Universidad de las Ciencias Informáticas

Facultad 4

Título: Información Adelantada de Pasajeros: Análisis y Diseño.

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

Autor (es):

Lisett Hidalgo Tamayo

Randy Plasencia Herrera

Tutor (es):

Ing. Liannis Soria Barreda

Ciudad de La Habana, Mayo del 2009

DECLARACIÓN DE AUTORÍA

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

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

Randy Plasencia Herrera

_____

Firma del Autor

Lisett Hidalgo Tamayo

_____

Firma del Autor

Liannis Soria Barreda

_____

Firma del Tutor

DATOS DE CONTACTO

Tutor: Ing. Liannis Soria Barreda.

- ✓ Ingeniera en Ciencias Informáticas graduada en el 2007 en la Universidad de Ciencias Informáticas.
- ✓ Titulo de Oro.
- ✓ Adiestrada de la Facultad 4 en la UCI.
- ✓ Correo electrónico lsoria@uci.cu**.**
- ✓ Tutora de dos trabajos de diplomas en el curso 2007 – 2008 en la Facultad 4.
- ✓ Analista principal del Polo Productivo Sistemas Tributarios.

AGRADECIMIENTOS

A nuestra tutora Liannis Soria Barreda, que sin su ayuda y sus consejos nos hubiera sido imposible la confección de este documento, por su paciencia a la hora de revisarnos la tesis y por las horas que tuvo que dedicarle en su tiempo extra.

Gracias por todo.

A Manuel quien también estuvo presente y aportó ideas muy buenas.

Gracias a ti también.

A la Revolución y a nuestro sistema social que nos permite a nosotros, jóvenes sin grandes recursos, hacer realidad nuestros sueños de convertirnos algún día en graduados universitarios.

A nuestros padres por su apoyo y confianza en nosotros.

A nuestras amistades, esas que son tantas y que por temor a pecar no podemos mencionar a algunos y a otros no. A ustedes que en algún momento nos ayudaron con un consejo o alguna idea.

A nuestros compañeros de aula, que nos alentaron y confiaron en nosotros.

A nuestros compañeros de proyecto, profesores y estudiantes

Y a todos aquellos que de una forma u otra contribuyeron a la realización de este trabajo.

Lisett y Randy

DEDICATORIA

A mis padres, Belkis Herrera Fleites y Godofredo Plasencia Rodríguez, a quienes les estaré eternamente agradecidos por darme esta oportunidad de estudiar y saber siempre que cuento con su apoyo desinteresado, en los momentos buenos y sobre todo, en los malos. A ustedes les digo que me esforcé todo lo que pude y que ya casi tengo un gran sueño realizado, el tener un título universitario, en una carrera que me gusta y se además, que este es un sueño de ustedes también. Sólo me queda decirles lo siguiente, lo cual he tratado de que se convierta en una línea durante toda mi vida:

"Que adorno más grande puede haber para un hijo que la gloria de un padre, o para un padre que la conducta honrosa de un hijo."

A mi hermano Adiel Plasencia Herrera, el cual siempre me dio su apoyo. A ti te digo que no importa cuántas veces la vida te haga caer, ella te está poniendo a prueba, lo realmente importante es levantarse y seguir con más bríos y fuerzas que antes. Te deseo culmines la Universidad con excelentes notas, se que tienes para eso y mucho más porque lo has demostrado más de una vez. Confío en ti.

A mis abuelos Orocia Rodríguez y Alfredo M. Plasencia a quienes quiero y aprecio mucho ya que siempre han estado presentes con su ejemplo y me han dado consejos en la vida a través de sus cuentos y sus regaños. Para ustedes también va dedicada esta tesis, y aunque quizás no puedan leerla completa sepan que los llevo en el corazón y su ejemplo está en mi recuerdo cada vez que escribo una letra.

A mi otra familia de Santa Clara, Raquel, Alexis, Katia, Dunia, Yunel, Carlos y su pequeño hijo Daniel, a los cuales los molesto de vez en cuando, cada vez que tengo que viajar. Ustedes, como familia al fin, también se han portado muy bien conmigo y yo se los agradezco de corazón. Les diré que aunque nos alejemos un poco siempre estaremos unidos, ya que los lazos de sangre pueden más que cualquier distancia que exista.

A mi tía de cariño Cecilia Plasencia, con la cual estaré eternamente en deuda, ya que me enseñó gran cantidad de cosas en mi etapa de niño. Te quiero.

A mi compañera de tesis Lisett Hidalgo Tamayo, sin la cual este documento quizás no tuviera la calidad requerida, gracias por tu apoyo constante y por permitirme ver la vida desde un punto de vista diferente al que yo la miraba. Gracias por soportarme todos estos años a tu lado, por ser la compañera comprensiva que eres.

A Eliecer, Mary, la niña, Lieen y Leidis con los cuales mantengo excelentes relaciones.
A mis amistades del grupo 9 de la vocacional Ernesto Guevara.
A mis amistades de San Diego del Valle, a aquellos que estudiaron conmigo en la etapa de Primaria y Secundaria.
Y a todas aquellas personas que han confiado en mí y en algún que otro momento me han tendido su mano solidaria, o me han dado algún consejo.

Randy

DEDICATORIA

Quisiera dedicar este trabajo especialmente
A mis padres Ana L. Tamayo Suárez y Wald A. Hidalgo Caballero
Sin su apoyo no sería quien soy
Los quiero mucho.

A mis hermanos Miguel Ángel, Luis Alberto y Pablito los quiero mucho también y recuerden que
No hay nada imposible, porque los sueños de ayer son las esperanzas de hoy y pueden convertirse en realidad mañana.

A toda mi familia que es muy grande y no puedo mencionarlos a todos
Los quiero mucho.

A Randy
Gracias por tu gran ayuda en estos 5 años. Te quiero mucho.

A la Revolución cubana

Lisett

RESUMEN

La Información Adelantada de Pasajeros y Tripulantes constituye uno de los principales flujos de trabajo para la Aduana General de la República de Cuba (AGR). Dicha información es de vital importancia para cualquier aduana debido al continuo incremento de la cantidad de personas que viajan en el mundo, por lo que existe la necesidad de procesar toda la información obtenida en el menor tiempo posible y así tomar las medidas necesarias que permitan garantizar la seguridad nacional. La AGR con vistas a lograr una mayor eficiencia en su gestión, ya que su meta es convertirse en una aduana moderna con todo lo que esto conlleva, se ha propuesto la automatización de todos sus procesos y la modernización de su tecnología, por lo que se hace imprescindible el desarrollo de un sistema que permita reemplazar las herramientas que se utilizan en la actualidad para procesar la Información Adelantada de los Pasajeros y Tripulantes que se recibe diariamente.

El presente trabajo tiene como objetivo modelar un sistema para la Información Adelantada de Pasajeros y Tripulantes que solucione los problemas antes expuestos, garantizando de esta forma el control de este proceso con la calidad requerida por la AGR y que le permita entregar dicha información en tiempo, sin errores y con el máximo de calidad posible. Para lograr este objetivo se utilizó como metodología de desarrollo el Rational Unified Process (RUP), teniendo en cuenta las modificaciones establecidas en el Polo Productivo Sistemas Tributarios para las disciplinas definidas por esta metodología: Modelo de Negocio y Análisis y Diseño; además se utilizó la herramienta de modelado Visual Paradigm usando como base el modelado de los procesos esenciales de la organización.

Palabras Clave

Aduana, Cuba, Información Adelantada, RUP

TABLA DE CONTENIDO

ÍNDICE DE TABLAS Y FIGURAS

INTRODUCCIÓN

En los últimos años en el mundo ha existido un incremento continuo del flujo de pasajeros y el movimiento de personas de un país a otro, acompañado a este fenómeno se ha detectado un aumento en las actividades delictivas, el contrabando y la práctica del terrorismo entre otros actos ilegales que ponen en peligro la seguridad de los vuelos, los pasajeros y las instalaciones aeroportuarias, atentando a su vez contra la seguridad nacional de los países. Por la importancia que tienen estos factores para la seguridad de la Aviación Civil resultó imprescindible establecer la entrega obligatoria de la Información Adelantada de Pasajeros y Tripulantes (API) a los países destinos.

Cuba no se encuentra exenta de esta situación, por lo que ha establecido que le sea entregada de forma obligatoria la Información Adelantada de Pasajeros y Tripulantes, para mantener un control sobre las personas que van a entrar al país. Para cumplir con esto los especialistas del Centro de Automatización para la Dirección e Información (CADI) de la AGR desarrollaron un sistema como parte del Sistema Único de Aduanas (SUA) que se encarga de procesar la Información Adelantada de Pasajeros y Tripulantes recibida. Este sistema se conoce como el Paquete de Programas Informáticos para el Procesamiento Publicación y Divulgación de la Información Adelantada de Pasajeros de la República de Cuba.

En el CADI, para los desarrolladores llevar a cabo su labor, utilizan lenguajes de programación y herramientas que no les proporcionan las facilidades necesarias para hacerlo con la calidad que se requiere, tornando este proceso muy engorroso y dificultando su mantenimiento. Esta situación se contrapone al objetivo que la AGR se ha planteado de convertirse en una aduana moderna y de referencia mundial en cuanto a la gestión aduanera, por lo que es imprescindible la migración del SUA a una nueva arquitectura mucho más flexible y que les permita la facilitación del comercio con el mayor control posible, proceso que ya ha sido puesto en marcha.

Dada esta situación, la AGR necesita reemplazar el sistema utilizado para recibir la Información Adelantada de Pasajeros y Tripulantes, por un sistema que gestione dicha información y que sea capaz de integrarse a la nueva arquitectura definida a la que se encuentra migrando el SUA, lo que le permitirá una

mayor flexibilidad, seguridad y rapidez en su desarrollo, una interfaz de usuario más amigable y mayores facilidades para su mantenimiento y reutilización.

Basándose en lo anterior se puede deducir que el problema a resolver es: La AGR no tiene un sistema para gestionar la Información Adelantada de Pasajeros y Tripulantes que se integre con la nueva arquitectura del SUA.

El objeto de estudio en el cual está enmarcado el problema es: Formato de intercambio de Información de Pasajeros.

El campo de acción de este trabajo está centrado en: Los procesos de recepción de Información Adelantada de Pasajeros y Tripulantes en las Aduanas.

Para solucionar el problema científico se tiene como objetivo general diseñar una solución informática para gestionar la Información Adelantada de Pasajeros y Tripulantes.

A partir del análisis del objetivo general se derivan los siguientes objetivos específicos:

- ✓ Describir los procesos de Información Adelantada de Pasajeros y Tripulantes.
- ✓ Identificar los requisitos.
- ✓ Modelar la solución Informática.

Para lograr de forma satisfactoria estos objetivos se plantearon las siguientes tareas:

- ✓ Caracterizar la notación para el modelado de procesos BPMN.
- ✓ Caracterizar los procesos de Información Adelantada de Pasajeros y Tripulantes en las aduanas.
- ✓ Modelar los procesos de Información Adelantada de Pasajeros y Tripulantes.
- ✓ Caracterizar los formatos de la información de pasajeros.
- ✓ Caracterizar las técnicas de captura de requisitos.
- ✓ Caracterizar la plataforma de trabajo Symfony.
- ✓ Diseñar la solución informática.

Con el cumplimiento de estas tareas se obtendrían los siguientes resultados:

- ✓ El modelo de procesos de Información Adelantada de Pasajeros y Tripulantes en las aduanas.
- ✓ Requisitos funcionales de la solución a desarrollar.
- ✓ El diseño de la solución informática.

Estructura del documento:

Este documento se encuentra estructurado en 4 capítulos que se describen a continuación:
En el capítulo 1 se describirá cómo ha evolucionado la Información Adelantada de Pasajeros y Tripulantes y algunas características de los sistemas que gestionan dicha información en el mundo. Se explicará además cómo se gestiona la información adelantada en las aduanas en estos momentos y las dificultades que existen que hacen necesario el desarrollo de un nuevo sistema que automatice estos procesos.

Con el capítulo 2 se comenzará a describir la solución propuesta en este trabajo proporcionando una visión más clara y profunda del sistema a desarrollar a partir del análisis de los procesos, la definición de los requerimientos del sistema y la descripción de los casos de uso identificados.

En el capítulo 3 se explicarán las técnicas, métodos y herramientas utilizadas para dar solución al problema planteado, especificando además la plataforma y lenguaje utilizado haciendo referencia a la arquitectura del SUA con la que deberá integrarse.

Por último en el capítulo 4 se detallará la solución propuesta para el sistema a desarrollar en función del rol de diseñador desempeñado por uno de los autores de este trabajo, basándose en el diagrama de clases del diseño y el modelo físico de los datos.

CAPÍTULO 1: DESCRIPCIÓN DETALLADA DEL PROBLEMA

La Información Adelantada de Pasajeros y Tripulantes, no es más que la información obtenida de los datos del pasaporte o el visado de las personas que viajan a un país. Estos datos son transmitidos por medios electrónicos a las autoridades competentes del país de destino luego de la salida del vuelo, donde se analizan dichos datos para la gestión de riesgos antes de su llegada, con el fin de acelerar el despacho y evitar brechas de seguridad.

En este capítulo se explicará el surgimiento de la Información Adelantada de Pasajeros y Tripulantes, así como también algunas características de los sistemas informáticos que gestionan dicha información en el mundo. Se estudiará además, el sistema que se utiliza en estos momentos para gestionar la Información Adelantada en la AGR y se modelarán los procesos para el tratamiento de esta información.

1.1. Surgimiento.

Desde fecha tan temprana como el 7 de diciembre de 1944 se firma en los Estados Unidos el convenio de la Aviación Civil Internacional, donde se expresa el compromiso de cada estado contratante de crear un sistema de información anticipada sobre los viajeros y su transportación por medios electrónicos hacia el país de destino. Este convenio se firma en Cuba el 10 de junio de 1949, aunque comienza a exigirse mucho tiempo después.

Dada la importancia de la información que se maneja con estos fines se torna necesario garantizar la seguridad de la misma cuando es transmitida, por lo que en febrero del año 1949 se fundó la “Sociedad Internacional de Comunicaciones Aeronáuticas” (SITA), como una vía de cooperación entre 11 líneas aéreas (Ver Anexo #1), creando una red privada que conectaba estos aeropuertos. Estas 11 aerolíneas fueron las primeras en gestionar el tráfico en tiempo real en una Red Conmutada por Paquetes (PSN). La Información Adelantada de Pasajeros y Tripulantes es transmitida desde entonces a través de esta red, la cual se ha hecho responsable de la seguridad de la información que se transmite y de brindar todos los servicios que la navegación aeronáutica necesite, así como de su soporte. [1] [2] [3]

Los formatos de intercambio de esta información también han evolucionado desde sus inicios hasta la actualidad. En sus primero años, hasta 1987, los mensajes eran enviados como un Intercambio Electrónico de Datos (EDI), el cual era un concepto de intercambio informatizado de datos estructurados de aplicación a aplicación, soportadas por ordenadores, basado sobre mensajes normalizados y preestablecidos de modo que la comunicación electrónica se caracterizaba por tener tres formatos fundamentales de intercambio. [4]

- ✓ ANSI X12 (1978), por el «American National Standard Institute » (ANSI).
- ✓ T DCC/EDIA, por el «Transportation Data Coordinating Committee ».
- ✓ Guidelines for Trade Data Interexchange (1981), por la «UNited Nations Economic Comission for Europe, Working Party 4 ».

En el año 1987 la ONU trabajó en la preparación de una norma de sintaxis internacionalmente aceptable para la transferencia de mensajes electrónicos, creándose el formato EDIFACT (Intercambio de datos electrónicos para la Administración, el Comercio y el Transporte) por sus siglas en inglés. Este formato surge para unificar toda la información en un formato estándar que permitiera contrarrestar la gran desorganización existente en esos momentos ya que en algunas aerolíneas se utilizaba un tipo de formato y en otras se usaba otro tipo.

En el año 1988 surge la norma de calidad ISO (Organización Internacional de Normalización) 9735, la cual comprende los estándares, guías y lineamientos para el intercambio electrónico de datos comerciales estructurados entre sistemas computadorizados independientes; esta norma fue aplicada al formato recién creado, surgiendo así el formato UNEDIFACT PAXLST norma ISO 9735. El primer envío oficial de un mensaje con este formato es en el propio año 1988. A partir del año 1988 y hasta el 1996 se trabajó en el mejoramiento de este formato UNEDIFACT PAXLST norma ISO 9735, al cual no se le han realizado modificaciones hasta la actualidad.

Con el paso del tiempo algunos países vieron la necesidad de integrarse a este movimiento de intercambio de información de pasajeros y la idea fue tomando auge sobre todo en los países más ricos, los cuales eran los únicos que podían pagar los costosos sistemas con este fin, aunque todavía la idea de enviar información adelantada no se encontraba totalmente arraigada en la mente del resto de los países, los cuales no le veían su fin práctico significando además un gran costo.

No fue hasta los atentados del 11 de septiembre del 2001 en los Estados Unidos cuando el mundo se da cuenta de la gran importancia que tenía el envío de la información adelantada de sus viajeros. En este momento se evidenció la vulnerabilidad de los países del mundo en cuanto a la seguridad de sus fronteras. Esta situación unida al aumento del número de pasajeros que viajan de un país a otro y al incremento de la cantidad de vuelos que traen consigo colas de espera más largas en los aeropuertos, debido a la poca velocidad en el tratamiento de la información de los viajeros, así como las amenazas de contrabando y terrorismo que son objetos muchos países, contribuyó de manera significativa a que la Información Adelantada de Pasajeros y Tripulantes cobrara un mayor auge e importancia para todos los países del mundo.

Estos atentados, unido a la situación coyuntural expresada anteriormente, le abrieron los ojos al mundo sobre la importancia de cuidar sus fronteras para saber quien entrará a su país, fue así que en marzo del año 2003 se reúnen la Organización Mundial de Aduanas (OMA), la Asociación Internacional de Transportistas Aéreos (IATA) y la Organización de Aviación Civil Internacional (OACI), obteniéndose como resultado la guía para la Información Adelantada de Pasajeros por la cual se rige el resto del mundo. En esta reunión se recomendó el uso del formato UNEDIFACT PAXLST norma ISO 9735 para el envío y recepción de la información de las personas que fueran a viajar. [5]

No fue hasta abril del año 2007 que en Cuba se establece de forma obligatoria el envío y recepción de dicha información a partir de la resolución conjunta No. 1/2007 del Presidente del Institución de la Aeronáutica Civil Cubana (IACC) y de la AGR que se firma el 19 febrero del 2007 y entra en vigor el 1 de abril de ese año, a pesar de haberse firmado este acuerdo desde 1949 en nuestro país. [6]

En la actualidad el tratamiento de la Información Adelantada es de vital importancia para todas las aduanas del mundo y existen varios sistemas que gestionan dicha información. A continuación se expondrán algunas características de estos sistemas.

1.2. Sistemas informáticos para la gestión de la Información Adelantada de Pasajeros y Tripulantes

- ✓ **US-VISIT (United States Visitor and Immigrant Status Indicator Technology)**

Este programa es el que se utiliza en Estados Unidos y usa algunos recursos de la biometría como el lector dactilar digital sin tinta y las fotografías digitales. La biometría ayuda a resguardar la identidad en caso de que los documentos de viaje se pierdan o sean robados. La información de los pasajeros que se presentará en forma adelantada para este sistema incluye: nombre completo; fecha de nacimiento; ciudadanía; género; número de pasaporte y país de expedición de este; país de residencia; número de registro de extranjero (cuando sea necesario) y domicilio completo mientras permanecen en los Estados Unidos.

La identidad de las personas que viajan a los Estados Unidos se verifica a cada paso para tratar de garantizar que quien está cruzando la frontera sea la misma persona que recibió la visa. Este sistema además permite un control a posterior a su entrada ya que se puede saber donde radicará la persona en el país días después de haber entrado. [7]

- ✓ **BMS (Border Management Systems)**

El Sistema de Administración de Fronteras es un software para las autoridades de inmigración / control fronterizo para administrar el movimiento de personas a través de fronteras nacionales. Este sistema se especializa exclusivamente en las fronteras para el control de pasaportes, así como en las misiones en el extranjero que tenga un país para la emisión de las visas. El software BMS ha sido implementado en 4 países y aprovecha la experiencia en sistemas de inmigración para Australia y Asia.

El control fronterizo incorpora una serie de características como son: lector de documentos, integración, autenticación de documentos, captura de datos de documentos, captura de datos de arribos y partidas, almacenamiento eficiente de imágenes, verificación de listas de alerta, vigilancia y control, así como advertencias de viaje irregular, historial de personas y funciones de investigación.

El BMS maneja pasaportes electrónicos utilizando datos de chip y, de ser necesario, apareamiento biométrico. Usa múltiples idiomas como español, inglés, francés, portugués y árabe entre otros. [8]

- ✓ **PAXIS**

El sistema de información adelantada del Canadá se denomina PAXIS (sistema de información sobre pasajeros). El PAXIS tiene dos elementos, la solución de adquisición de datos (DAS) y el análisis de los pasajeros (PA). El DAS es una versión adaptada del soporte lógico IDetect de SITA y el elemento PA es

una aplicación de computadora central que fue elaborado internamente por la Agencia de Aduanas e Ingresos de Canadá (CCRA). Este sistema fue diseñado de forma tal que acepte datos de información adelantada por vía del intercambio electrónico de datos, correo electrónico o internet. Una vez que se han recibido los datos, PAXIS realiza búsquedas automatizadas en las bases de datos de ejecución de la CCRA y de Ciudadanía e Inmigración de Canadá (CIC), para luego mostrar los resultados para su examen y análisis por parte de los oficiales autorizados de aduanas e inmigraciones. Los datos se almacenan en la base de datos PAXIS y quedan a disposición de usuarios que tengan acceso a la misma durante un período máximo de seis años. Este sistema cuenta con una gran seguridad ya que el acceso a los mismos es controlado a través de la identificación de los usuarios y sus claves, esto se encuentra limitado a un pequeño número de funcionarios autorizados que son responsables de las operaciones que realizan. También tiene la capacidad de examinar todas las operaciones y comprobar la relación entre el usuario, la terminal y los datos. [9]

- ✓ **IDetect**

Este software es promocionado por los empresarios de la red SITA al resto del mundo para gestionar la Información Adelantada. En el año 2005-2006 a nuestro país le es propuesto este sistema cuando sale al mercado internacional en busca de programas automatizados que permitan mantener un control sobre la Información Adelantada. IDetect es una solución para la recolección de datos de pasajeros empleada por los diferentes organismos y agencias del gobierno que tienen participación en el transporte aéreo. Tiene como característica fundamental que puede obtener los datos totales de los PNR (número de reservación del pasajero por sus siglas en inglés), pueden mostrar los datos del pasajero y cuál es su comportamiento (por ejemplo, cuando y como pagó por su viaje), detalles completos de su viaje (por ejemplo, a que otros países viajará o a que otros países viajó previamente) y cualquier asociación potencial con otros pasajeros (por ejemplo, quien más está viajando en rutas similares, quien se sienta cerca de él en el avión, Etc.). [25]

Beneficios para Cuba al usar IDetect:

- Permite que las diferentes agencias del gobierno tengan una base de informaciones detalladas sobre las personas que vienen en un vuelo particular y contar con una aplicación para interpretar estos datos.

- Contar con un solo proveedor de comunicaciones y aplicaciones, que tenga como finalidad la simplicidad en el manejo y administración del proyecto.
- Confianza en las capacidades de entrega. La solución está basada en plataformas probadas por SITA y sus miembros iTravelAutomate, iTravelConnect e iTravelDirect.

Este software, IDetect, como se puede apreciar, tiene una serie de características muy favorables para Cuba por lo que se pensó en su adquisición, pero cuando se le propuso su compra, el principal agravante que se encontró y por el cual se decidió no comprarlo fueron los altos precios que tiene. A partir de esta situación se decidió que los especialistas del CADI desarrollaran un sistema capaz de gestionar la Información Adelantada de Pasajeros y Tripulantes. A continuación se mostrarán algunas características del sistema utilizado para gestionar la Información Adelantada en las aduanas cubanas.

✓ **Paquete de programas informáticos para el procesamiento, publicación y divulgación de la Información Adelantada de Pasajeros (API) de la República de Cuba.**

Fue diseñado, programado y puesto en prueba experimental en los meses de marzo a agosto del 2006 por los especialistas del CADI. Para su desarrollo se guiaron por las normas que rigen la OMA, la IATA y la OACI, con la tecnología que disponían en aquel entonces.

Para desarrollar este sistema se utilizó una gran diversidad de herramientas que dificultan su mantenimiento e integración con la arquitectura del SUA, así como un alineamiento con la estrategia planteada por la AGR. Para el desarrollo de este sistema se utilizaron varios lenguajes de programación, entre estos se encuentra PHP, aunque la mayor parte se realizó con Pyton y C++. Para la organización de la información se utilizó la base de datos Oracle y para la ejecución automática de los procesos de validación, procesamiento y descarga de los mensajes se utilizaron Cron realizados en Batch.

Este sistema tiene una serie de funcionalidades las cuales se muestran a continuación:

- ✓ Permite el monitoreo de los pasajeros y tripulantes.
- ✓ Permite el control de los vuelos para tener una fecha aproximada de su llegada.

- ✓ El sistema se encuentra actualizado, permitiendo conocer los datos de los viajeros y del vuelo en general ya que el flujo de información de los mensajes API no se detiene en ningún momento desde la recepción hasta que son guardados.
- ✓ Permite conocer el estado de los mensajes que han llegado.

En la AGR se han detectado algunas deficiencias en el sistema utilizado como son:

- ✓ El sistema no es compatible con la nueva arquitectura del SUA.
- ✓ Pocas opciones de análisis de la información.
- ✓ Interfaz muy poco amigable para los usuarios.
- ✓ El mantenimiento del sistema es un proceso engorroso.

Después de hacer un estudio de los sistemas de Información Adelantada utilizados en el mundo, se puede concluir que ninguno satisface completamente las necesidades de la AGR, ya sea por los precios que presentan y/o por las pocas posibilidades de gestión de la información que ofrecen. Además, el estudio realizado de estos sistemas ha servido para conocer las funcionalidades mínimas que debería tener un sistema que se desarrollase con este fin, así como las funcionalidades que lo convertirían en un sistema muy deseable para cualquier aduana en el mundo.

El primer paso en el proceso de desarrollo de software es precisamente alcanzar cierto nivel de conocimientos sobre el problema en cuestión, para lograr esto es necesario realizar el modelo del negocio. Con el modelo del negocio se logra comprender la estructura y la dinámica de la organización en la cual se va a implantar el sistema, los problemas que existen y las mejoras potenciales que pudieran realizarse, además de servir para asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización, ayudando en la derivación de los requerimientos que el sistema va a soportar. [10]

Para modelar el negocio se realizará de la forma establecida en el Polo Sistemas Tributarios, que incluye utilizar RUP como metodología de desarrollo del software con algunas modificaciones para la disciplina definida por esta metodología: Modelo de Negocio. RUP propone que se modelen los procesos del negocio utilizando el lenguaje de modelado UML, mediante los casos de uso del negocio y diagramas de

actividades que detallan como ocurre el proceso; se considera que independientemente de los beneficios que esto puede traer, implica utilizar un mayor tiempo en explicarle al cliente cómo funciona el mismo y como entenderlo, factor muy valioso en el desarrollo de sistemas informáticos.

Modelando los procesos del negocio, utilizando la notación de modelado BPMN se ahorraría tiempo y se comprendería con mayor facilidad la dinámica de la organización, logrando un entendimiento común entre los clientes y el equipo de desarrollo del software. Es por esto que se modelarán el negocio a través de procesos utilizando esta notación de modelado que presenta mayores ventajas para el cliente a la hora de comprender el proceso del negocio modelado.

1.3. Procesos de Gestión de la Información Adelantada de Pasajeros y Tripulantes

Para llevar a cabo la gestión de la Información Adelantada de Pasajeros y Tripulantes existen dos procesos fundamentales. El primero protagonizado por las aerolíneas, que tiene como objetivo principal la creación de la Información Adelantada de los Pasajeros y Tripulantes y el envío de la misma hacia las aduanas destinos. El segundo protagonizado por la AGR que es la encargada de recibir y procesar la información recibida. Estos dos procesos son representados a continuación en la Figura #1.

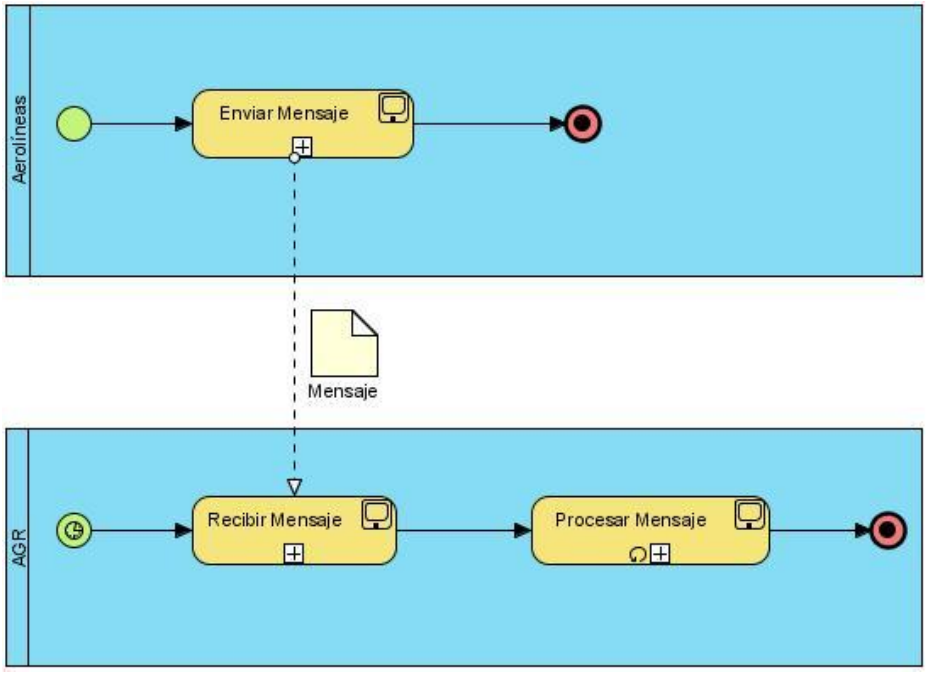

Figura #1 Procesos de Información Adelantada de Pasajeros y Tripulantes.

El envío de la información Adelantada de los Pasajeros y Tripulantes se puede realizar por la aerolínea al conformar un mensaje, el cual será generado en un fichero físico y puede ser enviado a las aduanas destinos o puede ser guardado para enviarlo en otro momento. En caso de que el mensaje se haya conformado previamente, este puede ser cargado y enviado a la AGR, así como en los casos que se necesite, el mensaje puede ser modificado antes de ser enviado, como se muestra en la Figura #2.

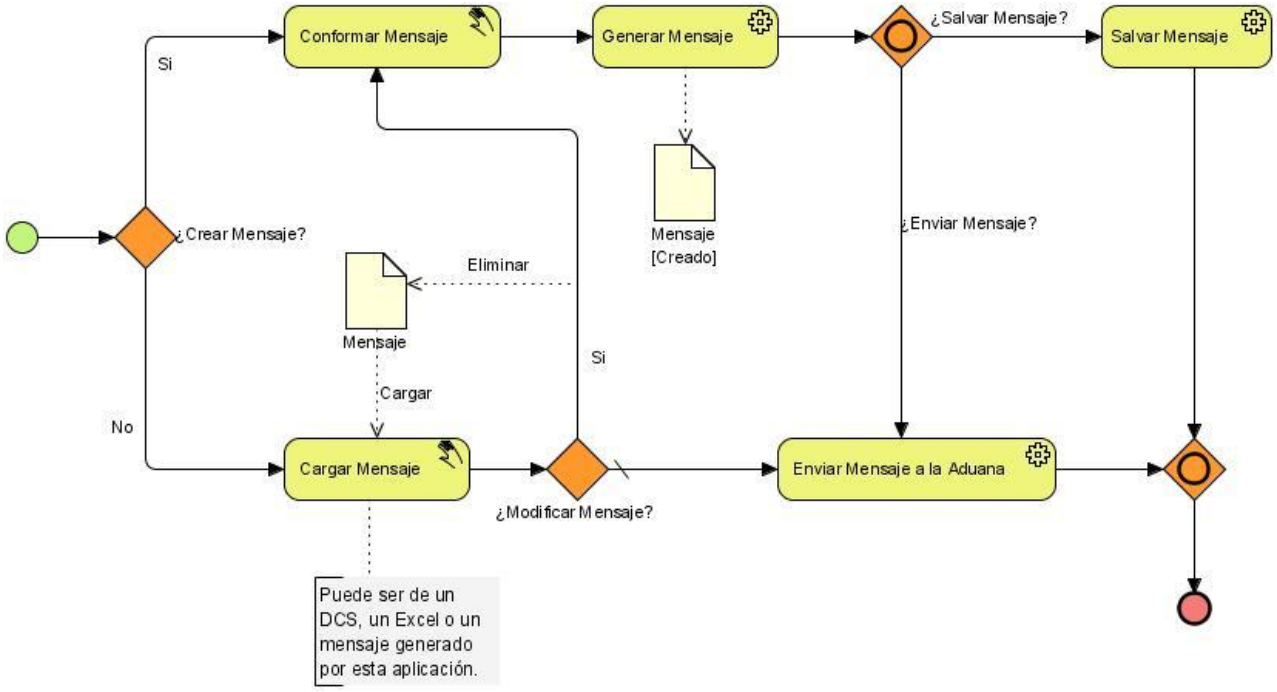

Figura #2 Enviar Mensaje.

Es importante aclarar que los mensajes pueden ser recibidos por tres vías:

- Red SITA
- Correo electrónico
- Internet

Al ser recibidos los mensajes en la AGR, estos son validados para comprobar si en realidad son mensajes API y no cualquier otro tipo de fichero que haya llegado por equivocación. Este proceso se encuentra representado en la Figura #3.

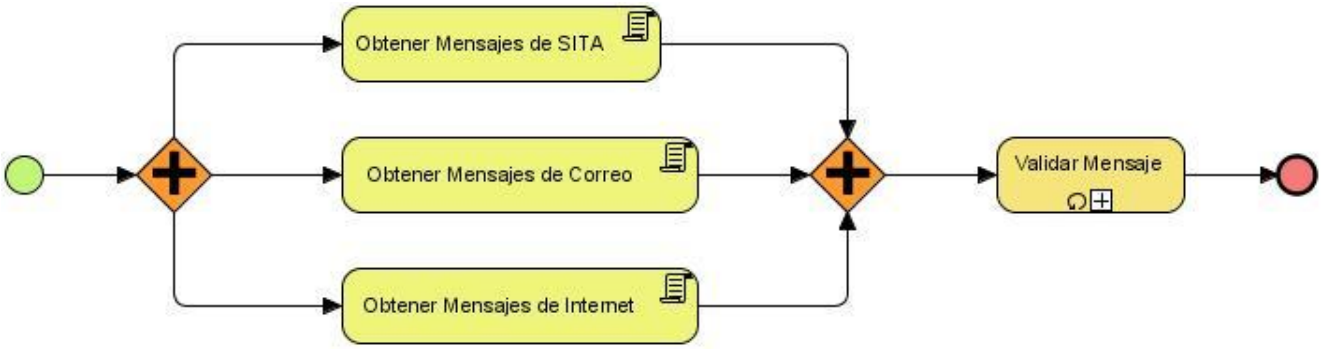

Figura #3 Recibir Mensaje.

Para determinar que un fichero es un mensaje API se verifica que sea de texto plano y que cumpla con el formato UNEDIFACT PAXLST norma ISO 9735 (Ver Anexo #5). En caso de cumplir con estas características se puede decir que es un mensaje API, en caso contrario se clasificarán como mensajes inválidos o elegibles según corresponda. Este proceso se encuentra representado en la Figura #4.

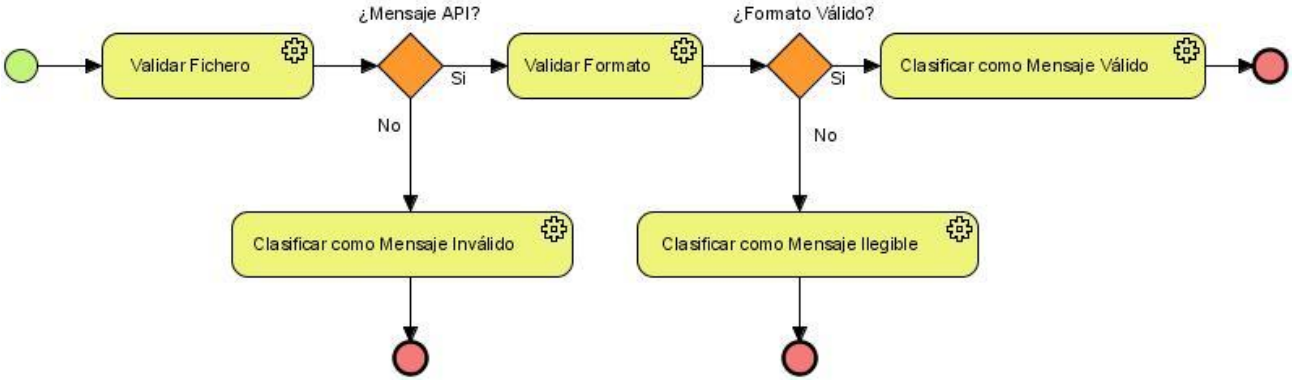

Figura #4 Validar Mensaje

Luego de que el mensaje se encuentre validado se pasa a procesar la información que este contiene, para esto se validan todos los datos y se le asigna un estado al mensaje según corresponda (Aceptado, Rechazado o Indefinido). Un mensaje es Aceptado si contiene todos los datos obligatorios establecidos por la aduana del país en el que está implantado el sistema. Si al mensaje le faltan datos relacionados con

el vuelo (origen, destino, hora de llegada, fecha de llegada, número de vuelo) el estado del mensaje será Indefinido. En caso de que al mensaje le faltan datos relacionados con los datos identificativos de las personas como (Nombre, primer Apellido, nacionalidad, Etc.) el mensaje se clasifica como Rechazado. Este proceso se encuentra representado en la Figura # 5.

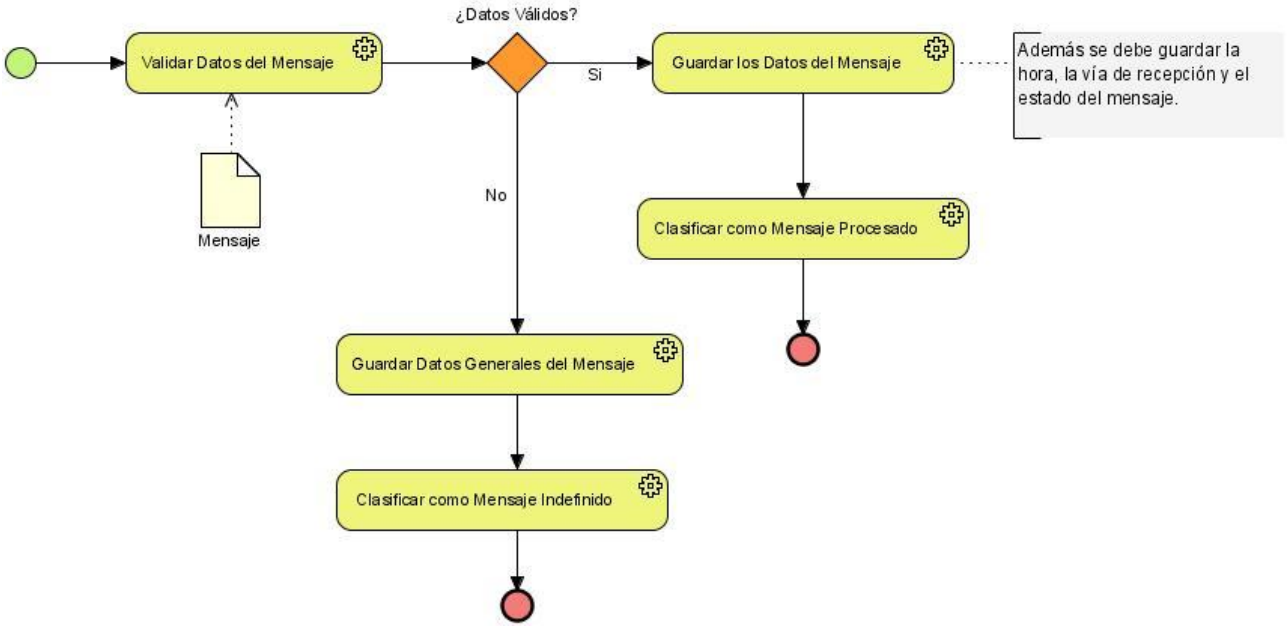

Figura #5 Procesar Mensaje.

Para obtener una mejor descripción de los procesos se puede consultar el documento Diagrama de procesos Información Adelantada de Pasajeros que se encuentra como material adjunto a esta tesis.

Después de comprender los procesos de Información Adelantada en las aduanas es preciso comenzar a definir lo que debe hacer el sistema, siendo necesario identificar y describir los requerimientos que el mismo debe cumplir.

CAPÍTULO 2: SOLUCIÓN DE SOFTWARE PROPUESTA

Para describir la solución de software que se propone desarrollar en este trabajo de diploma se realizará en el presente capítulo una exposición de los requisitos que debe cumplir dicho sistema para satisfacer las necesidades del cliente, además de mostrar una descripción textual reducida de los casos de uso del sistema y las ventajas que aporta la solución propuesta.

El propósito fundamental del flujo de trabajo de los requisitos es guiar el desarrollo hacia el sistema correcto. Esto se consigue mediante una descripción de los requisitos del sistema (es decir, las condiciones o capacidades que el sistema debe cumplir) que permita llegar a un acuerdo entre el cliente (incluyendo a los usuarios) y los desarrolladores sobre lo que debe y lo que no debe hacer el sistema. [11]

Producto de los encuentros sostenidos con los clientes y el análisis de los procesos de Información Adelantada se obtuvieron los requisitos funcionales que se describen a continuación, los cuales sirvieron de base para diseñar la solución propuesta en este trabajo.

2.1. Requisitos funcionales

Enviar mensaje API por Internet

1. Cargar un mensaje API desde un fichero.
2. Conformar un mensaje API.
3. Generar un mensaje API con los datos especificados.
4. Salvar el mensaje API elaborado en un fichero.
5. Enviar el mensaje API a la aduana destino.
6. Renombrar el fichero API.

Obtener mensaje API por Correo electrónico

7. Conectarse a una dirección de correo electrónico por IMAP o POP3.

8. Verificar cada cierto tiempo la llegada de nuevos correos.
9. Leer todos los correos nuevos recibidos.
10. Obtener el remitente de cada correo leído.
11. Obtener el fichero adjunto de cada correo leído.
12. Renombrar los ficheros adjuntos obtenidos de cada correo para su almacenamiento.
13. Eliminar los mensajes de correo electrónico luego de obtener la información que se necesita.

Recibir el mensaje por la red SITA

14. Obtener los mensajes enviados por la red SITA que se encuentran almacenados en otra computadora.
15. Renombrar los ficheros API obtenidos para su almacenamiento.

Proceso de análisis, validación y almacenamiento:

16. Validar que el fichero sea texto plano.
17. Validar que el fichero contenga el formato UNEDIFACT PAXLST.
    a. Buscar que contenga las cabeceras establecidas como obligatorias.
    b. Conformar la cabecera UNA en caso que el mensaje no contenga esta información.
18. Desglosar el fichero en varios Mensajes en caso de que el fichero contenga más de un mensaje API.
19. Clasificar los ficheros en dependencia de su contenido.
20. Leer la estructura del mensaje
21. Validar los datos que contenga el mensaje.
22. Asignar un estado al mensaje según la validez de los datos que contiene.
23. Almacenar los datos del mensaje.
24. Borrar la información de los ficheros que no cumplen con el formato de los mensajes API.

Información de interés:

25. Mostrar listado de mensajes recibidos por el aeropuerto de destino del pasajero o tripulante.
26. Mostrar la información referente a los vuelos en un rango de fechas definidas por el usuario.
27. Mostrar los datos de los viajeros pertenecientes a un vuelo específico.
28. Mostrar los mensajes en los que vino la información de un viajero y las veces que vino repetido.
29. Mostrar el estado y los mensajes pertenecientes a un vuelo.
30. Mostrar un resumen por nacionalidad de los viajeros pertenecientes a un vuelo.
31. Mostrar un resumen con las cantidades de menores, infantes y adultos que se encuentren en el vuelo.
32. Mostrar un resumen por el origen y destino de los viajeros.
33. Realizar búsquedas fonéticas de personas.
34. Conformar un parte por aerolíneas de los vuelos que han entrado al país organizado por aeropuerto, en un rango de fechas definidas por el usuario.
35. Mostrar los ficheros que no cumplen con el formato UNEDIFACT PAXLST norma ISO 9735.

2.2. Técnicas de validación de Requisitos.

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias. [22]

Las técnicas de validación de requisitos utilizadas fueron:

- ✓ **Reviews o Walk-throughs:** Está técnica consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos. Con ello solamente se validó la correcta interpretación de la información transmitida.
- ✓ **Prototipos:** Esta técnica se basa en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema. Esta técnica tiene el problema que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final (Ver Anexo # 6).

2.3. Definición de los actores del sistema

Actor	Descripción
Contacto	El contacto es el responsable por parte de la aerolínea de enviar el mensaje API. Este es el mayor interesado en que esta información llegue a tiempo.
Analista	Es la persona (aduanero) que controlará y analizará toda la información de los vuelos recibida.
Tiempo	El sistema tiene que realizar algunas funciones cada cierto tiempo. Para darle solución a esta tarea se creó un actor ficticio llamado Tiempo el cual es el encargado de avisarle al sistema la hora en que tiene que realizar estas acciones.

Tabla #1 Actores del sistema

2.4. Definición de los casos de uso del sistema

A continuación se mostrará el diagrama de casos de uso del sistema:

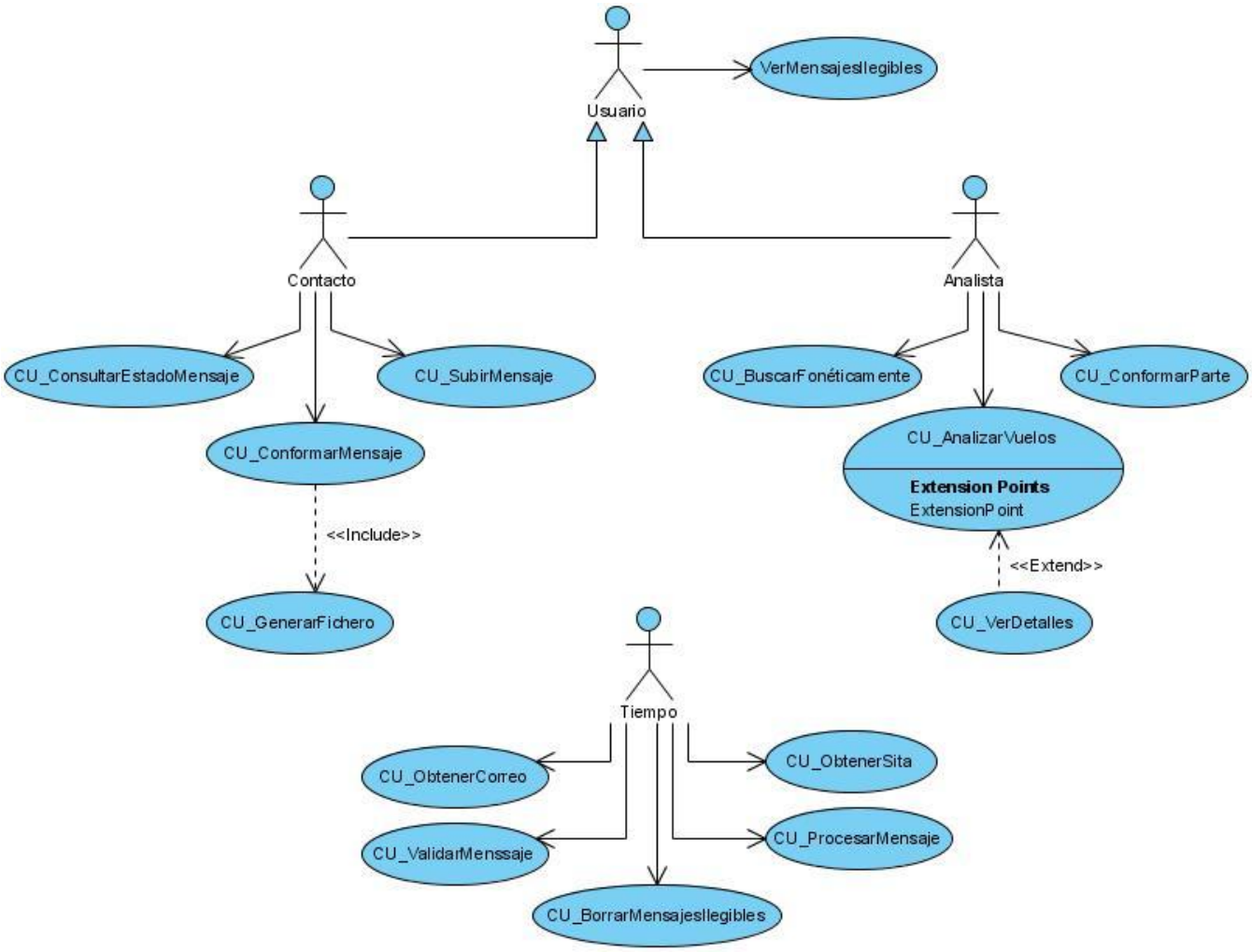

Figura #6 Diagrama de casos de uso del sistema

2.5. Descripción textual de los casos de uso del sistema

Caso de Uso	Subir Mensaje
Objetivo	Enviar un mensaje previamente elaborado.
Actores	Contacto
Resumen	Mediante este caso de uso el contacto puede enviar el fichero que contiene el mensaje API, conectándose directamente al sitio.

Complejidad	Simple
Nivel	Usuario
Trazabilidad	R1, R5, R6
Flujo Normal de Eventos	Ver expediente de proyecto, CU_SubirMensaje

Tabla #2 CUS Subir Mensaje

Caso de Uso	Conformar Mensaje
Objetivo	Elaborar un mensaje con los datos de cada pasajero y/o tripulante y el vuelo.
Actores	Contacto
Resumen	En este caso de uso se recogen los datos del vuelo, pasajeros y tripulantes necesarios para conformar el mensaje API. El sistema posibilita guardar el mensaje en la PC del actor, el mensaje será generado con el formato UNEDIFACT PAXLST.
Complejidad	Media
Nivel	Usuario
Trazabilidad	R2, R4, R5
Flujo Normal de Eventos	Ver expediente de proyecto, CU_ConformarMensajeAPI

Tabla #3 CUS Conformar Mensaje

Caso de Uso	Generar Fichero
Objetivo	Este caso de uso tiene como objetivo generar un fichero con el mensaje API.
Actores	Sistema
Resumen	El sistema genera un fichero con los datos entrados por el contacto en el caso de uso “Conformar Mensaje API”, este fichero se crea con las cabeceras establecidas en el formato UNEDIFACT PAXLST.

Complejidad	Media
Nivel	SUBFUNCIÓN
Trazabilidad	R3
Flujo Normal de Eventos	Ver expediente de proyecto, CU_GenerarFicheroAPI

Tabla #4 CUS Generar Fichero

Caso de Uso	Obtener Correo
Objetivo	Obtener los mensajes API que se encuentran en el servidor de correo.
Actores	Tiempo
Resumen	El sistema se conecta al correo cada cierto tiempo para obtener los mensajes API que se encuentran en el servidor.
Complejidad	Simple
Nivel	Usuario
Trazabilidad	R7, R8, R9, R10, R11, R12, R13
Flujo Normal de Eventos	Ver expediente de proyecto, CU_ObtenerCorreo

Tabla #5 Obtener Correo

Caso de Uso	Obtener SITA.
Objetivo	Obtener los mensajes que han llegado por la red SITA.
Actores	Tiempo.
Resumen	El sistema se conecta cada cierto tiempo al directorio donde se encuentran los mensajes que han llegado por la red SITA para obtener los ficheros API que se encuentren allí.
Complejidad	Simple
Nivel	Usuario
Trazabilidad	R14, R15

Flujo Normal de Eventos	Ver expediente de proyecto, CU_ObtenerSita

Tabla #6 Obtener SITA

Caso de Uso	Validar Mensaje
Objetivo	El objetivo de este caso de uso es verificar que el fichero que se ha enviado es realmente un mensaje API.
Actores	Tiempo
Resumen	Este caso de uso se encargará de Validar si el fichero que se ha enviado es un mensaje API, en caso de serlo se organiza el mensaje en una sola línea. Si el fichero contiene más de un mensaje, el sistema conforma nuevos ficheros con estos mensajes.
Complejidad	Media
Nivel	Usuario
Trazabilidad	R16, R17, R18, R19
Flujo Normal de Eventos	Ver expediente de proyecto, CU_ ValidarMensajeAPI.

Tabla #7 Validar Mensaje

Caso de Uso	Procesar Mensaje
Objetivo	El objetivo de este caso de uso es descifrar el mensaje para obtener su estado.
Actores	Tiempo
Resumen	En este caso de uso se verifica si el mensaje tiene los datos obligatorios establecidos por la Aduana del país donde se encuentre implantado el sistema, en relación con estos datos se le asigna un estado al mensaje y se guarda en la base de dato.
Complejidad	Complejo
Nivel	Usuario

Trazabilidad	R19, R20, R21, R22, R23
Flujo Normal de Eventos	Ver expediente de proyecto, CU_ProcesarMensaje.

Tabla #8 Procesar Mensaje

Caso de Uso	Analizar Vuelos
Objetivo	Analizar la información de los vuelos.
Actores	Analista
Resumen	El sistema muestra los vuelos que han llegado en la fecha actual. En caso de querer restringir la búsqueda de los vuelos, el especialista puede filtrar la información por fechas, aeropuerto de destino y número de vuelo.
Complejidad	Simple
Nivel	Usuario
Trazabilidad	R26
Flujo Normal de Eventos	Ver expediente de proyecto, CU_AnalizarVuelos.

Tabla #9 Analizar Vuelos

Caso de Uso	Mostrar Detalles
Objetivo	Analizar de forma detallada un vuelo.
Actores	Analista
Resumen	El sistema muestra una ventana emergente con los datos específicos del vuelo seleccionado.
Complejidad	Simple
Nivel	SUBFUNCIÓN
Trazabilidad	R27, R28, R29, R30, R31, R32
Flujo Normal de Eventos	Ver expediente de proyecto, CU_MostrarDetalles.

Tabla #10 Mostrar Detalles

Caso de Uso	Consultar Estado Mensajes.
Objetivo	Conocer el estado de los mensajes.
Actores	Contacto
Resumen	En este caso de uso el contacto puede consultar el estado de los mensajes enviados por la aerolínea a la que pertenece. La información puede ser filtrada por fecha de llegada del mensaje, vía de llegada y estado.
Complejidad	Simple
Nivel	Usuario
Trazabilidad	R25
Flujo Normal de Eventos	Ver expediente de proyecto, CU_ConsultarEstadoMensaje.

Tabla #11 Consultar Estado Mensajes

Caso de Uso	Mostrar Mensajes Ilegibles
Objetivo	Mostrar los mensajes indefinidos y los ficheros que no cumplen con el formato UNEDIFACT PAXLST.
Actores	Contacto analista
Resumen	El sistema muestra los datos de los mensajes indefinidos y los nombres de los mensajes que no cumplieron con el formato UNEDIFACT PAXLST.
Complejidad	Simple
Nivel	Usuario
Trazabilidad	R35
Flujo Normal de Eventos	Ver expediente de proyecto, MostrarMensajesIlegibles.

Tabla #12 Mostrar Mensajes Ilegibles

Caso de Uso	Conformar Parte.
Objetivo	Obtener un resumen de la cantidad de vuelos que entraron al país por aeropuerto, conocer la cantidad de aerolíneas que viajaron a ese aeropuerto y el total de todos los vuelos que entraron al país.
Actores	Analista.
Resumen	En este caso de uso se conforma un resumen de todos los vuelos que entraron al país, por aeropuerto y por aerolínea. El analista puede definir la aerolínea y/o aeropuerto a analizar. En caso de no seleccionar ningún parámetro (aerolínea o aeropuerto) se genera un parte con todos los vuelos que entraron al país en el rango de fecha especificado organizados por aeropuerto y aerolínea.
Complejidad	Simple
Nivel	Usuario
Trazabilidad	R34
Flujo Normal de Eventos	Ver expediente de proyecto, CU_ConformarParte.

Tabla #13 Conformar Parte

Caso de Uso	Buscar Fonéticamente
Objetivo	Verificar si una persona ha viajado con anterioridad.
Actores	Analista
Resumen	El sistema muestra a las personas que han viajado con anterioridad y sus datos coinciden fonéticamente con los datos especificados por el especialista de la aduana.
Complejidad	Simple
Nivel	Usuario
Trazabilidad	R33
Flujo Normal de Eventos	Ver expediente de proyecto, CU_BuscarFoneticamente.

Tabla #14 Buscar Fonéticamente

Caso de Uso	Eliminar Mensajes Ilegibles
Objetivo	Eliminar los ficheros que no cumplen con el formato del mensaje API.
Actores	Tiempo
Resumen	Este caso de uso de uso se encarga de eliminar mensajes ilegibles cada cierto tiempo, para no ocupar espacio innecesario en disco.
Complejidad	Simple
Nivel	Usuario
Trazabilidad	R24
Flujo Normal de Eventos	Ver expediente de proyecto, CU_EliminarMensajesIlegibles.

Tabla #15 Eliminar Mensajes Ilegibles

2.6. Ventajas del nuevo sistema informático.

Con la implementación e implantación en la AGR del sistema propuesto en este trabajo de diploma se podrán obtener un grupo de ventajas las cuales estarán presentes tanto en características visibles por el usuario, como en características no visibles; como son los procedimientos automatizados que se ejecutan periódicamente, los cuales serán indudablemente más legibles y estarán en un solo lenguaje de programación lo que permitirá mayor facilidad a la hora de futuras mejoras. Las ventajas a obtener con este sistema se detallan a continuación:

Interfaz amigable: Una ventaja del nuevo sistema para el procesamiento y monitoreo de la información adelantada será la de contar con una interfaz amigable, que permita a los usuarios finales (trabajadores de la aduana y especialistas de otros países) analizar con una mayor facilidad el estado de sus mensajes, así como la información a la que tenga acceso por su rol.

Independencia de una base de datos: No serán necesarios grandes cambios en el código de la aplicación si se desea o necesita trabajar con otra base de datos, solamente cambiando unos pocos archivos de configuración bastarán para que su sistema se adapte a la nueva base de datos y funcione correctamente. Esta importante ventaja se obtiene al utilizar Symfony como Framework de desarrollo.

Fácil mantenimiento del sistema: El sistema usará para la validación de los mensajes expresiones regulares que facilitarán el futuro mantenimiento del sistema, haciendo el código más legible y más fácil de mantener, evitando así tener que entrar en la estructura del mensaje API, complicando el código y haciéndolo más largo.

Como se utilizará Symfony como Framework de desarrollo este permite varias ventajas como la fácil migración del sistema de bases de datos que se utilice en la aplicación como se explica anteriormente y la separación del código en PHP del HTML, logrando con esto mayor legibilidad, además de ser una buena práctica de programación sobre todo para las futuras revisiones y mejoras. Además permite crear plantillas para crear efectos visuales las cuales se reutilizarán en las páginas permitiendo una rápida corrección ya que al modificar la plantilla no hay necesidad corregirlo página a página.

También se crearán clases específicas para cada acción de envergadura, haciendo que cada parte del sistema tenga una clase controladora que gestione y reutilice los objetos, permitiendo así que a la hora de cambiar algo sólo se necesite ir a la clase encargada de esto para corregirlo y no tener que hacerlo para el resto de la aplicación o donde quiera que se utilice esa llamada a ese método que se corrigió.

Proceso de validación de los datos configurable: Cada país tiene la libertad de escoger los datos que considere obligatorios dentro de los que se establecen como datos máximos a pedir en el mensaje API. Estos datos se encuentran recogidos en la guía para la Información Adelantada de Pasajeros conformada por las organizaciones mundiales como son la OMA, IATA y OACI. El sistema estará preparado para que cada país establezca estos datos y así a la hora de validar el mensaje, se rija por los datos seleccionados por el país donde se encuentre implantado el sistema.

Mayor claridad en el análisis de los datos: Para los trabajadores que se encuentran en el módulo estadístico del API se crearán una serie de reportes que permitirán conocer los vuelos que han llegado en un determinado período de tiempo y sus datos, incluyendo a cada una de las personas que vienen en estos vuelos. En la representación que se realizará de la información mostrada al cliente se permitirá la

utilización de filtros y búsquedas de datos específicos, lo que agilizará el proceso de análisis de la información disponible.

Alta Seguridad: Alta protección contra ataques por inyección SQL o Código Java script ya que el Framework es capaz de detectarlos y evitarlos.

Eliminación de los ficheros ilegibles: Se incluirá además un procedimiento para eliminar los documentos que se envían por error a la AGR que no constituyen un mensaje API, lo que permitirá eliminar información inútil de los servidores que sólo complejizaría las búsquedas a realizar.

CAPÍTULO 3: TÉCNICAS, MÉTODOS Y HERRAMIENTAS UTILIZADOS

En este capítulo se describirán las herramientas utilizadas para diseñar la solución informática propuesta, la arquitectura en la que se enmarca dicha solución y la notación empleada para el modelado de los procesos del negocio. Además se abordarán las técnicas que se utilizaron para la captura de los requisitos del sistema a desarrollar, así como los patrones de diseño que se tuvieron en cuenta para darle solución al problema planteado.

Esta sección culminará detallando el flujo de actividades asignadas a los roles en los que se desempeñaron los autores de este trabajo, los roles de analista y diseñador en el Polo Productivo Sistemas Tributarios, así como sus competencias y responsabilidades.

3.1 Arquitectura

La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. El objetivo principal de la Arquitectura del Software es aportar elementos que ayuden a la toma de decisiones y al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto. Para conseguirlo, la Arquitectura del Software construye abstracciones, materializándolas en forma de diagramas. [12]

La solución propuesta en este trabajo se ajustó a la arquitectura definida en el Polo Productivo para todos los sistemas que se desarrollen para el SUA. En la Figura #7 se muestra un diagrama que especifica la arquitectura utilizada en el Polo Sistemas Tributarios. A continuación se realizará una breve descripción de la misma para comprender como la solución desarrollada se enmarca en esta arquitectura.

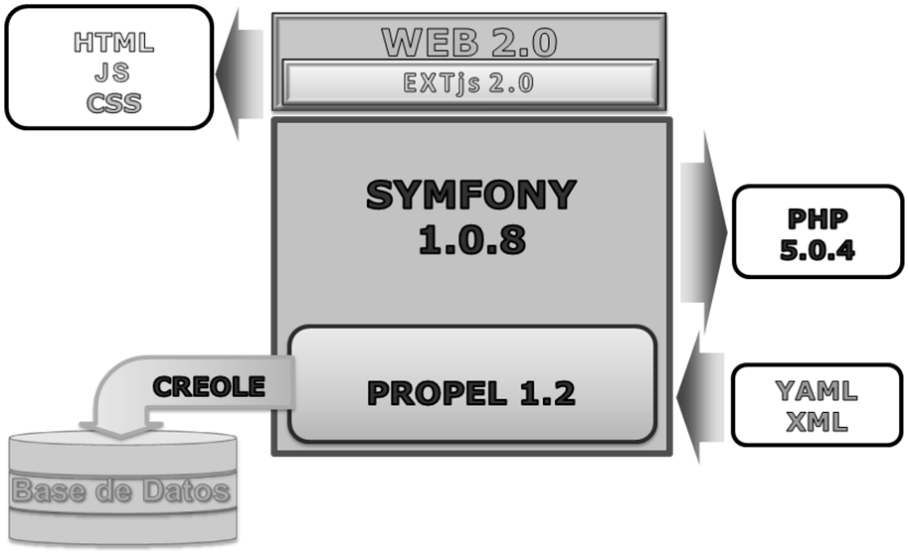

Figura #7 Arquitectura del SUA.

- ✓ **Uso de Symfony como Framework Arquitectónico**

Este Framework ofrece una serie de características que lo hacen muy útil en el desarrollo de aplicaciones Web, algunas de estas se exponen a continuación: [13]

- Facilita herramientas para desarrollar aplicaciones web de alta complejidad.
- Adopta buenas ideas de otros proyectos, reutilizando el código existente cuando está disponible.
- El código desarrollado es fácil de mantener.
- 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.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Separación de modelo, vista y controlador (implementación MVC).
- Utilización de patrones de diseño.

- Independiente del sistema gestor de bases de datos.
- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Linux).
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.

✓ **Propel como capa de persistencia de los datos**

Aunque Propel está desarrollado independiente a Symfony ambos se complementan muy bien, desarrollando una combinación muy rápida y flexible en cuanto al acceso a la información de una base de datos. En la solución se utiliza el Propel como capa de persistencia a la base de datos. [14]

✓ **EXTjs Presentación y Comunicación**

Es una librería de Java Script para construir páginas Web dinámicas, se utiliza en el desarrollo de la solución para mejorar la calidad de la aplicación desde el punto de vista del usuario final, para mejorar su entorno gráfico y que sea más amigable ya que tiene un conjunto de componentes desarrollados con este fin. También cuenta con validación de formularios, lo que permitirá una mayor seguridad para no introducir datos erróneos.

Otro elemento que se utilizará mucho será la posibilidad que brinda de soportar serialización de objetos mediante JSON, permitiendo que los datos enviados desde el controlador en respuesta a la vista ya no sean objetos sino un JSON que contenga sus propiedades pero no su comportamiento, permitiendo esto una mayor seguridad ya que minimiza el riesgo de que los objetos sean modificados erróneamente en la vista. [14]

✓ **Web 2.0. Una Actitud**

Este término no es más que una transición que ha ocurrido en los últimos años hacia aplicaciones que funcionan a través de la web, enfocada al usuario final. Se quiere reemplazar a las aplicaciones de escritorio con otras que ofrezcan colaboraciones y servicios. Esta solución es una actitud y no una tecnología. [14].

3.2 BPMN (Notación de Modelado para los Procesos del Negocio)

BPMN proporciona a los negocios la capacidad de entender sus procedimientos internos en una notación gráfica, facilitando a las organizaciones la habilidad para comunicar esos procedimientos de una manera estándar. Por tanto las principales características de BPMN de manera reducida son: [15]

- ✓ Proveer una notación que sea fácilmente entendida por todos los usuarios, desde el analista de negocio, el desarrollador técnico y hasta la propia gente del negocio.
- ✓ Crear un puente estandarizado para el vacío existente entre el diseño del proceso de negocio y su implementación.
- ✓ Asegurar que los lenguajes para la ejecución de los procesos de negocio puedan ser visualizados con una notación común.
- ✓ Es usado para comunicar una amplia variedad de información a una amplia variedad de audiencias.

Para la realización de los diagramas utilizando esta notación se utilizó la herramienta Visual Paradigm siguiendo la política establecida en la universidad para el desarrollo de sus productos. Esta herramienta permite de una manera muy fácil y eficiente desarrollar los diagramas necesarios para desarrollar la solución informática propuesta.

3.3 Visual Paradigm

Es una herramienta CASE (Ingeniería de Software Asistida por Ordenador) profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Esta herramienta tiene una serie de características entre las que se destacan: [16]

- ✓ Soporta la versión UML 2.1.
- ✓ Soporta Ingeniería Inversa.
- ✓ Soporta BPMN.
- ✓ Generación del modelo físico de datos.

- ✓ Documentación de la captura de requisitos.

3.4 Técnicas de captura de requisitos

Para desarrollar un producto lo más importante es saber los servicios y restricciones que debe cumplir el mismo, saber con exactitud lo que los clientes desean y como lo desean; a este proceso se le llama captura de requisitos. Al no existir fórmulas matemáticas ni algoritmos para que los desarrolladores se pongan de acuerdo con los clientes en cuanto a lo que estos necesitan, se hace muy difícil la captura de requisitos, además de tener una gran dependencia de las personas a las que se consulte en el proceso. Por todo esto, la Ingeniería de Requisitos ha trabajado arduamente en el desarrollo de técnicas que permitan realizar este proceso de una forma más eficiente. [22]

Por mucho que se ha trabajado en este campo, todavía no se puede decir que una técnica es efectiva por sí misma, estas requieren ser combinadas para lograr “una correcta” especificación de los requisitos. Para lograr una buena especificación de requisitos se debe tener en cuenta las siguientes consideraciones: [17]

- ✓ Ambiente de la Especificación de Requisitos. Debe de estar descrita de tal manera que no describa aspectos del área de diseño o de implementación.
- ✓ Características de los Requisitos. Los requisitos descritos deben ser:
  - ➢ Completos.
  - ➢ Implementación Independiente.
  - ➢ Consistente y no Ambiguo.
  - ➢ Preciso.
  - ➢ Verificable.
  - ➢ Que pueda ser leído.
  - ➢ Modificable.

- ✓ Aprobación del Cliente o patrocinador. Todos los requisitos descritos deben contar con esta aprobación. Hay que tener en cuenta que algunos requisitos pueden variar con el tiempo y hay que tener en cuenta esta variación. En este caso existen requisitos tales como:
  - ➢ Costo.
  - ➢ Fecha de Entrega.
  - ➢ Criterios de Validación y Verificación.

Las técnicas utilizadas para la captura de requisitos fueron:

- ✓ **Introspección:** Esta técnica recomienda que el ingeniero de requisitos se ponga en el lugar del cliente y trate de imaginar cómo el cliente desearía el sistema. En base a estas suposiciones se debe recomendar al cliente sobre la funcionalidad que el sistema debería presentar. El principal problema de esta técnica radica en que un ingeniero no es un tipo normal de cliente ya que posee un conocimiento técnico más elevado, por lo que se podrían recomendar cosas que el cliente no necesitara.
- ✓ **Entrevistas:** Existen diferentes tipos de entrevistas, a continuación se explicarán los tipos de entrevistas utilizados para el desarrollo del trabajo:
  - ➢ **Entrevistas en grupos de desarrollo:** Este tipo de entrevistas recomienda formar grupos específicos con el personal del cliente. Estos grupos tendrán en común algún área de trabajo o especialidad. El objetivo es poder contar con los expertos en cierta área de la empresa para poder llegar en conjunto a la especificación de requisitos. Luego de cada entrevista se redactaba la minuta de la reunión para ser analizada con posterioridad.
  - ➢ **Open Ended Interview (Entrevista abierta terminada):** Este tipo de entrevistas son del tipo que realizan los psicólogos. La idea es que el ingeniero de requisitos permita que el cliente le vaya platicando su problemática y el ingeniero de software lo va a ir guiando a través de la plática para ir determinando los requisitos del sistema.
- ✓ **VORD:** Esta técnica es utilizada para capturar requisitos en base a puntos de vista, ya que se tomaron los puntos de vista de los diferentes entrevistados para encontrar cosas en común. Es el

analista el encargado de unir estos puntos de vista y conformar los requisitos. Tiene cuatro pasos fundamentales:

- Identificación del punto de vista.
- Estructuración de dichos puntos de vista
- Documentación de puntos de vista (refinación)
- Trazado del punto de vista.

- ✓ **Análisis de Protocolo:** Esta técnica parte de la idea de que el cliente cuenta con un modelo mental preexistente del sistema deseado y en base a este modelo ya existente se puede analizar y obtener los requisitos del sistema. Es una técnica muy poco utilizada debido a que los clientes rara vez poseen una idea clara de lo que desean en su sistema, pero en este caso los clientes si tenían una idea del sistema bien definida.
- ✓ **Brainstorming (Tormenta de ideas):** Es también una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. Como técnica de captura de requisitos es sencilla de usar y de aplicar. Suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que se aplicó en los primeros encuentros. Por la importancia de estas reuniones quedó establecido que participaran el analista, el diseñador y el jefe de proyecto donde se analizaron los diferentes sistemas que se utilizan en el mundo para gestionar la información Adelantada de Pasajeros.

3.5 Patrones de Diseño utilizados

Un patrón, en términos generales es una solución a un problema en un contexto dado, es recurrente, lo que hace que sea relevante para otras soluciones. Es una solución a un problema de diseño no trivial que es efectiva y reusable.

Resumiendo, en términos generales, un patrón es un conjunto de información que proporciona una respuesta a un conjunto de problemas similares, es decir, un patrón es una solución a un problema en un contexto, donde: [18]

- ✓ Contexto son las situaciones recurrentes a las que es posible aplicar el patrón.
- ✓ Problema es el conjunto de metas y restricciones que se dan en ese contexto.

- ✓ Solución es el diseño a aplicar para conseguir las metas dentro de las restricciones.

Existen diversos patrones de diseño para desarrollar aplicaciones, estos se pueden agrupar en varias categorías como son los patrones GOF ("Gang of Four" o banda de los 4 debido a sus cuatro autores, los cuales fueron Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides) y los patrones GRASP (patrones generales de software para asignación de responsabilidades), estos últimos se consideran que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendables en el diseño de software.

A continuación se describirán los patrones utilizados para el diseño de la solución propuesta.

- ✓ **Patrones GOF implementados**

  - ➢ **Patrón FACADE**

Problemática: ¿Cómo puedo acceder a diferentes clases a través de una única clase?

Propósito: Simplificar el acceso a un conjunto de clases o interfaces. Proporcionar una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La "fachada" satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas.

Solución: Symfony mediante el sistema de configuración de archivos YML implementa este patrón permitiendo acceder a diferentes configuraciones del Framework desde un único lugar. Además de que el acceso a la aplicación es a través del controlador frontal que implementa este patrón.
Este patrón permite utilizar una interfaz común para un conjunto de interfaces del módulo, haciendo que este sea más fácil de usar.
Por ejemplo, en el módulo Internet del proyecto SARPIA se cuenta con el archivo view.yml, el cual provee la configuración del módulo, generando la interfaz seleccionada, con los accesos a las páginas o reportes seleccionados. [14]
Ejemplo:
javascripts: [sarpia/internet/reporteVuelosGeneral, sarpia/internet/json, sarpia/internet/infoReporteVuelo]

Explicación: Al agregar direcciones de archivos JavaScript a este archivo de configuración la interfaz a la hora de crearse buscará en estos antes de mostrarse y mostrará lo que se encuentre en estos archivos permitiendo así acceder a otros lugares de mi aplicación desde un único lugar. Además si existiera otro módulo este tendría su view.yml donde se podría configurar todo el acceso de este nuevo módulo.

- **Patrón DECORATOR**

Problemática: ¿Cómo añadirle una nueva característica dinámicamente a una clase, pudiéndose así obtener una gran funcionalidad combinando piezas sencillas?

Propósito: Añadir responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades.

Solución: En Symfony la plantilla no es un documento XHTML válido ya que le falta la definición del DOCTYPE y las etiquetas <html> y <body>. El motivo de este problema es que estos elementos se encuentran en otro lugar de la aplicación, un archivo llamado layout.php que contiene el layout de la página. Este archivo, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde otro punto de vista, el layout *decora* la plantilla. [14]

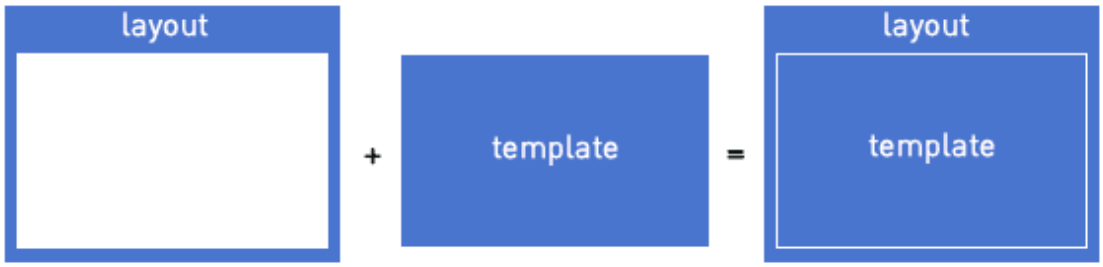

Figura #8 Ejemplo de Implementación del patrón Decorator en Symfony

- **Patrón SINGLETON**

Problemática: Obtener un punto de acceso global a una clase. (Instancia única).

Propósito: Asegurar que sólo exista una instancia de una clase específica en un sistema a desarrollar y la creación de un mecanismo de acceso global a dicha instancia.

Solución: El controlador frontal proporciona un punto de acceso global a la aplicación. En una acción, el método getContext() devuelve el mismo singleton. Se trata de un objeto muy útil que guarda una referencia a todos los objetos del núcleo de Symfony relacionados con una petición dada, y ofrece un método de acceso para cada uno de ellos. A continuación se muestran algunos ejemplos de la implementación de este patrón: [14]

1. sfController: El objeto controlador (->getController())
2. sfRequest: El objeto de la petición (->getRequest())
3. sfResponse: El objeto de la respuesta (->getResponse())
4. sfUser: El objeto de la sesión del usuario (->getUser())
5. sfDatabaseConnection: La conexión a la base de datos (->getDatabaseConnection())
6. sfLogger: El objeto para los logs (->getLogger())
7. sfI18N: El objeto de internacionalización (->getI18N())

Se puede llamar al singleton sfContext::getInstance() desde cualquier parte del código.

- **Patrón ADAPTER**

Problemática: ¿Cómo tener una única interfaz de acceso a múltiples bases de datos?

Propósito: Convertir la interfaz de una clase para que se adapte a lo que el cliente que la usa necesita, permitiendo así que trabajen juntas clases cuyas interfaces son incompatibles.

Solución: Symfony da la posibilidad de cambiar a otro sistema de base de datos completamente diferente a mitad de desarrollo, sólo basta con configurar un archivo YML. La capa de abstracción utilizada

encapsula toda la lógica de los datos. El resto de la aplicación no tiene que preocuparse por las consultas SQL y el código SQL que se encarga del acceso a la base de datos es fácil de encontrar. [14] [19]

Ejemplo:

Los archivos YML se encuentran en la carpeta config del módulo de la aplicación y se denominan propel.ini y databases.yml. Un segmento del propel.ini que brinda la conexión a la base de datos sería el siguiente:

```
propel.database.createUrl = oracle://usuario:contraseña@BaseDatos
propel.database.url      = oracle://usuario:contraseña@BaseDatos
```

Un ejemplo de databases.yml sería:

```
all:
 propel:
 class:        sfPropelDatabase
param:
dsn:        oracle:// usuario:contraseña@BaseDatos
```

Estos son todos los ficheros necesarios para modificar a la hora de cambiar de una base de datos a otra.

- ✓ **Patrones GRASP implementados**
  - ➢ **Patrón BAJO ACOPLAMIENTO**

Problemática: ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?

Propósito: Aumentar la reutilización y eliminar las dependencias entre las clases para propiciar un fácil mantenimiento y entendimiento. 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. Acoplamiento bajo significa que una clase no depende de muchas clases. Acoplamiento alto significa que una clase recurre a muchas otras clases. Si no se usara este patrón los cambios en las clases afines ocasionarían cambios locales, generarían tantas dependencias que serían difíciles de reutilizar, además de que serían difíciles de entender por sí mismas al estar aisladas de sus dependencias. [20]

Solución: Symfony asigna a cada clase una responsabilidad para mantener pocas dependencias entre las mismas.

Ejemplo:
La clase Action hereda solamente de sfActions para lograr un bajo acoplamiento de clases. En caso de la tarea automática del procesar hay un bajo acoplamiento de clases ya que a la hora de guardar los mensajes y los datos de las personas y vuelos que vienen en ellos se le asigna la responsabilidad a MensaDAO y no a la controladora procesarAction.class reduciéndose la dependencia de esta última con el resto de las clases.

- **Patrón ALTA COHESION**

Problemática: ¿Cómo mantener la complejidad dentro de límites manejables?

Propósito: Cada elemento dentro del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. La cohesión 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. Una baja cohesión hace muchas cosas no afines y realiza trabajo excesivo por lo que si no se utilizara este patrón las clases fueran difíciles de comprender, difíciles de reutilizar, de conservar y las afectarían constantemente los cambios. [20]

Solución: Symfony agrupa las clases por funcionalidades que son fácilmente reutilizables, bien por su uso directo o por herencia. Este patrón permitirá tener clases fáciles de mantener, de entender y reutilizar.

Ejemplo:
Symfony permite asignar responsabilidades con una alta cohesión, por ejemplo la clase Actions tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instalar objetos y acceder a properties (propiedades), es decir, está formada por diferentes funcionalidades que se encuentran relacionadas proporcionando que el software sea flexible frente a grandes cambios.

- **Patrón CREADOR**

Problemática: ¿Quién debería ser responsable de crear una nueva instancia de alguna clase?

Propósito: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos.
El propósito fundamental de este patrón es encontrar un creador que se conecte con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

Solución: Asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos: [20]

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A o
- B utiliza especialmente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A). B es un creador de los objetos A.
- Si existe más de una opción, prefiera la clase B que agregue o contenga la clase A.

Ejemplo:
En la solución a la hora de procesar los mensajes se le dio la responsabilidad de crear los objetos que serían guardados en la base de datos a la clase MensajeDAO ya que esta era la que tenía la información referente a los mismos, por lo tanto ninguna mejor que ella para crearlos y guardarlos.
Por otra parte, la creación del objeto MensajeDAO se le asignó a la clase procesarAction.class que es una controladora que se ejecuta automáticamente cada cierto tiempo para procesar los ficheros que se encuentran validados.

- **Patrón CONTROLADOR**

Problemática: ¿Quién debería encargarse de un evento del sistema?

Propósito: Facilitar la centralización de actividades, delegar las actividades en otras clases con las que mantiene un modelo de alta cohesión.

Solución: Symfony implementa para cada vista de la aplicación una clase controladora que se encarga de atender el evento generado por la vista. Además todas las peticiones Web son manejadas por un solo controlador frontal (sfActions), 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. [14] [20] [18]

Ejemplo:
En la solución se utilizaron varias clases controladoras como son: sitaActions.clas.php, procesarActions.class.php, internetActions.class.php, todas ellas heredan de sfActions que es el controlador frontal de Symfony. Estas clases se van a encargar de determinadas funcionalidades según los objetivos para los cuales fueron creadas, por ejemplo procesarActions.class.php se encarga de procesar todos los mensajes API, pero todas estas clases tienen cosas en común como son atender los eventos generados y a los cuales tienen que darle solución. Symfony a través de su controlador frontal "sabe" a que clase darle determinada acción controlando así todo el tráfico de información de la aplicación, y este a su vez transmite esta acción a las páginas controladoras que se diseñaron y que son las responsables de atender y controlar la petición.

3.6 Tecnologías empleadas y aspectos novedosos utilizados

Para cumplir con algunos de los requisitos capturados resultó necesario utilizar algunas tecnologías y elementos poco frecuentes en los sistemas de gestión que tornan más eficiente el proceso a automatizar. A continuación se detallarán cada uno de estos elementos.

- ✓ **Crones**

El cron es un administrador regular de procesos en segundo plano, que ejecuta programas a intervalos regulares (por ejemplo, cada minuto, día, semana o mes) en un sistema operativo. En la solución se crearán crones para la ejecución automática de los procesos, tarea imposible de realizar manualmente por

su excesiva frecuencia. Estos se implementarán en PHP y brindarán un servicio vital para el sistema que permitirá que el mismo esté completamente actualizado en cuanto a la información de los vuelos, los reportes y en sentido general en todas sus funcionalidades (Ver Anexo #2).

- ✓ **Expresiones Regulares**

Englobar la estructura definida internacionalmente por la IATA, la OMA y la OACI de los ficheros API a través de expresiones regulares, permitirá elevar la calidad y la eficiencia del proceso de validación de los ficheros recibidos. Con el análisis realizado de los ficheros se llegó a la conclusión de la necesidad de utilizar dos expresiones regulares, una para el inicio del fichero y otra para el final del mismo, ya que por su extensión no se pueden unificar en una única expresión. Al usar estas expresiones el código se tornará mucho más legible para las revisiones a realizar, facilitando también su mantenimiento, así como se agilizará el trabajo con el fichero, requisito muy importante para que los clientes puedan contar con la información contenida en el menor tiempo posible. (Ver Anexo #3).

- ✓ **Búsqueda Fonética**

Este elemento le proporcionará a la solución propuesta una gran usabilidad a las búsquedas realizadas, en especial a lo que se refiere a las personas, permitiendo obtener un rango mayor de personas que una búsqueda normal, donde tiene que coincidir el dato de la persona que se está buscando con el que se encuentra almacenado. Con esta búsqueda fonética se mostrarán todas las personas que el sonido del dato especificado se parezca al que se está buscando, por ejemplo si se buscara una persona cuyo nombre es Adiel, aparecerían además de Adiel, Adalia, Adel, Adela, Adele, Adelia, Adelio, Adil, Adla, etc. ya que fonéticamente esas cadenas se parecen. Este proceso mejora indudablemente las búsquedas a realizar ya que en muchas ocasiones sólo se cuenta con un dato (nombre, apellido) que la persona cree que se dice de determinada forma y no tiene una certeza de cómo se escribe (Ver Anexo #4).

3.7 Flujo de trabajo del rol

Los roles en los que se desempeñaron los autores de este Trabajo de Diploma para darle solución a la problemática planteada fueron los roles de analista y diseñador. A continuación se mostrarán las responsabilidades, competencias y artefactos generados por cada uno de estos roles.

Analista: Este rol dirige y coordina la adquisición de requisitos esquematizando la funcionalidad del sistema y delimitándolo. [21]

Artefactos generados:

- ✓ Especificación de Requisitos.
- ✓ Modelo de Proceso.
- ✓ Modelo de Caso de Uso del Sistema.

Responsabilidades:

- ✓ Participar en la definición del proyecto.
- ✓ Modelar el negocio.
- ✓ Definir los requisitos de la aplicación.
- ✓ Realizar los prototipos de interfaz de usuario.
- ✓ Modelar el sistema.

Competencias:

- ✓ Conocimientos de RUP, UML, BPMN.
- ✓ Captura de requisitos y análisis de sistema.
- ✓ Trabajo con la herramienta de modelado Visual Paradigm.

Diseñador: Este rol dirige el diseño de un sistema, dentro de las restricciones de los requisitos, arquitectura y proceso de desarrollo para el proyecto. El diseñador identifica y define las responsabilidades, operaciones, atributos y relaciones de los elementos de diseño. Además de asegurarse de que el diseño sea coherente con la arquitectura del software y que esté detallado hasta un punto en que pueda proceder la implementación. El análisis describe el problema (el ¿qué hacer?) mientras que el diseño describe la solución (el ¿cómo hacerlo?). [21]

Responsabilidades:

- ✓ Definir los elementos del diseño.
- ✓ Asegurar la coherencia del diseño con la arquitectura.
- ✓ Integrar los componentes de la solución.
- ✓ Dirigir el trabajo de los programadores.

Artefactos:

- ✓ Modelo de Diseño del Sistema.

Competencias:

- ✓ Trabajo con la herramienta de modelado Visual Paradigm.
- ✓ Conocimientos de RUP, UML.
- ✓ Conocimientos de Symfony.
- ✓ Conocimientos de EXTjs 2.0.

CAPÍTULO 4: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

En este capítulo se detallará el diseño del sistema, para esto se explicarán los diagramas de clases del diseño con estereotipos Web realizados y se mostrará el modelo físico de la base de datos propuesto. Además se presentará la descripción de la solución propuesta como última tarea realizada de este trabajo de diploma.

4.1 Diseño

En el diseño se modela el sistema para que cumpla con todos los requisitos incluyendo los no funcionales y con otras restricciones que le suponen bajo la arquitectura aprobada. Con el diseño se cumplen con una serie de propósitos como son: ser capaces de dibujar y reflexionar sobre el diseño utilizando una notación común y ser capaz de dividir el proyecto en partes manejables que puedan ser atendidas por varios equipos de desarrollo; teniendo en cuenta siempre la unión de esas partes y su posible concurrencia. Otro propósito que se cumple es el de crear una entrada apropiada y un punto de partida para las actividades de implementación subsiguientes, capturando los requisitos o subsistemas individuales, interfaces y clases; adquiriendo una comprensión en profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia, tecnologías de interfaz de usuario y tecnologías de gestión de transacciones, entre otras. [23]

4.2 Diagrama de clases del diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso, centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. El modelo de diseño sirve de abstracción a la implementación y a su código fuente, es de ese modo, utilizado como una entrada fundamental a las actividades de implementación. El mismo puede contener: diagramas, clases, interfaces, relaciones, colaboraciones, atributos, realizaciones de los casos de uso, entre otros elementos que se puedan considerar necesarios para el correcto entendimiento del sistema a implementar. [23]

El diagrama de clases para las Aplicaciones Web, difiere un poco del resto de las aplicaciones que se acostumbra a construir, como las aplicaciones de escritorio, puesto que están enfocadas a distintos fines. En el caso de las aplicaciones Web es más importante la modelación de la lógica y estado del negocio que los detalles de presentación, para darle solución al diseño se utilizarán los diagramas de clase con estereotipos Web. [27]

Para obtener un nivel correcto de abstracción y detalle que permita obtener un resultado final de elevada calidad se recomienda modelar todos los artefactos del sistema, es decir, modelar las páginas, los enlaces entre estas, así como el contenido dinámico de las mismas una vez que estén en el navegador del cliente.[27]

Para diseñar la solución propuesta se contó con la ayuda del arquitecto de la base de datos y el diseñador principal del Polo Productivo, con el propósito de lograr una correcta integración con los sistemas ya implementados en el SUA, haciendo coincidir los datos que se iban a manejar en este nuevo sistema con los datos ya existentes, garantizar la utilización de las librerías definidas y encontrar los puntos comunes del negocio a implementar.

Por ejemplo, el sistema de Recursos Humanos que se encuentra implementado en el SUA se encargó de definir cómo iban a ser tratadas todas las Personas que tuvieran contacto con la AGR. Al analizar la definición realizada se pudo observar que las Personas a gestionar enviadas en un mensaje API coincidían, por lo que no se tuvo que realizar un nuevo diseño de la Persona para este sistema, sino que se pudo utilizar el ya existente. Con esta reutilización se puede lograr un control mucho más eficiente de las personas sin duplicar sus informaciones, por ejemplo, en el caso de que una persona que trabaje en la AGR realice un viaje al exterior, el SUA contaría al instante con toda su información personal lo que agilizaría su tratamiento y permitiría realizar un mejor control.

A continuación se muestran los diagramas de clases realizados para cada caso de uso del sistema definido anteriormente.

4.2.1. Caso de uso Subir Mensaje

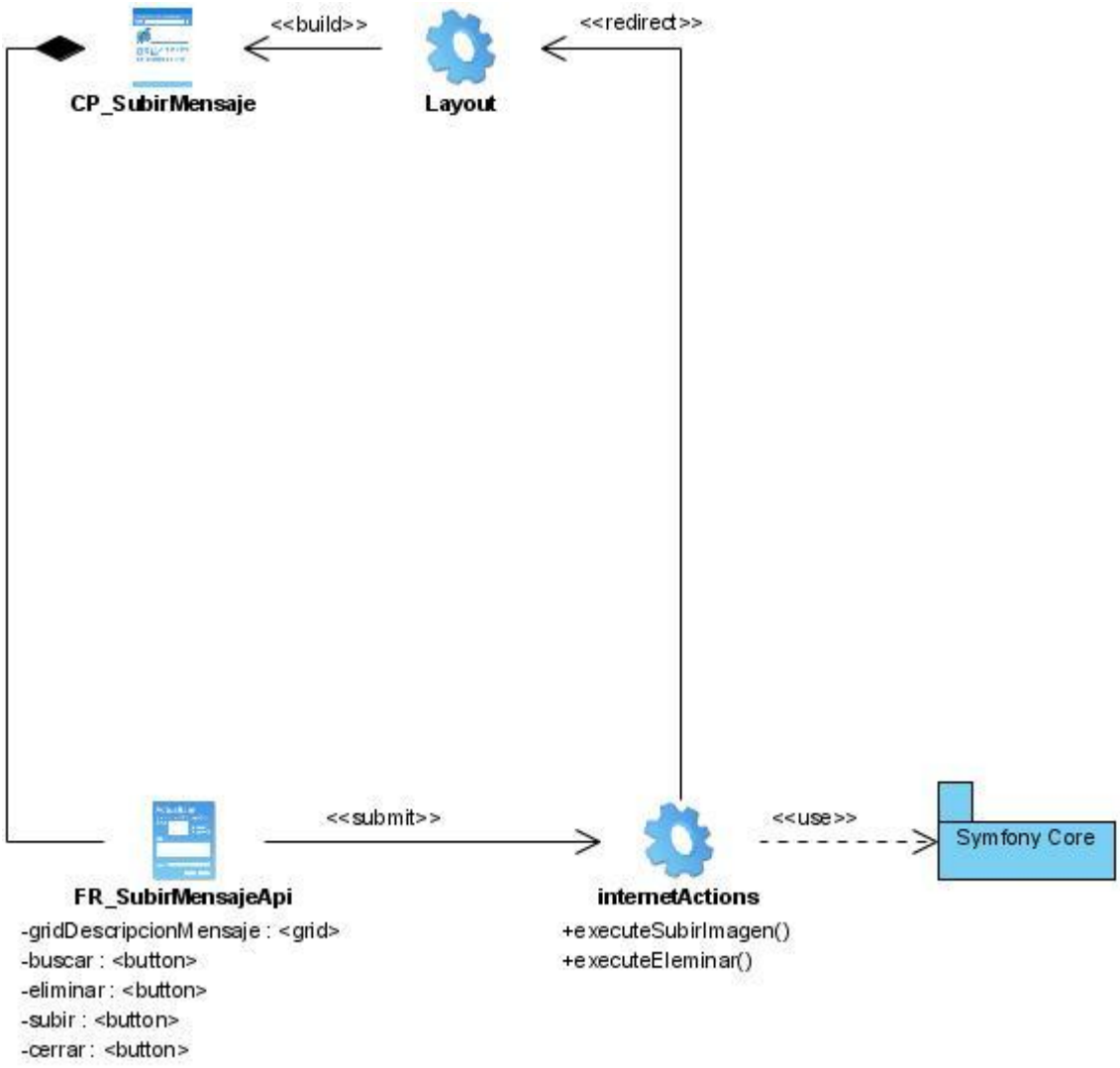

Figura #9 Diagrama de clases del diseño CU Subir Mensaje.

Para mayor descripción ver el documento Subir Mensaje Modelo de Diseño v1.0 que se encuentra en las memorias del proyecto.

4.2.2. Caso de uso Conformar Mensaje

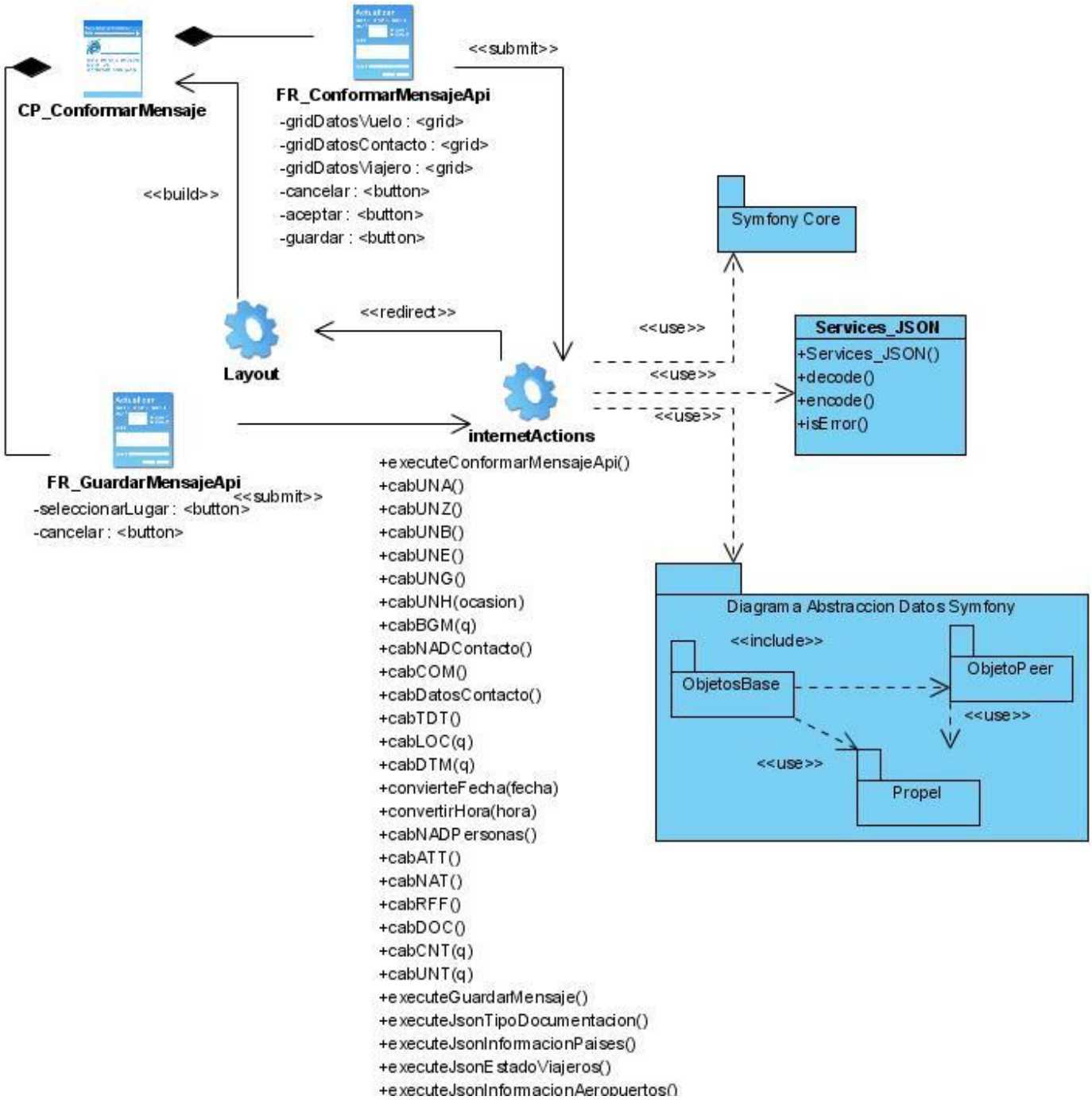

Figura #10 Diagrama de clases del diseño CU Conformar Mensaje.

Para mayor descripción ver el documento Conformar Mensaje Modelo de Diseño v1.0 que se encuentra en las memorias del proyecto.

4.2.3. Caso de uso Obtener Correo

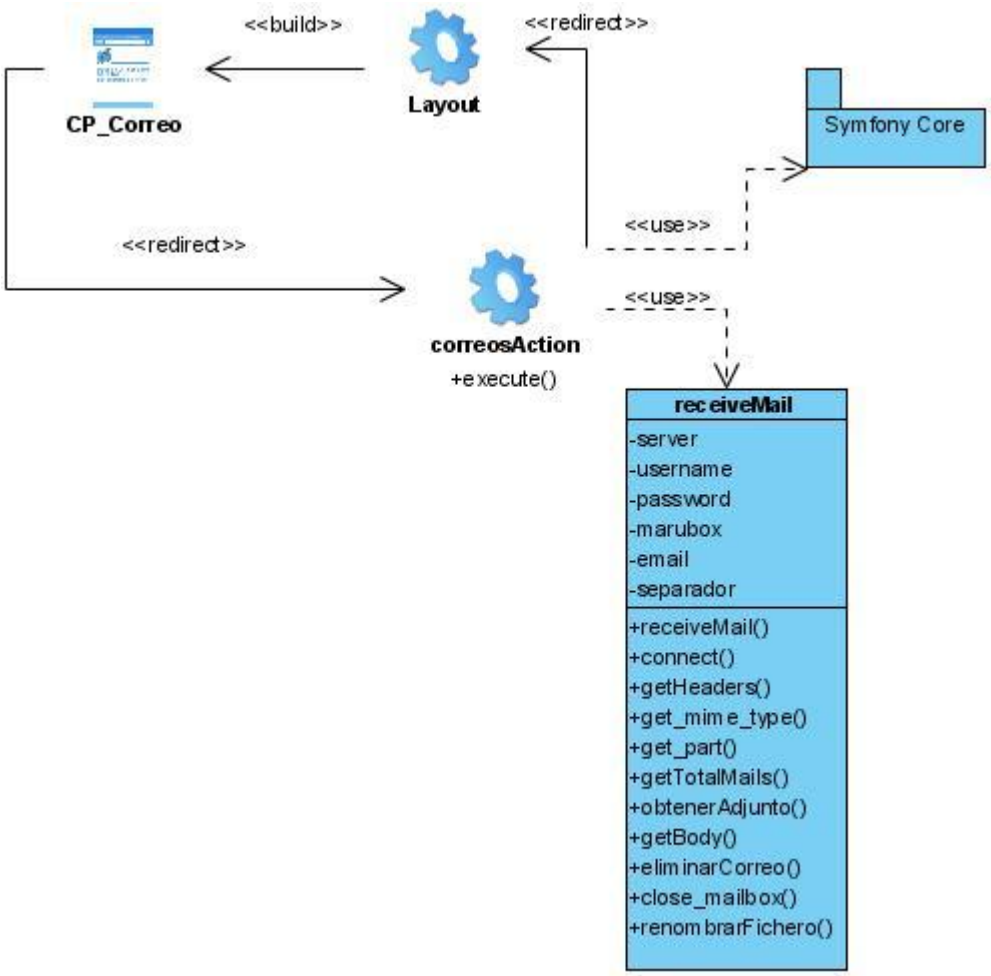

Figura #11 Diagrama de clases del diseño CU Obtener Correo.

Para mayor descripción ver el documento Obtener Correo Modelo de Diseño v1.0 que se encuentra en las memorias del proyecto.

4.2.4. Caso de uso Obtener Sita

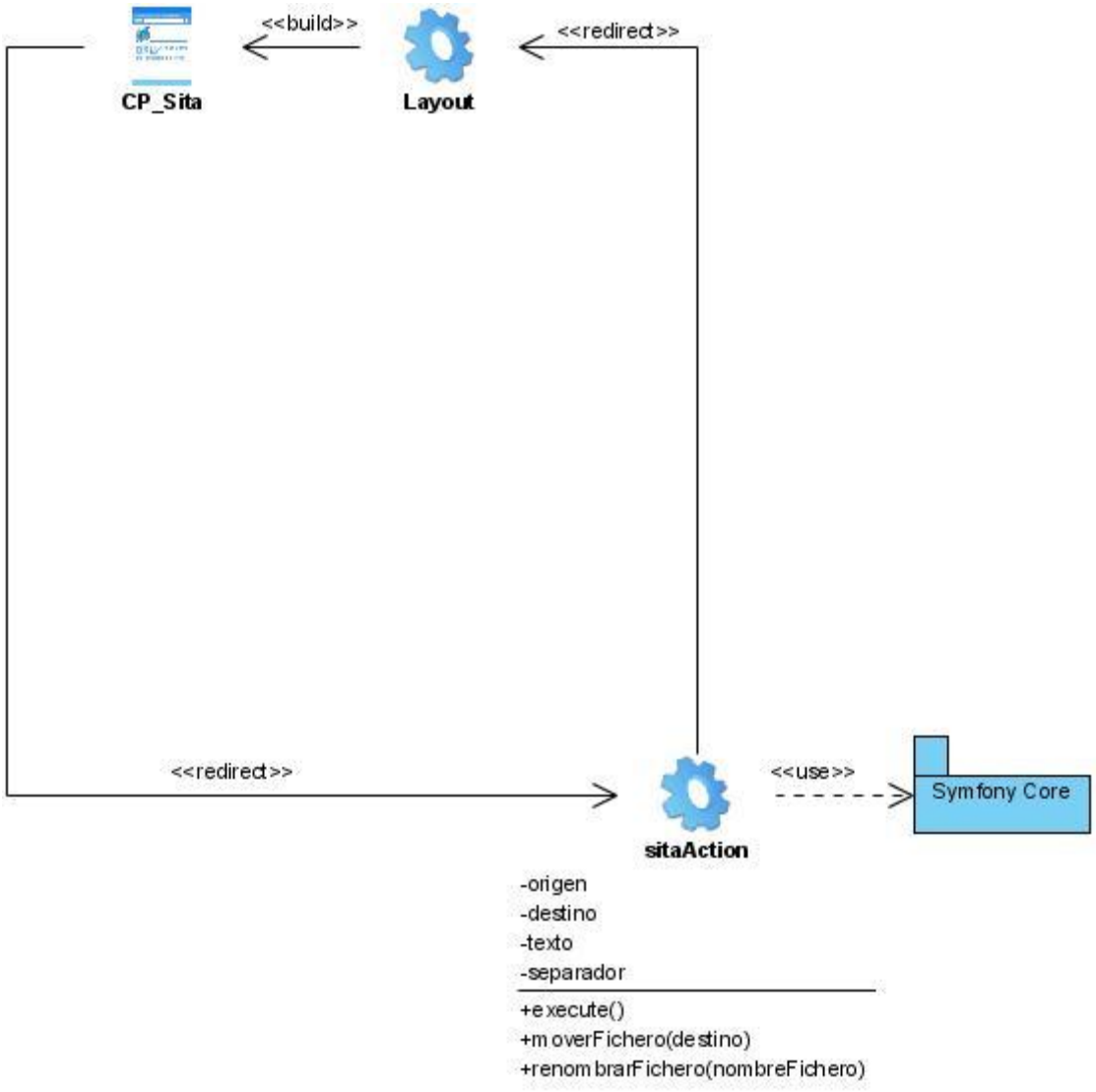

Figura #12 Diagrama de clases del diseño CU Obtener Sita.

Para mayor descripción ver el documento Obtener Sita Plantilla DCS Modelo de Diseño v1.0 que se encuentra en las memorias del proyecto.

4.2.5. Caso de uso Validar Mensaje

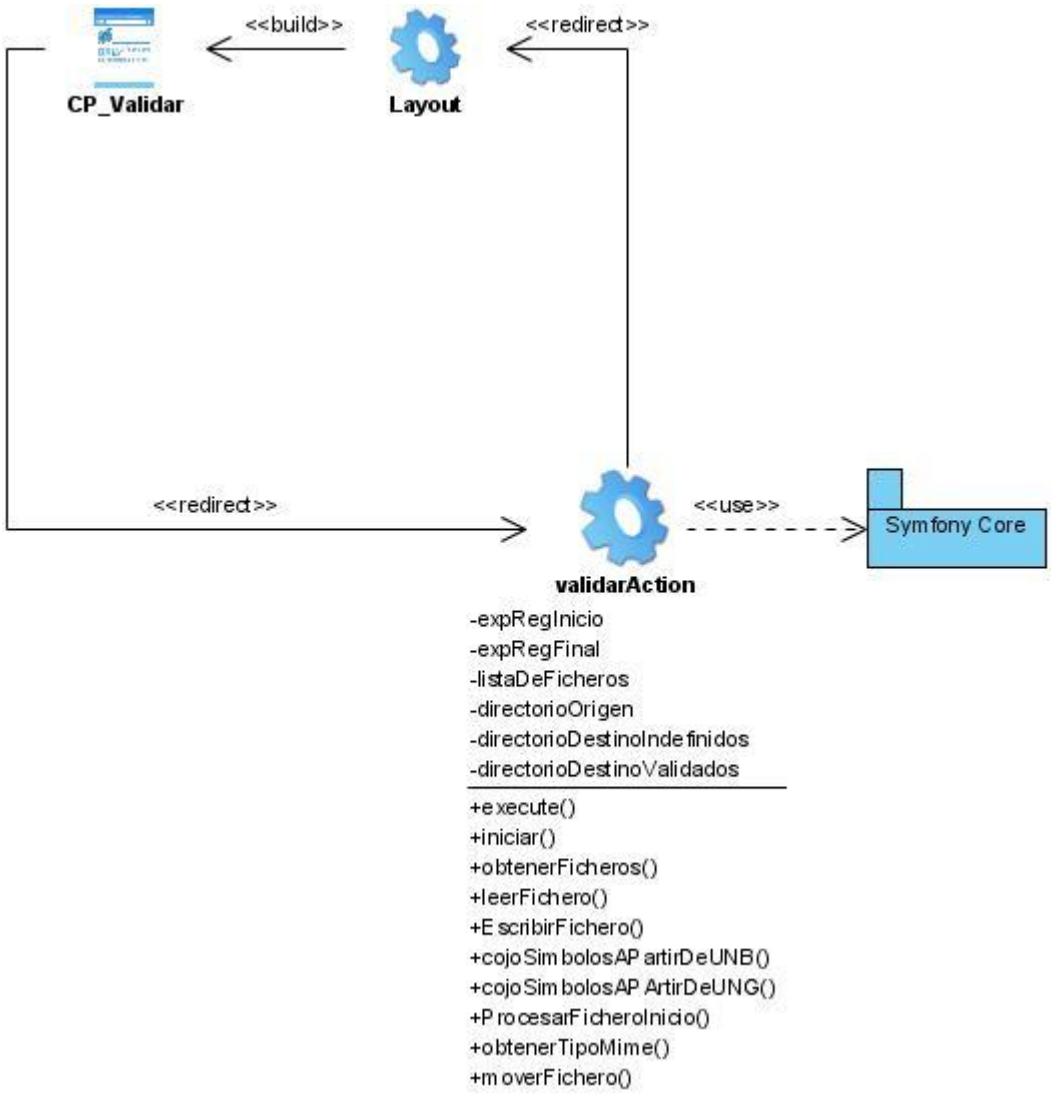

Figura #13 Diagrama de clases del diseño CU Validar Mensaje.

Para mayor descripción ver el documento Validar Mensaje - Modelo de Diseño v1.0 que se encuentra en las memorias del proyecto.

4.2.6. Caso de uso Procesar Mensajes

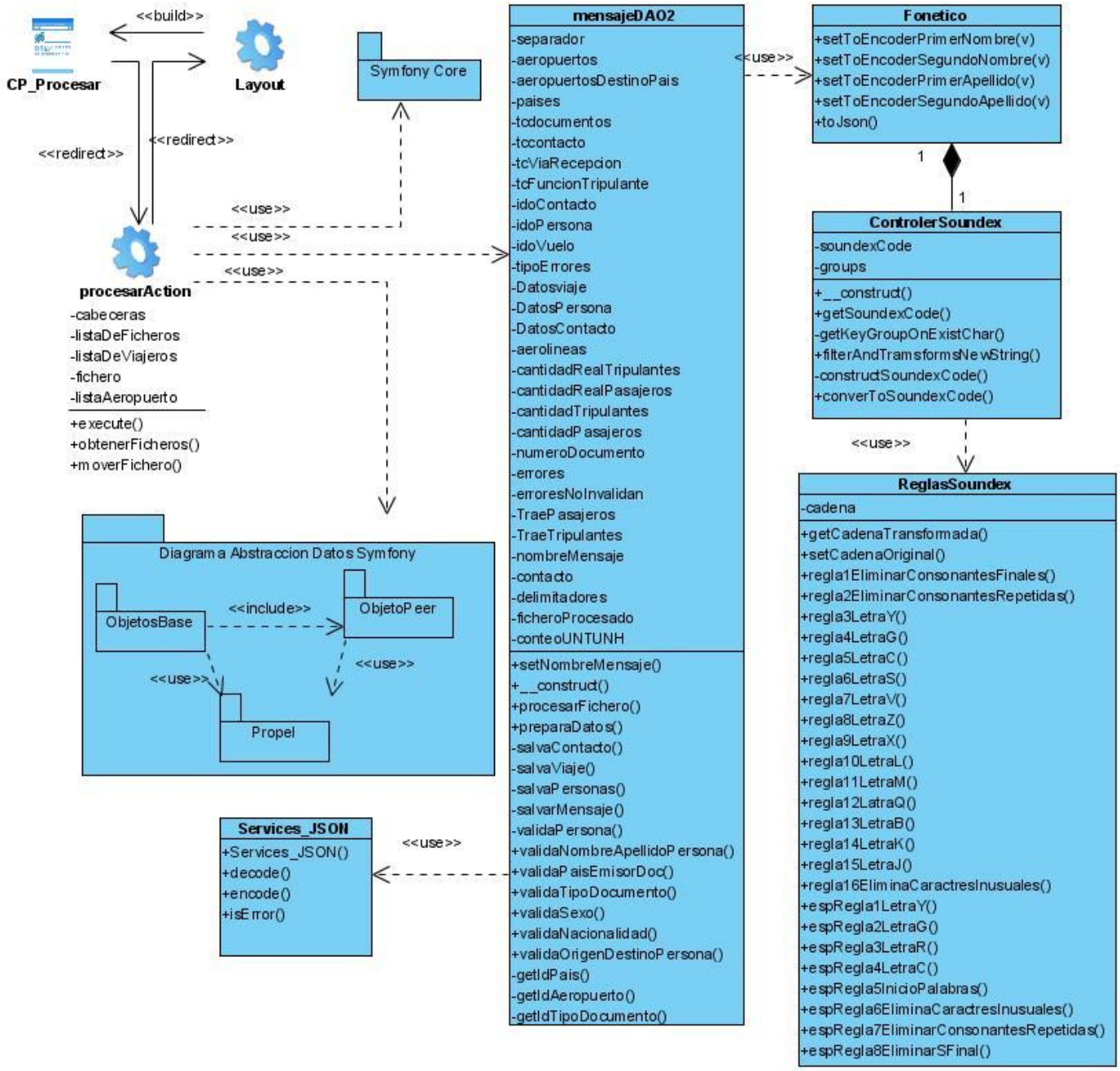

Figura #14 Diagrama de clases del diseño CU Procesar Mensajes.

Para mayor descripción ver el documento Procesar Mensaje Modelo de Diseño v1.0 que se encuentra en las memorias del proyecto.

4.2.7. Caso de uso Analizar Vuelos

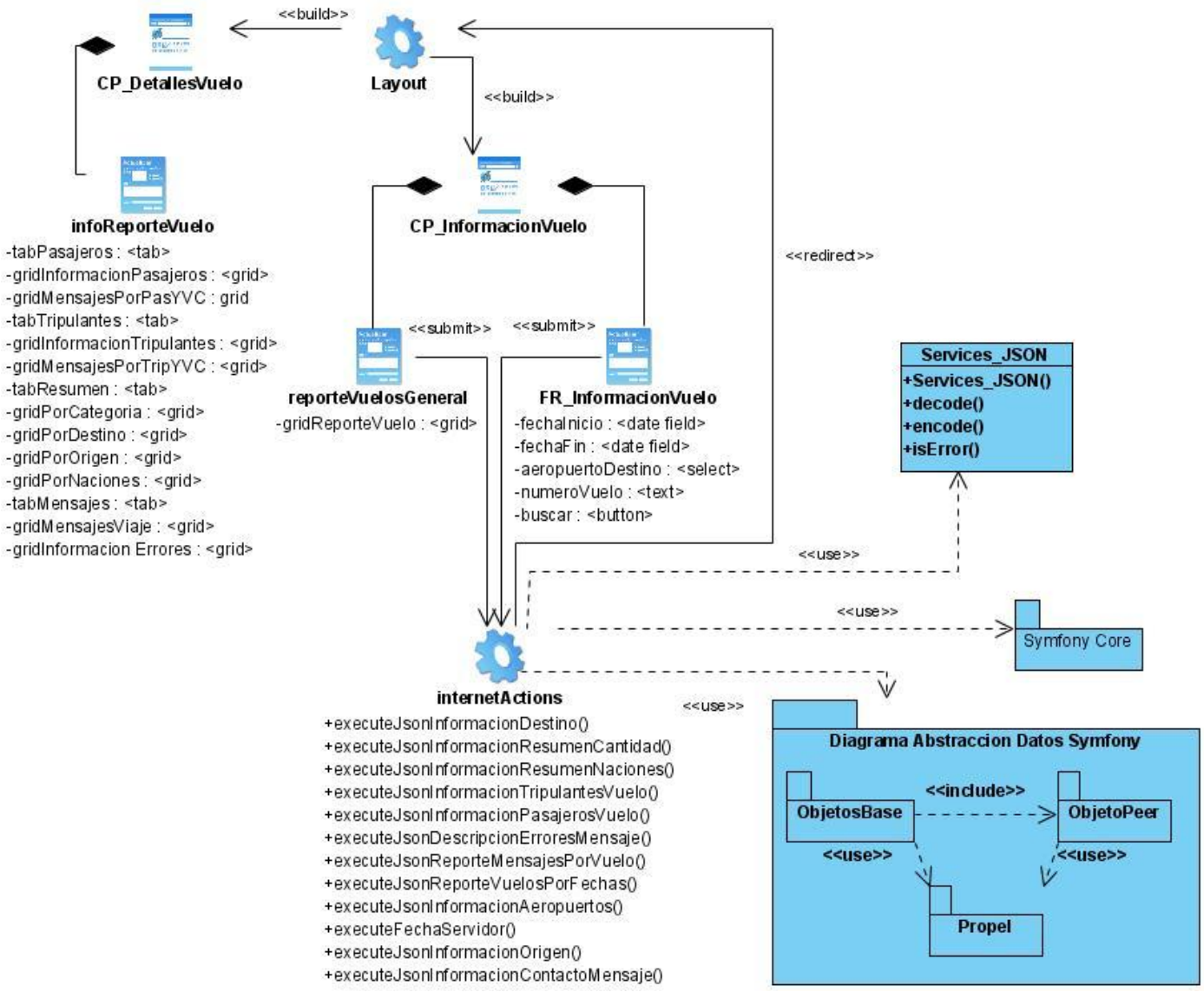

Figura #15 Diagrama de clases del diseño CU Analizar Vuelos.

Para mayor descripción ver el documento Analizar Vuelos Modelo de Diseño v1.0 que se encuentra en las memorias del proyecto.

4.2.8. Caso de uso Consultar Estado Mensaje

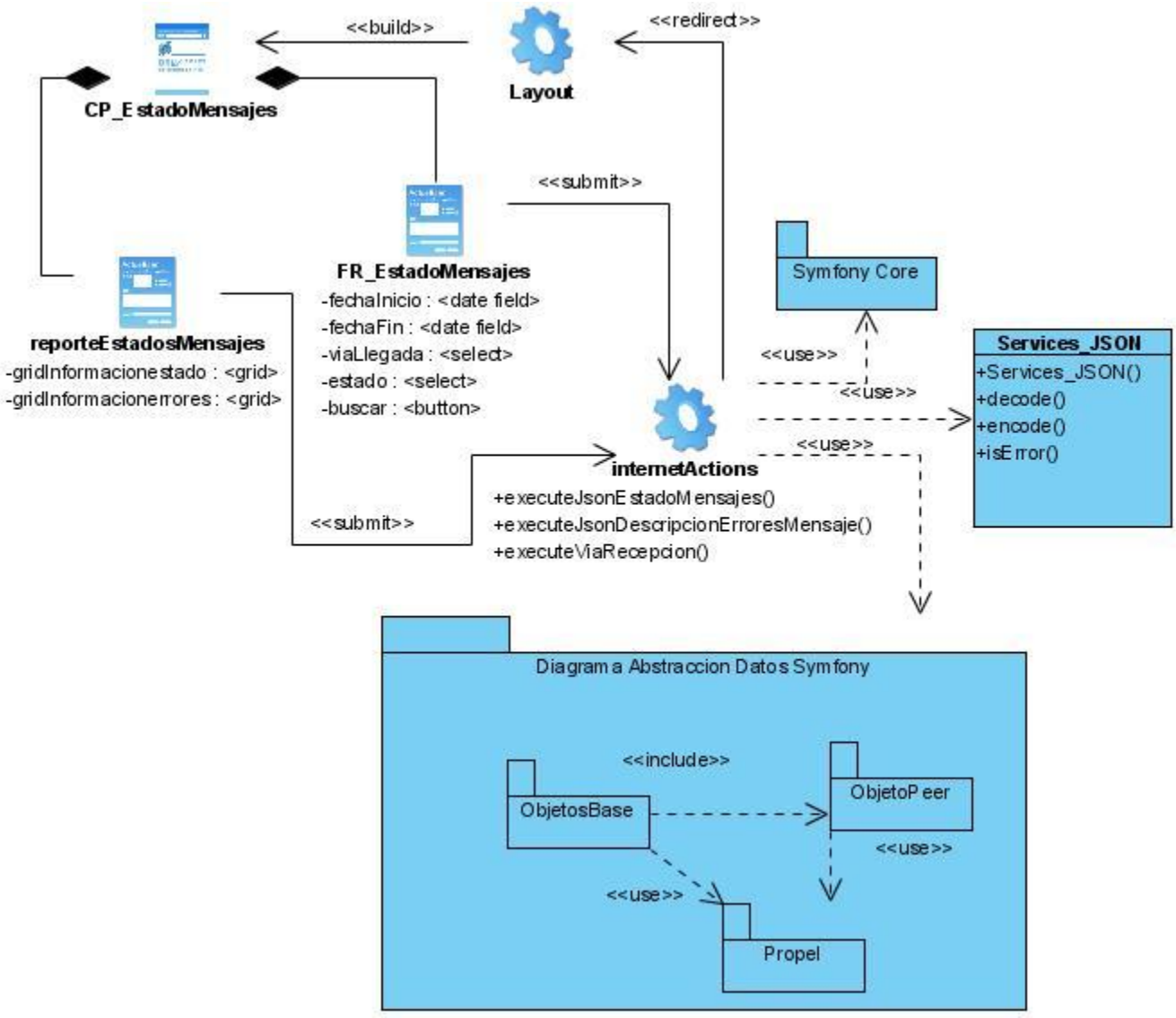

Figura #16 Diagrama de clases del diseño CU Consultar Estado Mensaje.

Para mayor descripción ver el documento Consultar Estado Mensajes Modelo de Diseño v1.0 que se encuentra en las memorias del proyecto.

4.2.9. Caso de uso Ver Mensajes Ilegibles

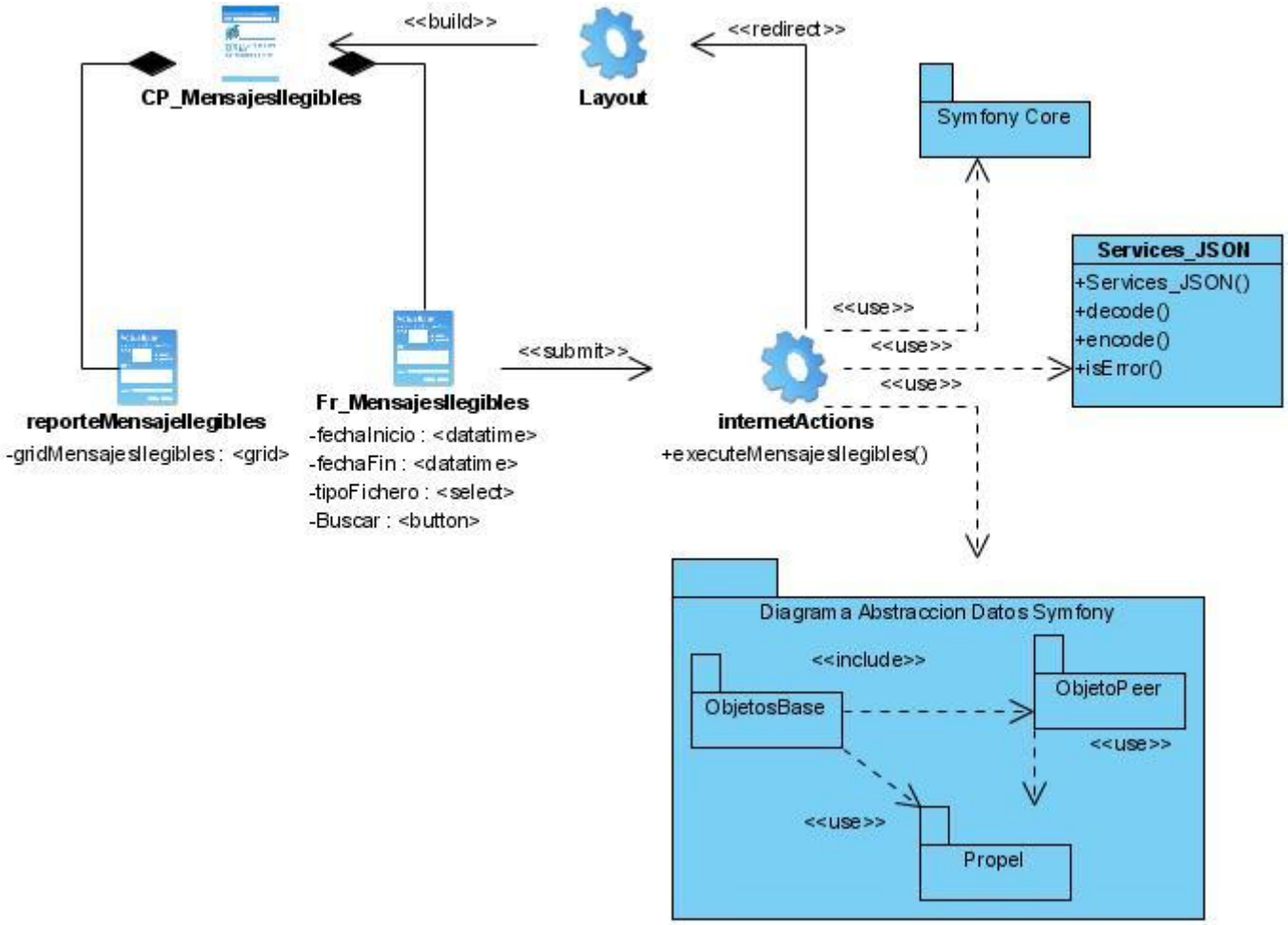

Figura #17 Diagrama de clases del diseño CU Ver Mensajes Ilegibles.

Para mayor descripción ver el documento Ver Mensajes Ilegibles Modelo de Diseño v1.0 que se encuentra en las memorias del proyecto.

4.2.10. Caso de uso Conformar Parte

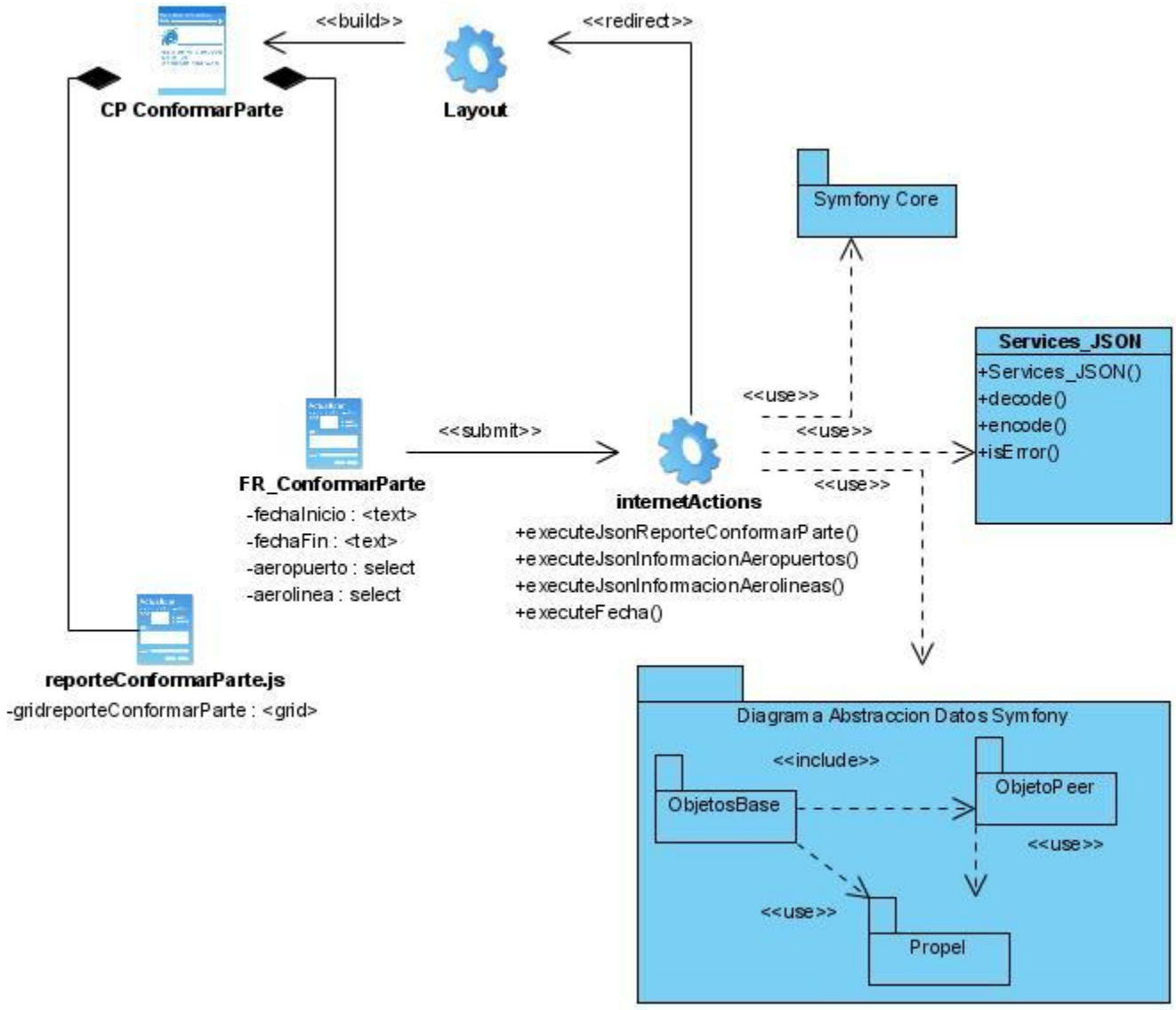

Figura #18 Diagrama de clases del diseño CU Conformar Parte.

Para mayor descripción ver el documento Conformar Parte Modelo de Diseño v1.0 que se encuentra en las memorias del proyecto.

4.2.11. Caso de uso Buscar Fonéticamente

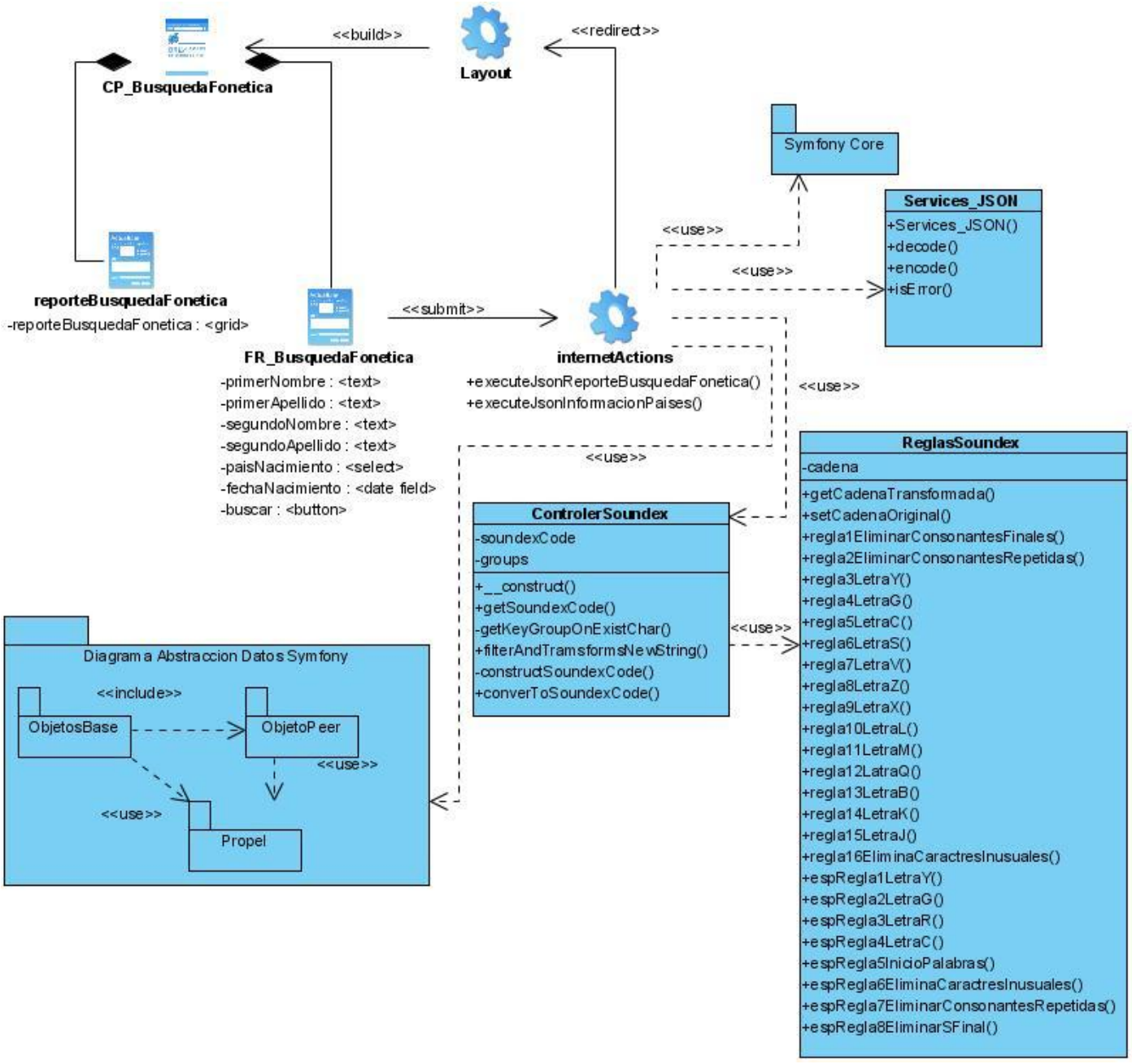

Figura #19 Diagrama de clases del diseño CU Buscar Fonéticamente.

Para mayor descripción ver el documento Búsqueda Fonética Modelo de Diseño v1.0 que se encuentra en las memorias del proyecto.

4.2.12. Caso de uso Eliminar Mensajes Ilegibles

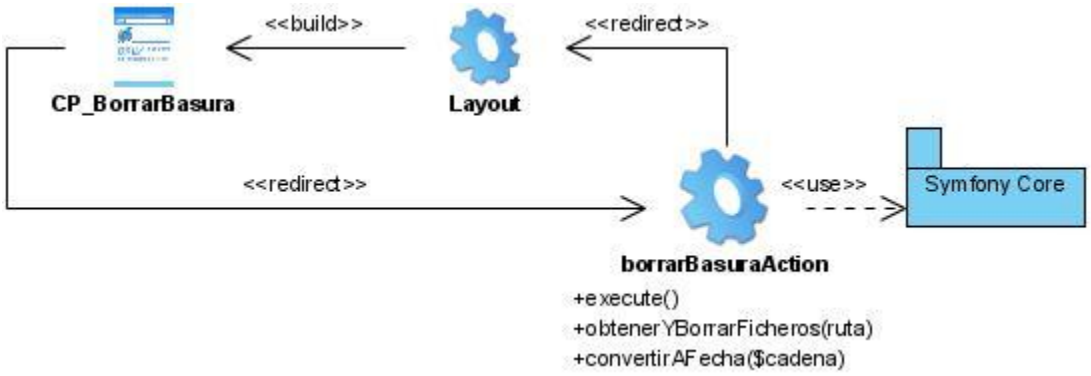

Figura #20 Diagrama de clases del diseño CU Eliminar Mensajes Ilegibles.

Para mayor descripción ver el documento Borrar Mensajes Ilegibles Modelo de Diseño v1.0.doc que se encuentra en las memorias del proyecto.

Se debe agregar también que al utilizar Symfony como Framework de desarrollo, este permite una abstracción de la base de datos, pero para realizar esto, necesita contar con una relación de clases que le permitan realizar este fin. Para esto Symfony genera 5 clases por cada clase del modelo Físico de Datos o sea, de la base de datos.

Para entender mejor, se pondrá un ejemplo con la clase Persona, estas serían las clases generadas:

- ✓ Persona: Esta clase se utiliza para crear algún método que se necesite extra a la hora de guardar una persona nueva. En la solución en esta clase no se necesita crear ninguno nuevo, pero en el caso de la clase Fonético si se necesitaría crear algunos ya que se necesitaría codificar los datos entrados por el mensaje API para que la Persona tenga un fonético asociado al que se le pueda realizar la búsqueda. Esta clase hereda de BasePersona.

- ✓ PersonaPeer: Esta clase será la usada para los reportes en sentido general, por ejemplo para la búsqueda fonética se agrega un método que se denomina jsonBusquedaFonetica, ya que a la hora de realizar la búsqueda lo que se haría es buscar todas las personas que cumplen con los datos pedidos. Esta Clase hereda de BasePersonaPeer.

- ✓ BasePersona: Esta clase se utiliza para modificar algún dato de una persona, tiene implementados los Get () y Set () de todos los atributos de la Persona. Por lo tanto su clase hija (Persona), los tiene y se podría poner en cualquier parte del código de otra clase:
  $a =new Persona ();
  $a->setPrimerNombre ('Randy');
  $a->Save ();

En el ejemplo se puede crear una persona, cambiarle un atributo y guardarla en la base de datos sin necesidad de sentencias SQL. Esta clase implementa una serie de métodos como son Save (): salva los datos que tenga en el objeto, delete (): borra la persona, entre otros. Es recomendable no cambiar ni agregar ningún método o atributo a esta clase ya que si algún día se cambia la base de datos esta clase cambiaría a esa nueva configuración. Esta clase hereda de una clase del núcleo de Symfony.

- ✓ PersonaMapBuilder: Se utiliza para la conexión a la base de datos, es usada por el Framework para su trabajo. Se recomienda no cambiar nada de esta clase.

- ✓ BasePersonaPeer: Es otra clase utilizada por el Framework para el acceso a los datos de la base de datos. Se recomienda no cambiar nada de esta clase.

Para una mejor comprensión de las relaciones existentes dentro del diagrama generado por Symfony se deben ver las Figuras #21 y #22 que se muestran a continuación. En estos diagramas se muestran las relaciones de 4 de estas 5 clases ya que las referentes a PersonaMapBuilder no se incluyen para ganar en legibilidad ya que esa clase es puramente del Framework e invisible a la lógica del diseño. En la Figura #21 se muestra una vista general de las clases Base del sistema. Esta relación representa como queda la unión de las clases con su multiplicidad y representa una aproximación generalizada de las clases principales. En la figura #22 se muestran las clases Peer que representan las clases encargadas de las consultas de los reportes.

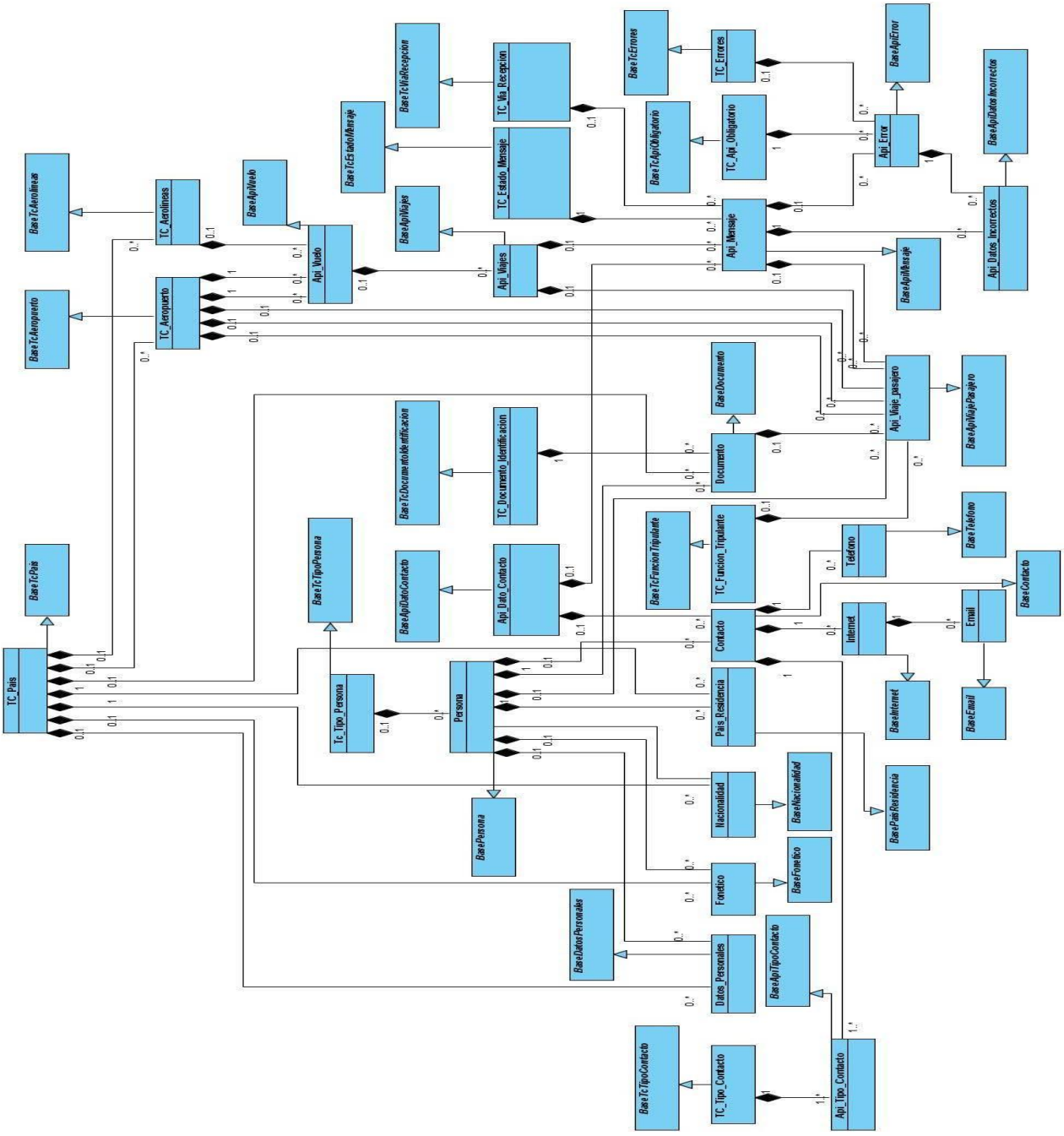

Figura #21 Diagrama de clases de Objetos Base de la solución propuesta

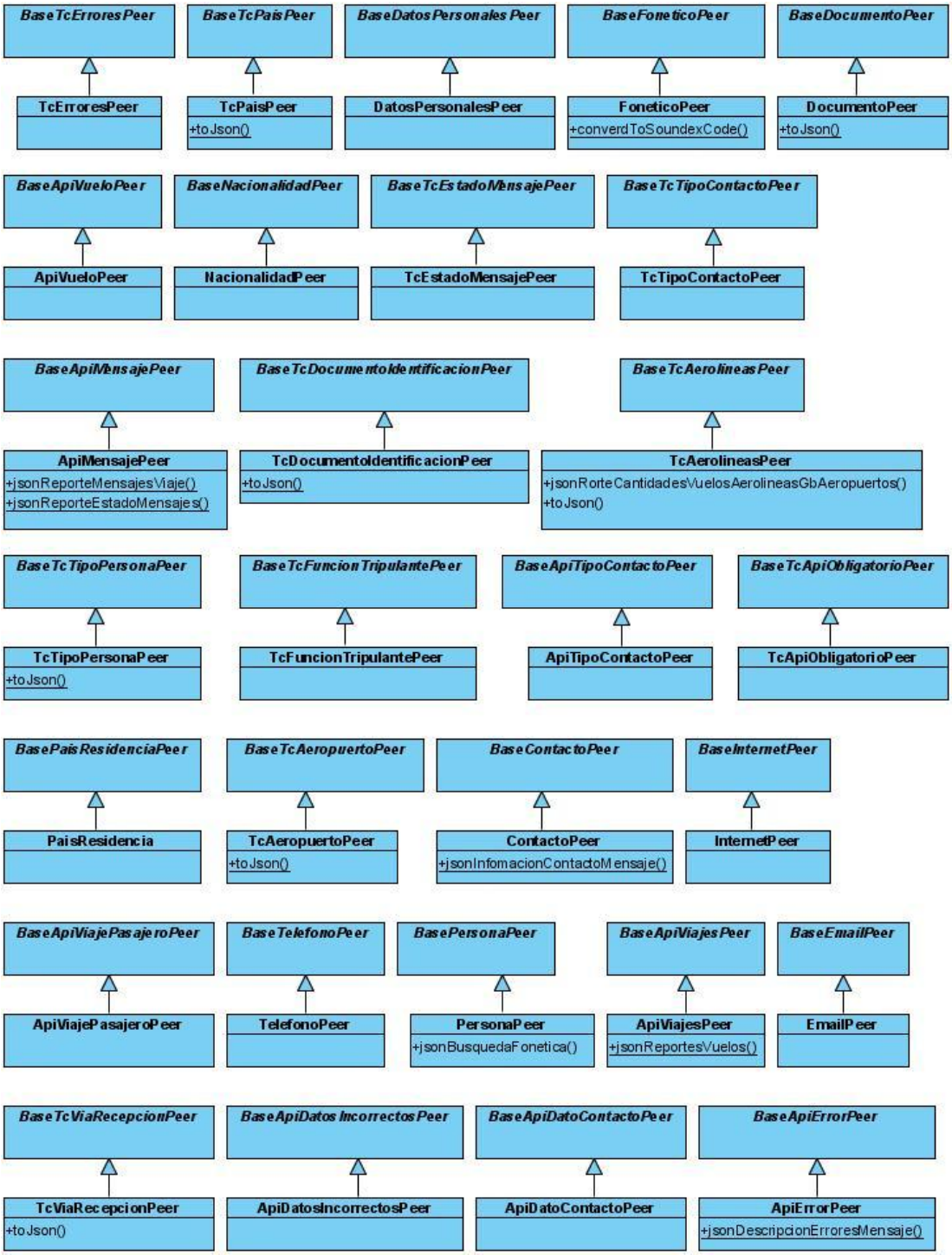

Figura #22 Clases de Objetos Peer.

A partir de este momento se debe saber que para realizar consultas a la base de datos el Framework utiliza las clases PersonaPeer, DocumentoPeer, TCPaisPeer, etc., mientras que para guardar datos nuevos utiliza las clases Persona, Documento, País etc., como se puede apreciar en la explicación anterior.

A continuación se muestra el paquete “Diagrama Abstracción Datos Symfony” que engloba la relación de las clases persistentes con el paquete Propel utilizado por Symfony para la persistencia de los datos.

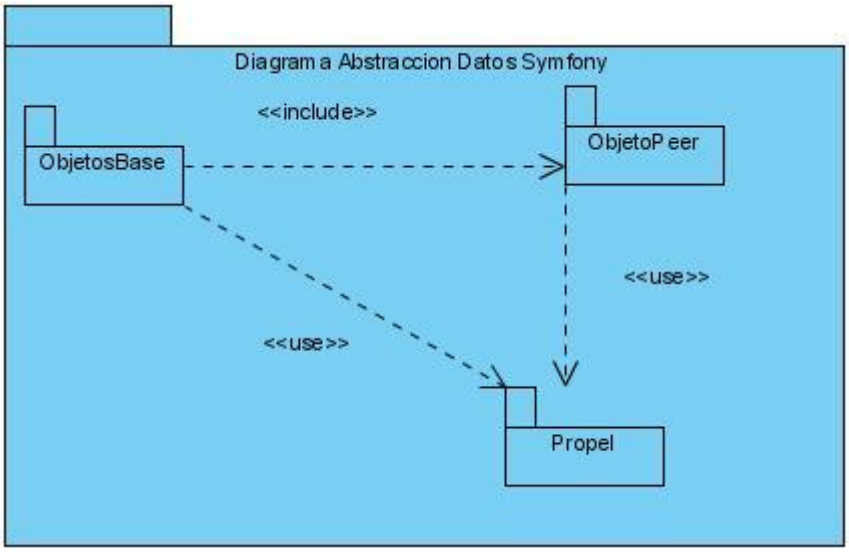

Figura #23 Paquete de abstracción de datos

- ✓ **Modelo Físico de datos**

Este modelo describe los datos en el nivel más bajo de abstracción y permiten identificar algunos detalles de implantación para el manejo del hardware de almacenamiento, es el resultado final cuando se quiere diseñar una base de datos. A partir de este modelo se crea la base de datos propiamente dicha. [26]

Para el diseño de la base de datos del API se utilizó como paso final el modelo físico de datos, a través del cual se diseñaron las clases persistentes con sus relaciones y se modeló la distribución de las tablas. El diseño propuesto satisface las necesidades de persistencia de los datos que el módulo requiere en cumplimiento de sus requerimientos funcionales y permite la integración con el resto del sistema.

A continuación se muestra el modelo físico de los datos (Figura #24)

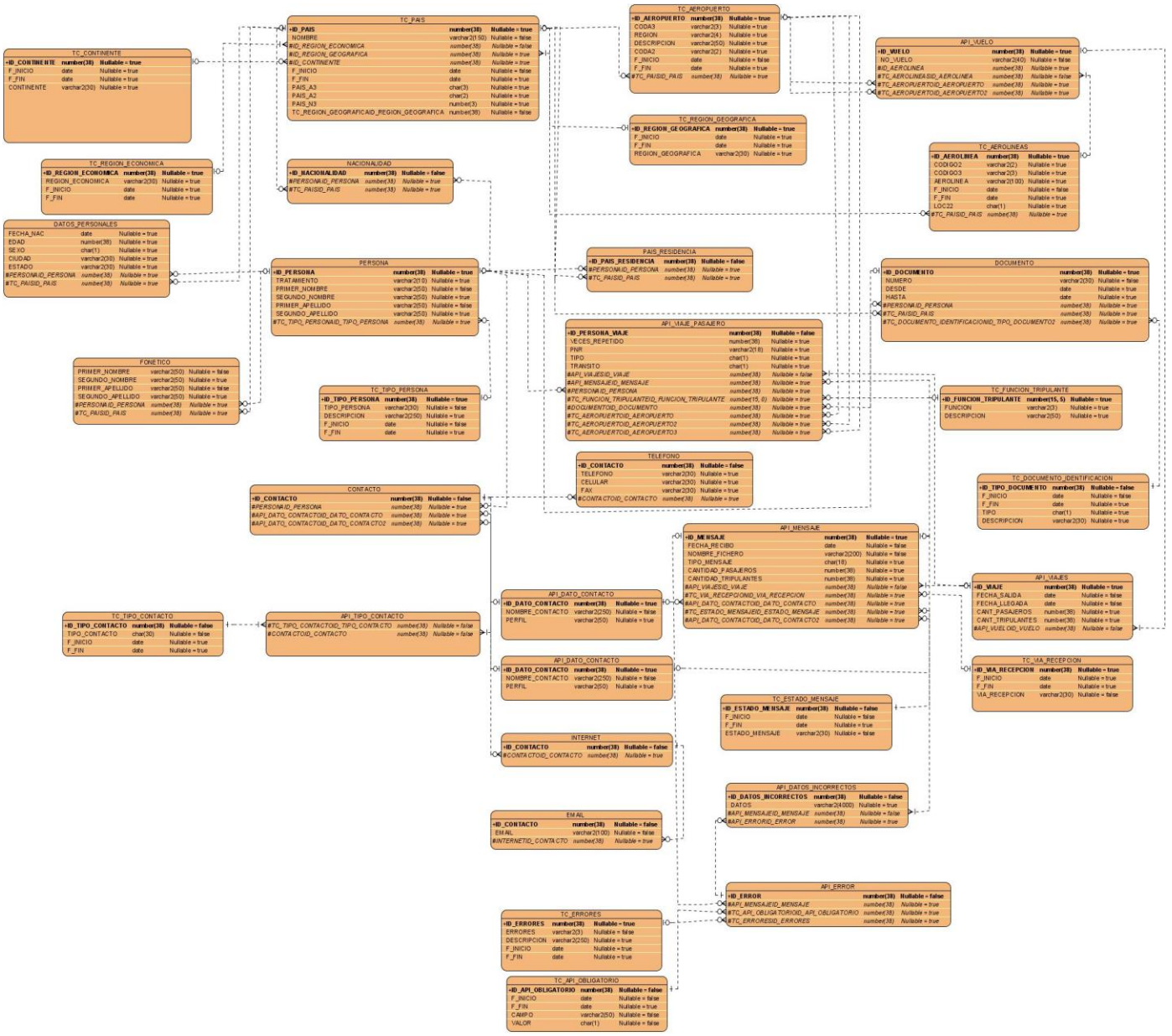

Figura #24 Modelo Físico de datos

CONCLUSIONES

Para el desarrollo de esta investigación se realizó un estudio sobre los sistemas existentes de Información Adelantada con el objetivo de conocer cómo marcha la gestión de esta información en el mundo, analizar si alguno se podría aplicar en Cuba y las características más importantes que debería tener un sistema informático que se desarrollase con este propósito. Con este estudio se determinó que ninguno de los sistemas estudiados satisfacen las necesidades de la Aduana General de la República por lo que se decidió realizar un sistema que cumpla con los requisitos identificados.

Para conocer los requisitos que debía tener el sistema se realizó un estudio de los procesos de Información Adelantada y como estos debían llevarse a cabo en la AGR. Este estudio sirvió como base para la captura de requisitos realizada, la cual estuvo basada en las técnicas siguientes, las cuales son muy usadas en la actualidad:

- ✓ Las entrevistas en grupos de desarrollo.
- ✓ Las Entrevistas abiertas terminadas.
- ✓ La introspección.
- ✓ La VORD.
- ✓ La tormenta de ideas.
- ✓ El análisis de protocolo.

Estas técnicas permitieron capturar las funcionalidades del sistema, las cuales fueron especificadas en los casos de uso y modeladas en el diseño del sistema a través de los diagramas de clases con estereotipos web. Con toda la información obtenida se propuso además el modelo físico de los datos que debía utilizarse en la aplicación que se desarrolle.

Para el modelado del sistema se utilizó RUP como metodología de desarrollo con las modificaciones establecidas en el Polo Productivo Sistemas Tributarios y el lenguaje de modelado UML y la notación BPMN.

Con los artefactos generados para el desarrollo de la Tesis se lograron cumplir los objetivos propuestos para el presente trabajo de diploma, ya que se lograron describir los procesos de Información Adelantada,

se identificaron los requisitos, se describieron los casos de uso del sistema y se modeló la solución llegando así al diseño del nuevo sistema a desarrollar. Se cumplieron además las tareas asignadas para el cumplimiento de la tesis y se generaron los artefactos establecidos para los roles de analista y diseñador.

RECOMENDACIONES

Con el desarrollo del nuevo sistema de Información Adelantada se garantiza su interrelación con la nueva arquitectura a la que quiere migrar la AGR y se logra un mayor análisis de la información recibida en el mensaje, por lo que se recomienda la implementación del mismo para su implantación en el menor tiempo posible en las aduanas del país.

Además de pensar en la integración de la aplicación con el sistema de Despacho de los Medios de Transporte Internacionales y el sistema de Enfrentamiento con el objetivo de lograr un mayor alcance en el análisis de la información lo que permitirá una gestión más eficiente en la AGR.

BIBLIOGRAFÍA

1. **J.H, Rech, W.M, Konig y G.J, Chretien.** *THE SITA NETWORK.* Septiembre 9-15, 1973.

2. **Lara, Rafael Jesus.** *La Historia de la Internet v1.0.* Venezuela : Facultad de Ciencias Universidad Central de Venezuela.

3. SITA Specialists in air transport communications and It solutions. [En línea] SITA. [Citado el: 3 de febrero de 2009.] http://www.sita.aero/content/sita-history.

4. Free Logistics. [En línea] 01 de Agosto de 2008. [Citado el: 14 de febrero de 2009.] http://www.free-logistics.com/index.php/EDI/View-category.html.

5. **WCO, IATA and ICAO.** *Guidelines on Advance Passenger Information (API).* 2003. p. 80.

6. **Acevedo González, Rogelio y Pupo Pérez, Pedro Ramón.** *Gaceta Oficial de la República de Cuba Ministerio de Justicia.* Ministerio de Justicia. La Habana : s.n., 2007. Resolución Conjunta Instituto de Aeronáutica Civil de Cuba y Aduana General de la República.

7. **U.S. Department of Homeland Security.** Homeland Security. [En línea] [Citado el: 15 de Diciembre de 2008.] http://www.dhs.gov/xlibrary/assets/usvisit/US-VISIT_Spanish_Web_Pamphlet.pdf.

8. **Coombe, Mr David.** Sistema de Administración de Fronteras. [En línea] [Citado el: 15 de Diciembre de 2008.] http://bordersystems.com/es/index.htm.

9. *EL PROGRAMA CANADIENSE DE INFORMACIÓN ANTICIPADA.* OACI. 2004. pág. 5, REUNIÓN DEPARTAMENTAL DE FACILITACIÓN (FAL).

10. **Teleformación.** *Fase de Inicio. Modelo del Negocio.* Ingenieria de Software, Universidad de las Ciencias Informáticas. La Habana : s.n., 2008. Conferencia.

11. **Jacobson, Ivar, Booch, Grady y Rumbauch, James.** *El Proceso Unificado De Desarrollo De Software.* Madrid : PEARSON EDUCACIÓN.S.A, 2000. pág. 464. Número de páginas de la información: 107.

12. **Casanovas, Josep.** Usabilidad y arquitectura del software. [En línea] 09 de 09 de 2004. http://www.desarrolloweb.com/articulos/1622.php.

13. **Alba, Alfonso.** Symfony: framework para el desarrollo de aplicaciones web en PHP. [En línea] 15 de 05 de 2008. http://www.volatt.es/articulos/articulo/ver/2008/mayo/titulo/framework-para-el-desarrollo-de-aplicaciones-web-en-PHP.html.

14. **Cobo Rodríguez, José Antonio.** *Línea Base Arquitectónica para el Polo Sistemas Tributarios y de Aduanas.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2009. pág. 137, Tesis. Páginas consultadas: 44 a la 54.

15. **Hernández Aguilar, Ing Violena.** *Modelado de Proceso del Negocio.* La Habana : s.n., 2008.

16. Visual Paradigm para UML (SE) [Linux] (Visual Paradigm for UML (SE) [Linux]). [En línea] 05 de 03 de 2007. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(SE)_%5Bcuenta_de_Linux_14736_p/.

17. **Admin.** Captura de Requisitos. [En línea] 03 de 10 de 2007. http://jcgm.ei.uvigo.es/jcmoreno/?page_id=2.

18. **Teleformación.** *Patrones GRASP.* 2009. Conferencia.

19. **Teleformación.** *Arquitectura y Patrones de Diseño.* Ingeniería de Software, Universidad de las Ciencias Informáticas. La Habana : s.n., 2009. Conferencia.

20. **Visconti, Marcello y Astudillo, Hernán.** *Fundamentos de Ingenieria de Software.* s.l. : Universidad Técnica Federico Santa María {visconti,hernan} en inf.utfsm.cl, 2008.

21. **Rational, Ayuda del.** Analista de sistemas. 2007.

22. **Escalona, María José y Koch, Nora.** *Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo.* Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla. 2002. pág. 26. Escuela Técnica Superior de Ingeniería Informática.

23. **Teleformación.** *Continuación del FT Análisis y Diseño. Modelo de Diseño.* Ingeniería de Software, Universidad de las Ciencias Informáticas. La Habana : s.n., 2009. Conferencia.

24. **Fábrega Martínez, Pedro Pablo.** Configuración de cron: Una guía básica Programación de tareas El demonio crond. [En línea] http://dns.bdat.net/documentos/cron/x50.html.

25. **SITA.** *Electronic Border Management For Cuba.*

26. **Giraldo, Gloria Lucía.** *Ingeniería de Requisitos y Modelos Conceptuales.* Universidad Nacional de Colombia. Conferencia.

27. **Teleformación.** *Extensiones WEB.* 2009. p. 10, Conferencia.

ANEXOS

Anexo #1: Diferentes Líneas Aéreas.

Las líneas aéreas que se conectaron para formar la Sociedad Internacional de Telecomunicaciones Aeronáuticas fueron: Air France, KLM, Sabena, Swissair, TWA, Corporación de Vías aéreos británica europea (BEAC), Corporación de Vías aéreos británica De ultramar (BOAC), Vías aéreos británicas sudamericanas (BSAA), A.G.Aerotransport sueco, Det Danske Luftfartselskab A/S danés y Noweigan Det Norske Luftfartselskap.

Anexo #2: Crones.

Los crones son programas que se ejecutan cada cierto tiempo y que ejecutan determinadas tareas. Por ejemplo para el caso del procesar mensajes se utiliza un cron que se ejecuta cada 1 minuto y que lo que hace es buscar en la carpeta “validados” y de encontrar mensajes los procesa. Para hacer esto primero hay que agregar la tarea a que se ejecute todos los minutos, para lograr esto hay que abrir el crontab que es el programa encargado de “manejar” las tareas automáticas, para ello se pone en cualquier consola de un sistema en Linux: crontab –e. [24]

Acto seguido se abre una pantalla donde se agrega una línea como la siguiente:

***/1 * * * * /var/www/sua/batch/procesar.php**

Los campos que describen el instante de ejecución son por orden:

- Minuto 0-59.
- Hora 0-23.
- Día del mes 0-31.
- Mes 0-12 (o su nombre con las tres primeras letras en inglés).
- Día semana 0-7 (o su nombre con las tres primeras letras en inglés).

La dirección /var/www/sua/batch/procesar.php significa que en este lugar dentro del directorio del proyecto es que se va a guardar la tarea programada en PHP, la cual se ejecutará cada 1 minuto.

Anexo #3: Expresiones Regulares.

Una expresión regular, a menudo llamada también patrón, es una expresión que describe un conjunto de cadenas sin enumerar sus elementos. Por ejemplo, la cadena:

UNA:+.? '
UNB+UNOA:4+IBCKI:ES+ADCUAPIS:ZZ+081106:1721+08110617210020++APIS'
UNG+PAXLST+IBERIA L?.A?.E?.:ES+ADCUAPIS:ZZ+081106:1721+1+UN+D:02B'
UNH+08110617210020+PAXLST:D:02B:UN:IATA+IB6621/081106/2100+20'

representa un ejemplo del inicio de un fichero API, donde la cabecera UNA significa el inicio del mismo y los símbolos que le siguen a continuación son los elementos que se van a utilizar para delimitar el mismo. Este fichero tiene una estructura definida la cual es esta : UNA a continuación UNB después UNG y luego UNH, como esta estructura es así siempre para que cumpla con el formato UNEDIFACT y se le considere un mensaje API esta no puede variar y por ello es posible aplicarle una expresión regular como puede ser por ejemplo :

"/UNA[\W|\w|\s|\S|#]{6}[\s]{0,}UNB.UNOA..[\W|\w|\s|\S]{10,200}[\s]{0,}[UNG[\W|\w|\s|\S]{20,200}]{0,1}[\s]{0,}UNH."

Para entender esta expresión regular hay que estudiar más profundamente sobre ellas pero para tener una idea significa que ella acepta cadenas que comiencen con la palabra UNA y seguido tienen que venir 6 caracteres los cuales pueden ser espacios en blanco"\s", caracteres alfanuméricos "\w", cualquier cosa que no sean caracteres alfanuméricos "\W", o cualquier cosa que no sea un espacio en blanco "\S". A continuación tiene que venir obligatoriamente la cadena UNB y luego un carácter cualquiera, a continuación la cadena UNOA y así hasta el final.

Para el final del fichero también se puede aplicar una expresión regular, a continuación se muestra un ejemplo de un final de fichero:

UNT+021+CM0247-080208P'
UNE+1+0802081540'
UNZ+1+0802081540'

La expresión regular correspondiente sería:

"/UNT[\W]{1}[\w]{0,6}[\W]{1}[\w]{0,14}[\W]{1}[\s]{0,}UNE[\W]{1}[\d]{0,6}[\W]{1}[\w]{0,14}[\W]{1}[\s]{0,}UNZ[\W]{1}[\d]{0,6}[\W]{1}[\w]{1,14}\W

Donde significa que comienza con UNT a continuación un carácter que no sea alfanumérico “\W” luego de 0 a 6 caracteres alfanuméricos y así hasta pasar por la cadena UNE para luego de otra serie de caracteres llegar a la cabecera UNZ.
Los ficheros que cumplan con estas expresiones regulares se consideran mensajes API, ya que contienen el formato UNEDIFACT PAXSLT norma ISO 9735. Si el día de mañana cambia algo en la estructura del mensaje sólo habría que cambiar la expresión regular correspondiente.

Anexo #4: Búsqueda Fonética.
El proceso de buscar fonéticamente una persona es muy simple realmente, el problema radica a la hora de guardarle el fonético de la persona.
La base de datos está preparada para guardarle a cada persona su fonético y a la hora de hacer una búsqueda lo único que hay que hacer es buscar las personas que tengan el mismo fonético, pero para que cada persona tenga su fonético hay determinadas reglas propias de cada lenguaje. Se pondrá un ejemplo:

En el español las palabras que terminan en ‘s’ como por ejemplo Lisis y Yixis no difieren fonéticamente de Lixi y Yixi respectivamente, es por esto que una regla del idioma español es quitar en las palabras que terminan en ‘s’ la última letra y desecharla.

Otra regla sería para la s, c, k ya que por ejemplo en el caso de Cesar la primera C no siempre “suena” como S ya que en el caso de Carlos esa misma C ahora suena como K, para el primer caso habrá una regla que dirá que todas las C que estén delante de E o I se cambiarán por S ya que la S siempre suena igual frente a cualquier vocal y para el segundo caso , el de Carlos, se cambiarán todas las C que estén delante de A, O, y U por la letra K ya que esta siempre mantiene su sonido delante de cualquier letra.

Como se puede apreciar la palabra original va sufriendo transformaciones a través de estas y otras reglas que están definidas ya internacionalmente para el idioma español, y al final va a quedar una cadena que sólo contiene consonantes fuertes. Entonces hay una tabla definida internacionalmente donde se le va codificando, por ejemplo la cadena Krl, que pudo pertenecer a Karlos, Kerli etc, para la k le corresponde una secuencia alfanumérica cualquiera, para la r otra y para la l otra, quedando la palabra por ejemplo de

la siguiente forma: QW$·FE, entonces este valor es el que se guarda como fonético de Karlos o el nombre que haya sido.

A la hora de hacer la búsqueda es solamente codificar el nombre que estoy buscando y buscar los que son iguales a él.

Anexo #5: Estructura del mensaje API.

A continuación se muestra la estructura del formato UNEDIFACT, a través del cual deben venir los mensajes API, para esto ver las figuras 25 y 26 que se detallan a continuación.

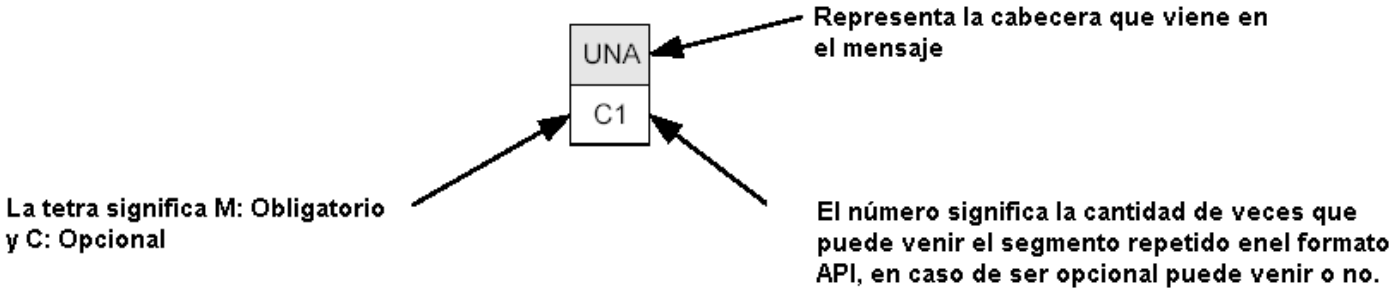

Figura #25 Cabeceras del mensaje API.

- ✓ El nivel 0: Significa las cabeceras de inicio y fin del mensaje.
- ✓ El nivel 1: Representa a los distintos mensajes que pueden venir dentro de un mismo fichero.
- ✓ El nivel 2: Representa los datos del nombre del contacto y el de los pasajeros y tripulantes.
- ✓ El nivel 3: Representa los datos específicos de estos, sexo, aeropuerto de origen, aeropuerto de destino, lugar de despacho, nacionalidad entre otros, estos datos hacen referencia al pasajero o tripulante delimitado en el nivel 2.
- ✓ El nivel 4: Representa los datos del documento oficial de viaje.

La figura # 27 representa en ejemplo completo de un mensaje, en el ejemplo se explica cada parte para una mejor comprensión del mismo.

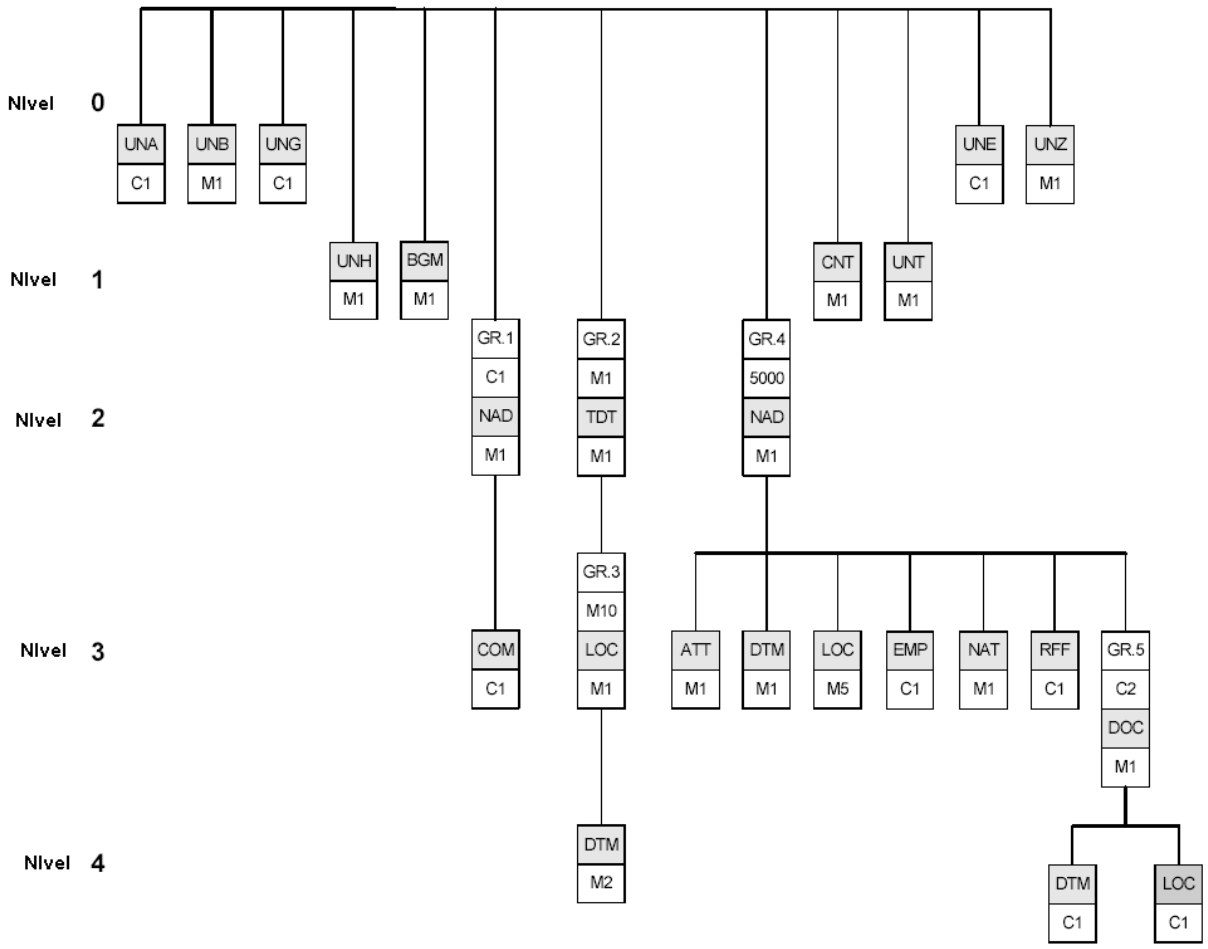

Figura #26 Árbol que representa la estructura del mensaje API.

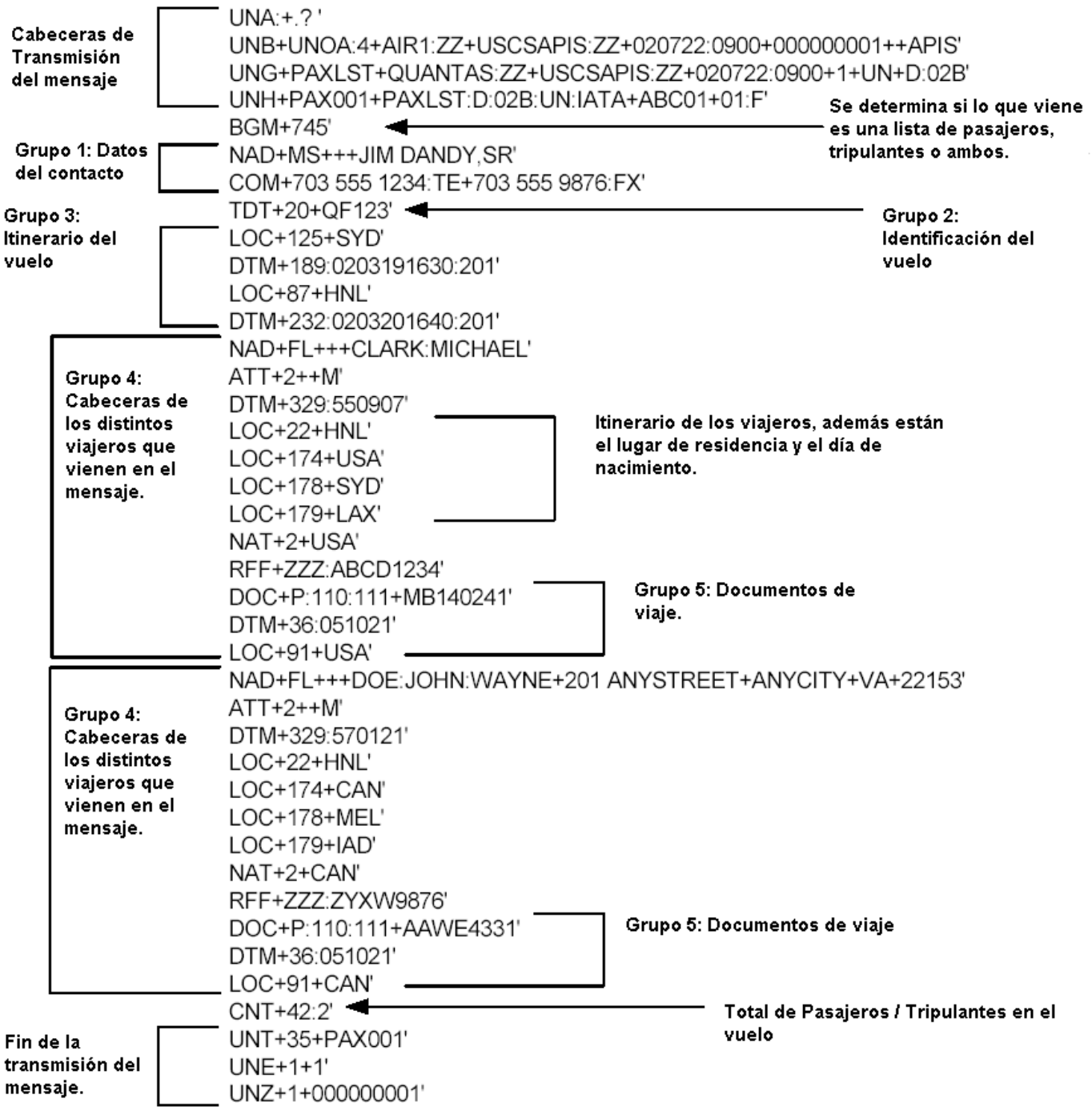

Figura #27 Mensaje API. Formato UNEDIFACT

Anexo #6: Prototipos de Interfaz de Usuario.

- ✓ Prototipo de Interfaz Caso de Uso Subir Mensaje.

Figura #28 Prototipo de Interfaz CU Subir Mensaje

- ✓ Prototipos de Interfaz Caso de Uso Conformar Mensaje.

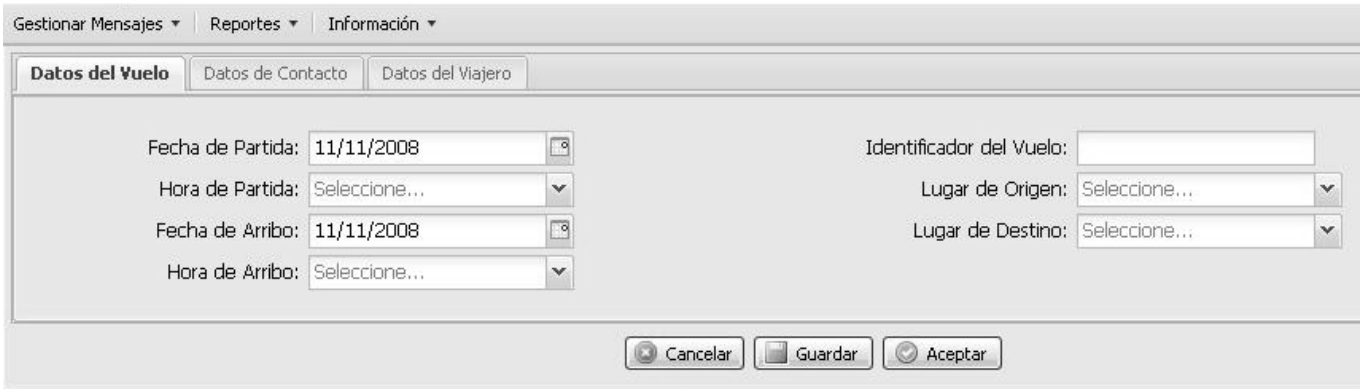

Figura #29 Prototipo de Interfaz CU Conformar Mensaje: Datos de Vuelo

Figura #30 Prototipo de Interfaz CU Conformar Mensaje: Datos de Contacto

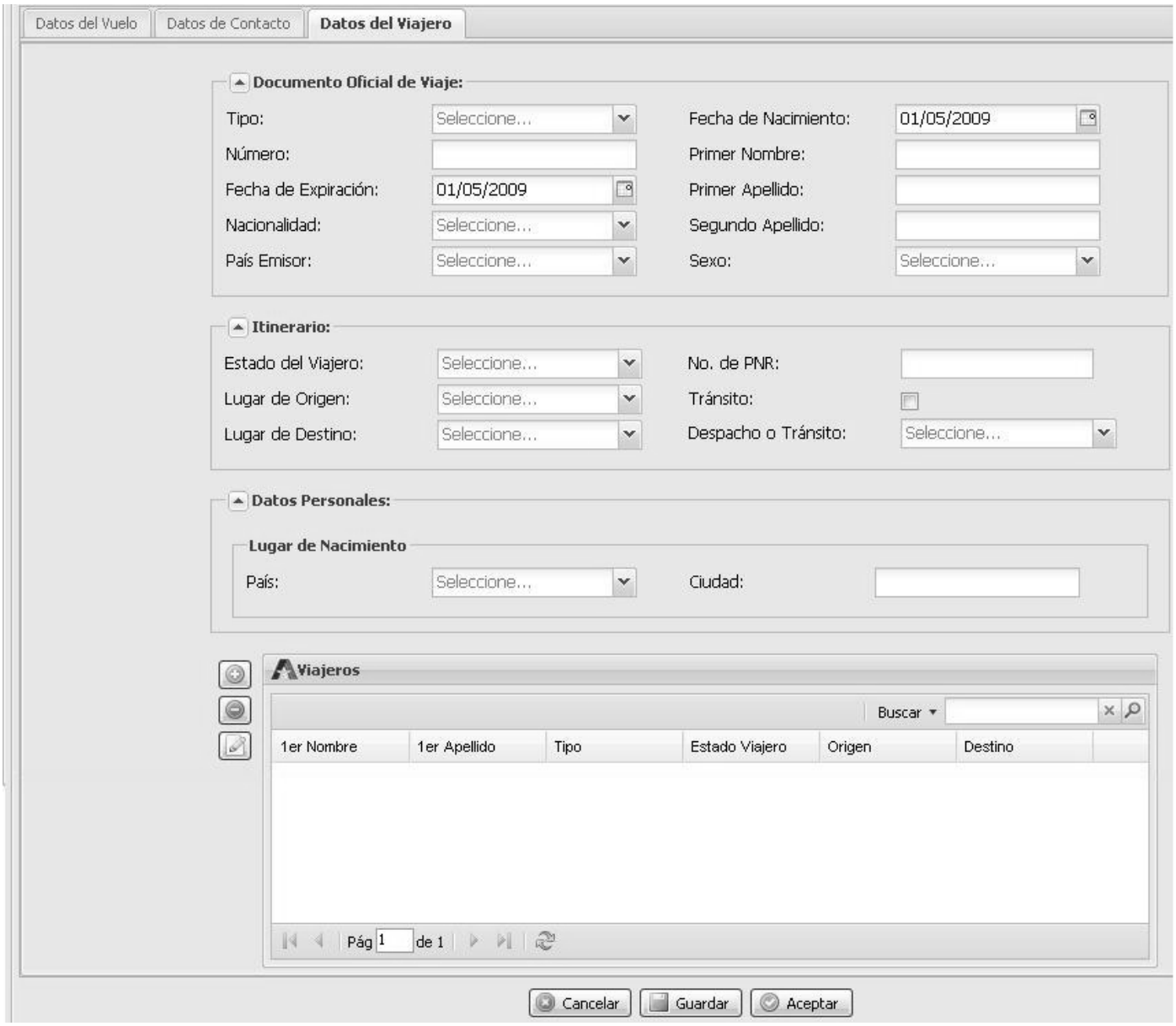

Figura #31 Prototipo de Interfaz CU Conformar Mensaje: Datos del Viajero

- ✓ Prototipos de Interfaz Caso de Uso Analizar Vuelos.

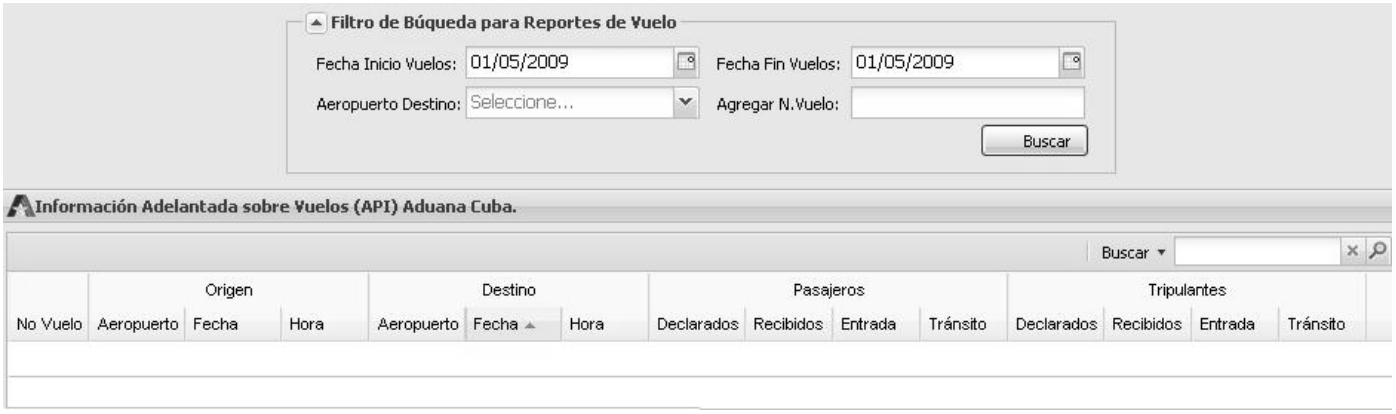

Figura #32 Prototipo de Interfaz CU Analizar Vuelos

- ✓ Prototipos de Interfaz Caso de Uso Mostrar Detalles.

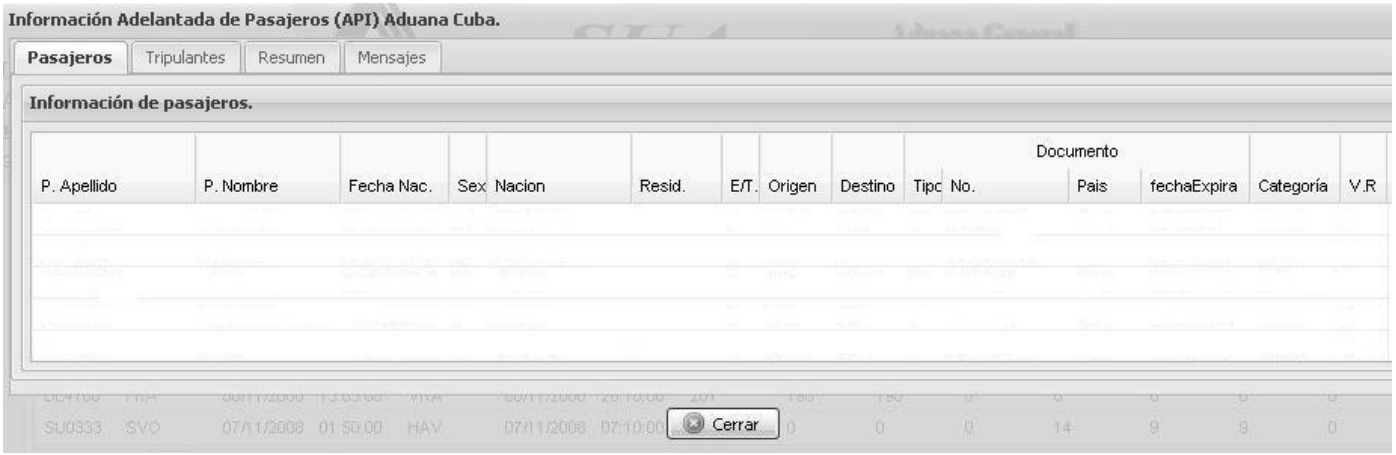

Figura #33 Prototipo de Interfaz CU Mostrar Detalles: Pasajeros

Figura #34 Prototipo de Interfaz CU Mostrar Detalles: Tripulantes

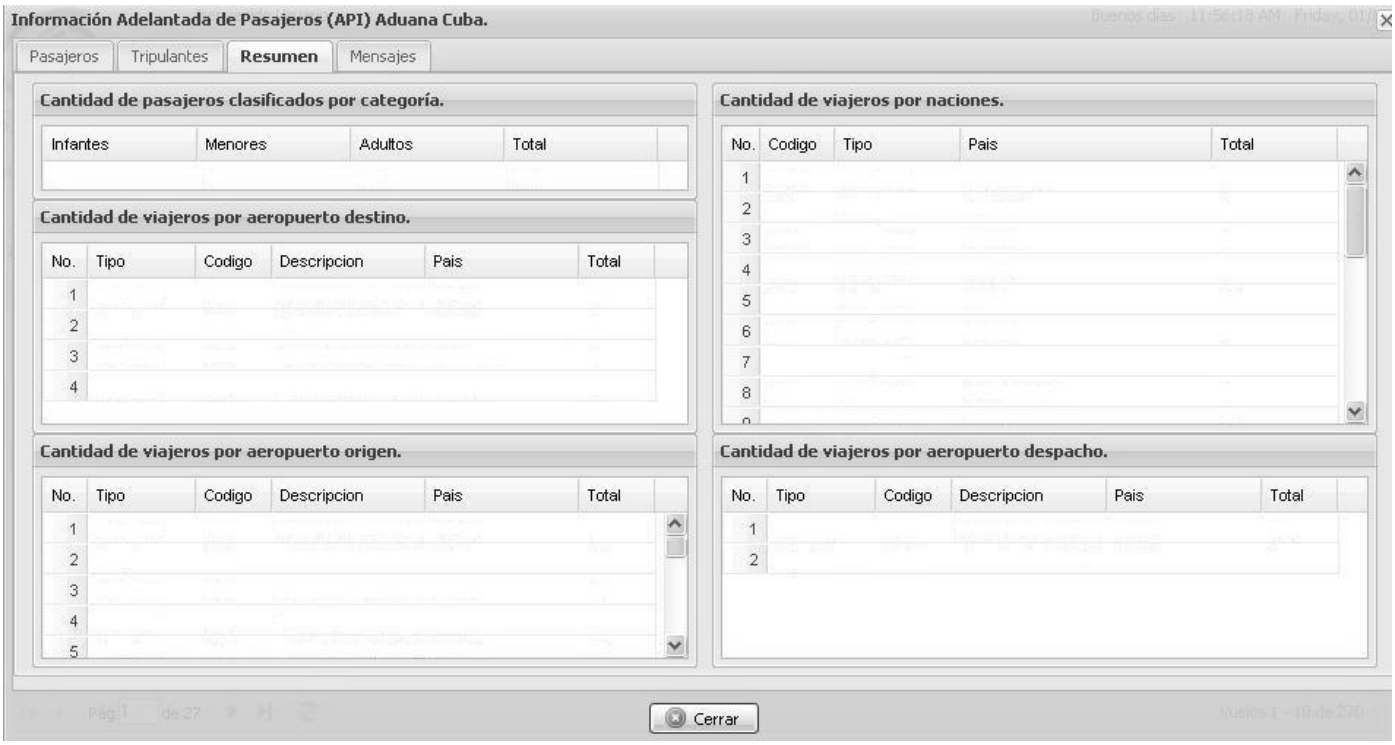

Figura #35 Prototipo de Interfaz CU Mostrar Detalles: Resumen

Figura #36 Prototipo de Interfaz CU Mostrar Detalles: Mensajes

- ✓ Prototipos de Interfaz Caso de Uso Consultar Estado Mensaje.

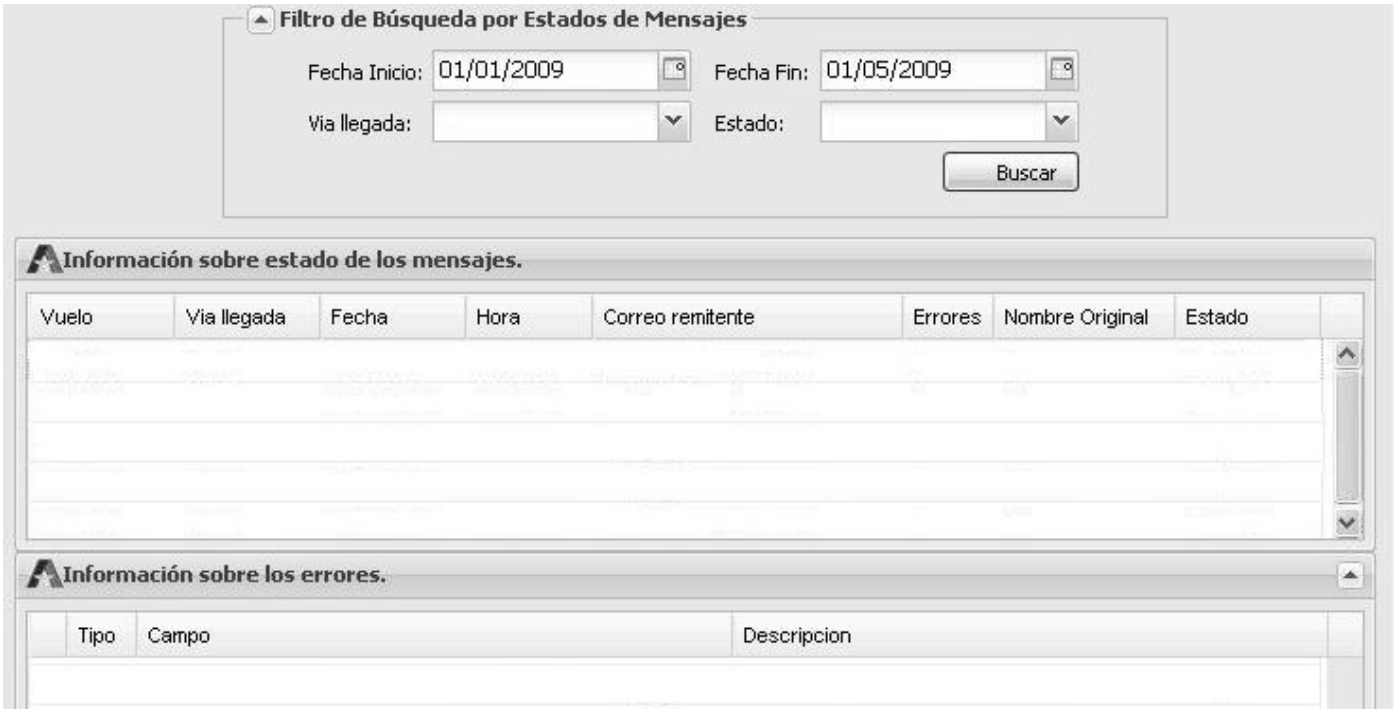

Figura #37 Prototipo de Interfaz CU Consultar Estado Mensaje

- ✓ Prototipos de Interfaz Caso de Uso Mostrar Mensajes Ilegibles.

Figura #38 Prototipo de Interfaz CU Mostrar Mensajes Ilegibles

- ✓ Prototipos de Interfaz Caso de Uso Conformar Parte.

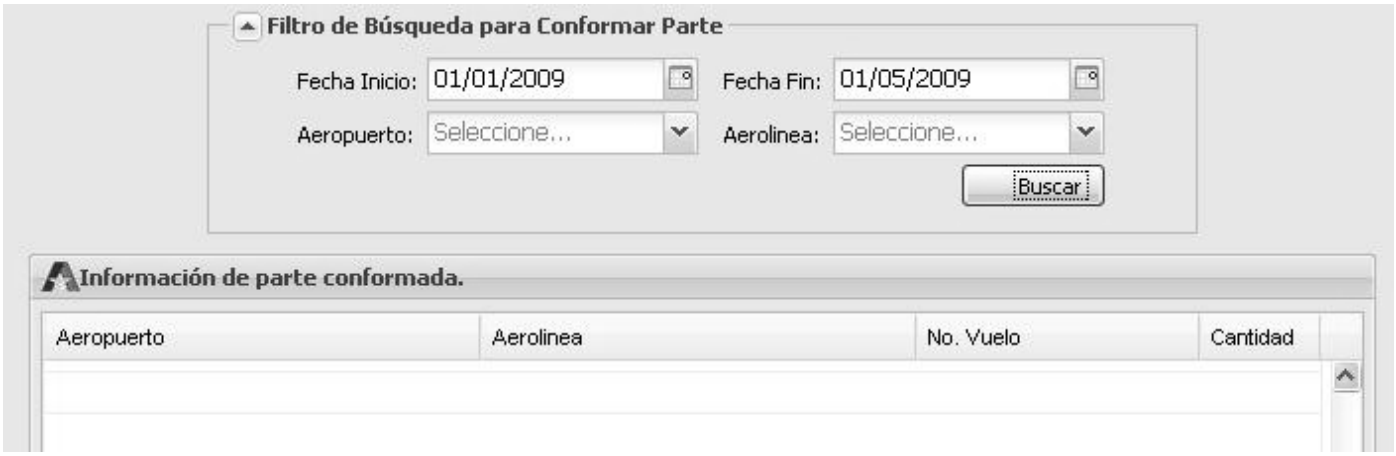

Figura #39 Prototipo de Interfaz CU Conformar Parte

- ✓ Prototipos de Interfaz Caso de Uso Buscar Fonéticamente.

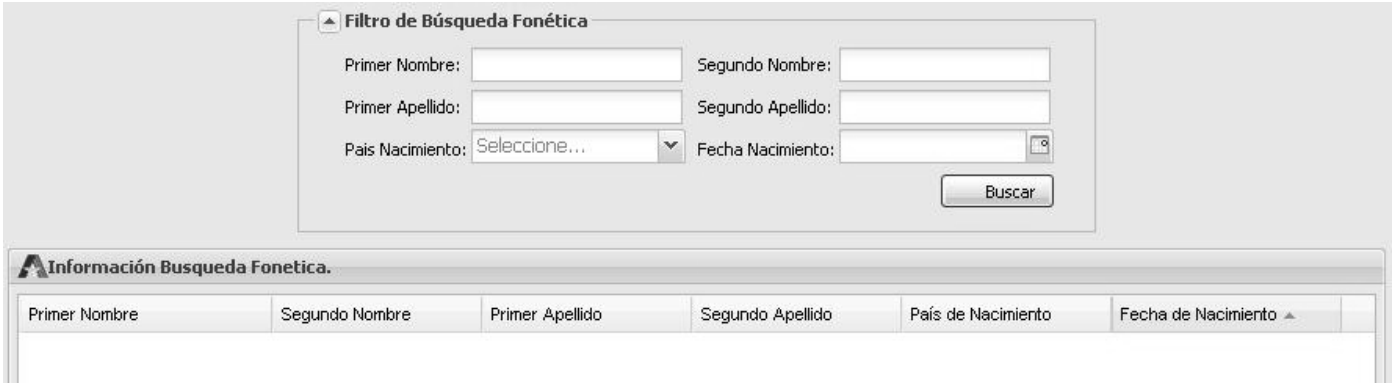

Figura #40 Prototipo de Interfaz CU Buscar Fonéticamente

GLOSARIO DE TÉRMINOS

AGR:

Aduana General de la República de Cuba. Es el máximo órgano aduanero del país.

ANSI X12:

También conocido como "ANSI X12 y X12 ASC," es un protocolo de la American National Standards Institute (ANSI) para el intercambio electrónico de datos (EDI). X12 fue el principal estándar de América del Norte para la definición de transacciones EDI.

API:

Advance Passenger Information o Información Adelantada de Pasajeros, se refiere a la información obtenida de los datos del pasaporte o visado de las personas que viajan a un país. Estos datos son transmitidos por medios electrónicos a las autoridades competentes del país destino luego de la salida del vuelo.

CADI:

Centro de Automatización para la Dirección e Información.

C / C++:

C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel.

Herramienta CASE:

Por sus siglas en ingles significa “Computer Aided Software Engineering” o Ingeniería de Software asistida por Ordenador. Bajo este nombre existen diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, Implementación de parte

del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

HTML:

Acrónimo inglés de HyperText Markup Language (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

Gracias a Internet y a los navegadores del tipo Internet Explorer, Opera, Firefox o Netscape, el HTML se ha convertido en uno de los formatos más populares que existen para la construcción de documentos y también de los más fáciles de aprender.
Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

IATA:

International Air Transport Association o Asociación de Tráfico Aéreo Internacional.

Mensaje API:

Fichero que contiene la Información Adelantada de Pasajero, esta información viene con el formato UNEDIFACT y contienen la información referente al vuelo, los viajeros (pasajeros y tripulantes) y los datos del contacto.

Mensaje Aceptado:

Mensaje API que contiene todos los datos obligatorios establecidos para la aduana en que está implantado el sistema.

Mensaje Rechazado:

Mensaje que le faltan datos relacionados con los datos identificativos de las personas como (Nombre, primer Apellido, nacionalidad, Etc.).

Mensaje Indefinido:

Mensaje que le faltan datos relacionados con el vuelo (origen, destino, hora de llegada, fecha de llegada, número de vuelo) pero cumplen con el formato UNEDIFACT PAXLST.

Mensajes Ilegibles:

Ficheros que no cumplen con el formato UNEDIFACT PAXLST.

MVC:

Modelo Vista Controlador.

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

OACI:

Organización de Aviación Civil Internacional.

OMA:

Organización Mundial de Aduanas.

PHP:

Personal Home Page.

Es un lenguaje interpretado de alto nivel embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl, con solamente un par de características PHP específicas. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil. Es un lenguaje que se ejecuta del lado del servidor.

RUP:

El Proceso Unificado de Rational (RUP, en inglés Rational Unified Process) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Proceso de ingeniería de un software desarrollado por IBM Rational. Incorpora las mejores prácticas de la ingeniería.

SITA:

Sociedad Internacional de Comunicaciones Aeronáuticas. Red privada de la Aviación Internacional.

UML:

Unified Modeling Language o Lenguaje Unificado de Modelado. Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

Viajero:

Todo pasajero y/o tripulante del vuelo.