UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

"Facultad 3"

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

Título: Versión 2.0 del sistema informático para la gestión de las solicitudes del Despacho Comercial en la Aduana General de la República de Cuba.

Autores: Lien Burgos Pérez
Maylevis Morejón Valdés

Tutores: Ing. Lianet Pineda de la Nuez
Ing. Yordani Cruz Segura
Ing. Eilys Pacheco Rodríguez

Ciudad de La Habana, Junio del 2012

Año 54 de la Revolución

"…el futuro de nuestra Patria, tiene que ser necesariamente un futuro de hombres de ciencias… "

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

Declaramos que somos las únicas autoras de este trabajo y autorizamos al Centro de Informatización para la Gestión de Entidades (CEIGE) de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2012.

_____ _____

Lien Burgos Pérez — Maylevis Morejón Valdés

Firma del Autor — Firma de Autor

_____ _____

Ing. Lianet Pineda de la Nuez — Ing. Yordani Cruz Segura

Firma del Tutor — Firma del Tutor

_____

Ing. Eilys Pacheco Rodríguez

Firma del Tutor

DATOS DE CONTACTO

Ing. Lianet Pineda de la Nuez. (lpineda@uci.cu) Graduada de Ingeniería en Ciencias Informáticas desde el año 2009. Instructor. Actualmente labora como analista del módulo Despacho Comercial, cuenta con 2 años de experiencia en el tema.

Ing. Eilys Pacheco Rodríguez. (epacheco@uci.cu) Graduada de Ingeniería en Ciencias Informáticas desde el año 2009. Instructor. 2 Años de experiencia en el tema.

Ing. Yordani Cruz Segura. (ysegura@uci.cu) Graduado de Ingeniería en Ciencias Informáticas desde el año 2011. Recién Graduado en Adiestramiento. 2 Años de experiencia en el tema.

Lien:

A mis padres por el amor y la preocupación, por sus consejos, sus las palabras de aliento, por apoyarme en todo y prepararme para enfrentar los desafíos de la vida.

A mi tata, por su cariño, por siempre confiar en mí, por apoyarme en todo momento, y ser mi fuente de inspiración a pesar de la lejanía.

A mi novio por la confianza y el amor que he encontrado en él, por ver mis problemas como suyo, por ser esa persona especial que me da la fuerza de seguir adelante, por entenderme y tener esa gran paciencia, por convertirse en el angelito que me enseño a luchar por lo que sé quiere sin miedo a fracasar, convirtiéndose el sol que me despierta todos los días.

A la familia tan maravillosa que tengo que nunca dejaron de confiar en mí, que han ayudado a lograr que mis ilusiones se conviertan en realidad.

A mis tutores por su guía y apoyo para la realización de este trabajo.

A todo el equipo de trabajo del departamento de Soluciones aduanera en especial a Weeden, Lázaro, Fernando, Ricardo, Leodan, Reinier, Roberto, Daniel por su ayuda en todo momento.

A Mailyn, Gaby, Liudmila por su apoyo, comprensión y darme la confianza de poder contar con ellas en todo momento.

A mis compañeros de aula por compartir conmigo durante estos 5 años y enseñarme que cada uno tiene algo que lo hace especial.

A mi compañera de tesis Maylevis Morejón por darme la oportunidad de formar un gran equipo de trabajo donde reinó el apoyo y la consagración constante y demostrarme que las amigas se encuentran en todo momento de la vida.

A la Profe Rosalina por no poner límites y estar siempre dispuesta ayudar.

A la Revolución y a Fidel por darme la oportunidad de convertirme en profesional.

A todos muchas gracias.

Maylevis:

Agradezco a la Revolución y en especial a Fidel, el cual ha sido el creador de este gran sueño.

A mi familia presente siempre en lo bueno y lo malo, por hacerme sentir orgullosa de tenerlos cercas. En especial a mi papá brindándome todo el tiempo su apoyo incondicional, mi mamá porque sin su guía y amor no hubiese podido ser la mujer que soy hoy.

A mis compañeros de aula, especialmente a los vandálicos: Carlos, Daniel, Bernardo y Andy, con los que he compartido los mejores momentos mi vida en la UCI, a Dianela que a pesar las distancias he aprendido mucho de ella, Gretel por apoyarme mucho sin ella saberlo dándome consejos muy preciados en situaciones especiales. Odelsis y Yolaida con quienes vengo juntas de 1er año y con las cuales pasé momentos divertidos y Eddie por haber sido el único que aguantó mis ñoñerías.

Al equipo de trabajo, en especial a papá Lázaro que desde que comencé en el proyecto estuvo apadrinándome y ayudándome, a Fernando, Ricardo y Weeden que nunca nos dieron de lado cuando necesitábamos ayuda. Roberto, Leodan, Reinier y Eddie que en muchas ocasiones dejaban lo que estaban haciendo para atendernos.

A Osmel y Lucia que son personas de las que he aprendido y quiero mucho, que me enseñaron a ser fuerte y a la vez delicada, a valerme por mi misma sin tener que depender de otras personas para alcanzar mis metas.

A mis tutores por su ayuda paciente e desinteresada

A mi novio, que en el poco tiempo que hemos compartido me ha brindado su apoyo incondicional, por convertirse en este tiempo en una persona muy especial, por entenderme y sobre todo tener paciencia

A la profesora Rosalina dispuesta siempre a ayudar, brindando experiencia y apoyo.

Y por último y más importante a mi compañera de tesis, Lien, que me dio la posibilidad de conocer a una amiga.

A todos muchas gracias.

Dedico el fruto de mis horas de estudio e investigación a mis abuelos que aunque ya no esté a mi lado, sé que hice su sueño realidad y que hoy estarían orgullosos de mí. A mi papa por su ejemplo de hombre honrado y excelente persona. A mi mama por ser una mujer sacrificada e íntegra en sus principios y sus convicciones. A mi hermana por darme la oportunidad de tener un ejemplo digno a seguir para lograr todas mis metas y a mi novio por ser tan especial en mi vida.

De Lien.

Dedico este trabajo a tres personas muy especiales: a mi papá porque ante cada tropiezo de mi vida siempre tuve de él un gesto de preocupación, de apoyo y de mucho amor, a mi tía Yucemit que siempre ha estado presente brindándome su apoyo y ayuda, jugando papel de madre y en especial a mi mamá, que siempre ha luchado junto a mí en estos cinco años de la carrera, para ti el mejor de los regalos.

De Maylevis.

RESUMEN

La gestión de las solicitudes realizadas en materia aduanera por parte de los diferentes organismos externos a la Aduana, dedicados a la exportación e importación de mercancías al país, es un proceso que se lleva a cabo de forma manual actualmente. Lo que trae como consecuencia que exista un gasto considerable de tiempo, así como un insuficiente control sobre las solicitudes y se evidencie un aumento del riesgo de pérdida de la información.

Como objetivo principal se tiene desarrollar un sistema informático que garantice la rapidez en la gestión de los procesos de negocio correspondientes a las solicitudes del Despacho Comercial de la Aduana General de la República. Con este propósito se realizó la investigación a sistemas informáticos que manejan este tipo de información, así como también 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 análisis, diseño e implementación, lográndose medir la calidad del trabajo realizado aplicando métricas al diseño y pruebas funcionales a la aplicación obtenida.

Como resultado final se obtuvo un sistema que facilita la gestión de la información asociada al proceso de aprobación y rechazo de las solicitudes en el Despacho Comercial. Este garantiza un registro de las solicitudes presentadas en la Aduana, así como una mayor rapidez en la realización de los procesos. Además de que es compatible con las nuevas tecnologías desplegadas en la Aduana.

Palabras clave: Aduana, Declarante, Despacho Comercial, GINA, Solicitudes.

TABLA DE CONTENIDOS

INTRODUCCIÓN

El mundo se encuentra en constante evolución con respecto a las Tecnologías de la Información y las Comunicaciones (TIC). Estas tecnologías han provocado un rápido acceso y producción de la información, optimizando los recursos y mejorando las operaciones tanto en las organizaciones como en la sociedad.

Cuba ha potenciado el desarrollo de estas tecnologías en aras de obtener mejores resultados. Con la creación de la Universidad de las Ciencias Informáticas (UCI) se pretende que la misma sirva de soporte a la industria cubana de la informática, llegando a tener actualmente el liderazgo en la producción de software para la informatización de todo el territorio cubano. Una de las instituciones nacionales más importantes que se encuentra estrechamente vinculada con la UCI, en la realización de diversas soluciones informáticas es la Aduana General de la República de Cuba (AGR).

La AGR es un Órgano de la Administración Central del Estado responsable de dirigir, ejecutar y controlar la aplicación de la política del Estado y del Gobierno en materia aduanera para el tráfico internacional de medios de transporte, mercancías y viajeros, con el objetivo de garantizar la protección y seguridad de la sociedad socialista y de la economía nacional.

La AGR está compuesta por diferentes áreas entre las cuales se puede destacar por su relevancia el Despacho Comercial (DC), encargado de regular el tráfico de mercancías de importación y exportación para brindar las estadísticas del comercio exterior.

En el área de DC se maneja la información referente a las solicitudes, las cuales son los permisos requeridos por parte de los diferentes organismos externos a la Aduana. Este proceso comienza cuando la persona, ya sea declarante[1] o director de la entidad interesada en solicitar estos permisos, lo hace de forma manual a través de un documento donde expone los motivos de sus intereses. Luego de elaborada la solicitud, el individuo debe dirigirse personalmente a la AGR para la presentación de la misma. Una vez entregado el documento, el responsable de recibirlo debe revisarlo porque en muchos casos no se completan todos los campos requeridos o los elementos que exponen contienen errores o incongruencias. De acuerdo a los resultados obtenidos, se procede a aceptar o rechazar dicha solicitud. Este proceso constituye por su forma de ejecución un gasto considerable de tiempo, así como un insuficiente control sobre las solicitudes y evidencia un aumento del riesgo de pérdida de la información.

En la UCI, específicamente en el Centro de Informatización para la Gestión de Entidades (CEIGE), se está realizando la informatización de todas las áreas de la AGR, con el desarrollo del Sistema de

[1] **Declarante:** representante de la entidad ante la Aduana.

Gestión Integral de Aduanas (GINA). En el año 2011 se incorporó el módulo Solicitudes al GINA, para el mismo se definieron e integraron 3 tipos de solicitudes referidas como: Facilidad, Liberación de cotejo y Régimen temporal. Después de realizar un profundo estudio de los procesos, se detectó la necesidad de incluir 11 nuevas solicitudes para darle continuidad al proceso de informatización de las entidades aduaneras. Por otra parte, en la solución que actualmente posee GINA, existe una interacción directa por parte del declarante con la aplicación, donde los niveles de acceso y permiso son definidos solamente para el personal aduanero. En aras de darle solución a esta última vicisitud, surge el sistema de Ventanilla Única (VU) con el objetivo de tener un único punto de entrada de información digital y estandarizada para las operaciones comerciales controladas bajo regímenes aduaneros. La VU tendrá una total retroalimentación y comunicación con el GINA y a su vez con el módulo Solicitudes.

Teniendo en cuenta la problemática anteriormente expuesta se puede arribar al siguiente **problema a resolver:** los procesos de negocio correspondientes a las solicitudes que se realizan actualmente en la Aduana General de la República afectan la rapidez de los procesos del Despacho Comercial.

Trazando como **objeto de estudio:** los procesos de negocio correspondientes a las solicitudes.

Enfocándose en el siguiente **campo de acción:** los procesos para la gestión de las solicitudes en el Despacho Comercial.

Para dar solución al problema mencionado anteriormente se plantea como **objetivo general:** desarrollar un sistema informático que garantice la rapidez de la gestión de los procesos de negocio, correspondientes a las solicitudes del Despacho Comercial de la Aduana General de la República de Cuba.

Para darle cumplimiento al objetivo general previamente descrito, se definen los siguientes **objetivos específicos:**

- Elaborar el marco teórico relativo a la investigación, para la construcción del estado del arte.
- Modelar la ingeniería de requisitos para garantizar el diseño del sistema que se desea desarrollar.
- Diseñar e implementar la solución modelada de manera que cumpla con las necesidades del cliente.
- Validar la solución mediante pruebas aplicadas al sistema para garantizar la calidad del software.

Los objetivos específicos perfilados en este trabajo, pueden desglosarse en las siguientes **tareas de investigación:**

- Elaboración de la fundamentación teórica y la revisión bibliográfica.
- Estudio de las herramientas para darle cumplimiento a los objetivos.
- Modelado de procesos, descripción de procesos e identificación de reglas de negocio del módulo Solicitudes de Aduana en su versión 2.0.
- Elaboración y descripción del modelo conceptual del módulo Solicitudes de Aduana en su versión 2.0.
- Especificación de los requisitos correspondientes a la gestión del módulo Solicitudes de Aduana en su versión 2.0.
- Elaboración del modelo de datos del módulo Solicitudes de Aduana en su versión 2.0.
- Elaboración del diagrama de clases del diseño de las solicitudes en su versión 2.0.
- Elaboración de los diagramas de secuencias orientados a actividades del módulo Solicitudes de Aduana en su versión 2.0.
- Elaboración del diagrama de paquetes.
- Diseño e implementación de las interfaces del módulo Solicitudes en su versión 2.0.
- Implementación de las clases del negocio de las solicitudes en su versión 2.0.
- Elaboración del diagrama de componentes del módulo Solicitudes en su versión 2.0.
- Realización de pruebas funcionales al sistema.
- Realización de la validación con los clientes de la solución obtenida.
- Documentación de las pruebas realizadas.

Con la finalidad de proporcionar un mejor desarrollo del presente trabajo durante la investigación, se utilizarán varios métodos científicos:

Analítico – sintético: este método es utilizado con el fin de recopilar a través de un estudio detallado, la de información más importante a utilizar.

Lógico – histórico: este método permite estudiar la evolución y desarrollo de la información y los conocimientos que serán necesario emplear durante toda la realización del trabajo.

Modelación: este método tiene como objetivo fundamental hacer una reproducción simplificada de la realidad. Un ejemplo fehaciente del mismo se manifiesta en todos los modelos y diagramas a presentar durante el análisis, diseño e implementación del módulo Solicitudes de Aduana en su versión 2.0.

Entrevista: este método es uno de los más importantes ya que permite una relación directa con el cliente; tiene como objetivo captar los detalles no explícitos en la documentación a consultar, además de precisar las necesidades y demandas del cliente en función de la solución al problema que es presentado.

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. (1) Este método es uno de los más practicados pues será empleado durante el proceso de implementación para obtener el sistema que controle las solicitudes realizadas a la AGR.

Para un mejor entendimiento de esta investigación se estructura el documento de la siguiente forma:

Para un mejor entendimiento de esta investigación se estructura el documento de la siguiente forma:

Capítulo 1: Fundamentación teórica

Este capítulo incluye un estado del arte de algunos sistemas que se utilizan para la gestión de solicitudes tanto a nivel nacional como internacional. Además de realizar un estudio de las herramientas, lenguajes y metodologías de desarrollo utilizadas en el proyecto.

Capítulo 2: Descripción de la solución propuesta

Se identifican y describen los procesos de negocio de forma detallada, incluyendo las reglas del negocio, además de la especificación de los requisitos funcionales. Se muestran los aspectos principales del diseño de la solución propuesta.

Capítulo 3: Implementación y validación del sistema

En este capítulo se abordan temas correspondientes al desarrollo de la solución. Realizándose pruebas al producto y dando a conocer el impacto que han tenido en la solución del problema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Tanto por el valor de las tecnologías, como por los volúmenes de información que se manejan en la sociedad moderna, resulta importante organizarla de forma eficiente con el fin de manipularla con mayor facilidad. Aún después de organizarse la información es de vital importancia la rapidez con que se pueda recuperar la misma, de esta forma el hombre tiene que recurrir a los avances de la ciencia para poder procesar, organizar y recuperar grandes volúmenes de información con el mínimo costo y tiempo posibles.

En el capítulo se realiza un estudio sobre los sistemas informáticos tanto del ámbito nacional como internacional que gestionan solicitudes, con el objetivo de sustentar la solución en desarrollo con las mejores características de los mismos, en función de complementar las especificidades requeridas por el cliente. Se explican además, los resultados de la investigación realizada, las herramientas, modelo de desarrollo utilizado y lenguajes analizados para el desarrollo del sistema.

1.2 Sistemas informáticos para la gestión de las solicitudes.

Existen varios sistemas informáticos utilizados en diferentes organismos y empresas que gestionan diversos tipos de solicitudes. Cada uno de ellos están orientados específicamente a las necesidades y requisitos de las distintas entidades en las cuales son empleados, ejemplo de estos sistemas son los presentados a continuación:

1.2.1 Sistema Informático de Solicitud de Convalidaciones de la Dirección Nacional de Gestión Universitaria (SISCo)

SISCo es un sistema informático utilizado por el ministerio de educación de Argentina. (2)

Está dirigido a profesionales extranjeros provenientes de alguno de los países con los que el ministerio posee un convenio de reconocimiento recíproco de títulos universitarios y deseen acceder a otros estudios o ejercer la profesión. (3)

SISCo tiene como ventaja la gestión de documentos entrantes y documentos salientes, brinda información interna y externa. Mantiene actualizada la información sobre gestores de las universidades. (3)

La principal desventaja que presenta este producto a la hora de considerarlo como solución al problema que se plantea es que está solamente concebido para solicitudes de títulos universitarios, por lo que no cumple con los requisitos planteados por la AGR; además es un programa privativo y

provocaría una contradicción con las políticas del país y de la Universidad en aras de fomentar el software libre.

1.2.2 Supportickets

Supportickets es un programa de gestión de solicitudes de soporte y conocimientos. Ofrece un módulo de administración para solicitudes y correo electrónico y proporciona métodos de gestión para manejar varias clases de bases de conocimientos. (4)

Este sistema es desarrollado con PHP (acrónimo de PHP: Hypertext Preprocessor) y MySQL[2]. Presenta tres interfaces y además un módulo de administración desde el cual se puede responder pedidos, asignar permisos y obtener estadísticas. Es multiplataforma, bajo licencia comercial Shareware[3]. (4)

Como ventajas, Supportickets permite a los usuarios mantener organizados los documentos, agregar adjuntos, notas, descargar y guardar páginas web; además puede construir fácilmente una base con los conocimientos de toda la organización. (4)

Para muchos clientes resulta ventajoso generar actualizaciones de solicitudes por medio de correo electrónico o formularios web en línea, pero para la AGR es una desventaja puesto que se requiere que la realización de las solicitudes sea a través del sistema de VU.

1.2.3 Sistema informático de gestión de ayudas, becas y premios del Ministerio de Cultura (SAB)

SAB es un sistema informático para la tramitación y gestión de ayudas, becas y premios del Ministerio de Cultura de España. Permite la presentación de solicitudes por vía electrónica y realiza una gestión conjunta de todas las solicitudes de una determinada convocatoria de subvención, con independencia de la vía de entrada de la solicitud al sistema. (5)

SAB brinda soporte a todo el proceso, tanto desde el punto de vista del ciudadano como del gestor. Utiliza los servicios proporcionados por otros departamentos de la administración para la validación de certificados electrónicos, notificaciones electrónicas y verificación de datos de identidad, y facilita la consulta a la Agencia Tributaria y Seguridad Social de datos de situación de deuda de los solicitantes. (5)

[2] **MySQL:** es un sistema de gestión de bases de datos.
[3] **Shareware:** esta licencia se expide gratuitamente con el fin de probar la librería antes de adquirirla.

Ventajas: (5)

- Uniformidad en la gestión de todas las convocatorias de subvenciones, en cuanto a normalización de formularios y procesos.
- Flexibilidad de configuración para dar respuesta a necesidades más específicas.
- Automatización de todas las fases del procedimiento.
- Centralización de la información en un único repositorio de datos, facilitando la explotación de datos y el uso de herramientas de toma de decisiones por parte de la dirección.

Teniendo en cuenta las características de SAB, se determina que el mismo no puede formar parte de la solución porque maneja un conjunto de información ajena a los objetivos de este trabajo como es el caso de las descargas de archivos, además de que entre sus requisitos se encuentra: "Es preciso la utilización de la plataforma @firma[4] del Ministerio de Presidencia, para la validación de certificados electrónicos[5]", donde el uso de esta plataforma no es requerido por la AGR. Además permite la realización de las solicitudes a través de correo electrónico, siendo la VU el sistema destinado para realización de las solicitudes aduaneras.

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

El SCOD-CUBA es capaz de gestionar las solicitudes de Certificación de Origen Digital (COD) y permite informatizar procesos para la confección del mismo y un ahorro por tramitación. (6)

Este sistema 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. (6)

Este sistema, 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 AGR. (1)

1.3 Lenguajes, modelo de desarrollo y herramientas

Con el fin de obtener un buen desarrollo del software se debe tener en cuenta qué tecnologías se utilizarán en aras de alcanzar buenos resultados y lograr las metas propuestas. El Departamento de

[4] **@firma:** solución tecnológica que se basa la implementación de la plataforma de validación y firma electrónica del Ministerio de Presidencia de España.

[5] **Certificados electrónicos:** documento firmado electrónicamente por un prestador de servicios de certificación a un firmante y confirma su identidad.

Soluciones para la Aduana perteneciente al centro CEIGE tiene definidas las tecnologías a usar para su desarrollo, basándose en las necesidades del Departamento, las cuales son las que se profundizan a continuación:

1.3.1 Modelo de desarrollo de software

Un modelo de desarrollo es un conjunto de actividades dentro del marco de trabajo para realizar los procesos de software. Ejecuta cada actividad con un conjunto de acciones de ingeniería de software y define cada acción en cuanto a un número de tareas que identifique el trabajo que debe completarse para alcanzar las metas de desarrollo.(39)

El Departamento de Soluciones para la Aduana tiene su propio modelo de desarrollo. El ciclo de vida de este modelo está estructurado de la siguiente forma: Estudio Preliminar, Modelación del Negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas Pilotos, Pruebas Internas, Pruebas de Liberación y Despliegue. Cada una de estas fases está descrita en el documento del Modelo de Desarrollo del Departamento Soluciones para la Aduana. Además se especifica la estructura del proyecto así como los roles y responsabilidades a desempeñar. (26)

1.3.2 Notación para el modelado de procesos de negocio

BPMN (Business Process Modeling Notation por sus siglas en inglés) es una notación gráfica que describe la lógica de los pasos de un proceso de negocio. El principal objetivo de BPMN es proveer una notación estándar que sea fácilmente entendible por parte de todos los involucrados e interesados del negocio (ya sean los analistas o los usuarios del negocio). Tiene la finalidad de servir como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación. BPMN define un Diagrama de Procesos de Negocio (BPD), basado en la técnica de diagramado de flujos, que ajusta modelos gráficos de operación de procesos de negocio. Un modelo de procesos de negocio será una red de objetos gráficos, correspondientes a actividades y controles de flujo que definen su orden de ejecución. (8)

Los diagramas de procesos de negocio se estructuran con un grupo de elementos gráficos. Específicamente se divide en las cuatro categorías básicas de elementos siguientes: (8)

- **Objetos de flujo:** son los principales elementos gráficos que definen el comportamiento de los procesos. Se agrupan en: Eventos, Actividades y Nodos de Decisiones o Compuertas (Gateways).
- **Objetos de conexión:** son los elementos usados para conectar dos objetos del flujo dentro de un proceso. Dentro de esta categoría se encuentran líneas de secuencia, asociaciones y líneas de mensaje.

- **Canales**: son elementos utilizados para organizar las actividades del flujo en diferentes categorías visuales que representan áreas funcionales, roles o responsabilidades. Los mismos son las Piscinas (Pool) y Calles (Lane).
- **Artefactos:** los artefactos son usados para proveer información adicional sobre el proceso. Los mismos son Objetos de datos, Grupo y Anotación de texto.

1.3.3 Lenguaje de modelado

UML (Unified Modeling Language o Lenguaje Unificado de Modelado) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Es importante recalcar que UML no es una guía para realizar el análisis y diseño orientado a objetos sino es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. (7)

Es además, un lenguaje que proporciona un vocabulario y reglas para permitir una comunicación. Se centra en la representación gráfica de un sistema e indica cómo crear y leer los modelos. Otro objetivo de este modelado visual es que sea independiente del lenguaje de implementación. (7)

Entre sus principales funciones se destacan:(9)

- **Visualizar**: permite expresar de una forma gráfica un sistema para que otro lo pueda entender.
- **Especificar**: permite especificar las características de un sistema antes de su construcción.
- **Construir**: a partir de los modelos especificados se puede construir los sistemas diseñados.
- **Documentar**: los propios elementos gráficos sirven como documentación del sistema desarrollado, sirviendo para su futura revisión.

1.3.4 Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering por sus siglas en inglés) son 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. Dichas herramientas son utilizadas en la actualidad por muchas empresas, con el fin de modelar los aspectos claves de todo el proceso de desarrollo de un sistema.

Una de estas herramientas es el Visual Paradigm para UML la cual soporta el ciclo de vida completo de desarrollo de software: análisis y diseño orientado a objetos, construcción, prueba y despliegue.

Dentro de las características distintivas se pueden mencionar: (10)

- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Disponibilidad de integrarse en los principales IDEs (Entorno de Desarrollo Integrado o Integrated Development Environment, por sus siglas en inglés).
- Disponibilidad en múltiples plataformas.

1.3.5 Lenguaje de programación

PHP es un lenguaje interpretado[6] de alto nivel[7], diseñado originalmente para la creación de páginas web dinámicas, es usado principalmente para la interpretación del lado del servidor. PHP ha sido especialmente creado para el desarrollo de páginas Web dinámicas y puede ser incluido con facilidad dentro del código HTML (Hyper Text Markup Language o lenguaje de marcado de hipertexto). Ha alcanzado gran popularidad y existe una amplia comunidad de desarrolladores y programadores que continuamente implementan mejoras en su código. (11)

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, es que permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones. (12)

PHP cuenta con cuatro grandes características que lo distingue entre los demás lenguajes de programación: (13)

- **Velocidad:** PHP no solo es rápido al ser ejecutado sino que no genera retrasos en la máquina, por esto no requiere grandes recursos del sistema. PHP se integra muy bien junto a otras aplicaciones, especialmente bajo ambientes Unix.
- **Estabilidad:** PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.
- **Seguridad:** PHP provee diferentes niveles de seguridad, que pueden ser configurados por archivos que se encuentran en el servidor.

[6] **Lenguaje interpretado:** cada vez que se usa el programa debe utilizarse un traductor llamado "intérprete" que se encarga de traducir ("interpretar") las instrucciones del programa original ("código fuente") a código máquina según van siendo utilizadas. Para el funcionamiento del programa siempre es necesario disponer del código original y del intérprete

[7] **Lenguaje de alto nivel:** son aquellos que se encuentran más cercanos al lenguaje natural que al lenguaje máquina. Son independientes de la arquitectura del ordenador porque se puede migrar de una máquina a otra sin problema.

- **Simplicidad:** se les debe permitir a los programadores generar código productivamente en el menor tiempo posible. Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente.

Otra característica que cabe citar además es la conectividad, PHP dispone de una amplia gama de librerías, y extensiones para ampliar su alcance. Esto le permite al PHP ser utilizado en muchas áreas diferentes, tales como encriptado, gráficos y XML (Extensible Markup Language o lenguaje de marcas extensible).

1.3.6 Marcos de trabajo

1.3.6.1 Symfony

Symfony es un marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web. Este 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, informatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (14)

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas Unix, Linux y Windows. (14)

Symfony se diseñó para que se ajustara a los siguientes requisitos: (14)

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

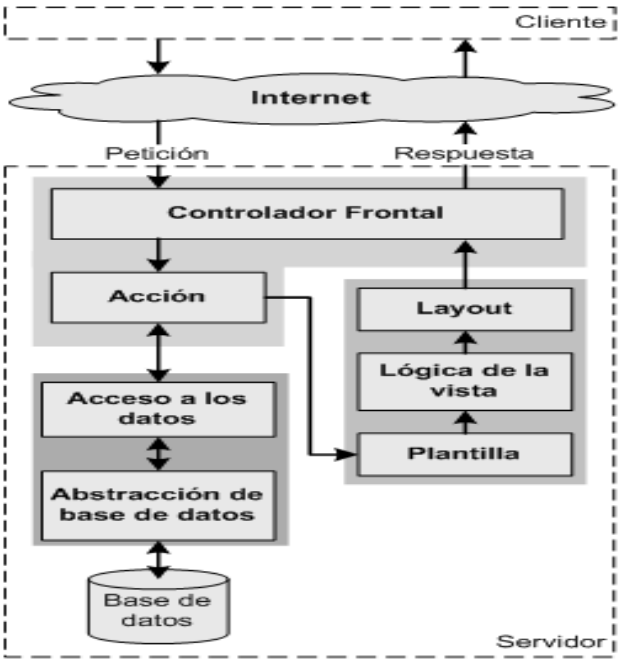


Figura 1.1 El flujo de trabajo de Symfony (14)

1.3.6.2 Ext JS

Ext JS es un marco de trabajo diseñado para la creación de páginas web dinámicas del lado del cliente, basado en lenguaje JavaScript para el desarrollo de aplicaciones web interactivas que además de flexibilizar el manejo de componentes de la página como el DOM (Document Object Model por sus siglas en inglés), Peticiones Ajax (acrónimo de Asynchronous JavaScript And XML), DHTML (Dynamic HyperText Markup Language por sus siglas en inglés), tiene la gran funcionalidad de crear interfaces de usuario bastante funcionales. (15)

Este marco de trabajo incluye:

- Componentes de interfaz gráfica de usuario del alto rendimiento y personalizables.
- Modelo de componentes extensibles.
- Licencias Open Source (GPL[8]) y comerciales.

Algunas ventajas que presenta:(16)

- Permite crear aplicaciones complejas utilizando componentes predefinidos.
- Evita el problema de tener que validar el código para que funcione bien en cada uno de los navegadores (Firefox, Internet Explorer, Safari, entre otros).

[8]**GPL:** licencia que 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.

- El funcionamiento de las ventanas flotantes lo pone por encima de cualquier otro.
- Relación entre Cliente-Servidor balanceado: Se distribuye la carga de procesamiento, permitiendo que el servidor pueda atender más clientes al mismo tiempo.
- Eficiencia de la red: Disminuye el tráfico en la red pues las aplicaciones cuentan con la posibilidad de elegir que datos desea trasmitir al servidor y viceversa (Criterio este que puede variar con el uso de aplicaciones de pre-carga).

1.3.7 IDE de desarrollo

El NetBeans IDE es una herramienta para programadores desarrollada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, es un producto libre, gratuito y de código abierto escrito sin restricciones de uso. El NetBeans IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles).

Este software contiene las herramientas necesarias para que los desarrolladores puedan crear aplicaciones de escritorio, empresariales, web, y aplicaciones móviles, con el lenguaje Java, así como también C/C++, PHP, JavaScript, Groovy, y Ruby. Posee un amplio soporte para el lenguaje PHP así como para los marcos de trabajo de trabajo Symfony y Ext JS.

1.3.8 Sistema Gestor de Base de Datos (SGBD)

Un SGBD es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Permite definir los datos a distintos niveles de abstracción y manipularlos, garantizando la seguridad e integridad de los mismos. (17)

Con Oracle 11g se reducen los costos informáticos y es posible ofrecer una calidad de servicio superior: (18)

- Consolidando las aplicaciones de negocio en redes de bases de datos rápidas, fiables y ampliables.
- Maximizando la disponibilidad y eliminando la redundancia del centro de datos inactivo.
- Comprimiendo datos en particiones de almacenamiento de bajo costo para un rendimiento más rápido.
- Protegiendo con seguridad la información y permitiendo el cumplimiento.

Entre las principales ventajas que Oracle presenta se destacan: (18)

- 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.
- 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.
- Las aplicaciones que utilizan objetos de Oracle son fáciles de entender y mantener porque soportan las características del paradigma orientado a objetos.
- Tiene buen rendimiento y hace buen uso de los recursos.
- Posee un rico diccionario de datos.
- Brinda soporte a la mayoría de los lenguajes de programación.
- Es un sistema multiplataforma, disponible en Windows, Linux y Unix.
- 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.

La AGR usa este software desde el año 1997, período en el cual ha obtenido experiencias positivas en cuanto al uso de dicho producto. Es la única herramienta privativa que se ha ganado un lugar en la AGR debido a sus insustituibles características.

1.4 Ingeniería de requisitos

La Ingeniería de Requisitos (IR) en sí cumple un papel primordial en el proceso de construcción de un software. Este estará basado en función de las necesidades planteadas por los clientes en un nivel muy general, donde se documentan, analizan y se definen los servicios o componentes del producto a desarrollar, además de las restricciones que tendrá dicha aplicación. Su principal tarea consiste en la definición del proceso a seguir en la construcción de un software, y de facilitar la comprensión de lo que el cliente requiera. La obtención correcta de los requisitos puede llegar a describir con claridad, sin ambigüedades, en forma consistente y compacta, el comportamiento de un sistema. (19)

Existen varios conceptos o significados acerca de la IR que proporcionan varios autores según su nivel de experiencia, el presente trabajo se basa en la definición propuesta por Roger S. Pressman en su libro Ingeniería del Software Un Enfoque Práctico:

- Proporciona el mecanismo apropiado para entender lo que el cliente quiere, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin

ambigüedades, validar la especificación y administrar los requisitos conforme ésos se transforman en un sistema operacional. (20)

1.4.1 Requisitos

Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. (19)

Los requisitos se pueden clasificar en funcionales y no funcionales:(21)

- Requisitos funcionales: son los que definen las funciones que el sistema será capaz de ejecutar, describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Es importante que se describa el ¿Qué? y no el ¿Cómo? se deben hacer esas transformaciones. Estos requisitos al tiempo que avanza el proyecto de software se convierten en los algoritmos, la lógica y gran parte del código del sistema.
- Requisitos no funcionales: tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad.

1.4.2 Técnicas de captura de requisitos

La captura de requisitos es de vital importancia debido a que permite gestionar las necesidades del proyecto de forma estructurada. Es un eslabón principal en la calidad del software pues si se obtienen todos los requisitos que desee el cliente se alcanzará la satisfacción de los mismos. A continuación se presenta un grupo de técnicas que se utilizan en el proyecto.

- **Entrevistas**: es una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Básicamente, la estructura de la entrevista abarca cuatro pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados. (22)
- **Observación**: permite obtener hechos que no los puede adquirir de ninguna otra forma. Además proporciona información en relación con la forma en que se llevan a cabo las actividades, no es lo mismo que se diga cómo se realiza un proceso determinado a que el analista esté presente en el mismo. (24)

- **Tormenta de ideas**: Es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios. Puede ayudar a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de captura, cuando los requisitos son todavía muy difusos. También se requiere participación intensiva del analista. (23)
- **Talleres**: son reuniones con sesiones intensivas y estructuradas concentradas en uno o dos días con las partes interesadas. Es precisa una preparación previa para definir con los participantes la finalidad del taller, facilitarles la información histórica, además de que el taller ha de ser dirigido por un experto para garantizar que todos los participantes aportan sus puntos de vista y no desviarse del propósito del mismo. (24)

1.4.3 Técnicas de validación de los requisitos

- **Revisiones del documento de requisitos:** Una revisión de requisitos es un proceso manual que involucra a varios lectores, tanto del cliente como de los contratistas. Se verifica el documento en cuanto a anomalías y omisiones. Estas revisiones pueden ser informales o formales. Los informales simplemente implican que los contratistas deben tratar los requisitos con tantos stakeholders[9] del sistema como sea posible. En la revisión formal el equipo de desarrollo debe conducir al cliente a través de los requisitos del sistema, explicándoles las implicaciones de cada uno. (19)
- **Prototipos:** Los prototipos son simulaciones del posible producto, que luego son utilizados por el usuario final, permitiendo conseguir una importante retroalimentación en cuanto a si el sistema diseñado con base a los requisitos recolectados le permite al usuario realizar su trabajo. (21)
- **Matrices de trazabilidad**: Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo los objetivos que cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos. (25)
- **Generación de casos de pruebas:** Los requisitos deberán ser probados. Si las pruebas pueden ser consideradas como parte del proceso de validación, esto frecuentemente permite describir los problemas en los requisitos y posteriormente encontrar soluciones acordes con los mismos. (19)

[9] **Stakeholders**: individuo, grupo u organización relacionado con el sistema.

1.5. Patrones

1.5.1. Patrones de diseño

Un patrón de diseño define un esquema de refinamiento de los subsistemas o componentes dentro de un sistema, o las relaciones entre estos. Describe una estructura común y recurrente de componentes interrelacionados, que resuelve un problema general de diseño dentro de un contexto particular. (27)

Debido a la utilización del marco de trabajo Symfony el sistema a desarrollar implementa varios patrones de diseño, ofreciendo a los desarrolladores la utilización de buenas prácticas en la implementación.

Entre los distintos patrones de diseño existentes, los utilizados en el desarrollo de la solución pertenecen a los GRASP (General Responsibility Asignment Software Patterns, en español Patrones Generales de Software para Asignar Responsabilidades) y a los patrones GOF (Gang of Four, en español Pandilla de los Cuatro, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides).

1.5.1.1. Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (28)

Patrones básicos de asignación de responsabilidades:

1. Experto

Solución: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. (28)

Problema que resuelve: ¿Cuál es el principio fundamental en virtud del cual asignan las responsabilidades a los objetos? (28)

Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades. Durante el diseño orientado a objetos, cuando se definen las interacciones entre los objetos, se toman decisiones sobre la asignación de responsabilidades a las clases. Si se hacen en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se presenta la oportunidad de reutilizar los componentes en futuras aplicaciones. (28)

2. Creador

Solución: Asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos: (28)

1. B contiene los objetos A.
2. B agrega los objetos A.
3. B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado.
4. B registra las instancias de los objetos A.
5. B utiliza específicamente los objetos A.

Problema: ¿Quién debería ser responsable de crear una nueva instancia de alguna clase? (28)

La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. (28)

3. Bajo Acoplamiento:

Solución: Asignar una responsabilidad para mantener bajo acoplamiento. (28)

Problema: ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización? (28)

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. (28)

Una clase con alto (o fuerte) acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente; presentan los siguientes problemas: (28)

- Los cambios de las clases afines ocasionan cambios locales.
- Son más difíciles de entender cuando están aisladas.
- Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

4. Alta Cohesión:

Solución: Asignar una responsabilidad de modo que la cohesión siga siendo alta. (28)

Problema: ¿Cómo mantener la complejidad dentro de los límites manejables? (28)

En la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. (28)

Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. No conviene este tipo de clases pues presentan los siguientes problemas: (28)

- Son difíciles de comprender.
- Son difíciles de reutilizar.
- Son difíciles de conservar.
- Son delicadas: las afectan constantemente los cambios.

Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos. (28)

5. Controlador:

Solución: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones: (28)

- La empresa u la organización global (un controlador de fachada).
- El "sistema" global (un controlador de fachada).
- Algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).
- Un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados “Manejador<NombreCasoUso>” (controlador de casos de usos).

Problema: ¿Quién debería encargarse de atender un evento del sistema? (28)

Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del sistema. (28)

Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. (28)

1.5.1.2. Patrones GOF:

Estos patrones tienen como características: (29)

- Son soluciones concretas. Proponen soluciones a problemas concretos, no son teorías genéricas.
- Son soluciones técnicas. Indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje.
- Se utilizan en situaciones frecuentes debido a que se basan en la experiencia acumulada al resolver problemas reiterativos.
- Favorecen la reutilización de código. Ayudan a construir software basado en la reutilización, a construir clases reutilizables. Los propios patrones se reutilizan cada vez que se vuelven a aplicar.
- El uso de un patrón no se refleja en el código. Al aplicar un patrón, el código resultante no tiene por qué delatar el patrón o patrones que lo inspiró. No obstante últimamente hay múltiples esfuerzos enfocados a la construcción de herramientas de desarrollo basados en los patrones y frecuentemente se incluye en los nombres de las clases el nombre del patrón en que se basan facilitando así la comunicación entre desarrolladores.
- Es difícil reutilizar la implementación de un patrón. Al aplicar un patrón aparecen clases concretas que solucionan un problema concreto y que no será aplicable a otros problemas que requieran el mismo patrón.

Los patrones GOF se clasifican en 3 categorías: creacionales, estructurales y de comportamiento. (30)

- **Creacionales**: abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados.
- **Estructurales**: se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades.
- **Comportamiento**: están relacionadas con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos.

El desarrollo del sistema empleando el marco de trabajo Symfony proporciona ventajas significativas para los desarrolladores de software. Symfony 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 el marco de trabajo los maneja.

A continuación se exponen varios patrones de diseño que han sido utilizados durante el desarrollo de la solución: (29)

1. Patrón Singleton

Es un patrón de Creación, el cual garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

2. Patrón Command

Patrón de comportamiento que encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.

3. Patrón Decorator

Es un patrón estructural, el cual añade funcionalidad a una clase dinámicamente.

1.5.2. Patrones arquitectónicos

El patrón arquitectónico es el nivel en el cual la arquitectura del sistema: (31)

- Define la estructura básica de un sistema, pudiendo estar relacionado con otros patrones.
- Representa una plantilla de construcción que provee un conjunto de subsistemas aportando las normas para su organización.

Ejemplos de patrones arquitectonicos: Capas, MVC (Modelo Vista Controlador), Tuberías y Filtros, Pizarra, entre otros.

Modelo Vista Controlador (MVC): es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario y es el más utilizado en aplicaciones Web, ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema, de forma simple y sencilla. (31)

MVC divide las aplicaciones en tres niveles de abstracción: (31)

- Modelo: representa la lógica de negocios. Es el encargado de acceder de forma directa a los datos actuando como “intermediario” con la base de datos.
- Vista: es la encargada de mostrar la información al usuario de forma gráfica y “humanamente legible”.

- Controlador: es el intermediario entre la vista y el modelo. Es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que esta, lo presente al usuario, de forma “humanamente legible”.

El funcionamiento básico del patrón MVC, puede resumirse en: (31)

- El usuario realiza una petición.
- El controlador captura el evento.
- Hace la llamada al modelo/modelos efectuando las modificaciones pertinentes sobre el modelo.
- El modelo será el encargado de interactuar con la base de datos, ya sea en forma directa, con una capa de abstracción para ello, un servicio web, entre otros y retornará esta información al controlador.
- El controlador recibe la información y la envía a la vista.
- La vista procesa esta información creando una capa de abstracción para la lógica (quien se encargará de procesar los datos) y otra para el diseño de la interfaz de usuario gráfica (GUI).

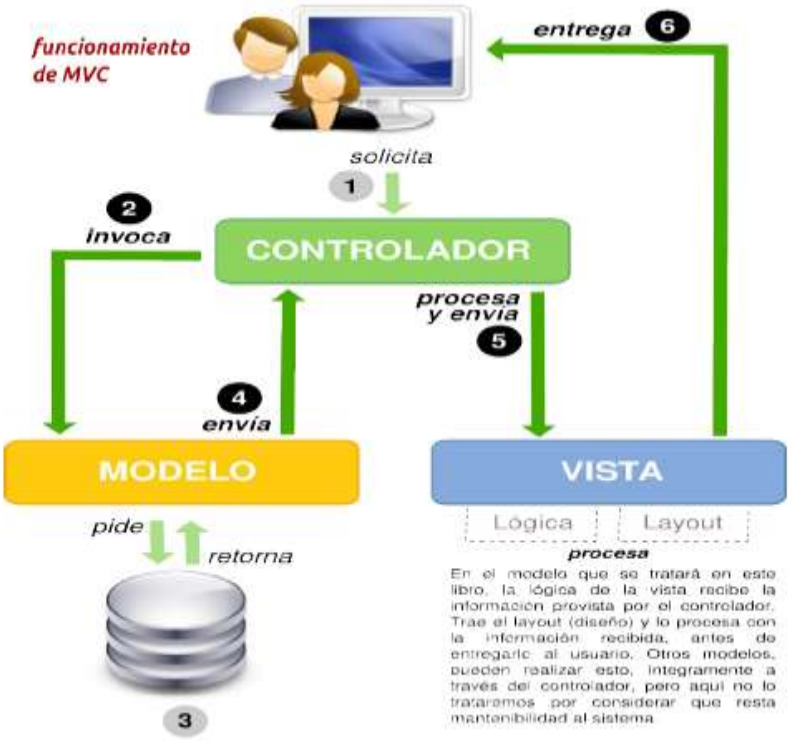


Figura 1.2 Funcionamiento del patrón Modelo Vista Controlador (31)

1.6. Conclusiones Parciales

En el capítulo se realizó una investigación sobre los sistemas que existen en Cuba y el mundo que gestionan solicitudes. Se analizó la factibilidad del uso de los mismos como solución al problema, evidenciándose que ninguno responde a las necesidades de la AGR por lo que no pueden formar parte plenamente de la solución. Sin embargo aportaron experiencias y conocimientos para la creación de una solución nueva que se ajuste a las características del producto GINA.

Con la premisa de obtener un sistema de alto nivel de calidad y funcionamiento, se analizaron diferentes temáticas para el desarrollo de un software como son: la ingeniería de requisitos, técnicas para la captura y validación de los requisitos, además de las herramientas y tecnologías a utilizar en el análisis, diseño e implementación de la aplicación.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

2.1. Introducción

Este capítulo describe la solución propuesta, mostrando características específicas del sistema mediante el modelado y descripción de los procesos de negocio; se especifican los requisitos funcionales definidos para el módulo Solicitudes así como las técnicas de captura y validación de requisitos utilizadas. Además se muestra el modelo conceptual, los diagramas de diseño requeridos; cuenta también con una breve descripción de los patrones de diseño utilizados en la solución de la aplicación y se presenta el modelo de datos.

2.2. Modelado de los Procesos de Negocio

A continuación se describe el proceso Autorizar Solicitudes y el subproceso Aprobar/Rechazar Solicitud, con el propósito de lograr un mayor entendimiento del negocio que posibilite el desarrollo de software con la calidad requerida.

2.2.1. Descripción del proceso Autorizar Solicitudes

El proceso puede iniciarse de cuatro formas (ver figura 2.1):

- Cuando es recibida la información ya validada referente a cada solicitud enviada desde VU, posteriormente si el tipo de solicitud es de Anulación se le envía al subsistema DC la declaración de mercancía (DM) para que verifique si se procede a regístrala. Si no es válida se notifica un mensaje de error a VU para que informe al representante de la entidad que su solicitud no puede ser anulada. Si es de Prórroga de Temporalidad o de Prórroga de Facilidad, se envía al despacho la DM para que notifique la fecha de vencimiento, comprobando que al solicitar estas prórrogas, si se encuentran fuera de término, se notifiquen los datos necesarios al componente persona para que registre la incidencia.

  Seguidamente, se procede a almacenar los datos de cada tipo de solicitud asignándole un número que la identificará, el cual será utilizado para futuras consultas, este número es enviado a VU. Se notifica los datos necesarios al subsistema de Auditoría para que esta acción sea auditada. Posteriormente, se ejecuta el subproceso de Aprobar/Rechazar Solicitud de acuerdo con los resultados obtenidos y se procede a informarle a VU el estado de la solicitud (Aprobar/Rechazar) y concluye el proceso.

- Cuando se actualiza el registro de la fecha de presentación del expediente en las solicitudes de Reintegro y Devolución de Derechos de Aduana que hayan presentado la documentación correspondiente y que sean de la Aduana especificada y concluye el proceso.

- El sistema comprueba, si se encuentran fuera de término (72 horas) las solicitudes de Devolución y Reintegro de Derechos de Aduana que estén aprobadas y no tengan el expediente correspondiente se deben anular automáticamente.
- Recibe petición del subsistema DC conteniendo los datos identificativos de la solicitud para notificarle la información.

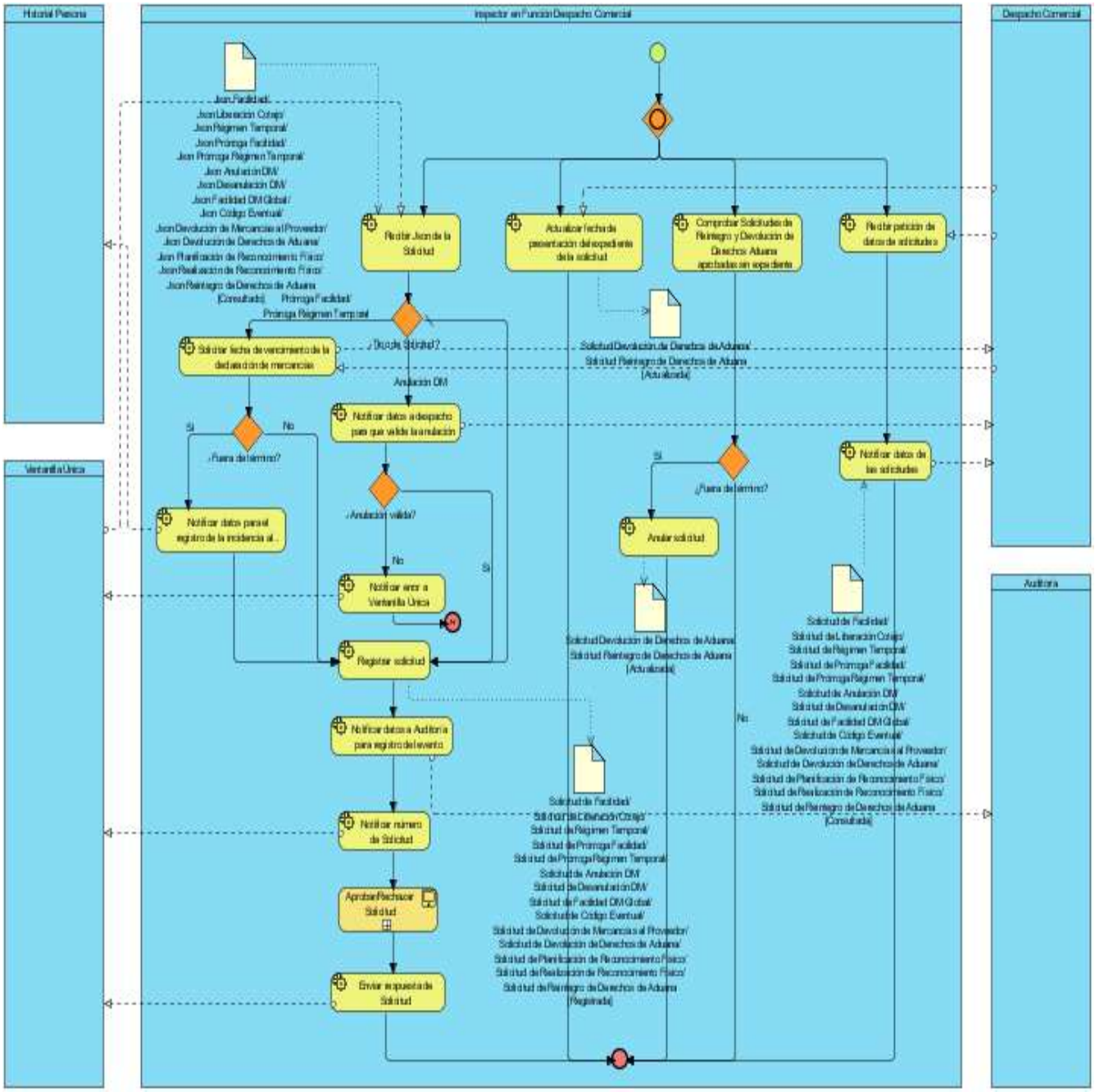


Figura 2.1 Proceso Gestionar Solicitudes

2.2.2. Descripción del subproceso Aprobar/Rechazar Solicitud

El responsable (ver Figura 2.2) de las solicitudes revisa la documentación necesaria, de acuerdo con los resultados obtenidos procede a aprobarla o rechazarla notificando al subsistema de Auditoría los datos necesarios para que esta acción sea auditada.

Se actualiza el Registro de Control de acuerdo al tipo de solicitud reflejando el estado (Aprobada/Rechazada), la fecha en que fueron realizadas cada una de estas acciones, además de los datos específicos de cada solicitud. Se le notifica al subsistema Despacho Comercial dependiendo del tipo solicitud que fue aprobada:

- Prórroga de Facilidad: los datos de la declaración y la prórroga otorgada.
- Prórroga de Temporalidad: los datos de la declaración de las mercancías aprobadas, la partida, el régimen y la prórroga otorgada.
- Anulación: los datos de la declaración anulada.
- Desanulación: los datos de la declaración desanulada.
- Realización de Reconocimiento Físico: los datos de la declaración y la planificación de reconocimiento otorgada.
- Planificación de Reconocimiento Físico: los datos de la declaración y la planificación otorgada.

Si el tipo de solicitud aprobada es de Código Eventual y contiene el escaque entidad, envía una petición al subsistema de Registro Central notificándole la entidad y la fecha de vigencia de la misma. Finalmente, después de aprobar o rechazar cada solicitud y hacer las notificaciones necesarias concluye el proceso.

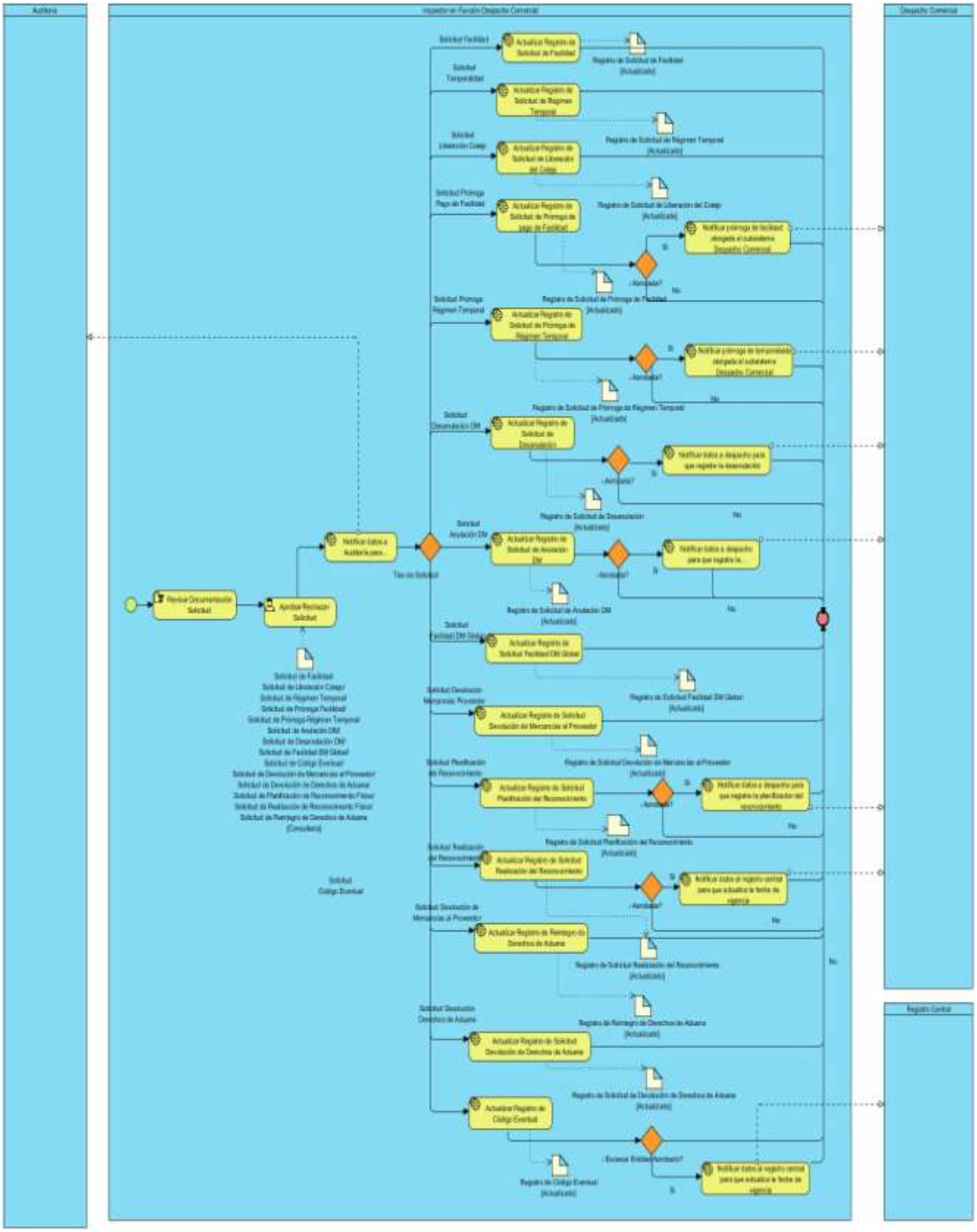


Figura 2.2 Subproceso Aprobar/Rechazar Solicitud

2.3. Especificación de los requisitos de software

Con la especificación de los requisitos de software se encuentra cada detalle a tener en cuenta para el desarrollo del sistema evitándose errores en posteriores etapas, por lo que se deben tener presentes las técnicas para la captura y validación de estos requisitos. A continuación se muestran las técnicas utilizadas y el listado de los requisitos funcionales definidos para el sistema.

2.3.1. Técnicas para la captura de requisitos

Para realizar la captura de requisitos se utilizaron las siguientes técnicas: las *entrevistas* que se le realizaron a Yamila Martínez, especialista del Departamento de Técnicas Aduaneras de la jefatura de la AGR, funcional con gran conocimiento sobre el tema tratado, lo que posibilitó la comprensión de los procesos de negocio existentes. También se realizaron *talleres* con Lianet Pineda de la Nuez analista principal del módulo DC con el objetivo de acumular la mayor información posible y aclarar dudas referentes a las solicitudes.

2.3.2. Requisitos funcionales (RF)

RF1 Registrar Solicitud**:** el sistema debe ser capaz de obtener la información proveniente de VU de cada tipo de solicitud y registrarla.

RF2 Gestionar Solicitud de Facilidad: el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF3 Gestionar Solicitud de Régimen Temporal**:** el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF4 Gestionar Solicitud de Liberación de Cotejo**:** el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF5 Gestionar Solicitud de Prórroga de Régimen Temporal**:** el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF6 Gestionar Solicitud de Prórroga para Pago de Facilidad**:** el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF7 Gestionar Solicitud de Anulación de la DM**:** el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF8 Gestionar Solicitud de Desanulación de la DM**:** El sistema debe ser capaz mostrar la información de la solicitud y actualizar algunos datos.

RF9 Gestionar Solicitud de Código Eventual**:** el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF10 Gestionar Solicitud de Facilidad de DM Global el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF11 Gestionar Solicitud de Devolución de Mercancías al Proveedor**:** el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF12 Gestionar Solicitud de Devolución de Derechos de Aduana**:** el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF13 Gestionar Solicitud de Planificación de Reconocimiento Físico**:** el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF14 Gestionar Solicitud de Realización de Reconocimiento Físico el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF15 Gestionar Solicitud de Reintegro**:** el sistema debe permitir aprobar/rechazar la solicitud y notificar al declarante la respuesta.

RF16 Notificar listado de solicitudes de Devolución de Derechos de Aduana: el sistema debe devolver los datos de las solicitudes de Devolución de Derechos de Aduana aprobadas, que hayan presentado la documentación correspondiente para la aduana especificada.

RF17: Notificar listado de solicitudes de Reintegro: el sistema debe devolver los datos de las solicitudes de Reintegro de Derechos de Aduana aprobadas, que hayan presentado la documentación correspondiente para la aduana especificada.

RF18: Actualizar fecha de presentación del expediente de la solicitud: el sistema debe permitir que se registre la fecha de presentación del expediente en las solicitudes de Devolución y Reintegro de Derechos de Aduana.

RF19: Notificar datos de solicitud: el sistema debe devolver los datos de las solicitudes de Facilidad, Liberación de Cotejo, Temporalidad, Código Eventual, Facilidad de DM Global y Devolución de Mercancías al Proveedor.

RF20: Anular Solicitud de Reintegro de Derechos de Aduana y Devolución de Derecho de Aduana: el sistema de garantizar que vencido el término de presentación de la documentación correspondiente en las solicitudes de Devolución y Reintegro de Derechos de Aduana se anule dicha solicitud.

2.3.3. Técnicas de la validación de los requisitos

Para la validación de los requisitos se utilizó la técnica de *prototipos de interfaz de usuario*, mediante la cual se comprobó que el sistema cubre las necesidades planteadas por el cliente, viendo reflejadas cada una de las funcionalidades en los prototipos. Además se realizaron diversas revisiones al documento de requisitos con la participación de los especialistas de la Aduana y los analistas del módulo DC para logar una correcta interpretación de la información transmitida con el objetivo de corregir cualquier inconformidad por parte de los clientes, los señalamientos planteados fueron recogidos y corregidos posteriormente.

2.4. Modelo conceptual

Un modelo conceptual es un lenguaje que se utiliza para describir esquemas conceptuales. El objetivo del diseño conceptual es describir el contenido de la información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información.

A continuación se observa un fragmento del modelo conceptual (Anexo 1) del negocio que presenta los conceptos más importantes del módulo Solicitudes y sus relaciones.

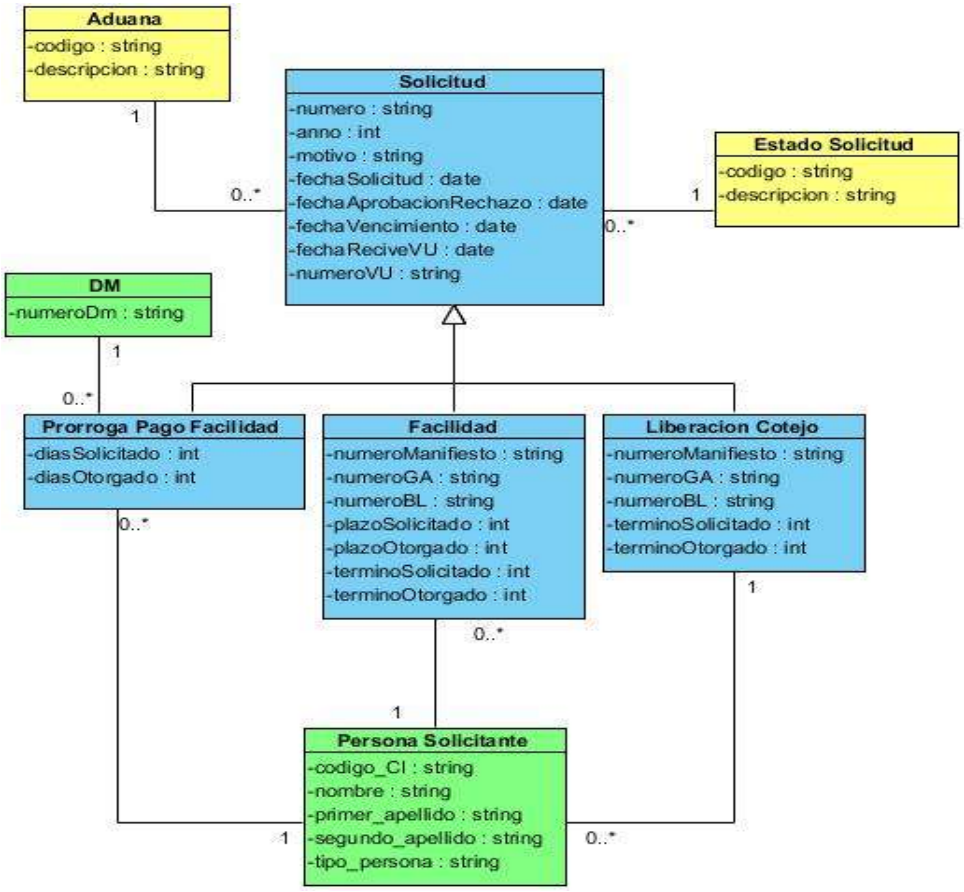


Figura 2.3 Fragmento del modelo conceptual del módulo Solicitudes

2.5. Diseño del sistema

En el desarrollo o ciclo de vida de un sistema informático, el diseño del sistema constituye un elemento fundamental del mismo. Se centra en proporcionar la funcionalidad del sistema a través de sus diferentes componentes. En esta fase se diseña una solución que convierte los requisitos del cliente en un sistema de información real.

El diseño debe proporcionar una completa idea de lo que es el software, enfocando los dominios de datos, funcional y comportamiento desde el punto de vista de la Implementación.

2.5.1. Patrones de diseño

- Patrones GRASP

Experto: este patrón es uno de los que más se utiliza cuando se trabaja con Symfony, con la inclusión de la librería Propel para mapear la Base de Datos. Symfony utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades. Las clases de abstracción de datos (Peer del Modelo) poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan.

Creador: en la clase Actions del módulo Solicitudes se encuentran las acciones definidas para el sistema y se ejecutan en cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que la clase Actions es “creador” de dicha entidad.

Bajo Acoplamiento: la clase Actions del módulo Solicitudes hereda únicamente de sfActions para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

Alta cohesión: Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase solicitudActions, la cual está formada por varias funcionalidades que están estrechamente relacionadas, siendo la misma la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, 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 manipuladas por un solo controlador frontal (dc_dev.php), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el

controlador frontal recibe una petición utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

- Patrones GOF

Singleton: utiliza el controlador frontal que se encarga de enrutar todas las peticiones que se hagan a la aplicación. Se evidencia en una acción con la utilización el método getConnection(),encargado de guardar una referencia a todos los objetos del núcleo de Symfony relacionados con una petición dada. Ejemplo: *Propel::getConnection('solicitudes');*

Command: este patrón se observa en la clase dc_dev.php, en el método dispatch(). Esta clase es la encargada de establecer el módulo y la acción que se va a usar según la petición del usuario. Este patrón se aplica además en la clase routing.yml, donde es parseada la URL con el objetivo de precisar los parámetros de la misma y de esta forma saber el procedimiento que debe responder a la petición.

Decorator: este método pertenece a la clase abstracta view.yml, padre de todas las vistas, que contiene un decorador para permitir agregar funcionalidades dinámicamente. El archivo nombrado layout.php es el que contiene el layout de la página. Este archivo, conocido también como plantilla global, guarda el código HTML que es usual en todas las páginas del sistema 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.

2.5.2. Patrón arquitectónico

- Patrón MVC

El uso del marco de trabajo que utiliza MVC obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el marco de trabajo. El código de la presentación se guarda en la vista, el código de manipulación de datos se guarda en el modelo y la lógica de procesamiento de las peticiones constituye el controlador.

Symfony toma lo mejor de la arquitectura MVC y la realiza de modo que el desarrollo de aplicaciones sea rápido y sencillo. En el controlador se encuentran las acciones, las cuales son el núcleo de la aplicación, pues contienen toda la lógica de la aplicación. Estas acciones utilizan el modelo y precisan las variables para la vista. Al realizarse una petición web en una aplicación Symfony, la URL define una acción y los parámetros de la petición.

La vista es la encargada de originar las páginas que son mostradas como resultado de las acciones, donde se encuentra el layout, que es común para todas las páginas de la aplicación. La vista en Symfony está conformada por varias partes, preparadas cada una de ellas especialmente para ser

fácilmente transformable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones.

En el modelo se encuentran las clases, que son generadas de forma automática según la estructura de la base de datos. En Symfony, el acceso y la modificación de los datos que se almacenan en la base de datos, se realiza mediante objetos. Propel es el motor generador que se encarga de esta generación automática para construir sus clases, creando la estructura y generando el código de las mismas.

2.6. Diagramas del diseño

2.6.1. Diagrama de paquetes

El diagrama de paquetes (ver figura 2.4) del módulo Solicitudes muestra los principales elementos que lo integran, representando las agrupaciones lógicas que reflejan las dependencias entre las mismas. Este está constituido por 4 paquetes internos, entre los que se encuentran: el *paquete JS* que hace referencia a las clases .js pertenecientes a las interfaces visuales de la aplicación, el *paquete Controlador* que contiene el controlador frontal, en este caso es dc_dev.php, *el paquete Lib* que tiene todas las clases del negocio de solicitudes y el *paquete Config* conformado por los ficheros propel.ini y database.yml.

Solicitudes está relacionado con varios paquetes externos utilizando los servicios que brindan para la obtención de datos, estos son: el *paquete TC* (Tablas de Control) que son nomencladores, los cuales son datos determinados que varían en el tiempo, el *paquete Contacto* gestiona la información referente a los contactos, el *paquete RC* (Registro Central) gestiona la información de las tablas declarante y entidad, el *paquete DC* gestiona formalidades aduaneras que se realizan cuando se comercializan mercancías, tanto en importaciones como exportaciones**,** el *paquete Contenedor* contiene la información de los contenedores, el *paquete Persona* gestiona la información referente a las personas, el *paquete Symfony Lib* representa las librerías de Symfony, el *paquete Symfony Core* representa el núcleo del framewok utilizado y por último el *paquete VU* que es el único punto de entrada y salida de la información digital y estandarizada.

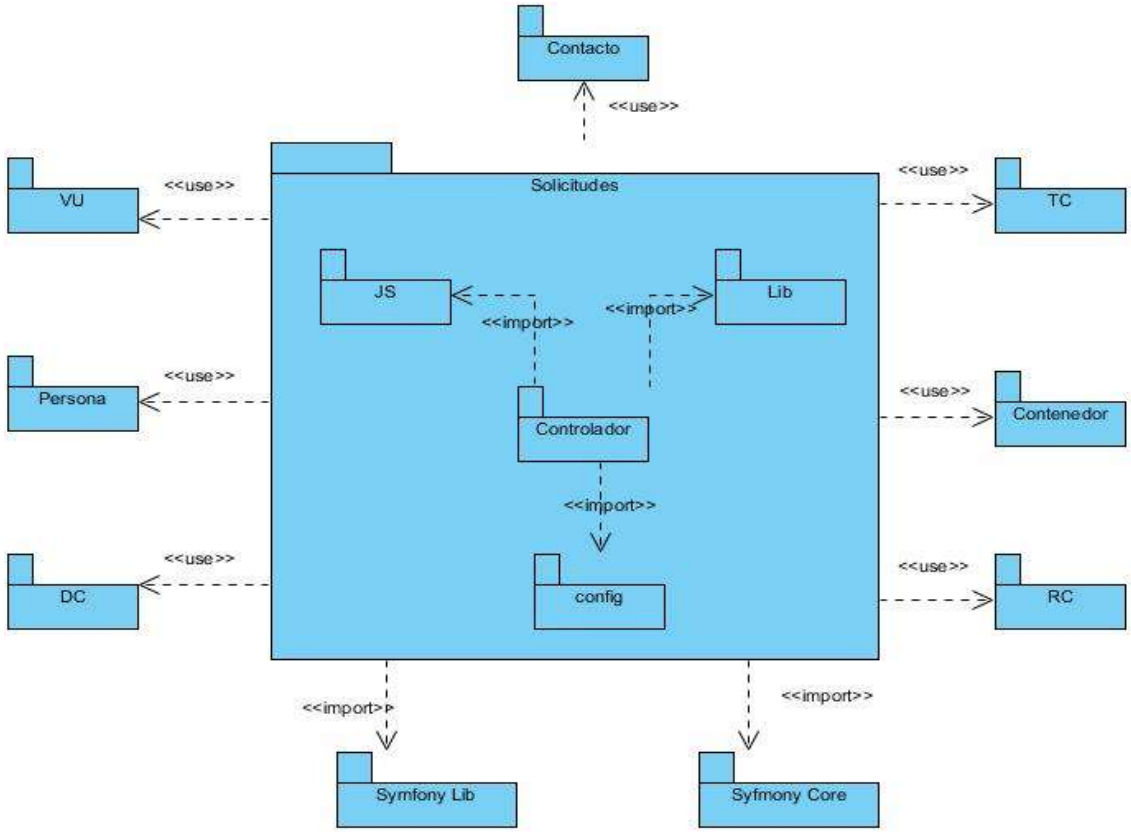


Figura 2.4 Diagrama del paquete módulo Solicitudes

2.6.2. Modelo de la base de datos

El modelo de datos (ver Anexo 2) está formado por un conjunto de conceptos que permiten describir la realidad que contiene las tablas y relaciones entre ellas que guardan la información referente a las solicitudes. La figura 2.5 muestra un fragmento del modelo de datos del módulo, el cual está compuesto por:

- Las tablas de color naranja son las más importantes arquitectónicamente debido a que en ellas se almacenan los datos principales de las solicitudes.
- Las tablas de color amarillo son nomencladoras, pertenecientes al esquema Tabla de Control (TC) de la base de datos, permitiendo el acceso de todos los subsistemas que conforman el GINA.
- Las tablas grises son de vital importancia debido a que contribuyen al completamiento de la información de las tablas del módulo Solicitudes.

El modelo de base de datos se encuentra en 3ra Forma Normal, ya que todas las tablas contienen una llave primaria, los atributos son atómicos y no existen dependencias parciales de la llave primaria ni dependencias transitivas. De esta forma se evita la redundancia de la información, reduciendo los

problemas que se puedan presentar con las actualizaciones de los datos en las tablas, protegiendo así la integridad de los mismos.

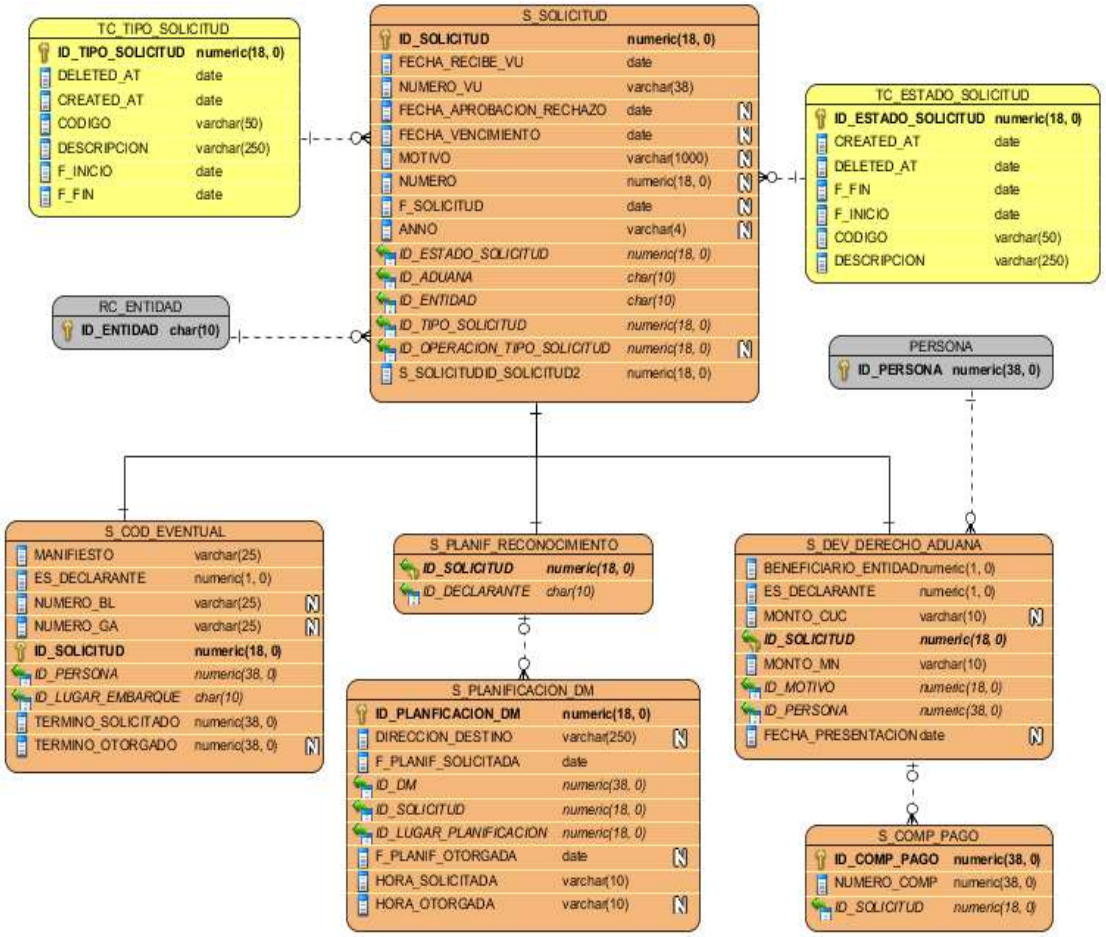


Figura 2.5 Fragmento del modelo de datos del módulo Solicitudes.

2.6.3. Diagramas de secuencia orientado a actividades

En el modelo de desarrollo utilizado en el Departamento de Soluciones para la Aduana en la fase análisis y diseño del ciclo de vida, define entre los artefactos a generar el diagrama de secuencia orientado a actividades, el cual es el encargado de describir el funcionamiento de cada uno de los RF definidos. 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. El objetivo principal de este diagrama es especificar la interacción entre las clases, así como el flujo a seguir de las actividades a realizar, experimentando un nivel de especificación tal que permita a los programadores lograr la implementación de la solución en menor tiempo y con mejor calidad. Se representa por calles, actividades y sus relaciones, incluyendo también

comentarios para mayor comprensión. Es considerado un híbrido entre los diagramas de actividades y los de secuencias propuestos por RUP en la etapa de diseño.

2.6.3.1. Diagramas de secuencia orientado a actividades del negocio

La figura 2.6 muestra un ejemplo de un diagrama de secuencia orientado a actividades del negocio, perteneciente al RF Gestionar Solicitud de Prórroga para Pago de Facilidad, específicamente de la funcionalidad Mostrar Datos de dicha solicitud. Como se puede observar está compuesto por cinco calles, cada una de ellas representa las áreas que abarca el negocio de esta funcionalidad. Se inicia cuando son obtenidos los datos en la clase controladora del módulo Solicitudes siendo a su vez la responsable de llamar a la funcionalidad *mostrarDatoProrrogaPadoFacilidadDatoSolicitud($idSolicitud)*, que se encuentra en la clase *SProrrogaPagoFacilidad.php*, en esta funcionalidad utilizando el identificador de la solicitud entrado por parámetro, se obtienen los datos pertinentes referente a dicha solicitud, utilizando para la obtención de los mismos los servicios brindados por los subsistemas del GINA. En este caso se utilizan los siguientes servicios:

- *devolverDmDadoID($prorrogaPagoFacilidad->getIdDm())* del Despacho Comercial pasándole como parámetro un identificador de la DM y devuelve un arreglo con los objetos que cumplan con dicha condición.
- *obtenerEntidadDadoId($idEntidad)*, *obtenerDeclaranteDadoId($IdDeclarante)* y *obtenerTrabajadorDadoIdPersona($idPersona)*, estos servicios pertenecen al subsistema Registro Central se le pasa por parámetro los identificadores de la entidad, del declarante y la persona respectivamente y se obtiene un arreglo con los objetos que satisfagan los anteriores parámetros.
- *obtenerRegistroXid($idMotivo)* y *obtenerDadoCriteria($criteria)* del subsistema TC, se le pasa por parámetro un arreglo especificando el nombre de la TC y el identificador de donde se desea obtener los datos y devuelve un arreglo con los objetos que cumplan con los parámetros establecidos. Al segundo servicio se le pasa por parámetro un arreglo especificando el nombre de la TC y el estado (vigente) y la consulta criteria[10] con el código que lo identifica y te devuelve un arreglo con los objetos que cumplan con los parámetros establecidos

Una vez obtenidos los datos, se retorna el JSON para posteriormente mostrárselo al usuario.

[10] **Criteria:** clase que permite la realización de consulta a la base de datos.

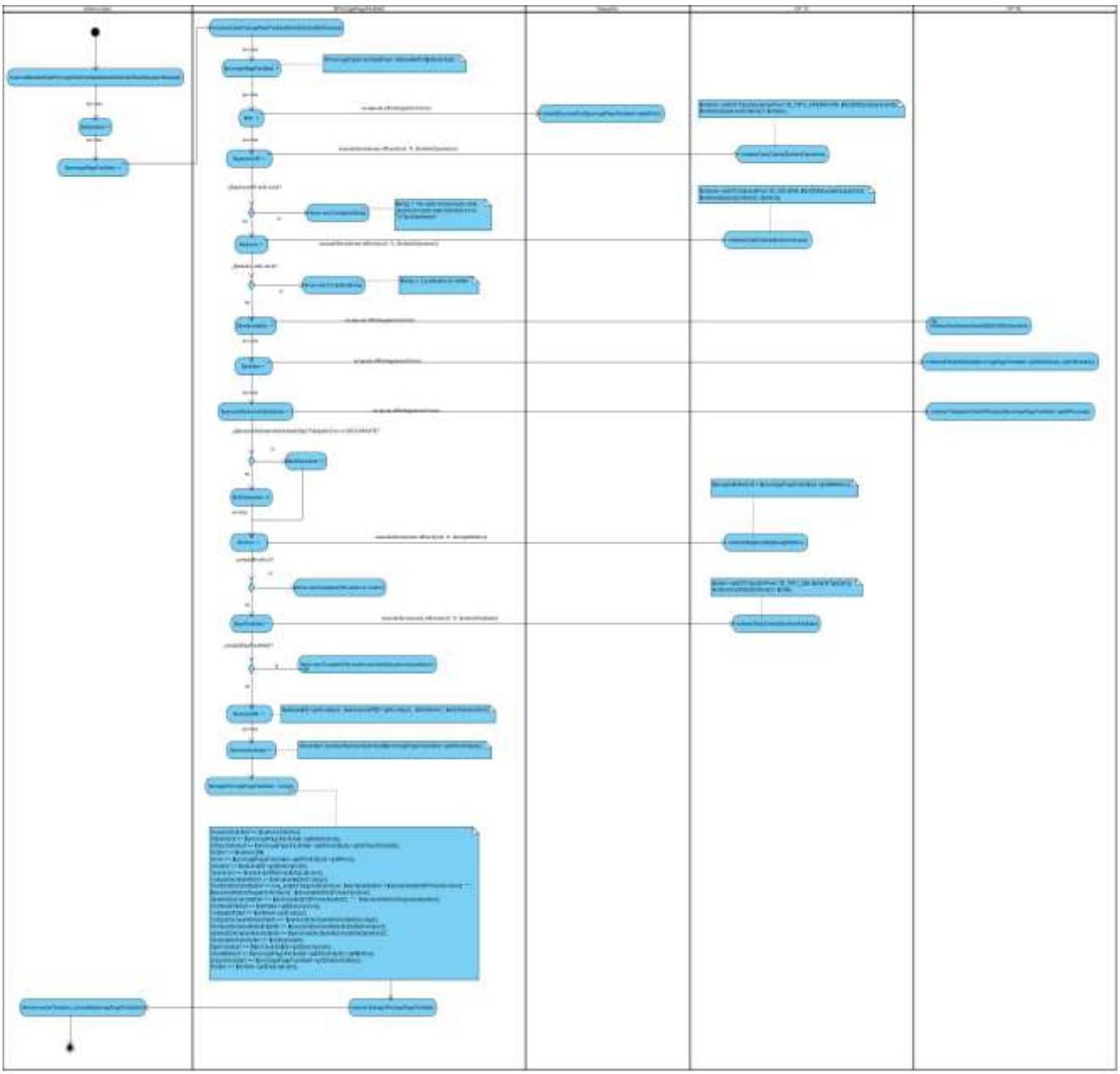


Figura 2.6 Diagrama de secuencia orientado a actividades del negocio de la solicitud de Prórroga para Pago de Facilidad

2.6.3.2. Diagramas de secuencias orientados a actividades de componentes visuales

Este diagrama tiene como principal objetivo 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 las notas de este diagrama se especifican los

parámetros que son útiles para el programador como es el caso de los eventos, nombre de la tabla o funcionalidad de la cual va a cargar o enviar la información, entre otros; garantizando una total correspondencia entre la información de este diagrama y su equivalente en el negocio.

En la figura 2.7 se muestra el diagrama perteneciente al formulario de la solicitud de Prórroga Pago de Facilidad. El mismo está formado por tres calles, una de ellas representa al actor que es el que comienza la funcionalidad al escoger una solicitud que se encuentre presentada. Posteriormente selecciona la opción "Ver detalles" donde se ejecuta la llamada a la acción *mostrarDatosProrrogaFacilidadDatoSolicitud()* mostrando la información correspondiente con la solicitud seleccionada y luego se procede a rechazar o aprobar dicha solicitud.

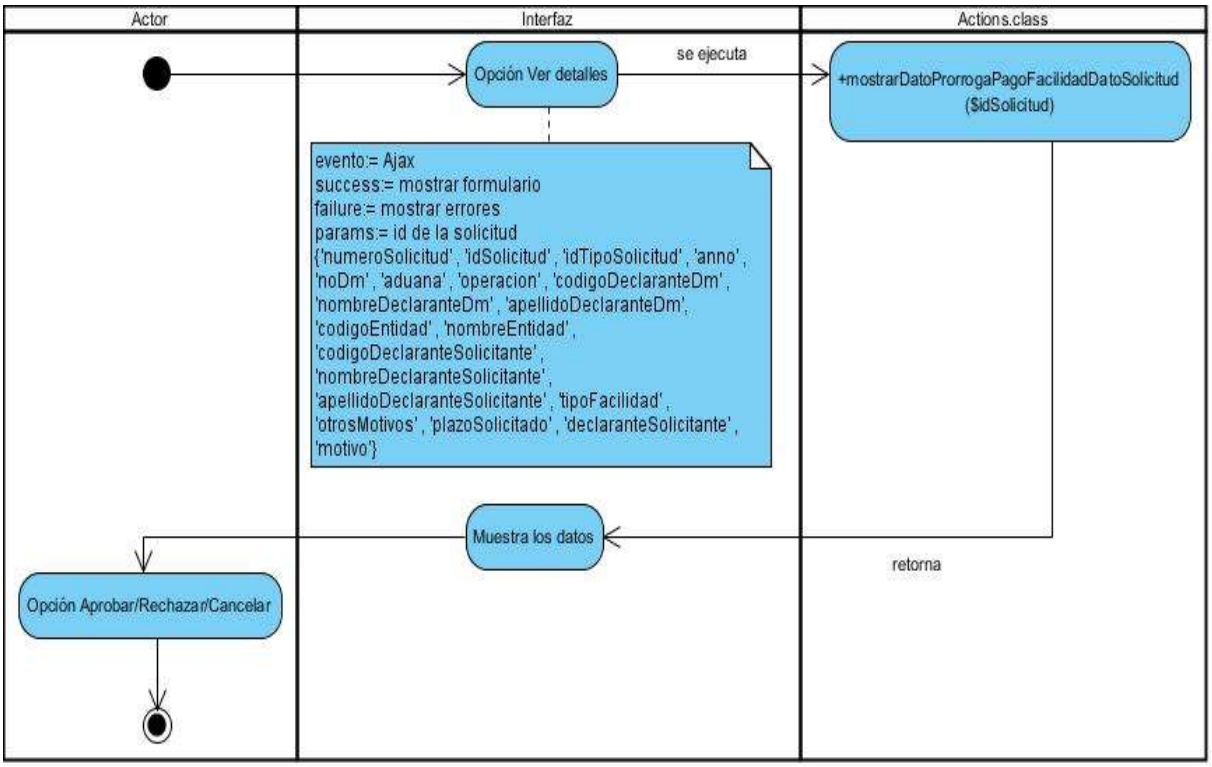


Figura 2.7 Diagrama de secuencia orientado a actividades para la funcionalidad de gestionar solicitud Prórroga de Pago de Facilidad.

2.6.4. Diagrama de clases del diseño

El diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación, ya que una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica; mostrando un conjunto de elementos que son estáticos, como las clases y tipos junto con sus contenidos y relaciones. Un diagrama de clases está compuesto por los siguientes

elementos: Clases: atributos, métodos y visibilidad. Relaciones: herencia, composición, agregación, asociación y uso. (32)

La figura 2.8 muestra un fragmento del diagrama de clases del módulo Solicitudes (ver Anexo 3). El mismo está constituido por 58 clases, de ellas 33 pertenecen al esquema del presente trabajo, representadas por el color gris. De estas clases se encuentran entre las más significativas *SSolicitud* que es contenedora de todas las funcionalidades y atributos comunes de las solicitudes, la relación que existe entre estas clases y debido a la utilización del Object-Relational Maping (ORM) Propel, es la simulación de herencia a través de composición, las mismas contienen toda la implementación principal que se necesita para la realización del módulo Solicitudes. De las restantes tablas, 19 son nomencladores que pertenecen al subsistema TC, de donde se obtiene información a través de servicios que brindan, las mismas son mostradas en color amarillo. Las tablas azules contribuyen al completamiento de la información de este módulo mediante los servicios que lo facilitan.

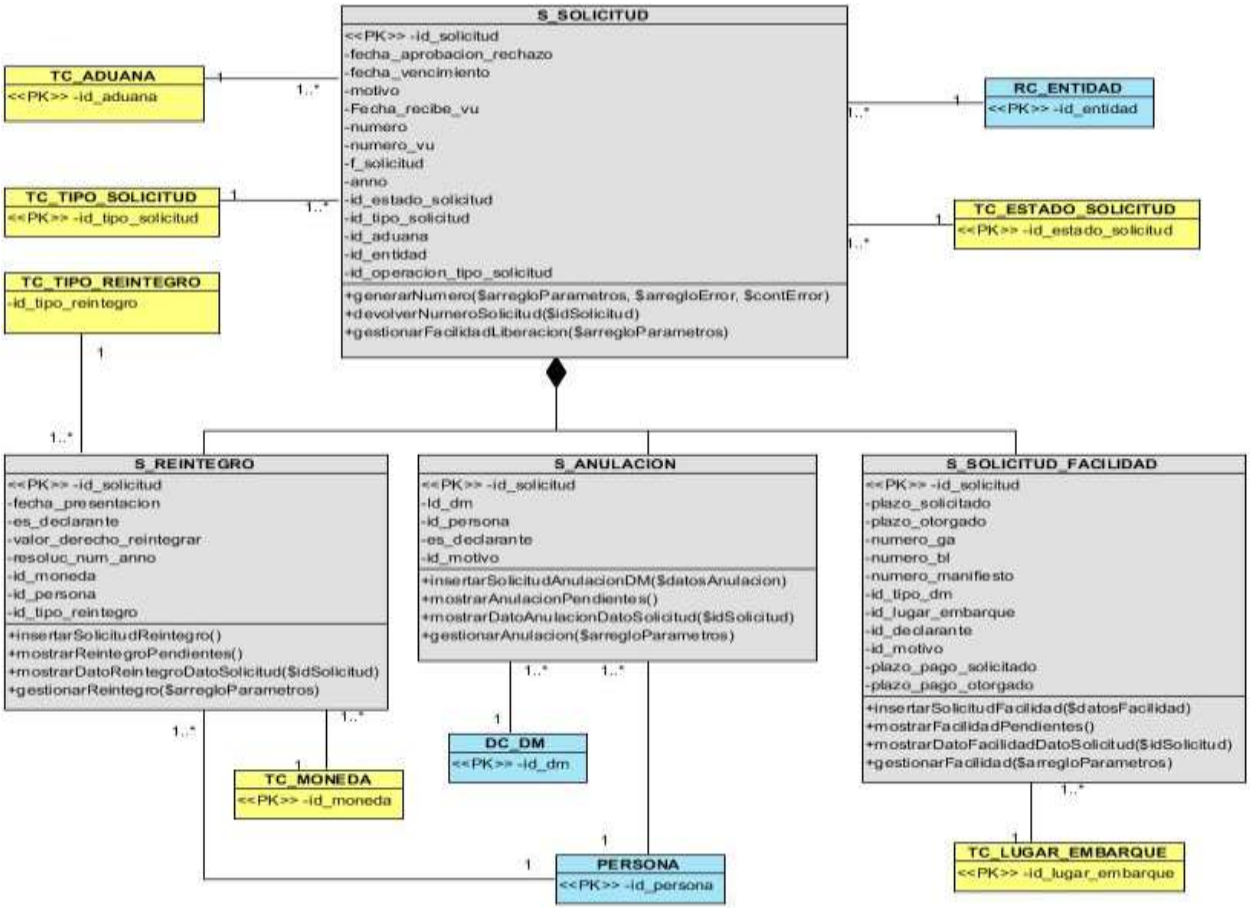


Figura 2.8 Fragmento del diagrama de clases del módulo Solicitudes.

2.7. Métricas para la evaluación del diseño

Aunque los términos medida, medición y métricas se utilizan a menudo indistintamente, existe gran confusión a la hora de referirse a ellos. Dentro del contexto de la ingeniería del software, una medida proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto. La medición es el proceso por el cual los números o símbolos son asignados a atributos o entidades en el mundo real tal como son descritos de acuerdo a reglas claramente definidas, también es definida como el acto de determinar una medida. (33)

2.7.1. Métricas del modelo de diseño Orientados a Objetos

Los objetivos principales de las métricas Orientadas a Objetos (OO) son los mismos que los existentes para las métricas surgidas para el software estructurado: (20)

- Comprender mejor la calidad del producto.
- Estimar la efectividad del proceso.
- Mejorar la calidad del trabajo realizado en el nivel del proyecto.

Berard [Laranjeira ‘90] define cinco características que dan lugar a unas métricas especializadas: (20)

- Localización: indica la forma que se concentra la información dentro de un programa. En el contexto OO, la información se concentra mediante el encapsulamiento tanto de datos como de procesos dentro de los límites de una clase u objeto.
- Encapsulamiento: comprende las responsabilidades de una clase, incluyendo sus atributos y operaciones, y los estados de la clase, según se definen mediante valores específicos de atributos. Ocultamiento de información.
- Herencia: hace posible que los compromisos de un objeto se difundan a otros objetos.
- Técnicas de abstracción de objetos: permite al diseñador centrarse en los detalles esenciales de algún componente de un programa (tanto si es un dato como si es un proceso) sin preocuparse por los detalles de nivel inferior.

2.7.1.1. Métricas orientadas a Clases

La clase es la unidad principal de todo sistema OO. Por consiguiente, las medidas y métricas para una clase individual, la jerarquía de clases, y las colaboraciones de clases resultarán sumamente valiosas para un ingeniero de software que tenga que estimar la calidad de un diseño (20)

2.7.1.1.1. Métricas CK (Chidamber y Kemener)

Carencia de Cohesión en los Métodos (CCM). Cada método dentro de una clase C, accede a uno o varios atributos (también llamados variables de instancia), CCM es el número de métodos que accede a uno o varios de los mismos atributos. Si no existen métodos que accedan a los mismos atributos, entonces CCM = 0. (20)

Si CCM es elevado, los métodos pueden estar acoplados entre sí a través de atributos. Esto incrementará la complejidad del diseño de clases. En general, unos valores elevados para CCM implican que la clase podría diseñarse mejor descomponiéndola en dos o más clases distintas. Aún cuando existen casos en que es justificable un valor elevado de CCM, en donde es deseable mantener elevado un grado de cohesión, esto es mantener un valor bajo para CCM. (20)

Medidas o umbrales[11] aplicados:

Umbrales	Clasificación
0-20	Baja
21 a 41	Media
42 a 62	Alta

Tabla 2.1 Umbrales para aplicar CCM. (33)

Resultados obtenidos:

No	Clase	Cantidad de métodos	CMM	Clasificación
1	SSolicitud	4	4	Baja
2	SSolicitudFacilidad	3	3	Baja
3	SLiberacionCotejo	3	3	Baja
4	SSolicitudTemporalidad	5	5	Baja
5	SProrrogaPagoFacilidad	4	4	Baja
6	SProrrogaTemporalidad	4	4	Baja
7	SAnulacion	4	4	Baja
8	SDesanulacion	4	4	Baja
9	SCodEventual	4	4	Baja
10	SFacilidadDmGlobal	4	4	Baja
11	SDevMercancia	4	4	Baja
12	SDevDerechoAduana	4	4	Baja
13	SPlanificacionDm	4	4	Baja
14	SRealReconocimiento	4	4	Baja
15	SReintegro	4	4	Baja

Tabla 2.2 Clases a las que se les aplicó la métrica CCM.

Después de aplicada la métrica CCM, cuyos resultados se muestran en la tabla anterior, se llega a la conclusión que CCM es bajo, por lo que a su vez el diseño de clases es de complejidad baja. Por lo

[11] **Umbral:** permiten determinar una medida que valide el trabajo tanto de la persona que interviene directamente con la solución como cada una de las posibles fallas o aristas que se generen durante el desarrollo del diseño.

tanto no es necesario rediseñar las clases obtenidas y se ha logrado mantener un elevado nivel de cohesión.

2.7.1.1.2. Métricas propuestas por Lorenz y Kidd

Lorenz y Kidd en su libro[12] dividen las métricas basadas en clases en cuatro categorías, cada una con un diseño al nivel de componentes: (20)

- Tamaño: se centran en el conteo de atributos y de operaciones para una clase individual, y promedian los valores para el sistema OO en su totalidad.
- Herencia: se centran en la forma en que se reutilizan las operaciones a lo largo y ancho de la jerarquía de clases.
- Valores internos: buscan cohesión y asuntos relacionados con el código.
- Valores externos: examinan el acoplamiento y la reutilización.

Tamaño Operacional de Clase (TOC).

El tamaño general de una clase se puede determinar empleando las medidas siguientes: (20)

- El número total de operaciones (tanto operaciones heredadas como privadas de la instancia) que están encapsuladas dentro de la clase.
- El número de atributos (tanto atributos heredados como atributos privados de la instancia) que están encapsulados en la clase.

Si existen valores grandes de TOC éstos mostrarán que una clase puede tener demasiada responsabilidad, lo cual reducirá la reutilizabilidad de la clase y complicará la implementación y la comprobación. Por otra parte cuanto menor sea el valor medio para el tamaño, más probable es que las clases existentes dentro del sistema se puedan reutilizar ampliamente. Debido a estas características es que se emplea el TOC para evaluar el diseño. (20)

Parámetros de calidad	Valores Grandes de TOC
Reutilización	Reduce la reutilización de clases
Implementación	Complica la implementación
Complejidad de las pruebas	Hace compleja las pruebas del sistema
Responsabilidades	La clase debe tener bastante responsabilidad

Tabla 2.3 Parámetros de calidad para valores grandes de TOC. (34)

Medidas o umbrales aplicados:

[12] Lorenz, M. et al, 1994] Lorenz, M. & Kidd, J. "Object-Oriented Software Metrics". Prentice Hall. 1994. Texto recomendado para introducirse en la medición de atributos de entidades desarrolladas con metodología orientada a objetos.

Número de operaciones y/o atributos

TOC	Umbral
Pequeño	<= 20
Medio	> 20 y <= 30
Grande	> 30

Tabla 2.4 Umbrales para TOC (34)

Resultados obtenidos:

No	Clase	Operaciones	Atributos	Tamaño
1	SSolicitud	3	14	Pequeño
2	SSolicitudFacilidad	3	12	Pequeño
3	SLiberacionCotejo	3	8	Pequeño
4	SSolicitudTemporalidad	5	7	Pequeño
5	SProrrogaPagoFacilidad	4	6	Pequeño
6	SProrrogaTemporalidad	4	4	Pequeño
7	SAnulacion	4	4	Pequeño
8	SDesanulacion	4	5	Pequeño
9	SCodEventual	4	7	Pequeño
10	SFacilidadDmGlobal	4	9	Pequeño
11	SDevMercancia	4	7	Pequeño
12	SDevDerechoAduana	4	6	Pequeño
13	SPlanificacionDm	4	2	Pequeño
14	SRealReconocimiento	4	16	Pequeño
15	SReintegro	4	8	Pequeño

Tabla 2.5 Clases a las que se les aplicó la métrica TOC.

Como resultado se obtuvo que el tamaño promedio es pequeño por lo que se llega a la conclusión que la complejidad y la responsabilidad del diseño realizado es baja. No existirá problemas con la implementación y las clases existentes se podrán reutilizar ampliamente.

2.8. Conclusiones Parciales

En este capítulo se detallaron las características principales de la propuesta de solución. Con la elaboración de cada uno de los artefactos definidos por el Departamento de Soluciones para la Aduana se logró sentar las bases para la implementación del sistema. La aplicación de las métricas permitió evaluar el diseño y determinar con la utilización de la métrica TOC que no se tendrá problemas con la implementación y las clases existentes se podrán reutilizar ampliamente y con CCM, obteniendo como resultado que no es necesario rediseñar las clases obtenidas logrando mantener un elevado nivel de cohesión.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA

3.1. Introducción

En el presente capítulo se plasmará el estándar de codificación utilizado, además se detallará el trabajo acerca del tratamiento de errores, el diagrama de componentes y un ejemplo de la comunicación entre las capas con las que cuenta la solución. Se validará mediante pruebas, si el software responde a las necesidades del cliente, comprobándose que todos los requisitos obtenidos, fueron cumplidos satisfactoriamente.

3.2. Modelo de implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se pueden encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. (35)

3.2.1. Estándares de codificación

El estándar de codificación utilizado en la implementación del módulo Solicitudes está condicionado por el documento: “Propuesta de un estándar de codificación”, perteneciente al Departamento de Soluciones para la Aduana del CEIGE.

El objetivo de crear un estándar de codificación es que los programadores se rijan por el mismo a la hora de implementar. Este estándar debe servir para que el personal del proyecto pueda identificar de forma sencilla el objetivo y las funcionalidades que brinda cada una de las clases, funciones y demás componentes del software dada su nomenclatura, es necesario que esto se pueda identificar a simple vista; además debe servir de guía para los implementadores que tienen que continuar el desarrollo de las aplicaciones. (36)

Las aplicaciones deben tener nombres que dejen reflejado claramente el propósito de la misma, ya sea en una palabra o siglas. En caso de ser mediante siglas se pondrán todas en mayúsculas .Se debe evitar mientras sea posible la utilización de palabras compuestas o la utilización de varias palabras; en caso de que sea palabras compuestas se utilizará la notación UpperCamelCase[13].

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

Los nombres de las acciones deben estar en la nomenclatura CamelCase[14] comenzando por la palabra execute. Deben especificar con la menor cantidad de palabras el objetivo de la acción, de ser posible estar en infinitivo.

Todos los nombres de las clases deben estar expresados en notación UpperCamelCase. No se deben utilizar guiones bajos "_" y deben expresar con claridad el alcance y la responsabilidad de la clase. Se mantiene el uso de los sufijos "Peer" para las clases que se encargan de estas funciones en el modelo.

Las funciones definidas deben seguir la nomenclatura UpperCamelCase, reflejando claramente la acción que realiza. Se debe apoyar en la utilización de sufijos que ayuden a identificar el resultado final de la ejecución de un método y en el uso de prefijos para expresar la acción que realiza sobre un elemento determinado.

Los nombres de las variables deben expresar claramente el contenido de la misma. Pueden estar referidas en singular o plural. En caso de que no se le asigne un valor inicial a las variables se deben inicializar con un valor que indique el tipo de dato más general al que debe pertenecer.

3.2.2. Tratamiento de errores

Cuando se desarrolla un software es de vital importancia, para garantizar su correcto funcionamiento, identificar y controlar los posibles problemas que se pueden presentar a la hora de interactuar con el software.

Con el propósito de afrontar cualquier situación inesperada o excepcional que se presente mientras se ejecuta un programa, se utiliza como unos de los métodos para el control de excepciones las palabras claves *try*, *catch* (ver figura 3.1). Las acciones que pueden producir una excepción se ejecutan dentro del bloque *try* y al producirse una anomalía, el flujo de control salta inmediatamente al controlador de excepciones, utilizándose la palabra *catch* para definir dicho controlador. Con el objeto de excepción *throw* excepciones, conteniendo información detallada sobre el error.

```
public function executeGestionarProrrogaPagoFacilidad(sfWebRequest $request) {
    try {
        $arregloParametros = $request->getPostParameters();
        $con = Propel::getConnection('solicitud');
        $con->beginTransaction();
        $parametro = json_decode($arregloParametros['solicitudes']);
        SProrrogaPagoFacilidad::gestionarProrrogaPagoFacilidad($parametro);
        $con->commit();
        return $this->renderText("{success:true}");
    } catch (Exception $e) {
```

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

```
            $con->rollBack();
            throw new Exception($e->getMessage());
        }
    }
```

Figura 3.1 Segmento de código donde se manifiesta el tratamiento de errores mediante try, cath.

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 o modificar valores de la base de datos (ver figura 3.2). Se le asigna a cada elemento del formulario un nuevo valor que posteriormente mediante la utilización de la funcionalidad *bind($valoresSolicitud)* se validarán si los datos son correctos. En caso que lo sea se actualizan los valores utilizando *updateObject($valoresSolicitud)* y por último se realiza la salva. Si los datos no son válidos se muestra un mensaje de error y no permite la inserción o modificación de estos.

```
$valoresSolicitud['fechaAprobacionRechazo'] = Date('m/d/Y');
$valoresSolicitud['idTipoSolicitud'] = $solicitud->getIdTipoSolicitud();
$valoresSolicitud['motivo'] = $solicitud->getMotivo();
$valoresSolicitud['idOperacionTipoSolicitud'] = $solicitud->getIdOperacionTipoSolicitud();
$valoresSolicitud['fSolicitud'] = $solicitud->getFSolicitud();
$valoresSolicitud['numero'] = $solicitud->getNumero();

 $solicitudForm = new SSolicitudForm($solicitud);

 $solicitudForm->bind($valoresSolicitud);

 if (!$solicitudForm->isValid()) {
    $error = $solicitudForm->getErrorSchema();
    throw new Exception($error);
} else {
    $solicitudForm->updateObject($valoresSolicitud);
    $solicitudForm->getObject()->save();
}
```

Figura 3.2 Segmento de código donde se manifiesta el tratamiento errores mediante formularios de Symfony.

El tratamiento de errores se aplica también en la vista a la hora de la validación de la entrada de datos, permitiendo que no se inserten valores incorrectos. En la figura 3.3 se muestra un ejemplo de la utilización del marco de trabajo Ext JS al validar la entrada correcta de datos mediante la interfaz.

```
if (Ext.getCmp(obj.idForm).form.isValid()) {
    obj.record.set('terminoValidez', Ext.getCmp(obj.idTerminoValidez).getValue());
    obj.record.set('plazoPagoOtorgado', Ext.getCmp(obj.idPlazoOtorgado).getValue());
    obj.record.set('estado', 1);
    obj.record.commit();
    obj.destroy();
```

```
}else{
    Ext.Msg.show({
        title: 'Alerta',
        msg: 'Debe llenar los datos requeridos.',
        icon: Ext.Msg.ERROR,
        buttons: Ext.Msg.OK,
        width: 270
    });
}
```

Figura 3.3 Segmento de código donde se manifiesta el tratamiento errores mediante el JavaScript.

3.2.3. Comunicación entre capas

Symfony es el marco de trabajo que se aplica en el desarrollo del módulo Solicitudes. Este utiliza el patrón arquitectónico MVC separando el sistema en tres capas (modelo, vista y controlador) permitiendo una mayor organización del sistema además de incrementar su rendimiento.

La capa de la Vista es la encargada de obtener la información deseada y enviarla a través de sus formularios hacia la capa del controlador, por medio de peticiones Ajax para lograr mayor rapidez y a través de objetos JSON logrando el intercambio de información entre las capas de presentación y la del controlador.

La capa del controlador está compuesta por varios componentes que se utilizan para diversos propósitos y se encargan de obtener los datos según la petición, realizando la acción que le corresponde. El controlador frontal es el único punto de entrada a la aplicación, el mismo se encarga de la configuración y determina la acción a ejecutarse. Las acciones contienen la lógica de la aplicación, encargadas de verificar la integridad de las peticiones y preparar los datos requeridos por la capa de presentación.

La comunicación entre las capas del Controlador y el Modelo se pone de manifiesto en el marco de trabajo Symfony con la implementación de un ORM que en este caso es Propel, encargado de transformar las tablas de la base de datos en objetos y realizar el tratamiento de la información necesaria para manejar las peticiones del usuario o las acciones a realizar.

3.2.3.1. Ejemplo de la comunicación entre capas

Se muestra como ejemplo el RF Gestionar Prórroga de Pago de Facilidad con el propósito de explicar mejor la comunicación entre las capas y contribuir a la comprensión del funcionamiento de la solución.

Al entrar al sistema el inspector de la Aduana interactúa con la página principal (ver figura 3.4), mostrándose una barra de menú con los tipos de solicitud que cuenta la aplicación para llevar a cabo el proceso de aprobación o rechazo.

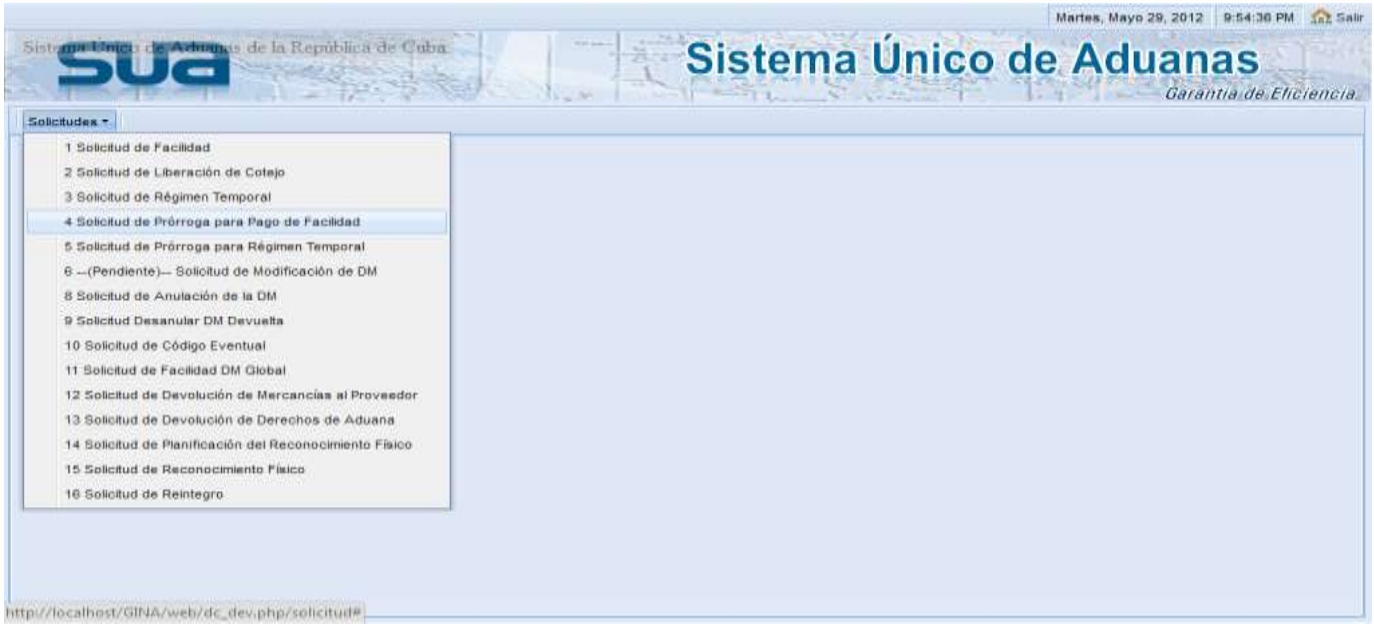


Figura 3.4. Pantalla principal de módulo Solicitudes.

Una vez seleccionada la opción, el sistema obtiene los datos a través del *store* del *grid* cargando la *url:'solicitud/mostrarProrrogaPagoFacilidadPendientes',* haciendo una petición a la clase controladora *solicitudAction*, específicamente al método *executeMostrarProrrogaPagoFacilidadPendientes()*, (ver figura 3.5).

```
public function executeMostrarProrrogaPagoFacilidadPendientes() {
        try {
            $respuesta = SProrrogaPagoFacilidad::mostrarProrrogaPagoFacilidadPendientes();
            return $this->renderText(json_encode($respuesta));
        } catch (Exception $ex) {
            throw new Exception($ex->getMessage());
        }
    }
```

Figura 3.5. Código del action de la solicitud Prórroga de Pago de Facilidad.

Al realizar la llamada a la funcionalidad mostrarProrrogaPagoFacilidadPendientes(), que se encuentra en la clase SProrrogaPagoFacilidad.php, se obtienen los datos a mostrar teniendo en cuenta el estado y el tipo de solicitud, además de la Aduana que se encuentra conectada, retornando así mediante el objeto JSON los datos que requiere la capas de presentación (ver figura 3.6)

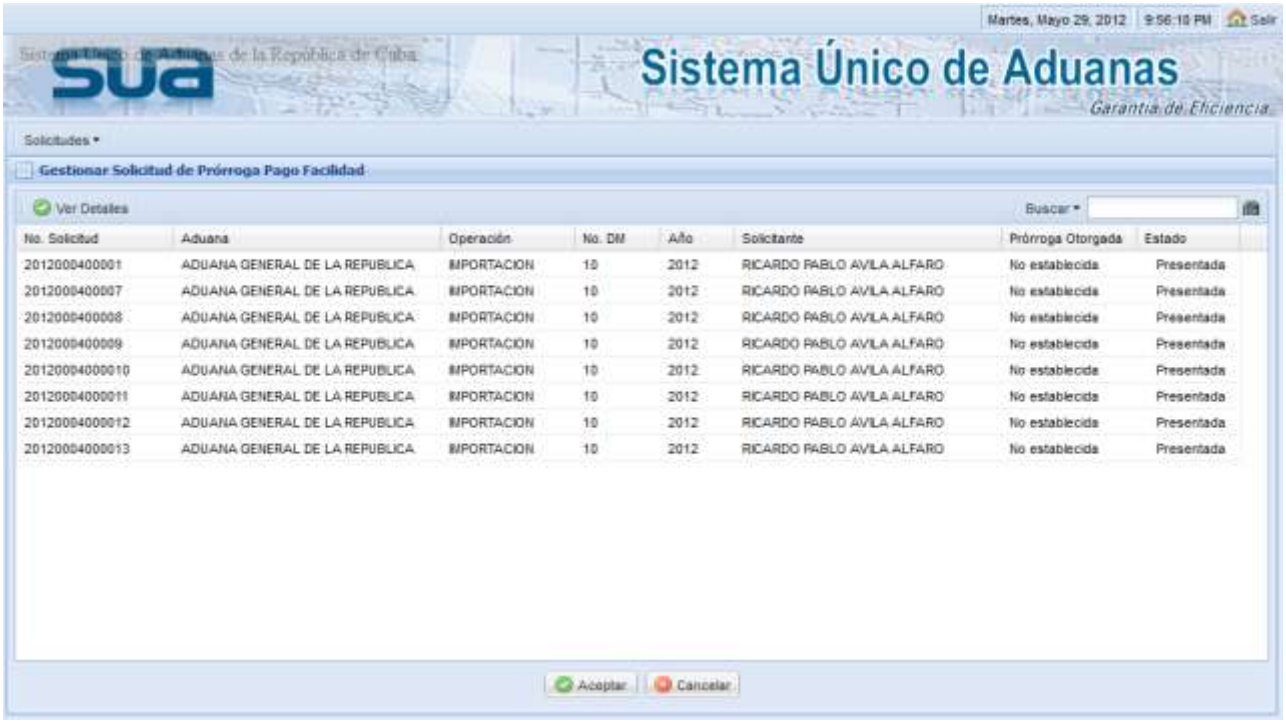


Figura 3.6. Pantalla Gestionar Solicitud de Prórroga de Pago de Facilidad.

Una vez seleccionada la solicitud a realizar el proceso de aprobación o rechazo, se ejecuta una petición Ajax (ver figura 3.7), permitiendo que se muestre un formulario con los datos de dicha solicitud(ver figura 3.3), menos el campo de prórroga otorgada, el cual lo asigna el inspector de la Aduana.

```
Ext.Ajax.request({
    url: 'solicitud/mostrarDatoProrrogaPadoFacilidadDatoSolicitud',
    success: function(response){
          var json = Ext.decode(response.responseText);
          new WGestionarProrrogaPagoFacilidad({
                record: record,
                json: json,
                storeGrid: obj.store
          }).show();
          Ext.Msg.hide();
    },
    params: {
          idSolicitud: record.data.idSolicitud
    },
    failure: function(form, action){
          FAILURE_SUBMIT(form, action);
          Ext.Msg.hide();
    }
});
```

Figura 3.7. Petición Ajax para el formulario de la solicitud Prórroga Pago de Facilidad.

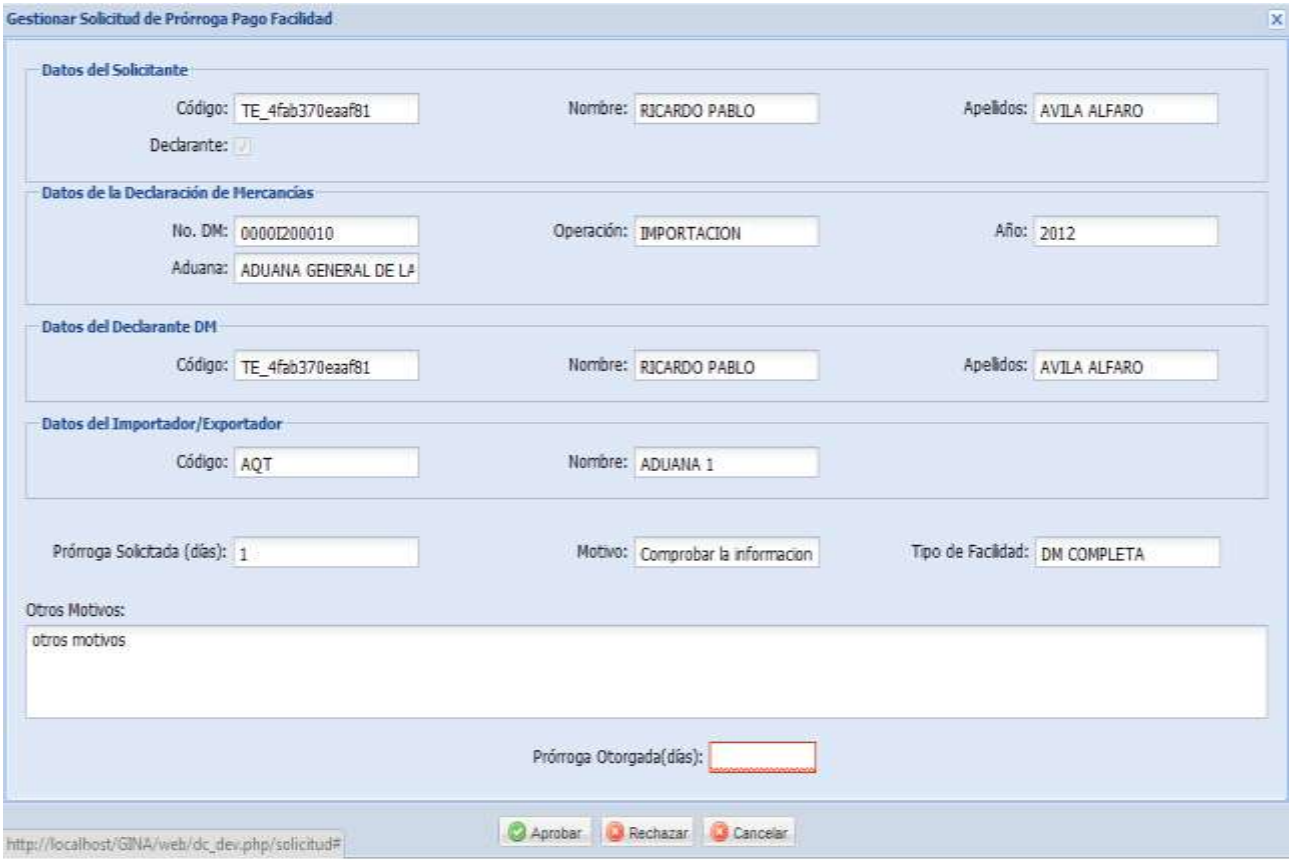


Figura 3.8. Formulario Gestionar Solicitud de Prórroga de Pago de Facilidad.

Para aprobar la solicitud 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 3.9 y 3.10.

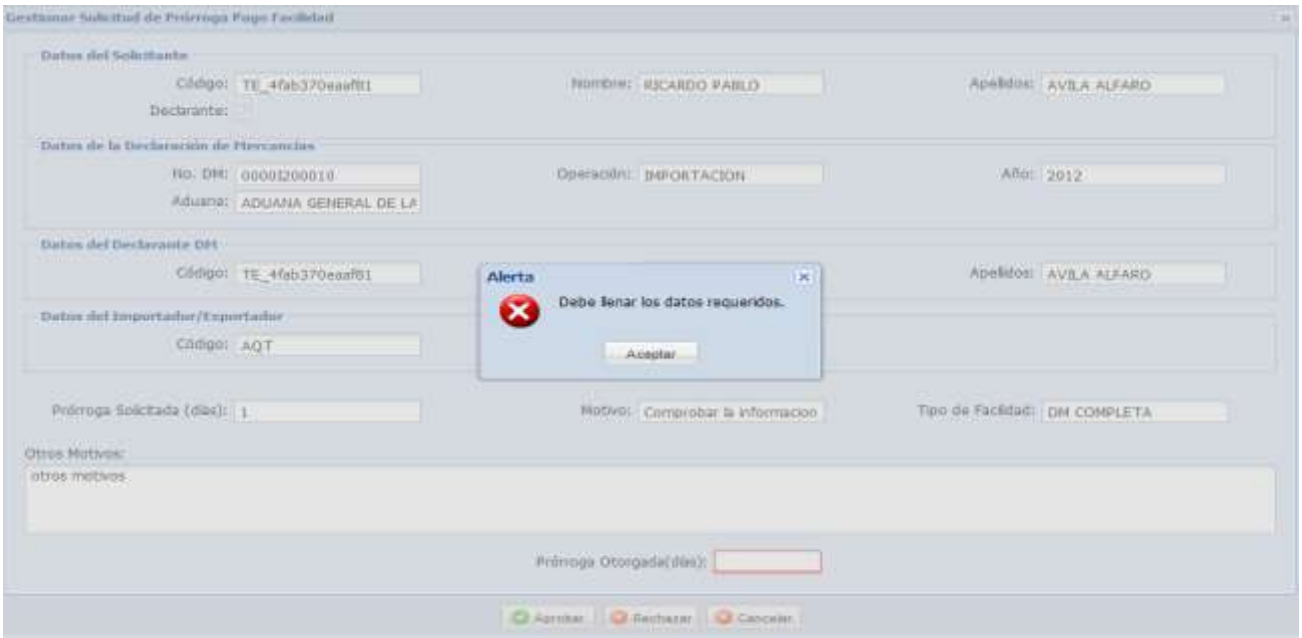


Figura 3.9. Mensaje que lanza cuando los datos no son correctos.

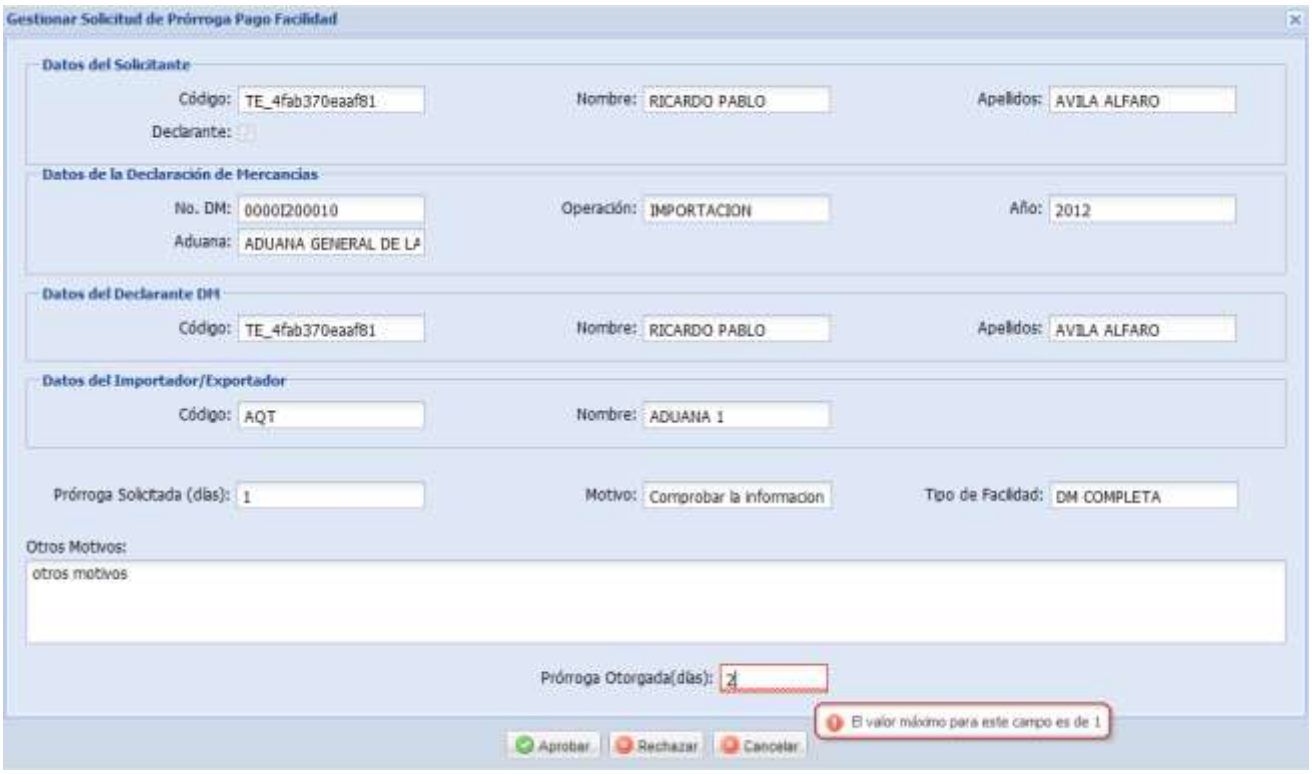


Figura 3.10 Mensaje que lanza cuando los datos no son correctos.

Una vez aprobada la solicitud se actualizan en el grid el estado y el término de validez otorgado (ver figura 3.11).

Figura 3.11 Pantalla de Gestionar Solicitud de Prórroga de Pago de Facilidad

Si el usuario selecciona el botón Aceptar se realiza una peteción Ajax enviando, mediante la utilización del método POST, los datos a la clase controladora específicamente al método *gestionarProrrogaPagoFacilidad($arregloParametros)* (ver figura 3.1).

Haciendo uso del método *getPostParameters()* se obtienen los parámetros enviados. Posteriormente se hace una llamada a la funcionalidad *gestionarProrrogaPagoFacilidad($arregloParametros)* que se encuentra en la clase *SProrrogaPagoFacilidad.php*, pasándole por parámetro los datos. Los mismos son obtenidos mediante sus llaves, realizándose las validaciones correspondientes para cada uno de ellos mediante la utilización de los formularios que genera Symfony (ver figura 3.2). Si no existen errores se actualizan los valores en la base de datos y se elimina del listado de las solicitudes pendientes a mostrar para realizar el proceso de aprobación o rechazo de la solicitud, en caso contrario los cambios previamente hechos en la base de datos son revertidos.

3.2.4. Diagrama de Componentes

En el diagrama que se muestra en la figura 3.12 se presentan los principales componentes que se utilizan en el módulo Solicitudes, describiendo los elementos físicos del sistema y sus relaciones. Al sistema se accede a través del *controlador frontal (dc_dev.php)* de la aplicación, el cual es el encargado de cargar la configuración y determinar la acción a ejecutarse. Este componente según la petición realizada se comunica con el *action.class.php*, con el *paquete Configuración* y además genera la interfaz de usuario según la petición realizada a través del *paquete de las interfaces*.

El *action.class.php* tiene como principal función establecer la lógica del negocio y la comunicación con las clases del acceso a datos, las cuales se ven reflejadas en el *paquete Abstracción de Datos de Symfony*, el cual se encuentra compuesto por los componentes *Objeto.php y ObjetoPeer.php* que utiliza *Propel* como ORM, el componente Objeto.php se relaciona con *TC*, *Persona* y *Contacto* a través de la interfaz[15] SysComponentsLocator, además del componente *DC* y *RC* a través de las clases integradoras sfDcIntegracionComun y sfRCIntegracionComun respectivamente y por último con el componente *Contenedor* a través de la interfaz sfContenedorManage.

El *paquete de las interfaces* es el encargado de vincularse directamente con el usuario, constituido por el *paquete Solicitudes*, el cual contiene todas las clases encargadas de recibir y por el *paquete Common* el cual está formado por los componentes *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 capa de la vista).

[15] **Interfaz de un componente:** contiene una colección de operaciones y se puede utilizar para especificar los servicios de un componente

El *paquete Configuración,* es el encargado de establecer las configuraciones respecto a la vista, a la seguridad del sistema, los servicios que este posee y las conexiones con la base de datos.

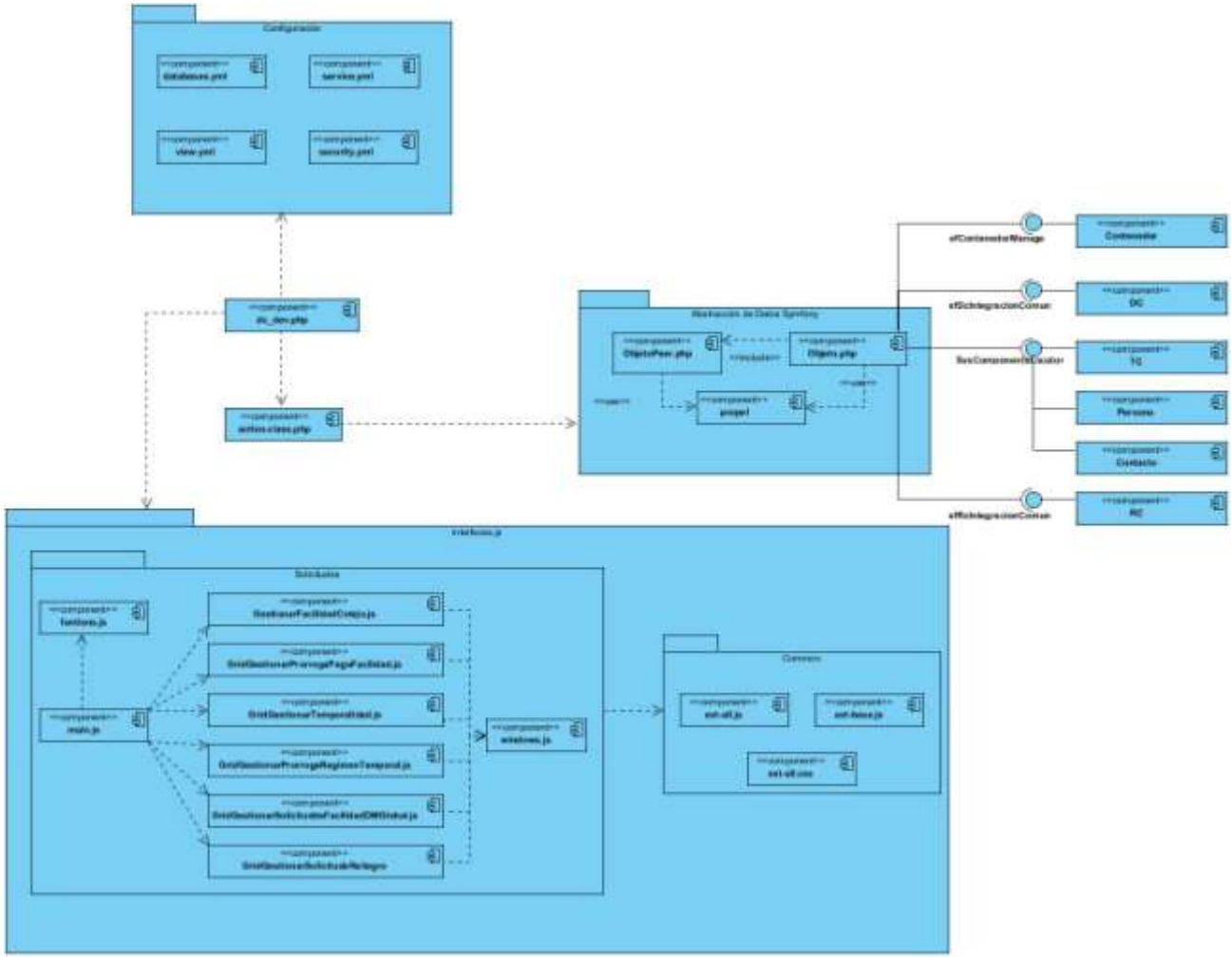


Figura 3.12. Diagrama de componentes del módulo Solicitudes.

3.3. Validación de la solución

3.3.1. Pruebas de Software

Las *pruebas del software*, conocidas también como *técnicas de evaluación dinámica* son un elemento crítico para la garantía de la calidad del sistema, representan una revisión final de las especificaciones del diseño y de la implementación. Su principal objetivo es diseñar prueba que, sistemáticamente, saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y de esfuerzo. (20)

Las pruebas no pueden asegurar la ausencia de errores; sólo puede demostrar que existen defectos en el software. (20)

3.3.2. Pruebas aplicadas.

3.3.2.1. Pruebas funcionales

Las pruebas funcionales se centran en validar las funciones que son objeto de pruebas. Con estas se ejecuta cada caso, flujo de caso de uso o función, usando datos válidos e inválidos para verificar que los resultados esperados ocurran cuando se usen datos válidos y que sean desplegados los mensajes apropiados de error y precaución cuando se usan datos inválidos. (38)

Las pruebas Funcionales están basadas en técnicas de *Cajas Negras* (38), también denominadas *pruebas de Comportamiento*, las cuales se centran en los requisitos funcionales, llevándose a cabo sobre la interfaz del software. (20)

Las mismas intentan encontrar errores de las siguientes categorías: (20)

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

Para la puesta en marcha de este tipo de pruebas se hace necesario la presencia de un buen diseño de casos de prueba, el cual es una técnica de particiones equivalentes que ha sido modificado y adaptado a los proyectos de la UCI. (38)

Los diseños de casos de pruebas recogen todas las combinaciones de escenarios posibles a ejecutar, contiene la descripción de todas las variables o datos de entrada al sistema. (38)

Aplicación de la prueba

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 contenga requisitos secundarios.

A continuación se muestra como ejemplo el diseño de caso de pruebas para el RF Gestionar Solicitud de Prórroga de Pago de Facilidad, el cual se encuentra distribuido en cuatro escenarios de pruebas.

Nombre del requisito	Descripción general	Escenarios de pruebas	Descripción del escenario	Flujos central
Gestionar Solicitud de Prórroga de Pago	Permite aceptar o rechazar las solicitudes de Prórroga de Pago	EC 1.1 Mostrar Solicitud Prórroga de Pago Facilidad.	Muestra un listado con todas las solicitudes que se	- Seleccionar opción Solicitud Prórroga de Pago de Facilidad. -Muestra una tabla con

Facilidad.	Facilidad.		encuentran presentadas.	las solicitudes que están presentadas.
		EC 1.2 Gestionar Solicitud.	Muestra un formulario con los datos de la solicitud.	-Seleccionar unas de las solicitudes mostradas e ir a la opción Ver Detalles.
		EC 1.3. Aprobar solicitud.	Es necesario insertar el campo correspondiente a la prórroga otorgada.	-Muestra los datos específicos de dicha solicitud.
		EC 1.4. Rechazar Solicitud.	No es necesario insertar valores.	-Se registra la fecha de aprobación o rechazo y la solicitud como aprobada o rechazada.

Tabla 3.1. Descripción del RF a probar.

Escenario	No.Solicitud	Nombre del Solicitante	Aduana	Estado	Operación	de Validez	No.Dm	Año											Respuesta del sistema
EC1.1	NA	NA	NA	NA	NA	NA	NA	NA											
	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío	Vacío											Informa que no hay elementos
	V	V	V	V	V	V	V	V											
	2,01E+12	RICARDO PABLO	Aduana General de la República	Presentada	IMPORTACION	No establecido	0000020000	2012											Muestra listado de las solicitudes presentadas.
	V	V	V	V	V	V	V	V											
	2,01E+12	RICARDO PABLO	Aduana General de la República	Rechazada	IMPORTACION	No establecido	0000020000	2012											Muestra las solicitudes que han sido rechazadas.
	V	V	V	V	V	V	V	V											
	2,01E+12	RICARDO PABLO	Aduana General de la República	Aprobada	IMPORTACION	2	0000020000	2012											Muestra las solicitudes que han sido aprobadas.
EC1.2	Código del Solicitante	Nombre del Solicitante	Apellidos del Solicitante	Declarante	Operación	Prórroga Otorgada	No.Dm	Año	Aduana Conectada	del declarant	del declarant	del declarante	Código de la entidad	Nombre de la entidad	a Solicita	Motivo	Tipo de Facilidad	Motivos	
	V	V	V	V	V	NA	V	V	V	V	V	V	V	V	V	V	V	V	
	TE_4fab370eaaf8f	RICARDO PABLO	AVILA ALFARRO	Si	IMPORTACION	Vacío	0000020000	2012	ADUANA GENERAL DE LA REPUBLICA	78ASAV	RICARDO PABLO	AVILA ALFARO	AGT	ADUANA1	1	Comprobar la informacion	DM COMPLETA	otros motivos	Informa que es obligatorio otorgar una prorroga.
EC1.3																			
	V	V	V	V	V	I	V	V	V	V	V	V	V	V	V	V	V	V	
	TE_4fab370eaaf8f	RICARDO PABLO	AVILA ALFARRO	Si	IMPORTACION	Vacío	0000020000	2012	ADUANA GENERAL DE LA REPUBLICA	78ASAV	RICARDO PABLO	AVILA ALFARO	AGT	ADUANA1	1	Comprobar la informacion	DM COMPLETA	otros motivos	Informa que el plazo otorgado debe ser menor o igual que el solicitado.
	V	V	V	V	V	I	V	V	V	V	V	V	V	V	V	V	V	V	
	TE_4fab370eaaf8f	RICARDO PABLO	AVILA ALFARRO	Si		2	0000020000	2012	ADUANA GENERAL DE LA REPUBLICA	78ASAV	RICARDO PABLO	AVILA ALFARO	AGT	ADUANA1	1	Comprobar la informacion	DM COMPLETA	otros motivos	
	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	
	TE_4fab370eaaf8f	RICARDO PABLO	AVILA ALFARRO	Si	IMPORTACION	1	0000020000	2012	ADUANA GENERAL DE LA REPUBLICA	78ASAV	RICARDO PABLO	AVILA ALFARO	AGT	ADUANA1	1	Comprobar la informacion	DM COMPLETA	otros motivos	Guarda la fecha y la solicitud como aprobada.
EC1.4																			
	V	V	V	V	V	NA	V	V	V	V	V	V	V	V	V	V	V	V	
	TE_4fab370eaaf8f	RICARDO PABLO	AVILA ALFARRO	Si	IMPORTACION	Vacío	0000020000	2012	ADUANA GENERAL DE LA REPUBLICA	78ASAV	RICARDO PABLO	AVILA ALFARO	AGT	ADUANA1	1	Comprobar la informacion	DM COMPLETA	otros motivos	Guarda la fecha y la solicitud como rechazada.

Figura 3.13 Caso de prueba del RF Gestionar Prórroga de Pago Facilidad.

Resultado de las pruebas

De esta forma se realizaron las pruebas de funcionalidad a los requisitos del sistema. Se realizó un total de 14 casos de prueba, los cuales se encuentran especificados en el documento de: Diseño de Casos de Pruebas del módulo Solicitudes correspondiente a cada uno de los requisitos funcionales, exceptuando el RF Registrar Solicitud.

No.	Caso de Prueba	Cantidad de escenarios	No conformidades
1	Facilidad	4	1
2	Liberación de Cotejo	4	0
3	Régimen Temporal	4	1
4	Prórroga de Pago Facilidad	4	0
5	Prórroga de Régimen Temporal	4	1
6	Anulación de la DM	2	0
7	Desanulación de la DM Devuelta	2	0
8	Código Eventual	4	0
9	Facilidad DM Global	4	0
10	Devolución de Mercancías al Proveedor	4	0
11	Devolución de Derechos de Aduana	2	0
12	Planificación de Reconocimiento Físico	4	0
13	Realización de Reconocimiento Físico	4	1
14	Reintegro	2	0
Total	**14**	**48**	**4**

Tabla 3.2 Resultado de los casos de pruebas

Luego de realizarse las pruebas funcionales al sistema llevadas a cabo, se identificó de la información recopilada 4 no conformidades (tabla 3.2), las mismas fueron sobre validaciones de campos en las interfaces y errores ortográficos. Estas inconformidades detectadas fueron resueltas en un cien por ciento, contribuyendo a la mejora del funcionamiento del sistema.

3.3.2.2. Pruebas Unitarias

Las pruebas unitarias aseguran que un único componente de la aplicación produzca una salida correcta para una determinada entrada. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular. (14)

Las pruebas unitarias de Symfony son archivos PHP cuyo nombre termina en Test.php y que se encuentran en el directorio test/unit/ de la aplicación. Su sintaxis es sencilla y fácil de leer. (14)

Aplicación de la prueba

La prueba unitaria fue realizada a cada una de las funcionalidades del RF Registrar Solicitud, las cuales consisten en retornar, dado un conjunto de datos que el usuario inserta, el número de la solicitud, el cual es de tipo String[16].

Para la realización de dicha prueba se utilizó el método *isa_ok($variable, $tipo, $mensaje),* el cual comprueba si la variable que se le pasa es del tipo que se indica, donde *$variable* es la funcionalidad a

[16]**String**: representa un tipo de un dato (conjunto de valores que puede tomar el dato durante el programa) en la programación, representa una cadena de caracteres, palabra o frase. En general es una sucesión de caracteres.

ejecutar, *$tipo* es el tipo de dato que se debe retornar y *$mensaje* es el mensaje que se muestra en caso de ejecutarse correctamente el método.

La prueba unitaria para la funcionalidad aplicada a la solicitud de Prórroga de Pago Facilidad se define de la siguiente manera:

```
$t = new lime_test(14, new lime_output_color());
$obj = new stdClass();
$obj->aduana = 1;
$obj->fechaRecibeVu = Date('m/d/Y');
$obj->motivo = 'muchos motivo ';
$obj->numeroVu = 1;
$obj->entidad = 1;
$obj->operacion = 1;
$obj->anno = date('Y');
$obj->diasSolicitado = 1;
$obj->idPersona = 1;
$obj->idMotivo = 1;

try {
    $t->isa_ok(SProrrogaPagoFacilidad::insertarSolicitudProrrogaFacilidad($obj), string, 'EXITO');
} catch (Exception $exc) {
    echo $exc->getMessage();
}
```

Figura 3.14 Prueba unitaria aplicada para la solicitud Prórroga de Pago de Facilidad.

Resultado de la prueba

Al ejecutar la prueba anterior, el resultado arrojado es el siguiente:

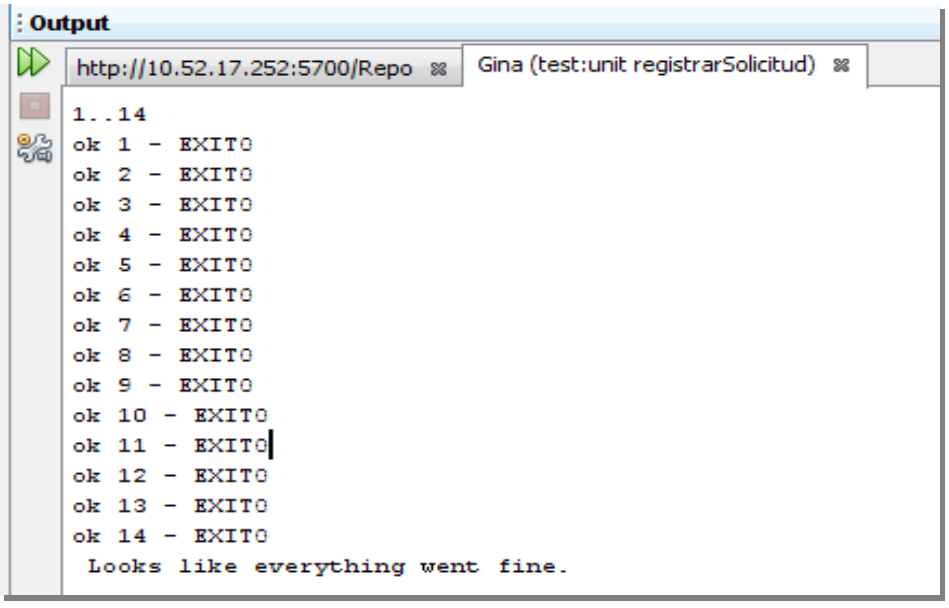


Figura 3.15 Resultado final de la prueba aplicada.

Como se observa, el resultado final muestra que la prueba realizada se ejecutó correctamente.

3.4. Impacto de la solución

Haciendo uso de herramientas y tecnologías libres, en su gran mayoría, fue posible obtener una solución que es capaz de integrarse al sistema GINA y adaptable a la integración con los subsistemas que serán implementados en un futuro.

Para realizar una prueba piloto, 12 de las solicitudes del sistema obtenidas, se pusieron en marcha en el CADI (Centro de Informatización para la Dirección e Informatización de la Aduana) desde principios del mes de abril del 2012. Luego de haber transcurrido dos meses se obtuvieron estadísticas de la aplicación con el objetivo de comparar cuán rentable es el uso de la misma.

Uno de los aspectos a tener 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 dos meses y las que fueron realizadas con el sistema en el mismo tiempo. La figura 3.10 refleja los resultados obtenidos en cuanto al indicador mencionado anteriormente con una muestra de 3 solicitudes, observándose un notable incremento en cuanto a la cantidad de solicitudes gestionadas con el sistema.

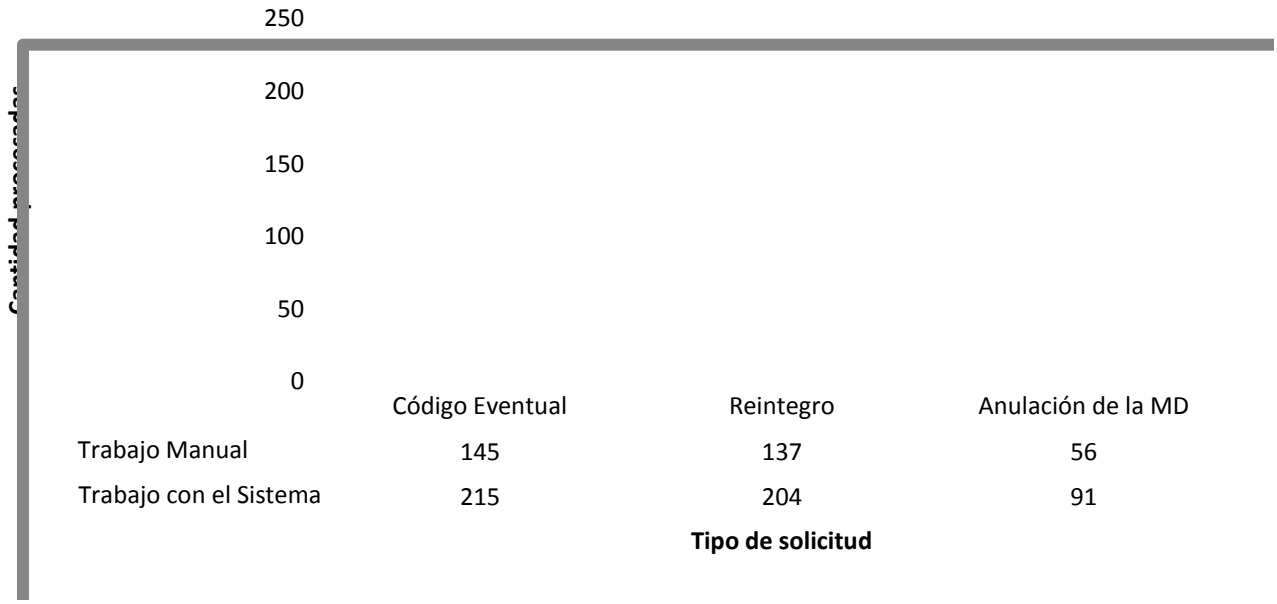


Figura 3.10 Resultado de la prueba piloto aplicada a tres solicitudes en un período de dos meses.

La utilización de la aplicación obtenida ha tenido un impacto positivo en el proceso de control manual de la información generada durante cada solicitud, además de que ha condicionado un ahorro en cuanto a materiales de oficina.

3.5. Conclusiones parciales

- Durante la fase de implementación se aprovechó al máximo las posibilidades brindadas por los marcos de trabajo utilizados, logrando un correcto tratamiento de errores y comunicación entre las capas previstas por el patrón arquitectónico MVC.
- La aplicación de las pruebas funcionales a la solución desarrollada permitió encontrar no conformidades, dando la posibilidad de que pudieran ser corregidos a tiempo y que se obtuviera un módulo que responda completamente a los RF.
- En el capítulo se pudo demostrar la correcta realización de un software con las características definidas en un inicio, potente, flexible, seguro y con una interfaz, permitiendo demostrar que el objetivo propuesto ha sido cumplido.

CONCLUSIONES

Luego de realizado el presente trabajo se llegó a las siguientes conclusiones:

- Las deficiencias identificadas en la gestión actual de la información asociada a las solicitudes realizadas por las distintas entidades, así como la ausencia de una solución que minimice las mismas, permitieron identificar la necesidad de realizar el presente trabajo.
- Para conformar la propuesta de solución se realizó un estudio del estado del arte y se tomaron las principales ventajas y experiencias de las herramientas estudiadas. Se realizó el análisis y diseño de la aplicación logrando un acercamiento a la misma para su posterior implementación.
- El módulo Solicitudes ha sido probado en el CADI logrando un alto grado de satisfacción del cliente.
- El subsistema web desarrollado, perteneciente al módulo Solicitudes del Sistema de Gestión Integral de Aduanas, facilita la gestión de la información asociada al proceso de aprobación y rechazo de las solicitudes, garantiza un registro de las solicitudes presentadas en la Aduana, además del envío de datos de estas al subsistema de DC, posibilita mayor rapidez en la realización de los procesos y es compatible con las nuevas tecnologías desplegadas en la Aduana.
- Por todo lo anterior expuesto se concluye que los objetivos propuestos para el presente trabajo han sido cumplidos satisfactoriamente ya que la aplicación da solución a la situación problémica que le dio origen.

RECOMEDACIONES

Con el fin de contribuir a un mayor desarrollo de la presente investigación se realizan las siguientes recomendaciones:

- Realizar la liberación de la aplicación por el equipo de calidad de software de la universidad.
- Garantizar la exportación de PDF (formato de documento portátil o Portable Document Format) de una de las solicitudes.
- Extender el uso del sistema a todas las aduanas de despacho del país.

REFERENCIAS BIBLIOGRÁFICAS

1. **Lundell, Mikael Berndtsson J.H. Björn Olsson y Björn.** *Thesis Projects. A Guide for Students in Computer Science.s.l. : Springer .* 2008.

2. [Online] [Cited: 11 10, 2011.] http://portal.educacion.gov.ar/universidad/gestion-universitaria.

3. [Online] [Cited: 11 10, 2011.] http://dngusisco.siu.edu.ar/aplicacion.php?ah=4eae98e5271a3&ai=convalidaciones%7C%7C14000331

4. Supportickets. [Online] [Cited: 11 16, 2011.] http://www.supportickets.com/.

5. [Online] [Cited: 11 16, 2011.] http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=PAE_PG_CTT_General&langPae=es&iniciativa=226.

6. **Dinza, Lic. Susana Enríquez Domínguez y Tec. Prof. Dennys López.** *SISTEMA DE CERTIFICACIÓN DE ORIGEN DIGITAL DE CUBA (SCOD-CUBA).*

7. *Introducción al Proceso Unificado de Desarrollo de Software (RUP) y al Lenguaje Unificado de Modelado (UML).* s.l. : EVA.

8. **Bizagi Process Modeler.** *BPMN Business Process Modeling Notation.*

9. CIENTEC. [Online] [Cited: 1 11, 2012.] http://www.cientec.com/analisis/ana-uml.html.

10. Ingeniería de Sofware. *Trabajando con Visual Parading for UML.* [Online] [Cited: 1 12, 2012.] http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf.

11. Mastros de web. [Online] [Cited: 1 12, 2012.] http://www.maestrosdelweb.com/editorial/phpintro/.

12. EcuRed. [Online] [Cited: 1 13, 2012.] http://www.ecured.cu/index.php/Php#Caracter.C3.ADsticas_de_PHP.

13. Programacion en catellano. [Online] [Cited: 1 16, 2012.] http://www.programacion.com/articulo/por_que_elegir_php_143. 13.

14. **Fabien Potencier, François Zaninotto.** *Symfony la guía definitiva.* 2008.

15. EcuRed. [Online] [Cited: 1 16, 2012.] http://www.ecured.cu/index.php/Sencha_Ext_JS#.C2.BFQu.C3.A9_es_ExtJS.3F.

16. Desarrollo en Web. [Online] [Cited: 1 17, 2012.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

17. CAVSI.com. *¿Qué es un Sistema Gestor de Bases de Datos?* [Online] [Cited: 1 19, 2012.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

18. Oracle. [Online] [Cited: 1 19, 2012.] http://www.oracle.com/us/index.html.

19. **Sommerville, Ian.** *Ingienería del Software.* s.l. : Miguel Martín Romo , 2005.

20. **Pressman, Roger S.** *Ingienería de Requisitos Un enfoque práctico.* 2005.

21. **Chaves, Michael Arias.** *La ingenieria de requisitoy su importancia en el desarrollo de protectos de software.* 2006.

22. **Informática, Escuela técnica Superior de Ingeniería.** *Ingeniería de Requisitos en Aplicaciones para la web.* s.l. : Universidad de Sevilla.

23. EcuReb. [Online] 1 27, 2012. http://www.ecured.cu/index.php/T%C3%A9cnicas_de_recopilaci%C3%B3n_de_requisitos.

24. **Martínez, Jenni Manso.** *Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE.* 2010.

25. EcuRe. [Online] 2 28, 2012. http://www.ecured.cu/index.php/Validaci%C3%B3n_de_Requisitos.

26. **Driggs, Idalberto Carlos Alonso.** *Modelo de Desrrollo del Departamento de Soluciones Para la Aduana.*

27. Guía de Patrones,Practicas y Arquitectura.Net. [Online] 1 10, 2012. http://sunshine.prod.uci.cu/gridfs/sunshine/books/PPArquitecturaNET.pdf.

28. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.*

29. EcuRed. [Online] 1 12, 2012. http://www.ecured.cu/index.php/Patrones_en_Symfony.

30. EcuRed. [Online] 1 13, 2012. http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_y_arquitectura.

31. **Bahint, Eugenia.** *El paradigma de la Programación Orientada a Objeto en PHP y el Patrón de Arquitectura de Software MVC.*

32. Ecured. [Online] [Cited: 03 26, 2012.] http://www.ecured.cu/index.php/Diagrama_de_Clase.

33. **Peña Lemus, Yudisleidys and Hernández Díaz, Yuniersy.** *SIMETSE – SIstema de METricas para evaluar el.* Ciudad de la Habana : Universidad de la Ciencias Informaticas, 2007.

34. **Trujillo Márquez, Dayán.** *Diseño del módulo de Selectividad del sistema de Gestión Integral de Aduanas .* La Habana : UCI, 2011.

35. **Leyet Frenández, Osmar and Rodríguez Lorenzo, Iosmel.** *Desarrollo de una herramienta generadora de ficheros de mapeo para la persistencia de esquema de objetos relacionales basada en NHibernate .* Ciudad de la Habana : UCI, 2008.

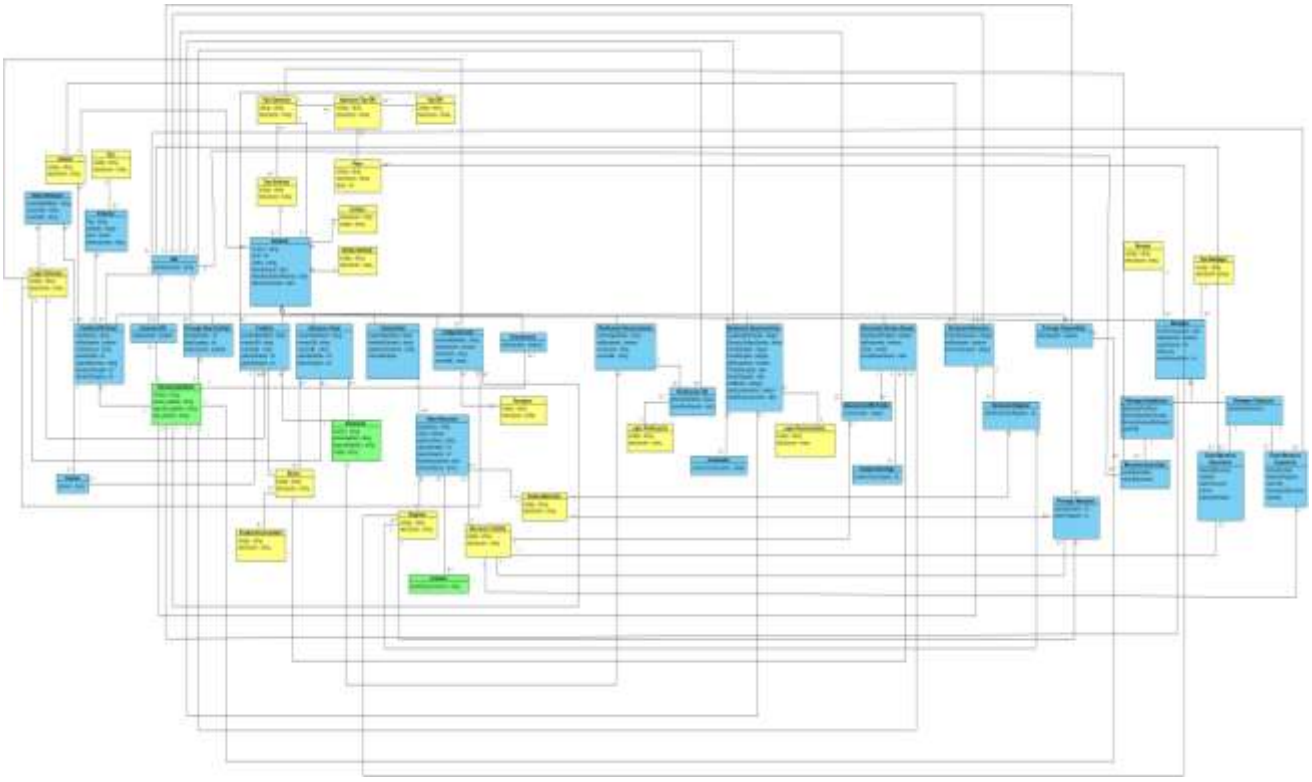
36. Merinde. [Online] [Cited: 4 2, 2012.] http://merinde.net/index.php?option=com_content&task=view&id=495&Itemid=291.

37. **CEIGE, Dpto Aduana.** *Propuesta de un Estándar de Codificación.*

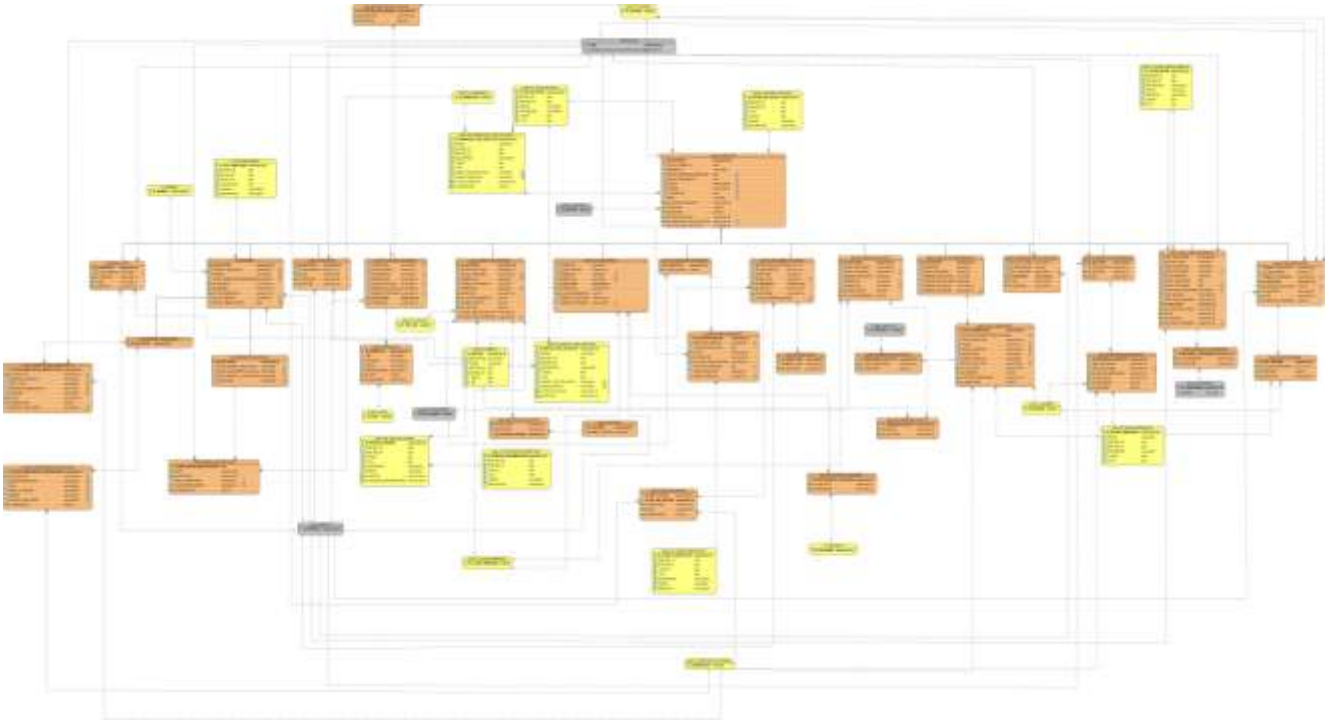
38. **Peña, Ing. Yadira Machado.** *Planificación, Diseñoy Ejecuciónde PruebasFuncionales.* Habana : Calisoft, marzo-2011.

39*. Scribd. [En línea] 14 de 6 de 2012. http://es.scribd.com/doc/19162245/Unidad-5-Modelo-Desarrollo-Software.*

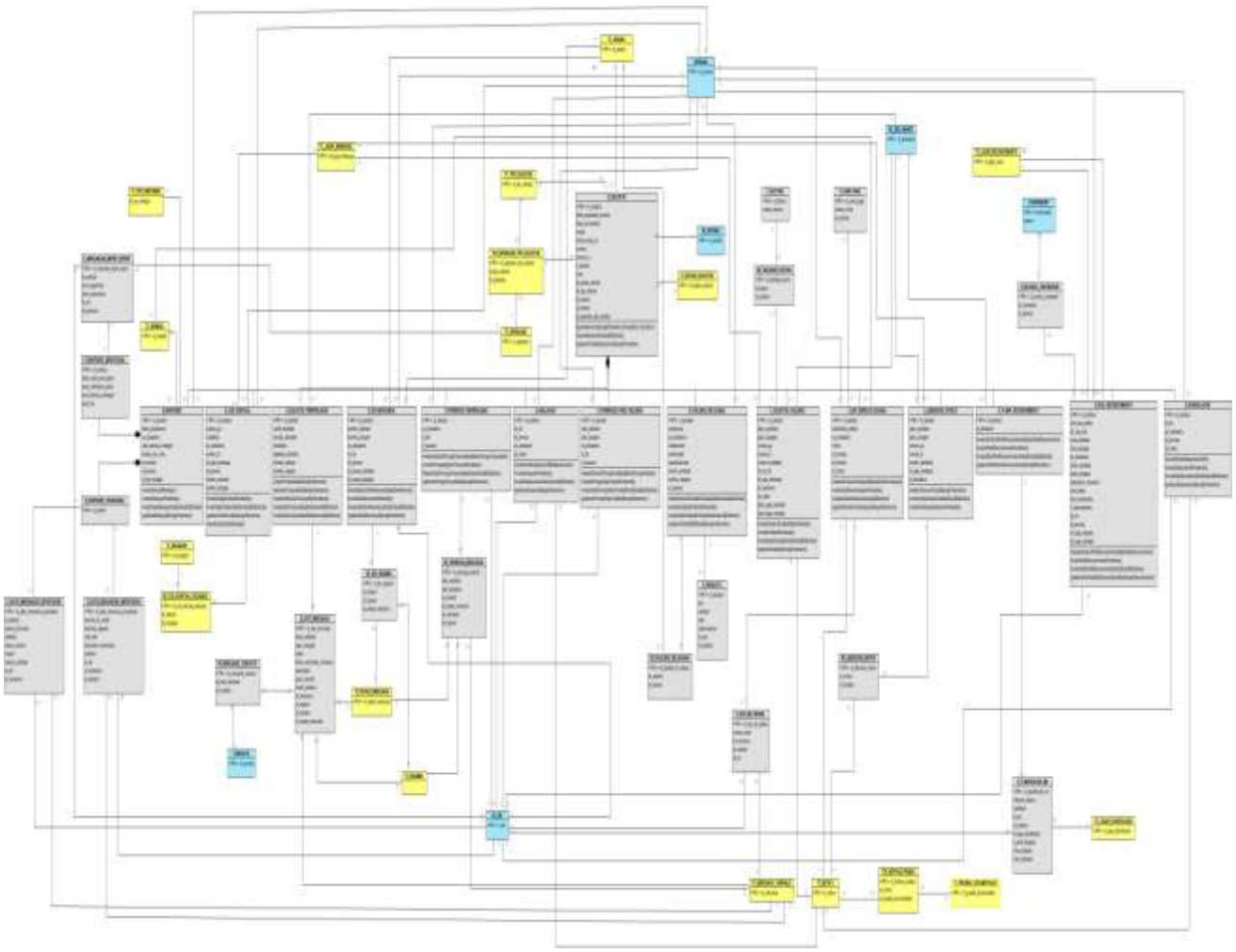
ANEXOS



Anexo 1: Modelo conceptual



Anexo 2: Modelo de datos del módulo Solicitudes



Anexo 3: Diagrama de clases del módulo Solicitudes

GLOSARIO DE TÉRMINOS

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.

Declaración de Mercancías: manifestación en la forma prescrita por la Aduana, por la que los interesados indican el régimen aduanero que se ha de aplicar a las mercancías y proporcionan los datos que la Aduana exige para la aplicación de este régimen.

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.

Grid: es una clase que representa la interfaz principal de un control basado en componente de la red para representar los datos en un formato tabular de filas y columnas.

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

IDE: (Integrated Development Environment o Entorno de Desarrollo Integrado) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código.

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 de Cuba: 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.

UNIX: es un sistema operativo multitarea, multiusuario que trabaja y funciona de manera similar a linux. se caracteriza principalmente por ser muy robusto y correr en diversas arquitecturas y plataformas.

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.

Solicitud de Anulación de la DM: solicitud que se presenta con la intención de que se anule la declaración de mercancías por motivos imputables al declarante.

Solicitud de Código Eventual: se presenta cuando las entidades van a tramitar una declaración de mercancías y no pueden completar determinada información de la misma como: proveedor, entidad y comprador que constituye información codificada en el país por los diferentes ministerios.

Solicitud de Desanulación de la DM: solicitud que se presenta con la intención de que se desanule una declaración de mercancías, previamente anulada como consecuencia de una devolución para la cual se venció el término establecido y no se volvió a presentar la declaración de mercancías corregida.

Solicitud de Devolución Derechos de Aduana: cuando por determinadas causas se modifica una declaración de mercancías que ya está pagada y se calculan los montos que por concepto de derechos y servicios de aduana debe pagar la entidad y resulta que este valor es menor que el que ya había pagado, presenta esta solicitud para que el Ministerio de Finanzas y Precio devuelva el monto correspondiente.

Solicitud de Devolución de Mercancías al Proveedor: los regímenes aduaneros de devolución garantizan que las mercancías sean devueltas al proveedor por no cumplir los requisitos contratados para que sean sustituidas por otras que se ajusten a los parámetros contratados u otras causas. Esta devolución sólo se podrá efectuar en el período de un año a partir de la fecha de registro de la

declaración de mercancías presentada bajo importación definitiva. Cuando se haya vencido este término las entidades podrán presentar esta solicitud a la aduana con el propósito de que se apruebe la utilización del régimen.

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 Facilidad DM Global: solicitud que se presenta con la intención de que se le autorice a una entidad formalizar los trámites de importación o exportación mediante declaraciones de mercancías de tipo global, ya que no posee la totalidad de la información que exige la aduana para la aplicación del régimen.

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 Planificación de Reconocimiento Físico: solicitud mediante la cual la entidad propone a la aduana una planificación para la ejecución de la revisión física de las mercancías.

Solicitud de Prórroga de Pago de Facilidad: se presenta por las diferentes entidades o empresas con la finalidad de que se le otorguen prórrogas para cumplimentar el pago de la facilidad de la declaración de mercancías, ya sea deuda informativa o documental.

Solicitud de Prórroga de Temporalidad: es la solicitud que se realiza por las diferentes entidades o empresas para que se le otorguen prórrogas con el propósito de continuar utilizando las mercancías bajo regímenes temporales.

Solicitud de Realización de Reconocimiento Físico: solicitud mediante la cual la entidad propone a la aduana que se le otorgue un canal de control de mayor rigor, con la finalidad de que se realice el examen documental y físico.

Solicitud de Reintegro: solicitud que permite, en ocasión de la importación de mercancías, obtener la restitución total o parcial de los derechos de aduanas que se han aplicado, sea a esas mercancías, a los productos contenidos en las mercancías exportadas o consumidas en el curso de su producción.

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.

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.

Store: es el objeto que se encarga de almacenar los datos que se van a mostrar en el grid.

TIC: 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.

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).