuesta de la Funcionalidad *insertarSolicitud()*

Figura 2.19. Pantalla que muestra el número de la solicitud insertada.

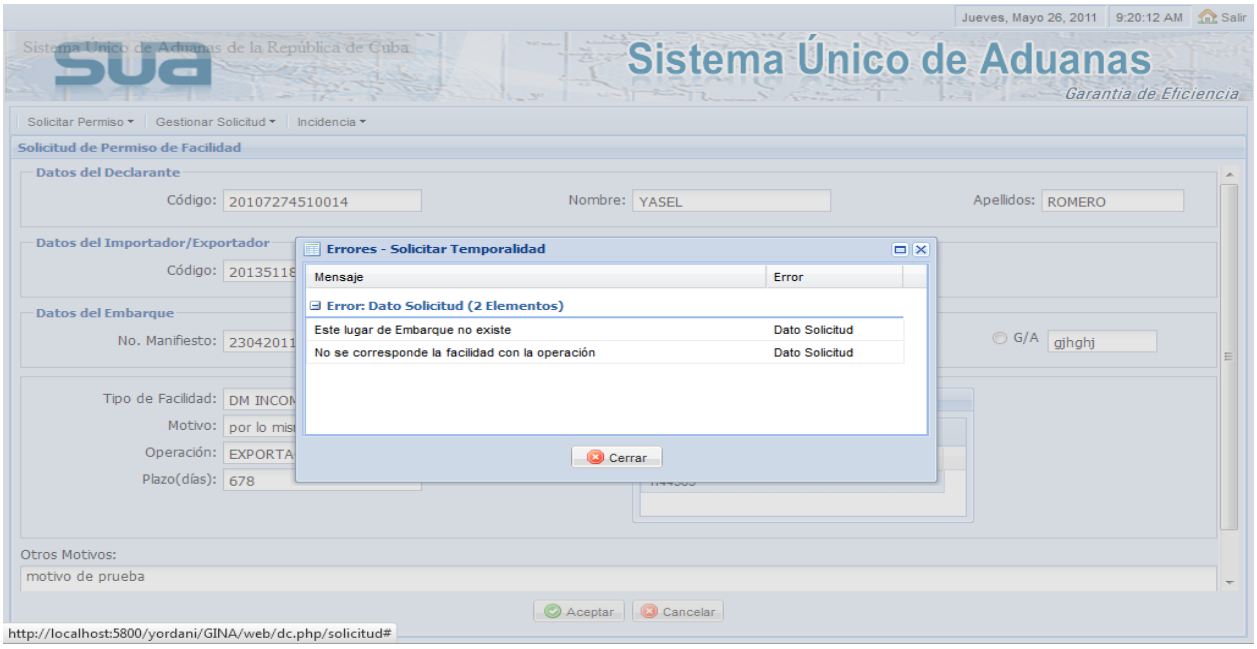


Figura 2.20. Pantalla que muestra los errores encontrado durante la Funcionalidad *insertarSolicitud()*.

2.6.4. Diagrama de Componentes

En la figura 2.21 se muestra el diagrama de componentes de la solución. El mismo está constituido por el componente que representa el controlador frontal de la aplicación, conocido como el único punto de entrada a la misma y encargado de cargar la configuración y determinar la acción a ejecutarse. El componente mencionado anteriormente se relaciona con el *action.class.php* teniendo entre sus funciones principales la comunicación entre las vistas y las clases del negocio. El *action* como componente se relaciona con dos paquetes, uno de ellos es el de las interfaces .js constituido por todas las clases encargadas de recibir y mostrar los datos al usuario y el otro es el de la abstracción de datos de *Symfony,* constituido por los componentes *BaseObjeto.php* y *ObjetoPeer.php* que a su vez utilizan *propel* como ORM. El último paquete de componentes mencionado se relaciona con el de configuración, constituido el mismo por el *databases.yml* que es el que contiene la información necesaria para la conexión a la base de datos y el *view.yml* encargado de definir las opciones entre las vistas. Además de relacionarse con el de configuración, también lo hace con tres importantes componentes pertenecientes al Sistema GINA: *sfCalendarioPlugin*, *TC* y *sfPersonaPlugin*.

Existen un paquete común entre el de configuración y el de las interfaces.js integrado por el componente *ext-all.js* (contiene todas las clases del Ext JS), el *ext-base.js* (encargado de configurar el acceso a las librerías del Ext JS) y el *ext-all.css* (tiene como función principal el control de la apariencia).

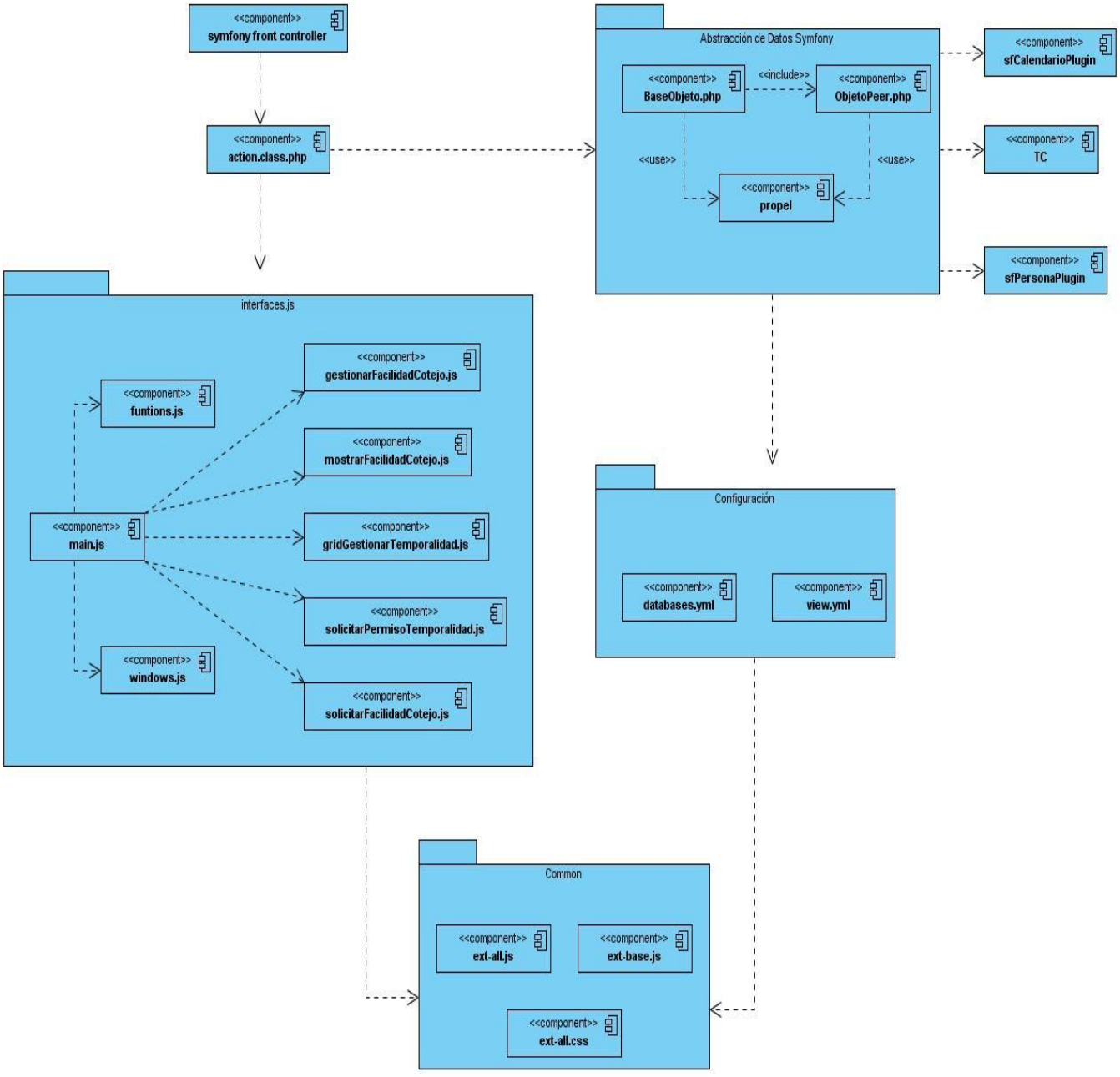


Figura 2.21. Diagrama de Componentes de la Solución.

2.7. Conclusiones Parciales

- Los patrones empleados durante el diseño contribuyeron a que se alcanzara en esta etapa un bajo acoplamiento y una alta cohesión entre las clases.
- Con la elaboración de cada uno de los artefactos definidos por el Departamento de Soluciones para la Aduana del CEIGE se logró sentar las bases para la implementación del sistema.
- La aplicación de la métrica TOC permitió evaluar el diseño y determinar que se cuenta con un 50% de las clases con un alto nivel de reutilización y el resto con un nivel medio.
- El diseño y la implementación de las interfaces y clases del negocio permitieron obtener una aplicación que fuera capaz de gestionar las solicitudes realizadas en la AGR.

Capítulo 3: Validación de la Solución

3.1 Introducción

En el presente capítulo se valida mediante pruebas, si el software realizado responde a las necesidades del cliente y se corresponde con los RF planteados durante la etapa del análisis. Para ello se realizaron un conjunto de casos de prueba y comparaciones entre el proceso de análisis y aprobación de las solicitudes realizadas manualmente y las ejecutadas con ayuda de la aplicación desarrollada.

3.2 Pruebas de Software

La prueba del software es un elemento crítico para la garantía de la calidad de la aplicación y representa una revisión final de las especificaciones del diseño y de la implementación. (29) El objetivo principal de la etapa de pruebas es garantizar la calidad del producto desarrollado. Es una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. La creciente percepción del software como un sistema y la importancia de los costos asociados a un fallo de este motivan a la creación de pruebas minuciosas y bien planificadas. (30)

3.2.1 Aplicación de Pruebas de Caja Negra

Las Pruebas de Caja Negra son aquellas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software. La misma se centra principalmente en los RF del software y permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los RF de un programa. En ellas se ignora la estructura de control, concentrándose en los RF del sistema y ejercitándolos. (31)
Entre los problemas que suelen detectar se encuentran (31):

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.

- Errores de rendimiento.
- Errores de inicialización y terminación.

Dentro de las pruebas de Caja Negra se incluyen las Pruebas de Funcionalidad que son las que se le aplicaron a la solución obtenida. Las Pruebas de Funcionalidad examinan si el sistema cubre sus necesidades de funcionamiento, acorde a las especificaciones del diseño. En ellas se debe verificar si el sistema lleva a cabo correctamente todas las funcionalidades requeridas y la validación de los datos, además se deben realizar pruebas de comportamiento ante distintos escenarios. Estas pruebas deben estar enfocadas a tareas, a límites del sistema, a condiciones planeadas de error y de exploración. (32)

Para la puesta en marcha de este tipo de pruebas se hace necesario la presencia de un buen diseño de casos de prueba, estos son un conjunto de condiciones o variables bajo las cuales se determina si el requisito de una aplicación es parcial o completamente satisfactorio. Se pueden realizar muchos casos de prueba para determinar que un requisito es completamente satisfactorio. Con el propósito de comprobar que todos los requisitos de una aplicación son revisados, debe haber al menos un caso de prueba para cada requisito a menos que un requisito tenga requisitos secundarios. En ese caso, cada requisito secundario deberá tener por lo menos un caso de prueba. En la tabla 3.1 se muestra el caso de prueba aplicado al RF Solicitar Facilidad de la presente solución.

Id del escenario	Escenario	Código del declarante	Nombre	Apellido	Código de la Entidad	Nombre	No. manifiesto	BL/GA	Lugar de embarque	Plazo(días)	Respuesta del sistema	Resultado de la prueba
EP 1.1	Solicitar Facilidad.	V	NA	NA	V	NA	V	V	V	V	Notificar el número otorgado a la solicitud de facilidad	Se guarda la información y se notifica el número de la solicitud
EP	Registrar	I	NA	NA	NA	NA	NA	NA	NA	NA	Marca el código del	La aplicación

1.2	código del declarante.										declarante como incorrecto y muestra los posibles errores encontrados en el mismo.	no carga los datos del declarante, notificando el error encontrado.
EP 1.3	Registrar código de la entidad.	NA	NA	NA	I	NA	NA	NA	NA	NA	Marca el código de la entidad como incorrecto y muestra los posibles errores encontrados en el mismo.	La aplicación no carga los datos de la entidad, notificando el error encontrado.
EP 1.4 Validar datos		NA	NA	NA	NA	NA	I	NA	NA	NA	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el error.
		NA	NA	NA	NA	NA	NA	I	NA	NA	Se validan los datos según el escenario.	El sistema no reconoce que la longitud de la cadena es mayor que 35.
		NA	NA	NA	NA	NA	NA	NA	I	NA	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos

											inválidos especificando el error.
	NA	NA	NA	NA	NA	NA	NA	NA	I	Se validan los datos según el escenario.	El sistema notifica que existen campos con datos inválidos especificando el error.

Tabla 3.1 Caso de Prueba del Requisito Funcional Solicitar Facilidad.

De esta forma se le realizaron las pruebas de funcionalidad a los requisitos del sistema. Se aplicaron un total de 6 casos de prueba (uno para cada RF, exceptuando al RF Consultar Historial de Declarante que ha sido probado por el Componente Historial de Personas) con sus Escenarios de Pruebas (EP) correspondientes como se muestra a continuación:

1. Caso de Prueba Solicitar Facilidad.
    - EP 1.1: Solicitar Facilidad.
    - EP 1.2: Registrar código del declarante.
    - EP 1.3: Registrar código de la entidad.
    - EP 1.4: Validar datos. Se detectó una no conformidad.
2. Caso de Prueba Solicitar Liberación de Cotejo.
    - EP 2.1: Solicitar Liberación de Cotejo.
    - EP 2.2: Registrar código del declarante.
    - EP 2.3: Registrar código del Importador/Exportador.
    - EP 2.4: Validar datos. Se detectó una no conformidad.
3. Caso de Prueba Solicitar Régimen Temporal.
    - EP 3.1: Solicitar Régimen Temporal.
    - EP 3.2: Registrar código de la entidad Importadora/Exportadora.

- EP 3.3: Validar datos del solicitante.
- EP 3.4: Registrar datos de la mercancía.

4. Caso de Prueba Gestionar Solicitud de Permiso de Facilidad.
    - EP 4.1: Gestionar Solicitud de Facilidad.
    - EP 4.2: Seleccionar solicitud.
    - EP 4.3: Aprobar solicitud.
    - EP 4.4: Rechazar solicitud.
    - EP 4.5: Validar plazo solicitado.
5. Caso de Prueba Gestionar Permiso de Liberación de Cotejo.
    - EP 5.1: Gestionar Solicitud de Liberación de Cotejo.
    - EP 5.2: Seleccionar solicitud.
    - EP 5.3: Aprobar solicitud.
    - EP 5.4: Rechazar solicitud.
    - EP 5.5: Validar plazo solicitado.
6. Caso de Prueba Gestionar Permiso de Régimen Temporal.
    - EP 6.1: Gestionar Solicitud de Régimen Temporal.
    - EP 6.2: Seleccionar solicitud.
    - EP 6.3: Ver datos de la partida.
    - EP 6.4: Aceptar mercancía.
    - EP 6.5: Rechazar mercancía.
    - EP 6.6: Validar plazo solicitado.

Durante la puesta en práctica de estos casos de prueba se reportaron un total de dos no conformidades que se centran fundamentalmente en la validación de los datos que son introducidos por los usuarios. Una vez detectadas las mismas fueron resueltas reduciendo esta cifra a cero.

Capítulo 3: Validación de la Solución

3.3 Impacto de la Solución Obtenida.

Después de finalizada la aplicación se hizo necesario implantarla en la AGR de manera que pudiera probarse su eficiencia en un entorno real, además de comprobar si en realidad permite resolver la gran problemática que existe en esta entidad con el manejo de las solicitudes.

Para realizar una prueba piloto el sistema obtenido se puso en marcha en dos importantes entidades aduaneras, localizadas en el Puerto de la Habana y en el Aeropuerto Internacional “José Martí Pérez” especializadas en el tráfico marítimo y aéreo respectivamente.

Luego de transcurrir un mes se obtuvieron las estadísticas arrojadas por la aplicación con el objetivo de comparar cuán rentable es el uso de la misma. Uno de los aspectos tenidos en cuenta a la hora de realizar dicha comparación fue la cantidad de solicitudes de cada tipo que se gestionaban manualmente en un período aproximado de un mes y las que fueron realizadas con el sistema en el mismo tiempo. Para el caso de las solicitudes de régimen temporal se tuvieron en cuenta las realizadas bajo el régimen 5400. En la figura 3.2 se muestran los resultados obtenidos en cuanto al indicador mencionado anteriormente, observándose un notable incremento en cuanto a la cantidad de solicitudes gestionadas con el sistema.

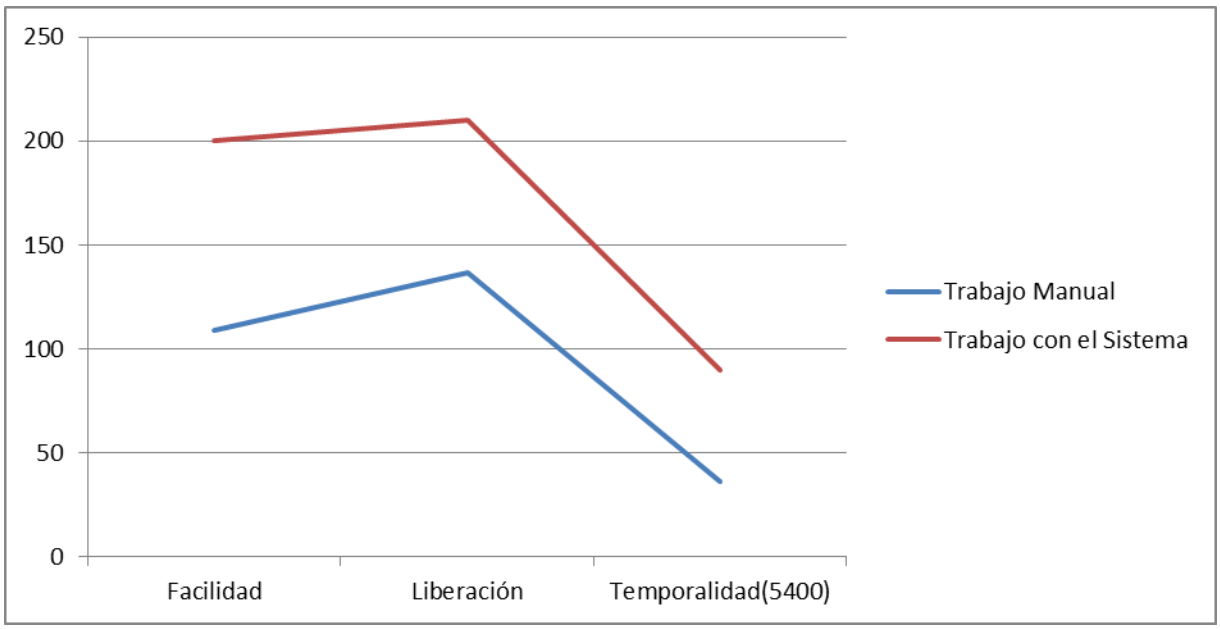


Figura 3.2. Gráfica que representa la cantidad de solicitudes realizadas de forma manual y con el sistema en un mes.

Capítulo 3: Validación de la Solución

Otro de los aspectos estudiados fue la cantidad de errores encontrado en las declaraciones de mercancías realizadas durante el proceso de las solicitudes ejecutado manualmente en el año 2010, esta información se muestra de manera clara en la figura 3.3.

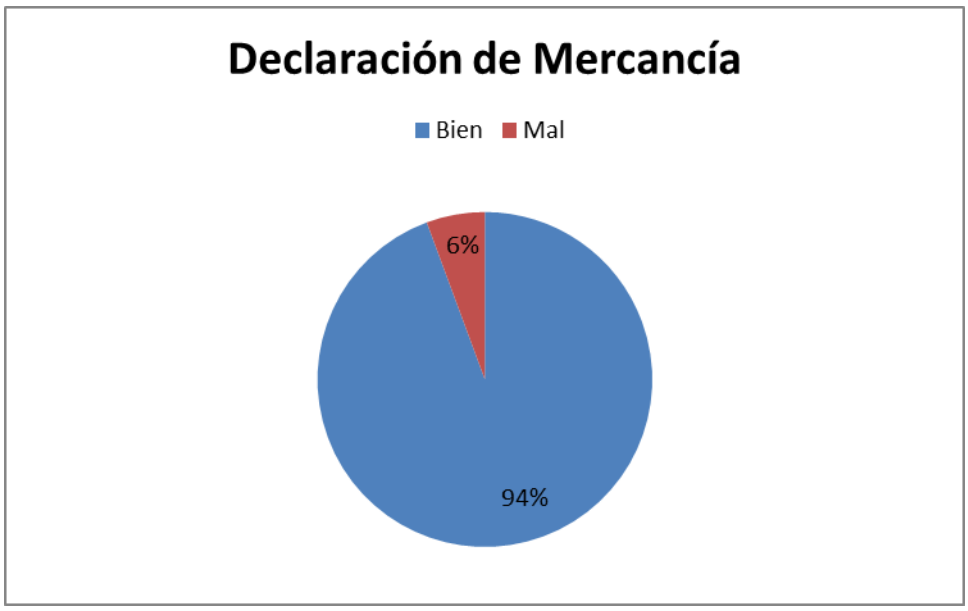


Figura 3.3 Resultados de las declaraciones de Mercancías en el año 2011.

Como se observa en la figura anterior, se detectaron un gran número de errores en las solicitudes realizadas en cuanto a las declaraciones de mercancías en el año 2010. De un total de 59497 declaraciones de mercancías presentadas en las dos entidades escogidas para la realización de la prueba del software, se detectaron 3570 declaraciones con errores lo que representa un 6% del total. Es válido destacar que con el uso de la herramienta desarrollada la cantidad de declaraciones con errores se reduce considerablemente.

La utilización de la aplicación obtenida ha tenido un impacto positivo en el proceso de control manual de la información generada durante cada solicitud. El proceso mencionado anteriormente se elimina completamente ganando en gran medida en cuanto a tiempo y eficiencia del proceso de gestión. El sistema también ha condicionado un ahorro en cuanto a combustible y materiales de oficina.

Capítulo 3: Validación de la Solución

3.4 Conclusiones Parciales

- La aplicación de pruebas funcionales a la solución desarrollada permitió encontrar errores que afectaban el funcionamiento del sistema, dando la posibilidad de que pudieran ser corregidos a tiempo y que se obtuviera un módulo que responde completamente a los RF definidos durante la etapa del análisis.

- El hecho de que la aplicación fuera probada en un entorno real demostró que se procesan una mayor cantidad de solicitudes, se reduce el margen de error y se ahorra combustible y materiales de oficina.

Conclusiones

En el presente trabajo se abordaron en un principio las problemáticas que originan los objetivos del mismo, argumentadas por las dificultades presentadas en el proceso de análisis y aprobación de las solicitudes realizadas por las diferentes entidades del país a la AGR para importar o exportar mercancías.

La identificación de un conjunto de sistemas informáticos que se encargan de gestionar solicitudes permitió analizar las características y el comportamiento de cada uno, determinándose que ninguno podía formar parte de la solución.

La utilización de patrones de diseño unido a la elaboración de cada uno de los artefactos definidos por el Departamento de Soluciones para la Aduana, permitió que se sentaran las bases para la implementación del sistema deseado.

El diseño y la implementación de las interfaces y las clases del negocio de la presente solución, permitieron que se obtuviera una aplicación que respondiera a los RF definidos durante la etapa de análisis

La aplicación de pruebas de funcionalidad posibilitó la detección de errores en el funcionamiento de la solución obtenida permitiendo que los mismos pudieran ser resueltos.

El hecho de que la solución pudiera ser probada en un entorno real demostró que el sistema permite gestionar una mayor cantidad de solicitudes y disminuye el margen de error en estos procesos.

Finalmente el diseño y la implementación realizada en el presente trabajo permitieron que se obtuviera una aplicación que fuera capaz de satisfacer las necesidades de los clientes del Módulo Solicitudes, dándole cumplimiento al objetivo general de la presente investigación.

Recomendaciones

Una vez cumplido con los objetivos de la investigación, teniendo en cuenta las experiencias obtenidas en la misma, se recomienda:

- La liberación de la aplicación por el equipo de calidad de software de la universidad.
- La implementación de reportes con el objetivo de favorecer el control y las estadísticas de las solicitudes realizadas.
- La implementación del resto de las solicitudes que han sido detectadas en el Proceso de Despacho Comercial de la AGR.

Bibliografía

1. **Hernández Sampieri, Roberto, Fernández Collado, Carlos y Baptista Lucio, Pilar.** *Metodología de la Investigación.* 2006.

2. **Berndtsson, Mikael, y otros, y otros.** *Thesis Projects. A Guide for Students in Computer Science.* s.l. : Springer, 2008.

3. **Supportickets.** Supportickets. [En línea] 2010. [Citado el: 16 de 1 de 2011.] http://www.supportickets.com/.

4. **iKom.** iKom. [En línea] [Citado el: 16 de 1 de 2011.] http://www.ikom.cl/Paginas/AG/AG1.aspx.

5. **Dinza, Tec. Prof. Dennys López y Enríquez Domínguez, Lic. Susana.** *SISTEMA DE CERTIFICACIÓN DE ORIGEN DIGITAL DE CUBA (SCOD-CUBA).* Ciudad Habana : s.n., 2010.

6. **Grupo de Soluciones de Innova.** Rational Unified Process. [En línea] [Citado el: 12 de 2 de 2011.] http://www.rational.com.ar/herramientas/rup.html.

7. **Jacobson Ivar, Booch Grady y Rumbaugh James.** *El Proceso Unificado de Desarrollo de Software.*

8. **Alarcón, Raúl.** *UML. Diseño Orientado a Objeto con UML.* Madrid. España : Grupo EIDOS.

9. **Ortiz, Antonio Moreno.** DISEÑO E IMPLEMENTACIÓN DE UN LEXICÓN COMPUTACIONAL PARA LEXICOGRAFÍA Y TRADUCCIÓN AUTOMÁTICA. *Estudios de Lingüística del Español.* [En línea] 2000. [Citado el: 10 de 6 de 2011.] http://ddd.uab.cat/pub/elies/elies_a2000v9/4-2.htm.

10. **Reyero, Eusebio.** Will Web For Food. [En línea] 15 de 6 de 2009. [Citado el: 18 de 1 de 2011.]

11. **García, Joaquín.** IngenieroSoftware. [En línea] 2005 de 5 de 27. [Citado el: 20 de 1 de 2011.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

12. **Larman, Craig.** *UML Y PATRONES. Introducción al análisis orientado a objetos.*

13. **Scribd.** Capítulo 1. Herramientas CASE. [En línea] [Citado el: 12 de 6 de 2011.] http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

14. Visual Paradigm International.UML, BPMN and Database Tool for Software Development. [En línea] [Citado el: 25 de 1 de 2011.] http://www.visual-paradigm.com/.

15. **Ochoa, Sergio.** *Introducción a los Patrones (Diseño y Arquitectura).* 2005.

16. **Potencier, Fabien y Zaninotto, François.** *Symfony la Guía Definitiva.* 2008.

17. *Manual de PHP.* 2010.

18. linuxcentro. [En línea] [Citado el: 28 de 1 de 2010.] http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.

Bibliografía

19. **Comunidad de Desarrollo de Ext JS.** Ext JS en Español. Comunidad de Desarrollo. [En línea] [Citado el: 5 de 2 de 2011.] http://extjs.es/.

20. **Escuela de Ingeniería Informática Universidad de Oviedo.** Entornos de Desarrollo Integrado. [En línea] 2008. [Citado el: 7 de 2 de 2011.] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

21. **Costilla, Carmen.** Características Objeto-Relacionales del Sistema de Gestión de Bases de Datos Oracle. [En línea] 2002. [Citado el: 8 de 2 de 2011.] Costilla, Carmen. Características Objeto-Relacionales del Sistema de Gestión de Bases de Datos Oracle. [En línea] 2002. http://sinbad.dit.upm.es/docencia/grado/curso0607/BDOR%20Primera%20Parte%20documento%20Oracle%20en%202002%E2%80%93Nov%202006.pdf..

22. **Oracle Corporation.** ORACLE. [En línea] [Citado el: 10 de 5 de 2011.] http://www.oracle.com/us/index.html..

23. **Guzmán, Rubén Darío García.** *El Proceso Unificado de Desarrollo de Software UP.* s.l. : Universidad de Caldas, 2011.

24. **IES Francisco Romero Vargas. Departamento de Informática.** *DISEÑO CONCEPTUAL: EL MODELO ENTIDAD-RELACION.*

25. **Carreón, Hugo, y otros, y otros.** sildeshare. [En línea] Marzo de 2010. [Citado el: 11 de Mayo de 2011.] http://www.slideshare.net/panchois/metricas-de-software.

26. **Ibarra González, Rosalina y Manresa Bernal, Annilié.** *Análisis y Diseño del Subsistema Contabilidad General del Sistema Integral de Gestión de Entidades.* Ciudad de la Habana : s.n., 2009.

27. **Dpto Aduana. CEIGE.** *Propuesta de Estándar de Codificación.*

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

29. **María Verónica, Lavieri y A. Carmen, Catellanos.** buenastareas. *Pruebas De Software.* [En línea] Noviembre de 2009. [Citado el: 15 de 5 de 2011.] http://www.buenastareas.com/ensayos/Pruebas-De-Software/312893.html.

30. **ELIZONDO, PERLA INÉS VELASCO.** *PRUEBA DE COMPONENTES DE SOFTWARE BASADAS EN EL MODELO DE JAVABEANS.* TLAXCALA : UNIVERSIDAD AUTÓNOMA DE TLAXCALA.

31. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico.* Sexta.

32. **Álvarez, José Luis Moreno.** UDLAP. *Aplicación de un Sistema Experto para el desarrollo de Sistema Evaluador del modelo Capability Maturity Model (CMM) niveles dos y tres. .* [En línea] 5 de 17 de 2004. [Citado el: 16 de 5 de 2011.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/moreno_a_jl/.

Bibliografía

33. **Aduana General de la República de Cuba.** SITIO WEB DE LA ADUANA CUBANA. [En línea] [Citado el: 10 de 1 de 2011.] http://www.aduana.co.cu/.

34. **Rosario, Jimmy.** La Tecnología de la Información y la Comunicación (TIC). Su uso como Herramienta para el Fortalecimiento y el Desarrollo de la Educación Virtual. *sibersociedad.* [En línea] 2005. [Citado el: 15 de 1 de 2011.] http://www.cibersociedad.net/archivo/articulo.php?art=218.

35. **García, José Manuel Rodríguez.** Informática para economistas. [En línea] 2005. [Citado el: 15 de 1 de 2011.] http://www.abecedario.com.es/editorial/novedades/fichas/capinfo.PDF.

36. **Campo, Dra. Nelsa María Sagaró del y Jiménez Paneque, Dra. C. Rosa.** Estado actual de la informatización de los procesos de evaluación de medios de diagnóstico y análisis de decisión clínica. [En línea] 2009. http://bvs.sld.cu/revistas/san/vol13_1_09/san12109.htm.

37. **Gamma, Erich, y otros, y otros.** *Design Patterns. Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley Pub Co, 95.

38. **Embarcadero Technologies.** *ER\Enterprise.*

39. **Oracle Corporation.** Portal del IDE Java de Código Abierto. [En línea] 2010. http://netbeans.org/index_es.html.

40. **González Henández, Yurisleydy.** *Implementación de la herramienta SeReq para el seguimiento de los requisitos en el Centro de Informatización de la Gestión de Entidades.* 2010.

41. **Rodríguez Pérez, Yisel y Céspedes Basteiro, Rafael Andrés.** *Diseño del Módulo Control de Personas del Sistema Único de Aduanas.* Ciudad de La Habana : s.n., 2009.

42. **Bravo Rodríguez, Julio Cesar.** *Sistema de Registro y Control de Portadores Energéticos.* Ciudad de la Habana : s.n., 2009.

43. **Manresa Bernal, Annilié y Ibarra González, Rosalina.** *Análisis y Diseño del Subsistema Contabilidad General del Sistema Integral de Gestión de Entidades.* Ciudad de la Habana : s.n., 2009.

44. **González Polanco, Liset y Pérez Betancourt, Yadian Guillermo.** *Análisis y Diseño del subsistema Préstamos del proyecto Modernización del Sistema Bancario Cubano.* La Habana : 2009.

ANEXOS

Anexo I Descripción del Modelo de Datos.

Nombre:S_Solicitud		
Descripción: Esta tabla es la encargada de almacenar los datos comunes de todas las solicitudes.		
Atributo	**Tipo**	**Descripción**
id_solicitud	numeric	Identificador de la Solicitud
fecha_aprobacion_rechazo	date	Fecha en la que se aprueba o rechaza la solicitud
fecha_vencimiento	date	Fecha en la que vence la solicitud
motivo	varchar	Motivo de la solicitud
fecha_solicitud	date	Fecha en la que se realiza la solicitud
anno	varchar	Año en el que se realiza la solicitud
id_estado_solicitud	numeric	Identificador del estado de la solicitud
id_tipo_solicitud	numeric	Identificador del tipo de solicitud
id_aduana	numeric	Identificador de la aduana desde la que se está solicitando.
id_entidad	numeric	Identificador de la entidad que está realizando la solicitud.
id_operacion_tipo_solicitud	numeric	Identificador de la relación entre solicitud y operación.

Tabla S_Solicitud

Nombre: S_Solicitud_Facilidad		
Descripción: Esta tabla es la encargada de almacenar los datos propios de las solicitudes de facilidad.		
Atributo	**Tipo**	**Descripción**
id_solicitud	numeric	Identificador de la Solicitud
plazo_solicitado	integer	Plazo solicitado
plazo_otorgado	integer	Plazo otorgado una vez que se aprueba la solicitud
numero_ga	varchar	Número de guía aérea que tiene la DM por la que se solicita
numero_bl	varchar	Número bl
numero_manifiesto	varchar	Número de manifiesto que tiene la partida
id_tipo_dm	numeric	Identificador de la DM por la que se solicita
id_lugar_embarque	numeric	Identificador del lugar de embarque
id_declarante	numeric	Identificador del declarante que solicita la facilidad.
id_motivo	numeric	Identificador del motivo de la facilidad

Tabla S_Solicitud_Facilidad

Nombre: S_Factura		
Descripción: Esta tabla es la encargada de almacenar los de las facturas asignadas a una solicitud de facilidad		
Atributo	**Tipo**	**Descripción**
id_factura	numeric	Identificador de la factura comercial
numero_factura	varchar	Número de la factura.
id_solicitud	numeric	Identificador de la Solicitud

Tabla S_Factura

Nombre: S_Liberacion_Cotejo		
Descripción: Esta tabla es la encargada de almacenar los datos propios de las solicitudes de liberación de cotejo		
Atributo	**Tipo**	**Descripción**
id_solicitud	numeric	Identificador de la Solicitud.
plazo_solicitado	integer	Plazo solicitado.
plazo_otorgado	integer	Plazo otorgado una vez que se aprueba la solicitud.
numero_ga	varchar	Número de guía aérea.
numero_bl	varchar	Número bl.
numero_manifiesto	varchar	Número de manifiesto.
id_lugar_embarque	numeric	Identificador del lugar de embarque.
id_declarante	numeric	Identificador del declarante que solicita la liberación.

Tabla S_Liberacion_Cotejo

Nombre: S_Liberacion_Motivo		
Descripción: Esta tabla es la encargada de almacenar los identificadores de los motivos de las solicitudes de liberación de cotejo correspondiente		
Atributo	**Tipo**	**Descripción**
id_liberacion_motivo	numeric	Identificador de la relación entre Liberación de Cotejo y motivo.
id_motivo	numeric	Identificador del motivo.
id_solicitud	numeric	Identificador de la Solicitud.

Tabla S_Liberacion_Motivo

Nombre: S_Solicitud_Temporalidad		
Descripción: Esta tabla es la encargada de almacenar los datos propios de las solicitudes de temporalidad.		
Atributo	**Tipo**	**Descripción**
id_solicitud	numeric	Identificador de la Solicitud.
carnet_identidad	varchar	Plazo solicitado.
nombre_solicitante	varchar	Plazo otorgado una vez que se aprueba la solicitud.
numero_manifiesto	varchar	Número de manifiesto.
apellido_solicitante	varchar	Identificador del lugar de embarque.

Tabla S_Solicitud_Temporalidad

Nombre: S_Dato_Mercancia		
Descripción: Esta tabla es la encargada de almacenar los datos propios de las solicitudes de temporalidad de cada mercancía.		
Atributo	**Tipo**	**Descripción**
id_dato_mercancia	numeric	Identificador de la mercancía en correspondencia con el régimen.
plazo_solicitado	integer	Plazo solicitado.
plazo_otorgado	integer	Plazo otorgado una vez que se aprueba la solicitud.
utiliza	bit	Se refiere a si la mercancía se utiliza o no.
fecha_vencimiento	date	Fecha en la que vence el plazo otorgado.
estado_mercancia	varchar	Se refiere al estado de la mercancía.
descripcion	varchar	Descripción de la mercancía.
quien_controla	varchar	Persona que controla la mercancía.
motivo_solicitud	varchar	Motivo por el que se solicita la mercancía.
id_mercancia	numeric	Identificador de la mercancía que se quiere solicitar.
id_regimen	numeric	Identificador del régimen para esa mercancía.
Id_solicitud	numeric	Identificador de la Solicitud.

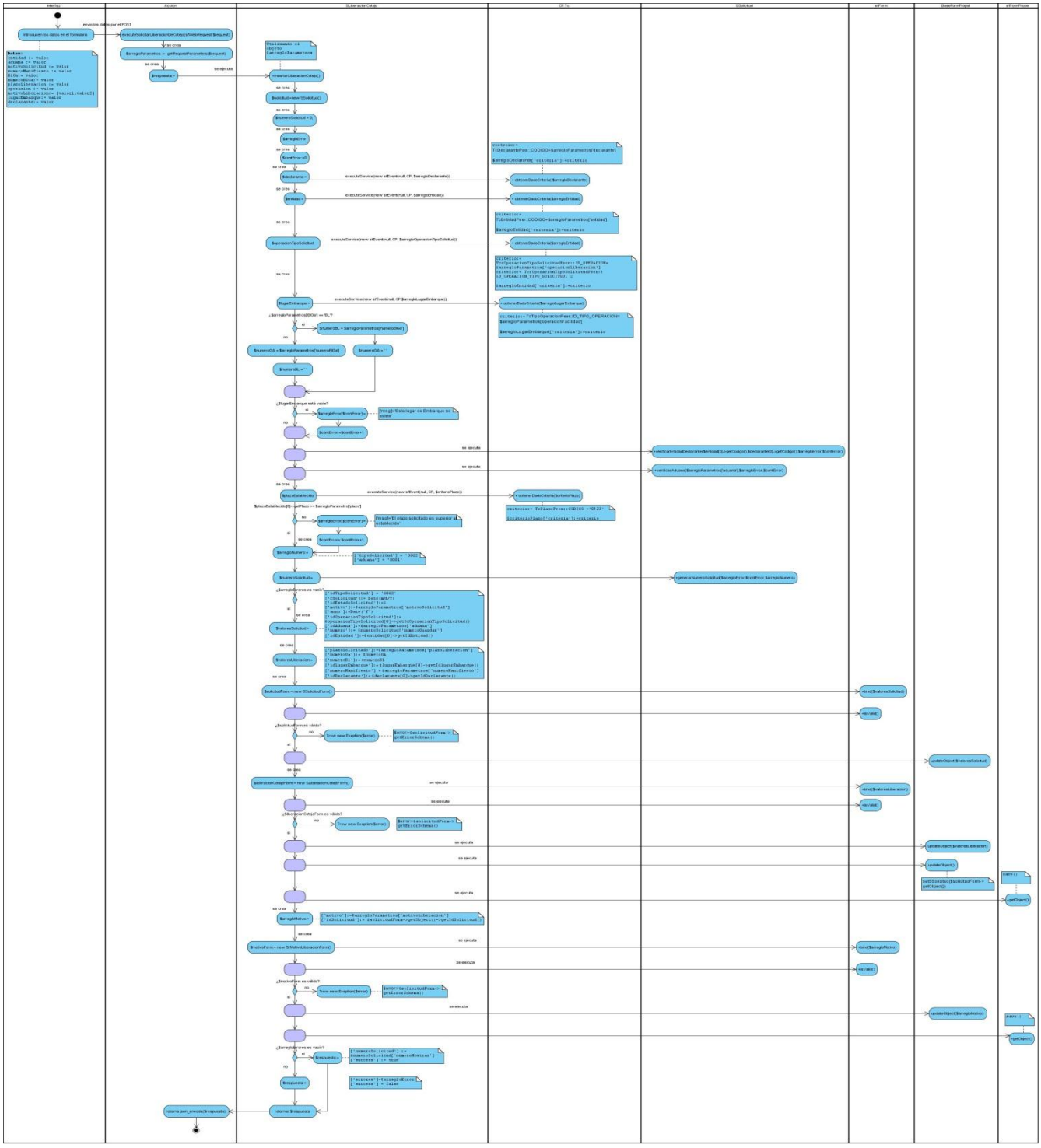
Tabla S_Dato_Mercancia

Nombre: S_Mercancia_Contacto		
Descripción: Esta tabla es la encargada de almacenar los identificadores de los contactos pertenecientes a las solicitudes de temporalidad realizadas para cada mercancía.		
Atributo	**Tipo**	**Descripción**
id_mercancia_contacto	numeric	Identificador de la relación entre contacto y la mercancía.
id_dato_mercancia	numeric	Identificador de la mercancía en correspondencia con el régimen.
id_contacto	numeric	Identificador del contacto.

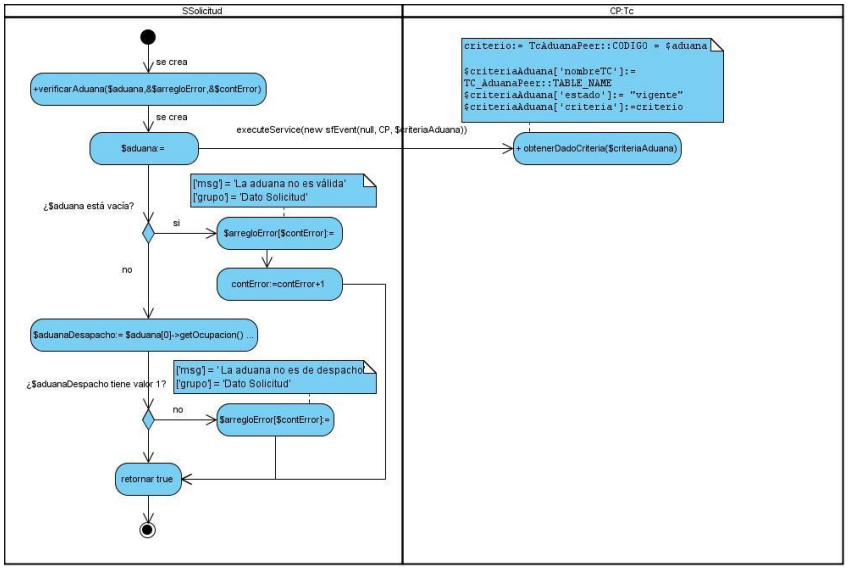
Tabla S_Mercancia_Contacto

ANEXOS

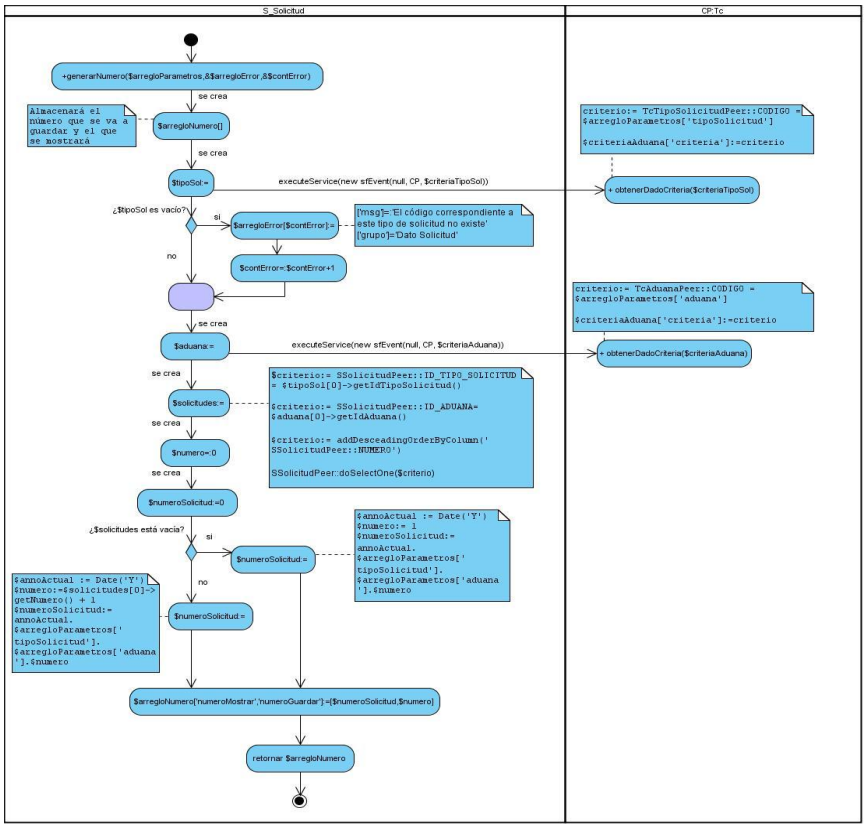
Anexo II Diagramas de Secuencia Orientados a Actividades del Negocio.



Diagramas de Secuencia Orientados a Actividades del Negocio del RF Solicitar Facilidad.

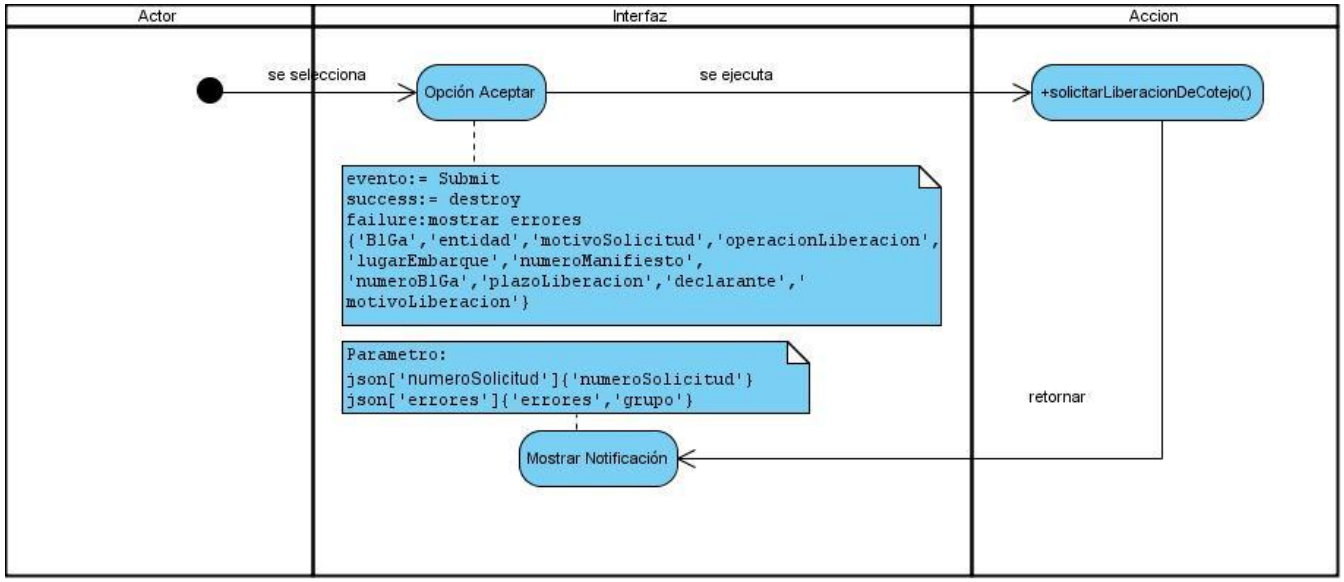


Diagramas de Secuencia Orientados a Actividades de la Funcionalidad Común Verificar Aduana.

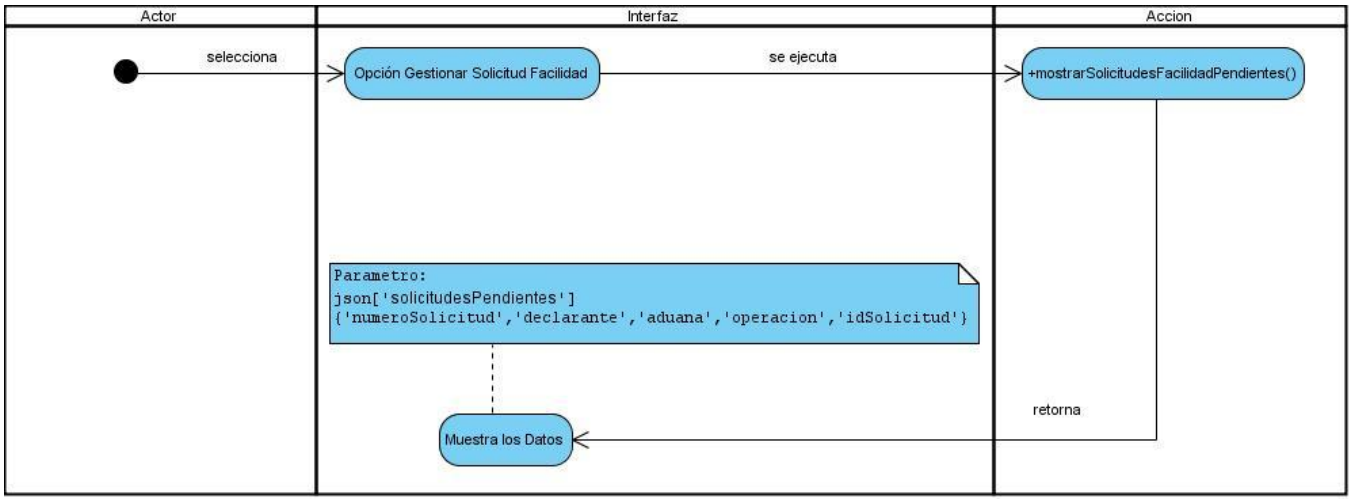


Diagramas de Secuencia Orientados a Actividades de la Funcionalidad Común Generar Número de Solicitud.

Anexo III Diagramas de Secuencia Orientados a Actividades de Componentes Visuales.



Diagramas de Secuencia Orientados a Actividades de Componentes Visuales del Componente Insertar Liberación de Cotejo.



Diagramas de Secuencia Orientados a Actividades de Componentes Visuales del Componente Mostrar Solicitudes de Facilidad Pendientes.

GLOSARIO

Aduana: La aduana es la oficina pública y/o fiscal que, a menudo bajo las órdenes de un estado o gobierno político, se establece en costas y fronteras con el propósito de registrar, administrar y regular el tráfico internacional de mercancías y productos que ingresan y egresan de un país.

Ajax: (*Asynchronous JavaScript And XML*) JavaScript asíncrono y XML, es una técnica de desarrollo web para crear aplicaciones interactivas. Esta se ejecuta en el cliente y mantiene comunicación asíncrona con el servidor en segundo plano. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

BPMN: Es una notación gráfica estandarizada que permite el modelado de procesos de negocio.

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.

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.

HTML: Siglas de *Hyper Text Markup Language* (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web.

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++.

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.

JSON: (*JavaScript Object Notation*) Notación de Objetos JavaScript, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La Aduana General de la República: Es la encargada de regular el control aduanero aplicable a la entrada, el tránsito, el cabotaje, el trasbordo, el depósito y la salida del territorio nacional de mercancías,

viajeros y sus equipajes, bienes y valores sujetos a regulaciones especiales, incluidas la flora y la fauna protegidas, así como los medios en que se transporten, además forma parte de la Administración del Estado y se subordina al Consejo de Ministros.

Linux: Es una implementación de libre distribución UNIX para computadoras personales (PC), servidores, y estaciones de trabajo.

.NET: Es un *framework* de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.

ORM: (*Object-Relational Mapping*) Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional.

POST: Es un método de petición que se refiere normalmente a la invocación de procesos que generan datos que serán devueltos como respuesta a la petición. Además se utiliza para aportar datos de entrada a esos programas.

Solicitudes en la Aduana: Se basa en el permiso solicitado por parte de los diferentes organismos externos a la aduana, dicha solicitud debe estar acompañada de un conjunto de documentación que respalde la misma, una vez realizada esta, los especialistas aduaneros se encargan de revisar toda la información correspondiente y deciden si la aprueban o rechazan. Fundamentando su decisión en políticas y restricciones definidas por el órgano aduanero correspondiente.

Solicitud de Facilidad: Es la solicitud de uno de los tipos de declaraciones de mercancías que existen en la aduana. La misma es realizada por una empresa cuando no cuenta con la información suficiente pero sí necesaria para importar o exportar la mercancía. En estos casos se solicita una facilidad y se le puede o no otorgar el permiso por un tiempo determinado en dependencia de que se cumpla con los parámetros requeridos por este proceso.

Solicitud de Liberación de Cotejo: El cotejo se basa fundamentalmente en la acción de comprobar la información referente a los datos del embarque de una mercancía determinada que es presentada por una entidad con el departamento de MTI (Medio de Transporte Internacional) de la aduana, de madera que se

compruebe si es válida o no la información presentada lo que determinará si es aprobada o rechazada la Solicitud de Liberación de Cotejo.

Solicitud de Temporalidad: Es la solicitud que se realiza por las diferentes entidades o empresas para utilizar una mercancía bajo uno de los regímenes temporales que existen en la aduana. Para realizar esta solicitud se debe presentar una serie de información necesaria de cada mercancía, la que es revisada por parte de los agentes aduaneros que son los que deciden si se aprueba o no el régimen temporal solicitado para cada una de las mercancías presentadas.

Tecnologías de la Información: Se denominan Tecnologías de la Información y las Comunicación (TIC) al conjunto de tecnologías que permiten la adquisición, producción, almacenamiento, tratamiento, comunicación, registro y presentación de informaciones, en forma de voz, imágenes y datos contenidos en señales de naturaleza acústica, óptica o electromagnética.

Unix: Es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

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.

WEB: Es un medio de comunicación de texto, gráficos y otros objetos multimedia a través de Internet, es decir, la web es un sistema de hipertexto que utiliza Internet como su mecanismo de transporte o desde otro punto de vista, una forma gráfica de explorar Internet.

Windows: Es el nombre de una serie de sistemas operativos desarrollados por Microsoft desde 1981.

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