UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

SISTEMA AUTOMATIZADO DE GESTIÓN DE VUELO PARA EL AVIÓN IL-96-300.

Trabajo de Diploma para optar por el Título de

Ingeniero en Ciencias Informáticas.

Autores: Ernesto Avila Domenech.
Yoandry Rodríguez Fernández.

Tutores: Ing. Maidelis Milanés Luque.
Ing. Leydis Esther Garzón Giro.

Ciudad de La Habana,
Julio 2008.
"Año 50 de la Revolución"

"La Revolución se lleva en el corazón no en la boca para vivir de ella."

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2008.

Yoandry Rodríguez Fernández.

_____

Autor

Ernesto Avila Domenech.

_____

Autor

Leydis Esther Garzón Giro.

_____

Tutor

Maidelis Milanés Luque.

_____

Tutor

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Sistema Automatizado de Gestión de Vuelo para el Avión IL-96-300.

Autor(es): Ernesto Avila Domenech.

Yoandry Rodríguez Fernández.

Los tutores del presente Trabajo de Diploma consideran que durante su ejecución los estudiantes mostraron las cualidades que a continuación se detallan:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Por todo lo anteriormente expresado considero que los estudiantes están aptos para ejercer como Ingenieros Informáticos y propongo que se le otorgue al Trabajo de Diploma la calificación de ___.

_____	_____	_____
Firma	Firma	Fecha

AGRADECIMIENTOS

Yoandry:

…A mi mamá por estar siempre allí.

…A ti Nany por apoyarme.

Ernesto:

…A mis padres por existir.

…A mi hermano por ayudarme tanto en todo este tiempo.

…A mis amigos en especial Edgar por ayudarme a superar los momentos difíciles.

…A Baby por aguantarme tanto.

…A Rosita, Humberto y Susana por su gran apoyo.

…A todos los que hicieron que este día fuese posible.

DEDICATORIA

Yoandry

A mis padres…

Ernesto

A mis abuelos en el cielo

y a mis padres en la tierra…

Resumen

La tripulación de Cubana de Aviación que opera los aviones IL-96 serie 300 tiene como principal objetivo la seguridad de sus vuelos. Actualmente están presentando algunas dificultades en los cálculos necesarios para la realización de estos, debido a la complejidad y exactitud de los cálculos para obtener el Plan de Vuelo y la Hoja de Peso y Balance. Por ello resulta de suma importancia para la tripulación la elaboración de un plan de vuelo confiable, que brinde los resultados en un breve período de tiempo, asegurando así la efectividad de los cálculos, cuya exactitud garantizaría la seguridad del vuelo. De igual forma debe tenerse en cuenta la ubicación de la carga y los pasajeros dentro del avión contemplados en la Hoja de Peso y Balance. Estos cálculos y otros que deben realizarse en muchas ocasiones manualmente por la tripulación del avión deben de estar exentos de errores humanos por lo que el principal objetivo de este trabajo es realizar un sistema que gestione la información de los vuelos del IL-96-300. El sistema desarrollado facilita estos cálculos del Plan de Vuelo y el Peso y Balance para un IL-96-300 lo que permite obtener una mayor autonomía y evita los posibles errores que pudieran cometerse en la confección manual de estos planes. Dicho sistema reportará beneficios significativos e importantes para la tripulación de estos aviones ahorrando una suma considerable de dinero al país.

Tabla de Contenidos

Tabla de Figuras

Introducción

El IL-96 es un avión de pasajeros de largo alcance diseñado y empezado a fabricar en la antigua Unión Soviética, actual Rusia. Surgió en principio como una mejora del IL-86, con la agregación de un ala totalmente transformada, motores silenciosos y de bajo consumo, una cabina de vuelo mucho más moderna, con un interior preparado para albergar distintas clases. En cuanto a las prestaciones del avión, se mejoró el régimen de crucero y de ascensos, los consumos, y unas mejoras considerables en despegues y aterrizajes, además de haberle añadido aletas marginales (winglets) en los extremos de las alas, que le otorgaban unas mejores condiciones aerodinámicas.

El IL-96-300 es la variante inicial, equipada con motores Aviadvigatel (Soloviev) PS90A turbofans. Su desarrollo comenzó a mediados de los años 80. Dicho avión posee una longitud de 55,3 m y una altura de 17,5 m, pesando estando vacío unas 183 t pudiendo llegar a 250 t. Su velocidad máxima es de 82 Mach y su mayor capacidad de combustible es de 150 000 L. Puede transportar la cifra de 262 pasajeros.

El primero de ellos arribó a La Habana el 31 de diciembre del 2005 y desde entonces realiza vuelos a países del continente americano. Esto ha sido posible por una gran preparación que a hecho la tripulación asignada por la empresa de Cubana de Aviación. En la actualidad es sumamente importante para la tripulación de estos aviones realizar los cálculos de peso y balance del avión, para lograr de este modo un vuelo satisfactorio, garantizando ante todo la seguridad de los pasajeros. Hoy en día estos cálculos han de hacerse de forma manual. Las pocas tablas existentes y que son sumamente necesarias para realizar la función, han sido hechas por el propio personal de la empresa, y el número de cálculos que se necesita es excesivamente grande. Debido a esto y a que en ocasiones los cálculos se realizan en condiciones de peligro, existe la posibilidad real de introducir errores humanos, y poner en peligro la vida de los pasajeros y de la tripulación. También necesitan realizar una serie de cálculos, datos, e informes que en la actualidad son obtenidos de empresas internacionales, y en el peor de los casos hechas manualmente. Por solo citar un ejemplo, para que el avión IL-96-300 pueda realizar vuelos, se deben solicitar previamente los Planes de Vuelo a SITA (Sociedad Internacional de Telecomunicaciones Aeronáuticas, con sede en Atlanta), pagando por ello 6 dólares por cada plan de vuelo, más un valor agregado de aproximadamente 10 dólares por concepto de comunicación.

Otro problema que presentan es que la mayoría de los procesos que tienen lugar antes de la salida o llegada de un vuelo se hacen actualmente en los aeropuertos cubanos de forma manual, haciendo engorroso todo este trabajo que tiene como premisa fundamental la rapidez y seguridad. Todos estos aspectos motivaron la necesidad de crear un sistema que diera solución a estas dificultades.

El **problema** de esta investigación esta dado por el alto volumen de información que se tramita para la realización de los vuelos, y se quiere contar con un sistema automatizado que permita gestionar la información de los vuelos de forma ágil y precisa. Tomando en consideración la **situación problémica** anteriormente expuesta, se puede plantear como **problema científico** ¿Cómo mejorar la gestión de la información de los vuelos del IL-96-300?

Para dar solución a la problemática planteada el **objeto de estudio** lo constituyen los procesos para la gestión de vuelos. Según lo planteado el **campo de acción** está conformado por los procesos ejecutados en la elaboración de los cálculos del Plan de Vuelo y del Peso y Balance para el avión IL-96-300.

Como **idea a defender** se plantea: si se confecciona el sistema automatizado de gestión de vuelo para el avión IL-96-300, entonces se brindará a los tripulantes del avión un sistema que posibilite un mejor servicio de gestión de vuelos.

El **objetivo general** es realizar un sistema que gestione la información de los vuelos del IL-96-300.

De ahí se derivan los siguientes **objetivos específicos**:

- Realizar la gestión de la información referente al Peso y Balance del Avión IL-96-300.
- Realizar la gestión de la información referente al Plan de Vuelo del Avión IL-96-300.

Para alcanzar dichos objetivos se planteó desarrollar las siguientes **tareas:**

- Realizar entrevistas con los clientes.
- Seleccionar y fundamentar la metodología y herramientas a utilizar para el desarrollo del sistema informático.
- Estudiar detalladamente el proceso de gestión de la información de los vuelos del avión IL-96-300.

- Realizar la implementación de los módulos de Peso y Balance y Plan de Vuelo del sistema Ilyushin.
- Desarrollar el documento de la investigación.

Entre los **resultados esperados** de este trabajo de diploma está la concepción de un sistema que gestione la información de los vuelos del IL-96-300.

El presente trabajo está dividido en cinco capítulos y un anexo, que recogen todos los temas abordados en la investigación y desarrollo del mismo, como la Fundamentación Teórica, Características del Sistema, Análisis y Diseño, Implementación y Estudio de Factibilidad del Sistema de Gestión de Vuelos del IL-96-300.

Capítulo1: Fundamentación Teórica, explica la metodología utilizada para la investigación, e incluye un estado del arte del tema tratado, a nivel internacional y nacional. Además fundamenta las tecnologías, metodologías y herramientas escogidas para el desarrollo de la solución a proponer.

Capítulo 2: Características del Sistema, describe el objeto de estudio, el problema y situación problémica, objeto de automatización, la propuesta del sistema. Además la especificación de los requisitos funcionales y no funcionales, definiéndose los casos de uso del sistema.

Capítulo 3: Análisis y Diseño del Sistema, se aborda los detalles relacionados con el análisis y diseño del sistema, representándose los diagramas de clases del análisis, los diagramas de colaboración y los diagramas de clases del diseño, con sus respectivas descripciones.

Capítulo 4: Implementación, se representan gráficamente el modelo de despliegue y los diagramas de componentes que representan la construcción del sistema propuesto.

Capítulo 5: Estudio de Factibilidad, donde se realiza la estimación de tiempo y el costo del proyecto para analizar si es viable, analizando los beneficios que proporcionará la construcción del sistema propuesto con las herramientas seleccionadas.

Capítulo 1: Fundamentación Teórica.

Introducción

En el presente capítulo se realizará un estudio del arte donde se abordarán de forma valorativa y crítica las tendencias actuales tanto a nivel internacional y nacional del sistema a construir. Se brindarán elementos de las posibles herramientas a utilizar, lenguajes de programación y metodologías que se podrán utilizar para la realización del sistema, teniendo en cuenta los sistemas que existen en la actualidad para realizar las tareas a implementar.

1.1 Estado del arte

La información necesaria para la realización de un vuelo está compuesta principalmente por el análisis de pista, el análisis del peso y balance del avión y por último la confección del plan de vuelo. Existen antecedentes de un sistema construido para la obtención del plan de vuelo para otros modelos de avión como YAK-40, YAK-42, AN-26 e IL-62. Es conocido como SAPLAV y fue construido en la Ciudad Universitaria José Antonio Echeverría (CUJAE) en el año 1993. En la actualidad no existe un único software que realice la gestión de vuelos del avión IL-96-300 a nivel nacional, sino que se cuenta con tres sistemas que de forma independiente gestionan el análisis del peso y balance, el análisis de pista y la confección del plan de vuelo respectivamente.

- Sistema automatizado para la obtención del Plan de vuelo del IL-96-300.

Este sistema facilita los cálculos para la obtención del plan de vuelo de forma autónoma evitando posibles errores. Para la planificación de los vuelos el sistema utiliza las especificaciones para este tipo de avión contenidas en el Manual Básico de Operaciones y el Manual de Vuelo, además hace un análisis de los datos meteorológicos y las Cartas de tiempo, además de las Cartas de vientos y temperaturas.

- Sistema de Automatización para el Análisis de pista del IL-96-300.

Este sistema es capaz de realizar los cálculos necesarios para crear los Análisis de Pista del IL-96-300 y de gestionar toda la información de los aeropuertos correspondientes a los Análisis de Pista. El

sistema además de gestionar la información referente a los aeropuertos, es capaz de realizar dos tipos de análisis de pista el análisis operativo y el análisis completo.

- Sistema de automatización para el peso y Balance del IL-96-300.

El sistema es capaz de realizar la ubicación de la carga y los pasajeros en el avión, de una forma balanceada. Realizar la Hoja de Peso y Balance, dar la posibilidad de insertar un nuevo avión al llegar a Cuba con su Peso Básico Operacional e índices correspondientes y ejecutar en forma óptima el centrado del avión. Incluye además la posibilidad de realizar cambios en el Performance del avión por un usuario autorizado. Los tripulantes del avión tienen conocimiento de un sistema en forma de calculadora que permite crear el análisis de pista y permite calcular el peso y balance del avión, pero este software es exclusivo de Rusia por lo que no se pudo acceder a él para analizar su funcionamiento.

1.2 Metodologías de Desarrollo de Software

En el mundo de la informática no se para de hablar de procesos de desarrollo, buscando el modo de trabajar eficientemente para evitar catástrofes que llevan a que un gran porcentaje de proyectos se terminen sin éxito. El objetivo de un proceso de desarrollo es aumentar la calidad del software (en todas las fases por las que pasa) a través de una mayor transparencia y control sobre el proceso. Por eso es de gran interés mostrar algunas metodologías de desarrollo que se pueden utilizar para llevar a cabo un software educativo sin importar el tipo que sea.

RUP

La metodología pesada RUP, llamada así por sus siglas en inglés Rational Unified Process, es uno de los procesos más generales de los existentes actualmente, ya que en realidad está pensado para adaptarse a cualquier proyecto, y no tan solo de software. RUP se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes.

Esta metodología utiliza el Lenguaje Unificado de Modelado (UML, Unified Modeling Language) para preparar todos los esquemas de un sistema de software. UML es una parte esencial del Proceso

Unificado, fueron desarrollados paralelamente por las mismas personas, haciendo que su integración sea un éxito.

Fig. 1: Proceso de Desarrollo de Software

"El Proceso Unificado es un proceso de desarrollo de Software. O sea es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto. El Proceso Unificado está basado en componentes, lo cual quiere decir que el software en construcción está formado por componentes software interconectados a través de interfaces bien definidas"

Los aspectos definitorios y a la vez que convierten en único al Proceso Unificado, se resumen en tres fases: dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental.

– **Dirigido por casos de uso:** Un caso de uso es el conjunto de acciones que debe realizar un sistema para dar un resultado de valor a un determinado usuario; se capta cuando se modela el proceso del negocio y se representa a través de los requerimientos. A partir de aquí todos los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

– **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

– **Iterativo e incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. Con su culminación se obtiene un producto con un determinado nivel, que irá creciendo incrementalmente en cada iteración.

El RUP incluye cuatro etapas importantes que son: Inicio, Elaboración, Construcción y Transición, cada una de ellas compuesta de una o varias iteraciones. Estas etapas revelan que para producir una versión del producto en desarrollo se emplean todas las actividades de ingeniería pero con diferente énfasis. Además contempla flujos de trabajo de soporte que involucran actividades de planificación de recursos humanos, tecnológicos y financieros. Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos.

Extreme Programing (XP)

Programación Extrema (XP) es una metodología reciente en el desarrollo de software. La filosofía de XP es satisfacer al completo las necesidades del cliente, por eso lo integra como una parte más del equipo de desarrollo.

Fue creada para el desarrollo de aplicaciones donde el cliente no sabe muy bien lo que quiere, lo que provoca un cambio constante en los requisitos que debe cumplir la aplicación. Por este motivo esta metodología se adapta a las necesidades del cliente y la aplicación se va reevaluando en períodos de tiempo cortos. Está diseñada para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño, donde la comunicación sea más factible que en grupos de desarrollo grandes. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes.

La metodología se basa en:

- Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándose en algo hacia el futuro, se pueda hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.
- Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.

La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

1.3 Herramientas

Rational Rose

Rational Rose Enterprise Edition se utiliza para la representación gráfica de los modelos que se obtienen durante el ciclo de vida de un software empleando como notación el lenguaje de modelado UML. Facilita la modelación de los procesos del negocio, captura de requisitos, análisis y diseño orientado a objetos, implementación del sistema mediante componentes y despliegue en las diferentes vistas, que son: vista de casos de uso, vista lógica, vista de componentes y vista de despliegue.

Es una herramienta compatible solamente con sistemas operativos de Microsoft que permite generar documentación y código fuente (de programas y bases de datos) a partir de los modelos para lenguajes como son: Java, C++, Ada, Visual Basic, entre otros. Además permite la ingeniería inversa (obtención de los modelos a partir del código fuente) para diferentes lenguajes, esta permite el ahorro de tiempo a los desarrolladores y disminuye las probabilidades de cometer errores.

Rational se integra con varios entornos de desarrollo, sobre todo con diversas versiones del Visual Studio. Usa un lenguaje estándar común para todo el equipo de desarrollo que facilita la comunicación. Acelera la implementación de sistemas con la calidad requerida.

Crystal Report

Crystal Reports es una herramienta para diseño de reportes creada mucho antes del nacimiento de la tecnología .NET; gracias a ésta una de las partes más tediosas e importantes del diseño de aplicaciones, el diseño de reportes, se hace más sencilla. Crystal Reports puede ser utilizado con diferentes bases de datos, entre ellas MS Access, SQL Server, Oracle, Informix, etc.

Crystal Report 10 es la herramienta de elaboración de informes estándar para Visual Studio .NET. Esta herramienta permite crear contenidos interactivos con calidad de presentación. Mediante Crystal Report se pueden crear informes en función de las necesidades de desarrollo:

- Seleccionar la opción de diseño de informe que le interese, desde informes estándar hasta cartas modelo, o crear una propia.
- Mostrar gráficos en los que los usuarios puedan profundizar con el fin de ver datos detallados de los informes.
- Calcular resúmenes, subtotales, y porcentajes de datos agrupados.
- Dar formato al texto y rotar objetos de texto cuando se cumplan determinadas condiciones.

Crystal Report 10 posee un fácil mecanismo para exportar los informes a varios formatos como Word y Excel, así como a formatos PDF, HTML.

1.4 Lenguaje de Modelado

UML

El Lenguaje Unificado de Modelado (UML, Unified Modeling Language) es la herramienta usada en la descripción y construcción de software reconocida por la industria como un estándar. Es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Este lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML es una notación con la cual se construyen sistemas por medio de conceptos orientados a objetos. Este prescribe un conjunto de notaciones y diagramas estándares, y describe la semántica esencial de lo que estos diagramas y símbolos significan.

1.5 Plataformas de Desarrollo

Antes de analizar una plataforma se debe comenzar por definir qué se conoce por plataforma:

Una plataforma es una combinación de hardware y software usada para ejecutar aplicaciones; en su forma más simple consiste únicamente de un sistema operativo, una arquitectura, o una combinación de ambos.

Plataforma Java

Java es una tecnología orientada al desarrollo de software con el cual se puede realizar cualquier tipo de programa. Hoy en día, la tecnología Java ha cobrado mucha importancia en el ámbito de Internet gracias a su plataforma J2EE. La tecnología Java está compuesta básicamente por dos elementos: el lenguaje Java y su plataforma. Es una plataforma sólo de software y se ejecuta sobre las otras plataformas de hardware.

La plataforma Java tiene dos componentes: la máquina virtual de Java (JVM) y la Interfaz de programación de aplicaciones (Java API, Application Programming Interface). El Java API es una gran colección de componentes de software que proporcionan muchas utilidades para el programador, por ejemplo, los API's para las interfaces gráficas. Los API's de Java están agrupados en librerías de ciertas Clases e interfaces, estas librerías son conocidas como paquetes.

El siguiente gráfico describe un programa que se está ejecutando sobre la plataforma Java. Como se puede observar, el Java API y la máquina virtual aíslan al programa del hardware.

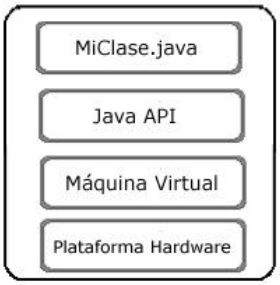


Fig. 2: Ejecución de un programa Java.

Plataforma .NET

La Plataforma .NET es una plataforma de desarrollo de software desarrollada por Microsoft que permite un rápido desarrollo de Aplicaciones tanto Desktop como Web, compitiendo en este ultimo caso con la plataforma Java de Sun Microsystem. Es una nueva tecnología con énfasis en

transparencia de redes, con independencia de plataforma, permitiendo un rápido desarrollo de aplicaciones, combinando la informática y las comunicaciones y ofreciendo las herramientas que se necesitan para transformar las aplicaciones Web.

.NET utiliza los Servicios Web como un medio para poder interactuar con distintas tecnologías. Permite conectar distintos sistemas operativos, dispositivos físicos, información y usuarios. Les da a los desarrolladores las herramientas y tecnologías para hacer rápidamente soluciones de negocios que involucran distintas aplicaciones, dispositivos físicos y organizaciones.

La idea central que se implementa con la plataforma .NET es la del uso de servicios con el objetivo de simplificar el desarrollo de aplicaciones Web. Más concretamente software como servicio y de cómo construir, instalar, consumir, integrar o agregar estos servicios para que puedan ser accedidos mediante Internet. La plataforma .NET permite usar Internet y su capacidad de distribución para que los usuarios accedan desde cualquier dispositivo, en cualquier sistema operativo y lugar a la funcionalidad que los Servicios Web proveen.

.NET está diseñada para que se puedan desarrollar componentes de software utilizando casi cualquier lenguaje de programación, de forma que lo que se escriba en un lenguaje pueda utilizarse desde cualquier otro de la manera más transparente posible. Esto es, en vez de estar limitados a un único lenguaje de programación, permitir cualquier lenguaje de programación, siempre y cuando se adhiera a unas normas comunes establecidas para la plataforma .NET en su conjunto.

En el corazón de .NET se encuentra el marco de trabajo .NET (.NET Framework). Este contiene dos componentes principales: Lenguaje común en tiempo de ejecución (CLR, Common Language Runtime) y la Biblioteca de Clases de Framework .NET, que incluye ADO.NET, ASP.NET y los formularios Windows Forms. Common Language Runtime es el fundamento de esta tecnología.

Fig. 3: Framework .NET

Common Language Runtime

El CLR es el motor encargado de gestionar la ejecución de las aplicaciones desarrolladas a las que ofrece numerosos servicios que simplifican su desarrollo y favorecen su fiabilidad y seguridad. Por esta razón, al código de estas aplicaciones se le suele llamar código gestionado, y al código no escrito para ser ejecutado directamente en la plataforma .NET se le suele llamar código no gestionado.

El código generado por los compiladores para la plataforma .NET no es código máquina para ningún tipo de Unidad Central de Procesos (CPU) en concreto, sino que generan código escrito en el lenguaje intermedio conocido como Lenguaje Intermedio de Microsoft (Microsoft Intermediate Language, MSIL), este es el código máquina del CLR que actúa como una máquina virtual, encargándose de ejecutar las aplicaciones diseñadas para la plataforma .NET. Es un lenguaje de un nivel de abstracción mucho más alto que el de la mayoría de los códigos máquina de las CPUs existentes, e incluye instrucciones que permiten trabajar directamente con objetos (crearlos, destruirlos, inicializarlos, llamar a métodos virtuales, etc.), tablas y excepciones (lanzarlas, capturarlas y tratarlas).

Las principales características y servicios que ofrece el CLR son:

- Modelo de programación consistente: A todos los servicios y facilidades ofrecidos por el CLR se accede de la misma forma: a través de un modelo de programación orientado a objetos. En los

sistemas operativos actuales (por ejemplo, los de la familia Windows), a algunos servicios se les accede a través de llamadas a funciones globales definidas en Biblioteca de enlace dinámico (Dynamic Link Library, DLL) y a otros a través de objetos (objetos COM en el caso de la familia Windows).

- Eliminación del “infierno de las DLLs”: La plataforma .NET elimina el problema conocido como "infierno de las DLLs" que consiste en que al sustituirse versiones viejas de DLLs compartidas por versiones nuevas puede que dejen de funcionar algunas aplicaciones si las nuevas no son 100% compatibles con las anteriores. En la plataforma .NET las versiones nuevas de las DLLs pueden coexistir con las viejas, de modo que las aplicaciones diseñadas para ejecutarse usando las viejas podrán seguir usándolas tras la instalación de las nuevas.
- Ejecución multiplataforma: El CLR actúa como una máquina virtual, encargándose de ejecutar las aplicaciones diseñadas para la plataforma .NET. Es decir, toda plataforma para la que exista una versión del CLR podrá ejecutar cualquier aplicación .NET.
- Integración de lenguajes: Desde cualquier lenguaje para el que exista un compilador que genere código para la plataforma .NET es posible utilizar código generado para la misma usando cualquier otro lenguaje tal y como si de código escrito usando el primero se tratase.
- Gestión de memoria: El CLR incluye un recolector de basura que evita que el programador tenga que tener en cuenta cuándo ha de destruir los objetos que dejen de serle útiles.
- Seguridad de tipos: El CLR incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, para evitar que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura.
- Aislamiento de procesos: El CLR asegura que desde código perteneciente a un determinado proceso no se pueda acceder a código o datos pertenecientes a otro, lo que evita errores de programación muy frecuentes e impide que unos procesos puedan atacar a otros.
- Tratamiento de excepciones: En el CLR todo los errores que se puedan producir durante la ejecución de una aplicación se propagan de igual manera: mediante excepciones. Anteriormente los sistemas Windows propagaban los errores de diferentes vías, mediante códigos de error en formato Win32, mediante HRESULTs y mediante excepciones.
- Soporte multihilo: El CLR es capaz de trabajar con aplicaciones divididas en múltiples hilos de ejecución que pueden ir evolucionando por separado en paralelo o intercalándose.

- Distribución transparente: El CLR ofrece la infraestructura necesaria para crear objetos remotos y acceder a ellos de manera completamente transparente a su localización real, tal y como si se encontrasen en la máquina que los utiliza.
- Seguridad avanzada: El CLR proporciona mecanismos para restringir la ejecución de ciertos códigos o los permisos asignados a los mismos según su procedencia o el usuario que los ejecute.
- Interoperabilidad con código antiguo: El CLR incorpora los mecanismos necesarios para poder acceder desde código escrito para la plataforma .NET a código escrito previamente a la aparición de la misma y, por tanto, no preparado para ser ejecutado. Estos mecanismos permiten tanto el acceso a objetos COM como el acceso a funciones sueltas de DLLs preexistentes (como la API Win32).

Bibliotecas de clases

Para programar una aplicación se necesitan realizar acciones como manipulación de archivos, acceso a datos, conocer el estado del sistema, implementar seguridad, etc. El Framework organiza toda la funcionalidad del sistema operativo en un espacio de nombres jerárquico de forma que a la hora de programar resulta bastante sencillo encontrar lo que se necesita.

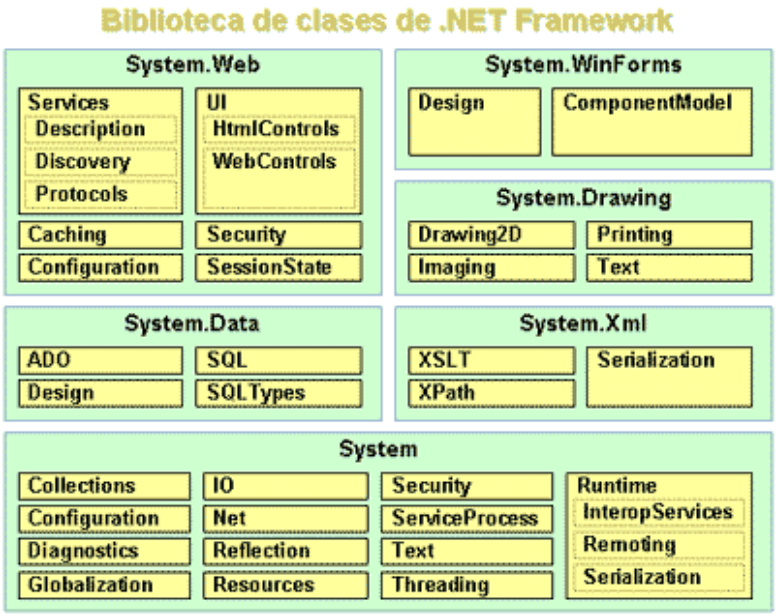


Fig. 4: Bibliotecas de Clases

Para ello, el Framework posee un sistema de tipos universal, denominado Sistema de Tipo Común (Common Type System, CTS). Este sistema permite que el programador pueda interactuar los tipos que se incluyen en el propio Framework (biblioteca de clases de .Net) con los creados por él mismo (clases). De esta forma se aprovechan las ventajas propias de la programación orientada a objetos, como la herencia de clases predefinidas para crear nuevas clases, o el polimorfismo de clases para modificar o ampliar funcionalidades de clases ya existentes.

1.6 Entorno de Desarrollo

Visual Studio .NET 2005

Visual Studio .NET 2005 es un conjunto completo de herramientas de desarrollo para la construcción de aplicaciones Web ASP, servicios Web XML, aplicaciones para escritorio y aplicaciones móviles. Visual Basic .NET, Visual C++ .NET, Visual C# .NET y Visual J# .NET utilizan el mismo entorno de desarrollo integrado (IDE), que les permite compartir herramientas y facilita la creación de soluciones en varios lenguajes.

Dichos lenguajes aprovechan las funciones de .NET Framework, que ofrece acceso a tecnologías claves para simplificar el desarrollo de aplicaciones Web ASP y servicios Web XML. Visual Studio.NET 2005 permite la creación de aplicaciones de consola, aplicaciones para Windows, archivos DLL, aplicaciones Web, XML Web Services y aplicaciones para dispositivos de bolsillo.

Visual Studio .NET 2005 es una herramienta que proporciona un entorno donde es posible crear aplicaciones avanzadas rápidamente. Algunas de estas características son:

Detección de errores automático: Es posible ahorrar muchas horas de trabajo al utilizar Visual Studio .NET para detectar los errores antes de intentar ejecutar su aplicación. Los errores potenciales se subrayan, de la misma forma que lo hacen algunos procesadores de texto mientras se escribe.

Herramientas de depuración: Visual Studio .NET 2005 mantiene sus herramientas de depuración legendarias que le permiten observar su código en acción y seguir la pista del contenido de las variables.

Diseño de formularios: Es posible crear formularios atractivos con la facilidad de arrastrar y soltar componentes de Visual Studio .NET 2005.

IntelliSense: Visual Studio .NET proporciona la finalización de sentencias de los objetos reconocidos y lista automáticamente la información sobre los parámetros de la función en útiles ToolTips.

1.7 Lenguajes de Programación

Lenguaje C#

C# es un lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg, éste último también conocido por haber sido el diseñador del lenguaje Turbo Pascal y la herramienta RAD Delphi. Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que C# carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el lenguaje nativo de .NET. La sintaxis y estructuración de C# es muy similar a la C++, ya que la intención de Microsoft con C# es facilitar la migración de códigos escritos en estos lenguajes a C# y facilitar su aprendizaje a los desarrolladores habituados a ellos. Sin embargo, su sencillez y el alto nivel de productividad son equiparables a los de Visual Basic.

Las principales características de C# son:

- **Sencillez**: Elimina muchos elementos que otros lenguajes incluyen y que son innecesarios. Por ejemplo:
    - El código escrito en C# es autocontenido, lo que significa que no necesita de ficheros adicionales al propio fuente tales como ficheros de cabecera o ficheros IDL (Lenguaje de Definición de Interfaces).
    - El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad del código.

  - Se eliminan elementos poco útiles de otros lenguajes, tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (.) para acceder a miembros de espacios de nombres (::).

- **Modernidad**: Incorpora en el propio lenguaje elementos que son muy útiles para el desarrollo de aplicaciones, como un tipo básico decimal que permite realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción foreach que permite recorrer colecciones con facilidad, la inclusión de un tipo básico string para representar cadenas o la distinción de un tipo bool específico para representar valores lógicos.
- **Orientado a objetos**: Es orientado a objeto ya que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos. Soporta todas las características propias del paradigma de programación orientada a objetos: encapsulamiento, herencia y polimorfismo con la excepción de que sólo admite herencia simple de clases pero si la implementación de múltiples interfaces. En lo referente al encapsulamiento es importante señalar que aparte de los típicos modificadores public, private y protected, C# añade un cuarto modificador llamado internal, que puede combinarse con protected e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado.
- **Orientación a componentes**: La sintaxis incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).
- **Instrucciones seguras**: Para evitar errores muy comunes, se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (==) con el de asignación (=); y todo caso de un switch ha de terminar en un break o goto que indique cuál es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.
- **Sistema de tipos unificado**: A diferencia de C++, todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada System.Object, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán “objetos”), esto también es aplicable a los tipos de datos básicos.

- **Extensibilidad de tipos básicos**: Permite definir, a través de estructuras, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos. Es decir, que se puedan almacenar directamente en pila y se asignen por valor y no por referencia.
- **Extensibilidad de modificadores**: Ofrece, a través del concepto de atributos, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente, información adicional a la generada por el compilador.
- **Eficiente**: En principio, todo el código incluye numerosas restricciones para garantizar seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad de procesamiento muy grandes.

Lenguaje Java

El lenguaje Java es de alto nivel que son aquellos en los que las instrucciones o sentencias son escritas con palabras similares a las de los lenguajes humanos (mayormente en inglés). Esto facilita la escritura y comprensión del código al programador, sus características más importantes son:

- Lenguaje orientado a objetos.
- Java es un lenguaje sencillo.
- Independiente de plataforma
- Brinda un gran nivel de seguridad.
- Capacidad multihilo.
- Gran rendimiento.
- Creación de aplicaciones distribuidas.
- Su robustez o lo integrado que tiene el protocolo TCP/IP lo que lo hace un lenguaje ideal para Internet.

Una de las principales características que favoreció el crecimiento y difusión del lenguaje Java es su capacidad de que el código funcione sobre cualquier plataforma de software y hardware. Esto significa que un mismo programa escrito para Linux puede ser ejecutado en Windows sin ningún problema.

Además es un lenguaje orientado a objetos que resuelve los problemas en la complejidad de los sistemas, entre otras.

De hecho, su versatilidad y eficiencia, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en una opción tecnología para su aplicación a redes, de manera que hoy en día, muchos dispositivos utilizan la tecnología Java. El lenguaje de programación Java ha sido totalmente mejorado, ampliado y probado por una comunidad de muchos desarrolladores de software.

1.8 Propuesta

El sistema Ilyushin tiene como objetivo gestionar la información de los vuelos realizados por los aviones Il-96-300. Este sistema está concebido realizarse a largo plazo y el personal que trabajará en el mismo puede salir del proyecto con facilidad, por lo que se necesita tener una documentación actualizada y abundante. Además el cliente no estará cerca de los desarrolladores para la construcción del mismo; por ello se propone como metodología de desarrollo de software Rational Unified Process (RUP).

Como herramienta de desarrollo se utilizará Rational Rose pues admite el lenguaje de modelado UML, que es el lenguaje por excelencia de RUP. Ejecuta chequeo semántico de los modelos, además de que permite realizar ingeniera inversa, desarrollo multiusuario y generar documentación.

Se plantea utilizar la plataforma de desarrollo .NET pues es una potente plataforma para implementar aplicaciones de escritorio ya que utiliza mecanismos robustos, seguros y eficientes. Otro de los beneficios de .NET es su soporte a múltiples lenguajes de programación, lo cual acelera la curva de aprendizaje de los desarrolladores permitiendo que cada uno elija en base a sus gustos personales. Además, la posibilidad de utilizar las mismas herramientas de programación y tener las mismas capacidades de acceso a la plataforma independientemente del lenguaje le proporciona una flexibilidad sin precedentes.

El lenguaje más utilizado en la plataforma .NET es el C#. Es muy poderosos debido a que es un lenguajes de alto nivel, que se beneficia y explota todas las funcionalidades que brinda el CLR, por tanto hace uso de todas las clases que componen la BCL. En consecuencia de esto se utilizará C# para desarrollar el sistema de gestión de vuelos del IL-96-300.

1.9 Estructura de los Módulos

Dada la explicación del cliente y de la forma de trabajo del mismo se decidió dividir el sistema en varios módulos, como son: Administración, Peso y Balance y Plan de Vuelo. Este trabajo de diploma se centra específicamente en los módulos de Peso y Balance y Plan de Vuelo.

Además de los módulos anteriores se decidió crear un módulo que contenga todos los procesos repetibles en los demás, a dicho módulo se le ha llamado “Común”. Es importante destacar que la manera más sencilla de rehusar código es copiarlo total o parcialmente desde el lugar donde ya se ha implementado hasta el lugar donde se volverá a usar; pero es trabajoso mantener múltiples copias del mismo código. Por lo general, para eliminar la redundancia se deja este código en un único lugar.

Fig. 5: Estructura de los módulos.

1.10 Arquitectura Propuesta

En la aplicación se utilizó la arquitectura en capas, dichas capas se han definido lo mejor posible teniendo en cuenta las características del sistema a implementar, esto reducirá hoyos de vulnerabilidad. Este tipo de arquitectura permite la reutilización de capas, facilita la estandarización, las dependencias se limitan solo entre las capas, permitiendo el mantenimiento y soporte de la aplicación más sencillo, brinda una mayor flexibilidad debido a que se pueden agregar nuevos módulos para dotar al sistema de nuevas funcionalidades. Además posibilita aplicaciones más robustas debido al encapsulamiento. Dentro de esta arquitectura la capa intermedia (capa de Negocio) ocupa un lugar de mucha importancia en la construcción de una infraestructura de software que permitirá el crecimiento y extensibilidad de los servicios para todas las aplicaciones existentes y futuras.

A continuación se describen las capas de la arquitectura escogida:

1. La capa de Presentación: Está formada por los formularios y los controles que se encuentran en los formularios. Es la capa con la que interactúa el usuario.
2. La capa de Negocio: Está formada por las clases que se ocupan de establecer todas las reglas que deben cumplirse. Esta capa se comunica con la de Presentación, para recibir las solicitudes y mostrar los resultados. Se comunica además con la de Acceso a Datos para mediante ella almacenar o recuperar datos de un determinado gestor de base de datos.
3. La capa de Acceso a Datos: Está formada por clases que interactúan con la base de datos y que reciben solicitudes de almacenamiento o recuperación de información desde la capa de Negocio.
4. La capa Dominio: Es una capa especial que no va alineada a las tres anteriores, es más bien una capa vertical que contendrá las clases del dominio. Esto permitirá pasar como parámetros objetos, por lo que será mucho más sencilla la comunicación entre las capas.

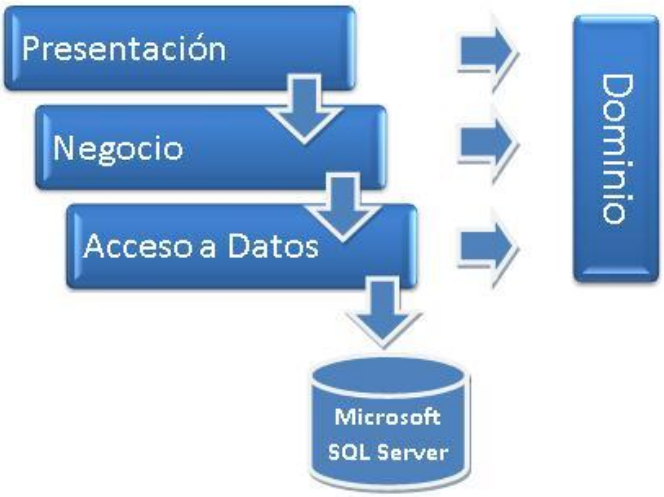


Fig. 6: Arquitectura del sistema.

Conclusiones

En este capítulo se realizó un análisis de algunos aspectos relacionados con los sistemas de gestión de vuelos. También se estudiaron las tecnologías a utilizar a lo largo del desarrollo del sistema propuesto. Se fundamentó la elección del lenguaje de programación, la metodología de desarrollo del software a utilizar y la herramienta CASE seleccionada para la modelación tanto de los procesos del negocio como del diseño de la aplicación.

Capítulo 2: Características del Sistema.

Introducción

En el siguiente capítulo se abordarán temas como los procesos que componen el negocio, características del sistema a implementar, con sus requisitos funcionales y no funcionales, de forma que se culminará con la conformación de la propuesta final de la aplicación. Se definirán además los actores y trabajadores, los casos de usos que serán automatizados en el sistema informático.

2.1 Objeto de estudio

Para la realización de un vuelo es necesario manejar por parte de la tripulación del avión ciertas informaciones, tales como:

Peso y Balance:

Actualmente el flujo de eventos se realiza de la siguiente forma:

El pasajero hace entrega del pasaporte y del equipaje en el mostrador al representante de tráfico, este último es el encargado de introducir los datos del pasajero y de hacer el pesaje del equipaje. Luego envía a la tripulación la cantidad total de adultos, cantidad total de niños y peso total de equipaje TTL (XPG).

El departamento de Carga y Correo es el responsable de enviar a la tripulación el peso de carga total TTL Carga, peso total de cantidad de correo y peso total de mercancías peligrosas.

El despachador hace el envío de todos estos datos a plan de vuelo, después de sus respectivos análisis, se entregará el peso total de combustible que necesita el avión.

Luego de este proceso la tripulación se encarga de realizar la hoja de Peso y Balance de forma manual, además de la distribución de la carga y los pasajeros en el avión.

Plan de vuelo:

Actualmente el flujo de eventos se realiza de la siguiente forma:

Cuando se le asigna un vuelo a la tripulación esta requiere de un plan de vuelo para ejecutarlo, siendo el navegante el encargado de la realización del mismo. Normalmente el navegante solicita a SITA el plan de vuelo; servicio que como se ha especificado anteriormente resulta costoso para el país; sin embargo en situaciones excepcionales el navegante se ve obligado a realizarlo manualmente debido a la imposibilidad de utilizar este servicio ya sea por falta de conexión, o alguna otra razón.

En estos casos el navegante realiza los cálculos necesarios partiendo de la información meteorológica, el peso de la carga de pago y teniendo en cuenta las restricciones impuestas por los NOTAMS y las vías del Atlántico disponibles en el día (NACKTRACK). Una vez confeccionado el plan de vuelo se discute con el piloto al mando. Luego del análisis y aceptación del plan de vuelo se elabora la hoja de Autorización de Despacho con los datos de combustible necesarios para el centrado y balance del avión, la cual se envía a Operaciones ECASA.

2.1.1 Objeto de automatización

Serán objeto de automatización los procesos para la confección del plan de vuelo y todo el proceso de cálculos de peso y balance del avión IL_96 300, los análisis de gráficos de donde se obtienen los principales datos para la obtención del peso máximo del avión, la obtención de datos de tablas tabuladas entre las que se incluyen las tablas de ascenso y descenso que junto a la información propia del avión se utilizan en la confección del plan de vuelo, distribución de la carga y los pasajeros en este, de forma tal que se encuentre balaceado, obtención de la hoja de Peso y Balance.

2.1.2 Información que se maneja

Para la confección de este sistema es necesaria la información contenida en el Manual Básico de Operaciones propio del modelo IL-96-300, Manual de Vuelo, siendo oportuno además el análisis de los datos meteorológicos y la información de los vientos que se brindan en las Cartas de tiempo significativos y las Cartas de vientos y temperaturas, así como la información referente a NOTAMS y NACKTRACKS, Manual para cálculos de Peso y Balance del IL-96-300, Hojas de Peso y Balance y tablas correspondientes a este proceso, las cuales poseen información de tipo confidencial, por lo cual los detalles de dicha información no puede ser reflejada en este documento.

2.2 Propuesta del sistema

Para acceder al sistema propuesto el usuario deberá autenticarse y de acuerdo a los permisos y privilegios que tenga accederá al módulo permitido. Una vez dentro del sistema el usuario podrá realizar solo las operaciones que se le estén permitidas.

Si el usuario es el encargado en la tripulación de realizar el peso y balance (Tripulación/PB) podrá gestionar (adicionar y modificar) los Galley, Crew, Aviones y Versiones. Podrá además realizar y eliminar los análisis de peso y balance que desee; dichos análisis podrán ser vistos por cualquier integrante de la tripulación.

Si el usuario es el encargado en la tripulación de realizar los planes de vuelo (Tripulación/PV) tendrá la posibilidad de gestionar (adicionar, modificar) un aeropuerto con la información necesaria para realizar su actividad, un punto de control, una ruta, un avión y la información referente al Crew y Versión a utilizar por los aviones, así como el PBO específico. Podrá crear también un plan de vuelo..

2.3 Modelo de Negocio.

2.3.1 Breve descripción del negocio.

Con la cantidad de pasajeros a transportar el navegante calcula el peso de equipaje según lo estipulado comercialmente (20 Kg. o 40 Kg. por pasajero), que unido a la información de la carga que debe trasladar la aeronave constituye el valor de la carga de pago. Conocida la información de las condiciones meteorológicas de los aeropuertos, el comportamiento de los vientos y las temperaturas en los diferentes niveles de vuelo y la carga de pago, el navegante confecciona el Plan de Vuelo, teniendo en cuenta las restricciones impuestas por los NOTAMS y las vías del Atlántico que estarán disponibles en el día.

Una vez confeccionado el **Plan de Vuelo** se discute con el piloto al mando. Luego del análisis y aceptación de este se elabora la hoja de Autorización de Despacho y se envía a Operaciones ECASA con los siguientes datos:

Imprescindibles:

- Combustible de quemada, combustible de reserva y combustible requerido.

Otros datos posibles:

- Aeropuertos de origen y destino.
- Tipo de avión.
- Niveles de vuelo.
- Tiempos de vuelo

En el caso del **Peso y Balance,** luego de que los pasajeros hacen entrega del pasaporte y del equipaje en el mostrador al representante de tráfico, este es el encargado de introducir los datos del pasajero y de hacer el pesaje del equipaje. Además envía al encargado en la tripulación del análisis de peso y balance la cantidad total de adultos, cantidad total de niños y peso total de equipaje TTL (XPG).

Carga y correo es el responsable de enviar a la tripulación el peso de carga total TTL Carga, peso total de cantidad de correo y peso total de mercancías peligrosas.

El cálculo de la carga y la posición de centro de gravedad en el avión se pueden dividir en dos etapas:

1. Determinación del peso de la carga útil transportable.

2. Ubicación de la carga útil, de modo que la posición del centro de gravedad en el avión sin el combustible y durante el despegue y el aterrizaje estén en el límite permisible.

Con dichos datos, el encargado del análisis de peso y balance (Tripulación/PB) realiza la distribución de la carga y de los pasajeros en el avión, además de la Hoja de Peso y Balance.

2.3.2 Actores del Negocio

Peso y Balance

Actor	Descripción
Tripulación.	Interesado en obtener la Hoja de Peso y Balance para la realización de un vuelo seguro con el menor gasto de combustible posible.

Plan de Vuelo

Actor	Descripción
Tripulación(Piloto al Mando)	Es el interesado en que se confeccione el plan de vuelo para saber como debe realizar el viaje (ruta a seguir, niveles de vuelo, aeropuerto alterno, cantidad de combustible, etc.), y como actuar ante una contingencia.

2.3.3 Trabajadores del Negocio

Peso y Balance

Trabajador	Descripción
Tripulación/PB.	Realiza los cálculos necesarios para determinar el peso y el balance con el cual va a despegar el avión y luego confeccionar la hoja de Peso y Balance.

Plan de Vuelo

Trabajador	Descripción
Tripulación/PV.	Integrante de la Tripulación encargado de confeccionar el Plan de Vuelo.

2.3.4 Diagramas de Casos de Uso del Negocio

Peso y Balance

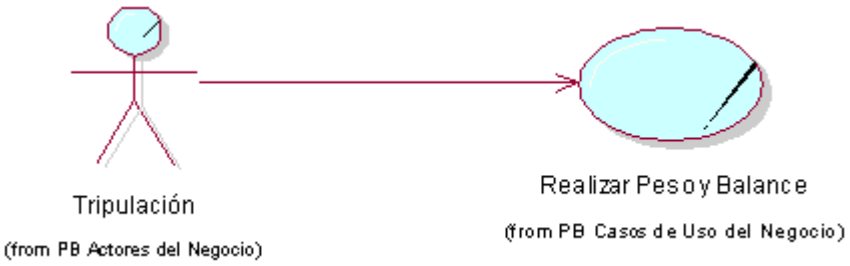


Fig. 7: Diagrama de Caso de Uso del Negocio Peso y Balance

Plan de Vuelo

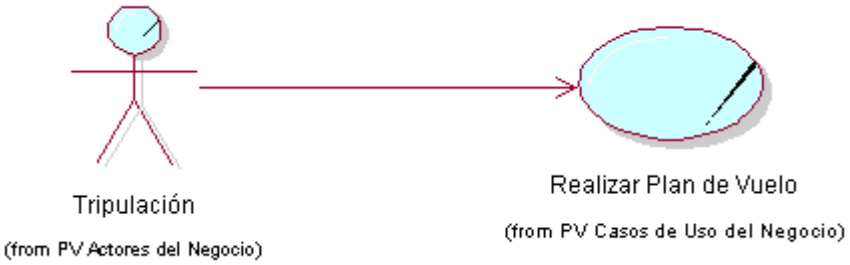


Fig. 8: Diagrama de Caso de Uso del Negocio Plan de Vuelo.

2.3.5 Especificaciones de Casos de Uso

Descripción del Caso de Uso: **Realizar Peso y Balance**

Caso de Uso:	**Realizar Peso y Balance**
Actores:	**Tripulación**
Trabajadores:	**Tripulación/PB**
Resumen:	Se inicia cuando existe la necesidad de realizar el vuelo por parte de la tripulación, para esta actividad es necesario el centrado del avión, por lo que se realiza la distribución de la carga y los pasajeros, de forma balanceada, además del combustible en los distintos tanques. Esto genera la Hoja de Peso y Balance, documento oficial necesario para la realización de cada vuelo.
Precondiciones:	Deben existir al menos un avión, crew, versión y galley en la base de

	datos.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. Solicita Hoja de Peso y Balance	2. Se realiza la distribución de los pasajeros, la carga y el combustible de forma balanceada.
	3. Se realiza la Hoja de Peso y Balance y posteriormente es entregada.
Poscondiciones	
Mejoras	
Prioridad	Crítico.

Descripción del Caso de Uso: **Realizar Plan de Vuelo**

Caso de Uso:	**Realizar Plan de Vuelo**
Actores:	**Tripulación**
Trabajadores:	**Tripulación/Navegante**
Resumen:	El caso de uso se inicia cuando se le asigna un vuelo a la tripulación para el cual necesita un plan de vuelo y no tienen conexión con SITA, por lo que es necesario hacerlo manualmente. El caso de uso finaliza con la elaboración del Plan de Vuelo.
Precondiciones:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Negocio**
1. La tripulación tiene asignado un vuelo.	2. El navegante confecciona el plan de vuelo con los datos de la carga de pago que se debe transportar, la información meteorológica y la ruta a seguir. Teniendo en cuenta la información de los aeropuertos y las restricciones de NOTAMS y los NACKTRACK.
	3. El navegante entrega el plan de vuelo al piloto al mando y juntos hacen un análisis de

	los resultados.
4. El piloto al mando de la tripulación recibe el plan de vuelo y lo analiza teniendo en cuenta las condiciones reales (las condiciones ambientales y técnicas).	
5. Si el plan de vuelo elaborado se puede cumplir bajo las condiciones establecidas se realiza el vuelo.	
Flujos Alternos	
Acción del Actor	**Respuesta del Negocio**
5. De no poderse cumplir el plan de vuelo elaborado se solicita al navegante que realice un nuevo plan de vuelo.	
	6. Se pasa al punto 2.
Poscondiciones	
Mejoras	
Prioridad	Crítico.

2.3.6 Diagramas de actividades

Diagrama de Actividades: **Realizar Peso y Balance**

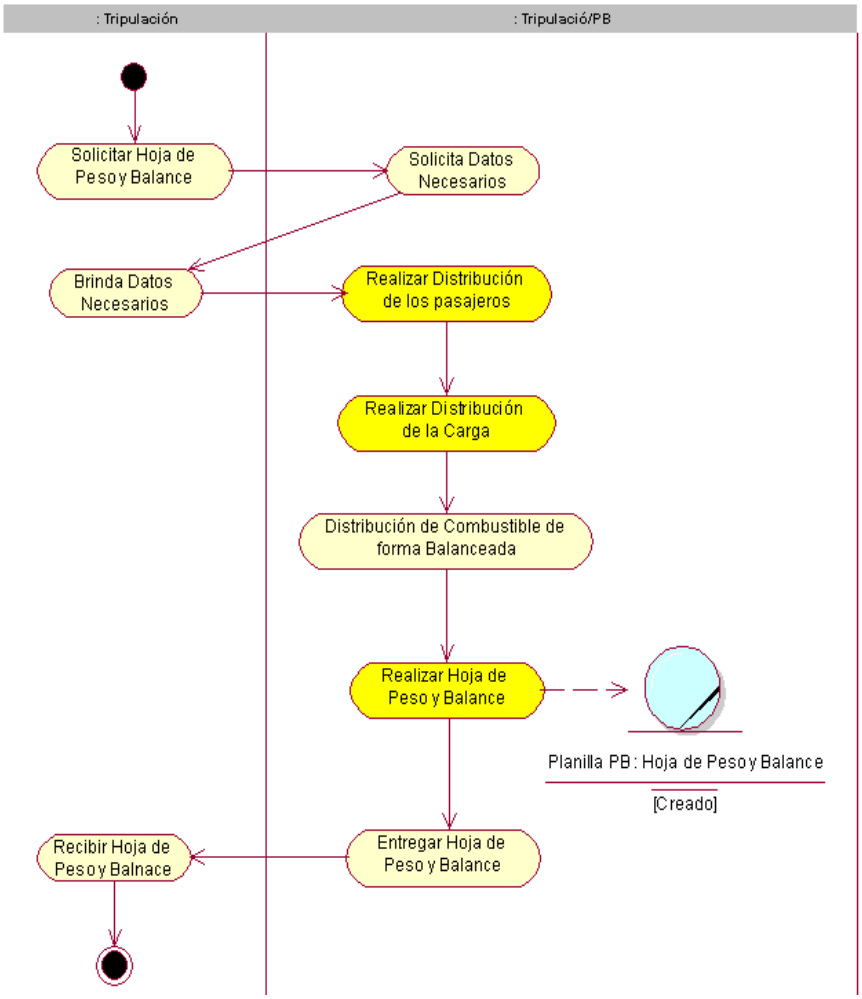


Fig. 9: Diagrama de Actividades CU Realizar Peso y Balance

Diagrama de Actividades: **Realizar Plan de Vuelo**

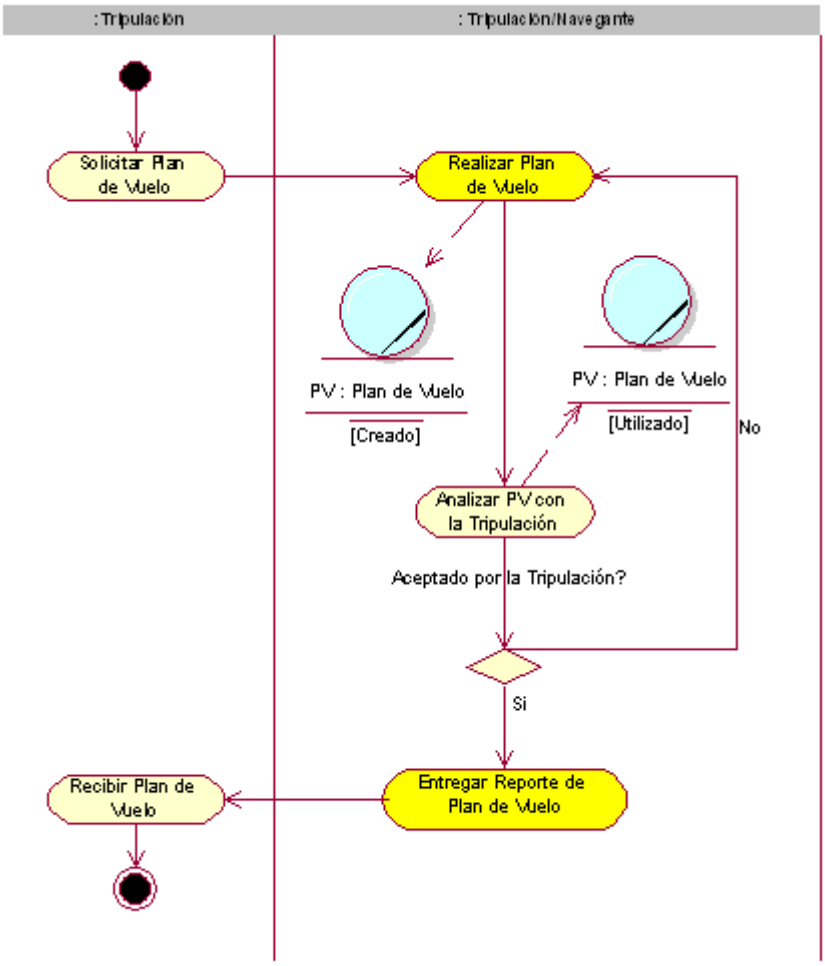


Fig. 10: Diagrama de Actividades CU Realizar Plan de Vuelo

2.3.7 Modelos de Objetos

Modelo de Objetos: **Realizar Peso y Balance**

Fig. 11: Modelo de Objetos CU Realizar Peso y Balance.

Modelo de Objetos: **Realizar Plan de Vuelo**

Fig. 12: Modelo de Objetos CU Realizar Plan de Vuelo.

2.4 Especificación de requisitos

2.4.1 Requisitos Funcionales

1. Gestionar Aeropuerto.
   1.1. Adicionar Aeropuerto.
   1.2. Modificar Aeropuerto.
2. Gestionar Hoja de Peso y Balance.
   2.1. Realizar Hoja de Peso y Balance.
   2.2. Eliminar Hoja de Peso y Balance.
3. Obtener Hoja de Peso y Balance.
   3.1 Obtener Hoja de Peso y Balance.
4. Gestionar Avión.
   4.1. Adicionar Avión.
   4.2. Modificar Avión.
5. Gestionar Crew.
   5.1. Adicionar Crew.
   5.2. Modificar Crew.
6. Gestionar Versión.
   6.1. Adicionar Versión.
   6.2. Modificar Versión.
7. Gestionar Galley.
   7.1. Adicionar Galley.
   7.2. Modificar Galley.
8. Generar Plan de Vuelo.
   8.1. Generar Plan de Vuelo.
9. Gestionar Puntos de Control.
   9.1. Adicionar Punto de Control.
   9.2. Modificar Punto de Control.
10. Gestionar Rutas.
    10.1. Adicionar Ruta.
    10.2. Modificar Ruta.

2.4.2 Requisitos No Funcionales

Apariencia o interfaz externa:

- El diseño de la interfaz externa del sistema debe ser profesional, sencillo, amigable, de fácil transición, familiar a los usuarios que han usado otras aplicaciones de escritorio en Windows, con el fin de lograr una eficiente interacción con el usuario, proporcionándole en todo momento una sensación de control sobre la aplicación. Para la construcción del sistema se deben seguir las normas convencionales de interfaz de usuario de Windows. Se debe mantener informado al usuario acerca de todo lo que sucede en la aplicación, los mensajes de esta deben estar dirigidos al usuario y por tanto, redactados en su idioma. La información se debe presentar de forma clara.

Usabilidad:

- El sistema está concebido para ser usado por la tripulación del avión IL-96-300 por lo tanto la dificultad dependerá del número de pasos, el conocimiento que el usuario debe tener del proceso y las decisiones que este debe tomar en cada paso. Para evitar errores, en los campos que se requiera el sistema brindará la opción de elegir el valor deseado en vez de que el usuario introduzca los datos.
- Se seguirán las guías de la UI para nombrar los menús, botones y las cajas de diálogo siempre que sea posible.
- Se debe informar al usuario en todo momento acerca de lo que sucede en la aplicación por lo que los mensajes deben ser evidentes y personalizados.

Rendimiento:

- El sistema operará con grandes volúmenes de información, por tanto, se hacen necesarios tiempos de respuestas cortos, al igual que la velocidad de procesamiento de la información.

Soporte:

- Debe ser de fácil instalación y mantenimiento. Se debe generar la inserción de nuevos módulos, sin negar lo realizado o afectar el buen funcionamiento del mismo.

- El sistema debe ser sometido a una etapa de adiestramiento previo donde se realicen pruebas en las cuales los usuarios se familiaricen con este y a la vez se puedan detectar posibles errores, o posibles cambios en las interfaces de manera que queden complacidos.

Portabilidad:

- El sistema funcionará sobre plataforma Windows, dicha plataforma es utilizada por el cliente en las demás aplicaciones que poseen, además tienen conocimiento pleno de esta plataforma.

Seguridad:

- Confiabilidad: la información manejada por el sistema debe estar protegida de acceso no autorizado y divulgación.
- Integridad: la información manejada por el sistema debe ser objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos.
- Disponibilidad: se les garantizará el acceso a la información solo a los usuarios autorizados evitando que los dispositivos o mecanismos utilizados para lograr la seguridad oculten o retrasen a los usuarios en la obtención de los datos deseados en un momento dado.

Todo esto se logra a través de la creación de grupos de usuarios los cuales tendrán asignados permisos de acción sobre cada información manejada por el sistema, para lo cual se requiere la autenticación del usuario. Si no se autentifica, es decir, no es un usuario reconocido por el sistema, no puede acceder al a mayoría de las opciones. Detallando aun más, entre las acciones a tener en cuenta para garantizar la seguridad se encuentran las siguientes:

1. La información será manejada únicamente por quien tenga los permisos suficientes para acceder a ella.
2. El sistema contará con protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

Interfaz:

Interfaz de software:

- Se realizará una aplicación de escritorio.
- La base de datos será independiente a la aplicación.

Confiabilidad:

- El sistema debe estar disponible todo el tiempo para trabajar de forma tal que se pueda acceder las 24 h. El sistema debe ser preciso en la información que le suministra al usuario para evitar cualquier tipo de error. El sistema debe estar bien documentado, para lograr que el tiempo de mantenimiento sea mínimo.

Ayuda y documentación:

- El sistema contará con la documentación completa de todas las tareas y operaciones que realiza el software, el glosario de términos y las planillas que especifican toda la Ingeniería de Software.
- Además tendrá una ayuda que garantice el asesoramiento e información al usuario acerca de los contenidos tratados. En esta deben quedar claramente reflejadas las funcionalidades del sistema y su manipulación.
- EL sistema requiere la construcción de un Manual de Usuario que describa detalladamente sus características y uso.

2.5 Modelo de Casos de Uso del Sistema

2.5.1 Actores del Sistema

Peso y Balance

Actor	Descripción
Tripulación	Reciben la Hoja de Peso y Balance para realizar sus vuelos. Pueden realizar búsquedas de Hoja de Peso y Balance que se hayan realizado con anterioridad.
Tripulación/PB	Son los responsable de gestionar los aviones, crew, versiones y galley. Son los que introducen los datos al sistema para obtener la Hoja de Peso y Balance.

Plan de Vuelo

Actor	Descripción
Tripulación/PV	Responsable de manejar los datos e introducirlos en el software con antelación al despegue. Son los responsables de cumplir con la planificación resultante del plan de vuelo.

2.5.2 Diagramas de Casos de Uso del Sistema

Peso y Balance

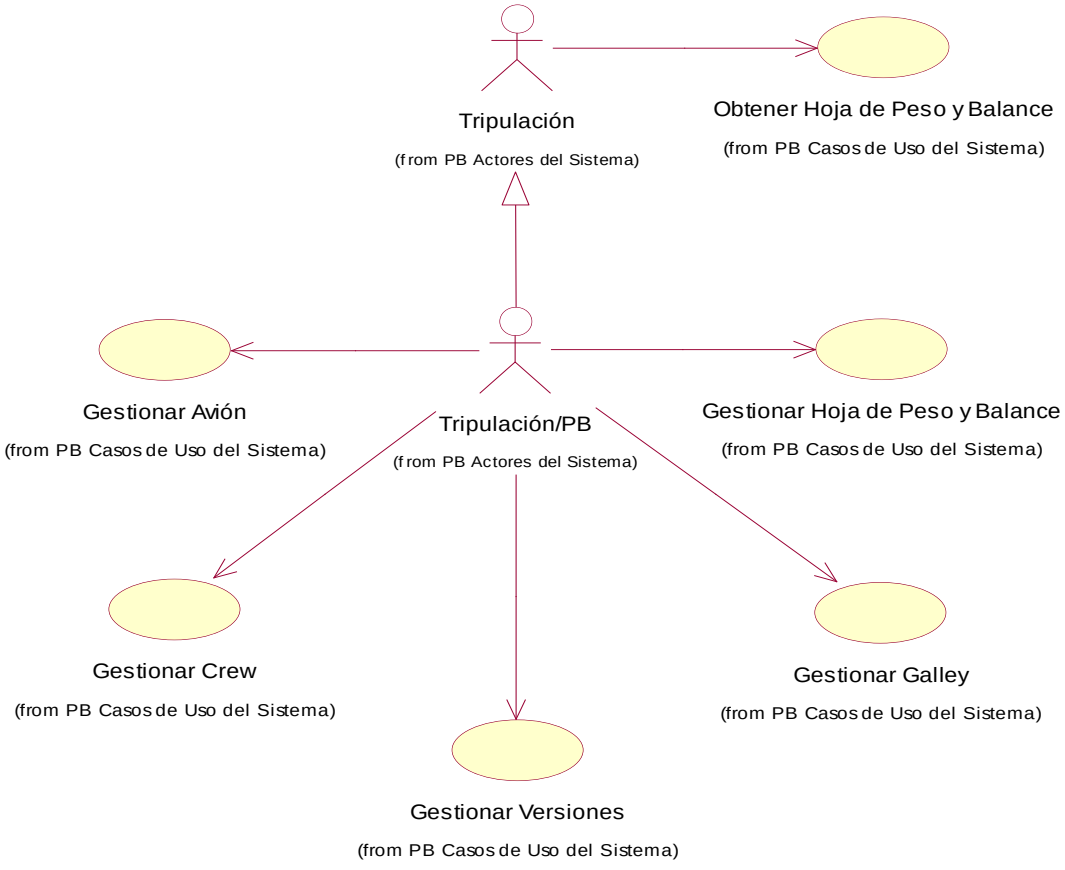


Fig. 13: Diagramas de Casos de Uso del Sistema Peso y Balance

Plan de Vuelo

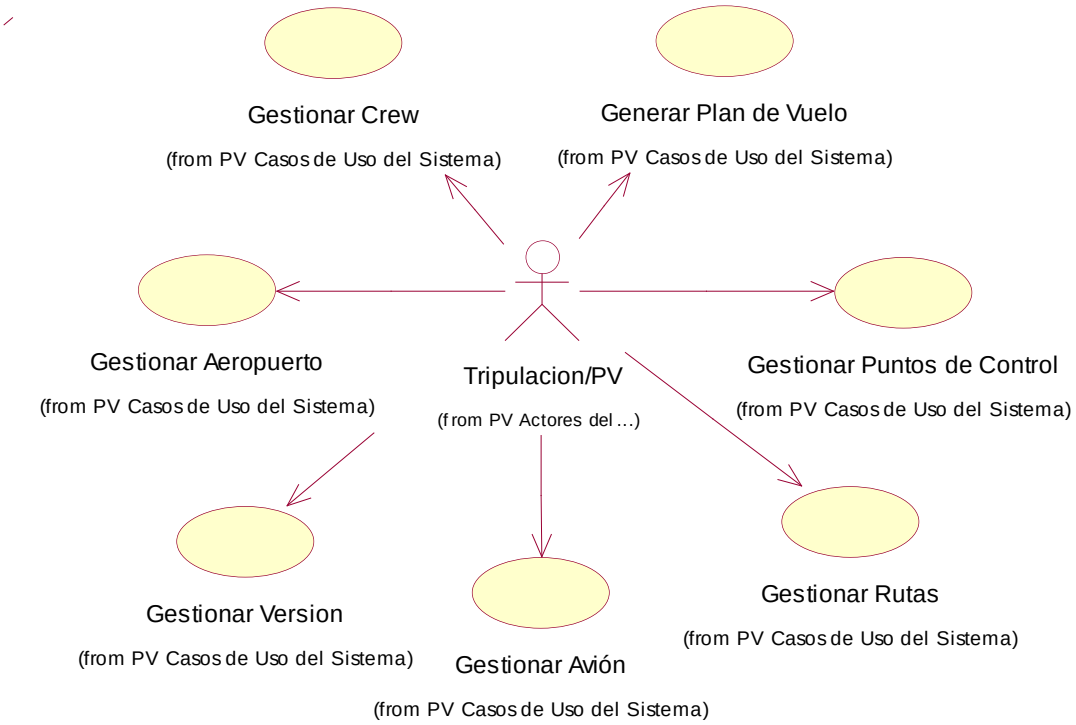


Fig. 14: Diagramas de Casos de Uso del Sistema Plan de Vuelo

2.5.3 Descripción de los Casos de Uso del Sistema

Descripción del Caso de Uso: **Gestionar Avión**

Caso de Uso:	**Gestionar Avión**
Actores:	**Tripulación/PB**
Resumen:	Agrupa funciones tales como adicionar o modificar un avión.
Referencia:	R4, R4.1, R4.2
CU asociados:	
Precondiciones:	Si lo que se desea es modificar, el avión sobre el que se va a trabajar debe estar creado con anterioridad.
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1- El usuario (Tripulación/PB) selecciona dentro de la interfaz principal de Peso y Balance la opción Gestionar Avión y dentro de ella la acción a realizar.	2- El sistema ejecuta alguna de las siguientes acciones: • Si decide adicionar un avión, ir a la sección “Adicionar”. • Si decide modificar los datos de un avión, ir a la sección "Modificar".
Sección “Adicionar”	
Acción del Actor	**Respuesta del Sistema**
3- El usuario (Tripulación/PB) introduce los datos del avión que desea adicionar.	4- El sistema verifica que todos los campos estén llenos.
	5- El sistema verifica que este avión no exista.
	6- El avión se almacena en el sistema.
	7- Se muestra un mensaje informándole al usuario que la operación ha sido realizada satisfactoriamente y finaliza el caso de uso.
Flujos Alternos	
	4- En caso de no haber llenado algún campo obligatorio se emite un mensaje para su llenado.
	5- Si el avión existe se muestra un mensaje informativo y finaliza el caso de uso.
Sección “Modificar”	
Acción del Actor	**Respuesta del Sistema**
	3- El sistema lee los datos de la Base de Datos y llena los campos de la interfaz de usuario correspondiente.
4- El usuario (Tripulación/PB) selecciona el avión a modificar y realiza las modificaciones deseadas.	5- Se verifica que los campos obligatorios estén llenos.

	6- Se actualiza la información y finaliza el caso de uso.
Flujos Alternos	
	5- En caso de no haber llenado algún campo obligatorio se emite un mensaje para su llenado.
Poscondiciones:	
Prioridad:	Crítico.
Especificaciones Complementaria:	

Descripción del Caso de Uso: **Gestionar Crew**

Caso de Uso:	**Gestionar Crew**
Actores:	**Tripulación/PB**
Resumen:	Agrupa funciones tales como adicionar o modificar un Crew.
Referencia:	R5, R5.1, R5.2
CU asociados:	
Precondiciones:	Si lo que se desea es modificar o eliminar, el Crew sobre el que se va a trabajar debe estar creado con anterioridad.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario (Tripulación/PB) selecciona dentro de la interfaz principal de Peso y Balance la opción Gestionar Crew y dentro de este la acción a realizar.	2- El sistema ejecuta alguna de las siguientes acciones: • Si decide adicionar un Crew, ir a la sección “Adicionar”. • Si decide modificar los datos de un Crew, ir a la sección "Modificar".
Sección “Adicionar”	
Acción del Actor	**Respuesta del Sistema**

3- El usuario (Tripulación/PB) introduce los datos del Crew que desea adicionar.	4- El sistema verifica que todos los campos estén llenos.
	5- El sistema verifica que este Crew no exista.
	6- El Crew se almacena en el sistema.
	7- Se muestra un mensaje informándole al usuario que la operación ha sido realizada satisfactoriamente y finaliza el caso de uso.
Flujos Alternos	
	4- En caso de no haber llenado algún campo obligatorio se emite un mensaje para su llenado.
	5- Si el Crew existe se muestra un mensaje informativo y finaliza el caso de uso.
Sección "Modificar"	
Acción del Actor	**Respuesta del Sistema**
	3- El sistema lee los datos de la Base de Datos y llena los campos de la interfaz de usuario correspondiente.
4- El usuario (Tripulación/PB) selecciona el Crew a modificar y realiza las modificaciones deseadas.	5- Se verifica que los campos obligatorios estén llenos.
	6- Se actualiza la información, se muestra un mensaje informando que la operación fue un éxito y finaliza el caso de uso.
Flujos Alternos	
	5- En caso de no haber llenado algún campo obligatorio se emite un mensaje para su llenado.
Poscondiciones:	
Prioridad:	Crítico.
Especificaciones	

Complementaria:	

Descripción de Caso de Uso: **Gestionar Galley**

Caso de Uso:	**Gestionar Galley**
Actores:	**Tripulación/PB**
Resumen:	El encargado de la tripulación de gestionar los Galley puede adicionar o modificar un Galley que no tenga vigencia.
Referencia:	R7, R7.1, R7.2
CU asociados:	
Precondiciones:	Si lo que se desea es modificar, el Galley sobre el que se va a trabajar debe estar creado con anterioridad.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario (Tripulación/PB) selecciona dentro de la interfaz principal de Peso y Balance la opción Gestionar Galley.	2- El sistema ejecuta alguna de las siguientes acciones: • Si decide adicionar un Galley, ir a la sección “Adicionar”. • Si decide modificar los datos de un Galley, ir a la sección "Modificar".
Sección “Adicionar”	
Acción del Actor	**Respuesta del Sistema**
3- El usuario (Tripulación/PB) introduce los datos del Galley que desea adicionar.	4- El sistema verifica que todos los campos estén llenos.
	5- El sistema verifica que este Galley no exista.
	6- El Galley se almacena en el sistema.
	7- Se muestra un mensaje informándole al usuario que la operación ha sido realizada satisfactoriamente y finaliza el caso de uso.

Flujos Alternos	
	4- En caso de no haber llenado algún campo obligatorio se emite un mensaje para su llenado.
	5- Si el Galley existe se muestra un mensaje informativo y finaliza el caso de uso.
Sección "Modificar"	
Acción del Actor	**Respuesta del Sistema**
	3- El sistema lee los datos de la Base de Datos y llena los campos de la interfaz de usuario correspondiente.
4- El usuario (Tripulación/PB) selecciona el Galley a modificar y realiza las modificaciones deseadas.	5- Se verifica que los campos obligatorios estén llenos.
	6- Se actualiza la información y finaliza el caso de uso.
Flujos Alternos	
	5- En caso de no haber llenado algún campo obligatorio se emite un mensaje para su llenado.
Poscondiciones:	
Prioridad:	Crítico.
Especificaciones Complementaria:	

Descripción del Caso de Uso: **Gestionar Versión**

Caso de Uso:	**Gestionar Versión**
Actores:	**Tripulación/PB**
Resumen:	El encargado de la tripulación de gestionar los Versiones puede adicionar o modificar una versión que no tenga vigencia.

Referencia:	R6, R6.1, R6.2.
CU asociados:	
Precondiciones:	Si lo que se desea es modificar, las versiones sobre el que se va a trabajar deben estar creadas con anterioridad.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario (Tripulación/PB) selecciona dentro de la interfaz principal de Peso y Balance la opción Gestionar Versiones.	2- El sistema ejecuta alguna de las siguientes acciones: • Si decide adicionar una versión, ir a la sección “Adicionar”.
Sección “Adicionar”	
Acción del Actor	**Respuesta del Sistema**
3- El usuario (Tripulación/PB) introduce los datos de la versión que desea adicionar.	4- El sistema verifica que todos los campos estén llenos.
	5- El sistema verifica que esta versión no exista.
	6- La versión se almacena en el sistema.
	7- Se muestra un mensaje informándole al usuario que la operación ha sido realizada satisfactoriamente y finaliza el caso de uso.
Flujos Alternos	
	4- En caso de no haber llenado algún campo obligatorio se emite un mensaje para su llenado.
	5- Si la versión existe se muestra un mensaje informativo y finaliza el caso de uso.
Sección “Modificar”	
Acción del Actor	**Respuesta del Sistema**
	3- El sistema lee los datos de la Base de Datos y llena los campos de la interfaz de usuario correspondiente.

4- El usuario (Tripulación/PB) selecciona la versión a modificar y realiza las modificaciones deseadas.	5- Se verifica que los campos obligatorios estén llenos.
	6- Se actualiza la información y finaliza el caso de uso.
Flujos Alternos	
	5- En caso de no haber llenado algún campo obligatorio se emite un mensaje para su llenado.
Poscondiciones:	
Prioridad:	Crítico.
Especificaciones Complementaria:	

Descripción del Caso de Uso: **Gestionar Hoja de Peso y Balance**

Caso de Uso:	**Gestionar Hoja de Peso y Balance**
Actores:	**Tripulación/PB**
Resumen:	Permite al usuario realizar el peso y balance del avión y mostrarlo, además de eliminar uno ya existente.
Referencia:	R2, R2.1, R2.2
CU asociados:	
Precondiciones:	El usuario debe estar registrado y autenticado. En el caso de eliminar alguna Hoja de Peso y Balance, debe estar creada con anterioridad.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario (Tripulación/PB) selecciona dentro de la interfaz de Peso y Balance la opción Gestionar Peso y Balance.	2- El sistema ejecuta alguna de las siguientes acciones: • Si decide realizar el peso y balance del avión, ir a la sección “Realizar Peso y Balance”. • Si decide eliminar una Hoja de Peso y Balance,

	ir a la sección " Eliminar Peso y Balance".
Sección "Realizar Peso y Balance"	
Acción del Actor	**Respuesta del Sistema**
3- El usuario (Tripulación/PB) desea Realizar Hoja de Peso y Balance.	4- El sistema muestra un formulario donde el usuario colocara los pasajeros y la carga de forma balanceada.
5- El usuario llena el formulario.	6- El sistema verifica que todos los datos hayan sido llenados y verifica que sean válidos.
	7- El sistema realiza los cálculos del %MAC para el ZFW, el LW y el TOW.
	8- El sistema Muestra la Hoja de Peso y Balance utilizando los cálculos realizados además de información contenida en la Base de Datos.
	9- El sistema guarda la Hoja de Peso y Balance.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	6- En el caso de existir algún error en la entrada de datos el sistema muestra un mensaje y regresa al paso anterior.
Sección "Eliminar Peso y Balance"	
Acción del Actor	**Respuesta del Sistema**
3- El usuario desea Eliminar Hoja de Peso y Balance	
	4- El sistema Muestra las hojas de Peso y Balance guardadas con antelación.
5- El usuario selecciona La hoja de Peso y Balance que desea Eliminar.	6- El sistema Elimina la Hoja de Peso y Balance.
Flujos Alternos	
	4- No existe ninguna hoja de Peso y Balance guardada en el sistema, muestra un mensaje y se

	termina el caso de uso.
Poscondiciones:	
Prioridad:	Crítico.
Especificaciones Complementaria:	

Descripción del Caso de Uso: **Obtener Hoja de Peso y Balance**

Caso de Uso:	**Obtener Hoja de Peso y Balance**	
Actores:	**Tripulación**	
Resumen:	El caso de uso se inicia cuando la tripulación desea obtener una Hoja de Peso y Balance previamente elaborado y termina cuando el sistema le muestra dicho documento.	
Referencia:	R3, R3.1	
CU asociados:		
Precondiciones:	La Hoja de Peso y Balance debe estar previamente elaborada.	
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
1- El usuario (Tripulación) selecciona dentro de la interfaz principal de Peso y Balance la opción de Obtener Peso y Balance.		2- El sistema lee los valores de la Base de Datos y llena los campos.
3- El usuario (Tripulación) selecciona los datos necesarios para la búsqueda, como son la fecha y el vuelo.		4- El sistema muestra las Hojas de Peso y Balance que existan con las características deseadas.
5- El usuario selecciona la Hoja de Peso y Balance que desea ver.		6- El sistema muestra la Hoja de Peso y Balance solicitada.
Poscondiciones:		
Prioridad:	Crítico.	
Especificaciones Complementaria:		

Descripción del Caso de Uso: **Gestionar Punto de Control**

Caso de Uso:	**Gestionar Puntos de Control**
Actores:	**Tripulación/PV**
Resumen:	Permite Gestionar la información referente a los puntos de Control, registrar nuevos Puntos de Control y Modificar la información de algún Punto de Control determinado.
Referencia:	R9, R9.1, R9.2
CU asociados:	
Precondiciones:	El Tripulante encargado de realizar la operación debe estar registrado y autenticado en el sistema, si la acción es modificar, dicho Punto de Control debe haber sido creado con anterioridad.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario selecciona dentro de la interfaz de Plan de Vuelo la opción Gestionar Puntos de Control.	2- El sistema ejecuta alguna de las siguientes acciones: • Si decide adicionar un Punto de Control, ir a la sección “Adicionar”. • Si decide modificar los datos de un Punto de Control, ir a la sección "Modificar".
Sección “Adicionar”	
Acción del Actor	**Respuesta del Sistema**
2- El Administrador del Sistema introduce los datos del Punto de Control que desea adicionar.	3- El sistema verifica que todos los campos estén llenos.
	4- El sistema verifica que este Punto de Control sea válido y no exista en el sistema.
	5- El Punto de Control se almacena en el sistema.
	6- Se muestra un mensaje informándole al usuario que la operación ha sido realizada

	satisfactoriamente y finaliza el caso de uso.
Flujos Alternos	
	4- Se emite un mensaje para que llene los campos obligatorios.
	5- Si el Punto de Control no es válido o existe se muestra un mensaje informativo y finaliza el caso de uso.
Sección "Modificar"	
Acción del Actor	**Respuesta del Sistema**
3- El usuario selecciona si va a modificar un Punto de Control.	4- El sistema lee los datos de la Base de Datos y llena los campos de la interfaz de usuario.
5- El usuario realiza las modificaciones deseadas.	6- Se verifica que los campos obligatorios estén llenos.
	7- Se actualiza la información y finaliza el caso de uso.
Flujos Alternos	
	7- Se emite un mensaje para que llene los campos obligatorios.
Poscondiciones:	
Prioridad:	Crítico.
Especificaciones Complementaria:	

Descripción del Caso de Uso: **Gestionar Ruta**

Caso de Uso:	**Gestionar Ruta**
Actores:	**Tripulación/PV**
Resumen:	Permite Gestionar la información referente a las Rutas de Vuelo, registrar nuevas Rutas y Modificar la información de alguna Ruta determinada.
Referencia:	R10, R10.1, R10.2

CU asociados:	
Precondiciones:	El usuario debe estar registrado y autenticado. Si lo que se desea es modificar, la ruta sobre el que se va a trabajar debe estar creada con anterioridad.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario selecciona dentro de la interfaz de Plan de Vuelo la opción de Gestionar Rutas.	2- El sistema ejecuta alguna de las siguientes acciones: • Si decide adicionar una ruta, ir a la sección “Adicionar”. • Si decide modificar los datos de una ruta, ir a la sección "Modificar".
Sección “Adicionar”	
Acción del Actor	**Respuesta del Sistema**
2- El Administrador del Sistema introduce los datos de la ruta que desea adicionar.	3- El sistema verifica que todos los campos estén llenos.
	4- El sistema verifica que esta ruta sea valida y no exista.
	5- La ruta se almacena en el sistema.
	6- Se muestra un mensaje informándole al usuario que la operación ha sido realizada satisfactoriamente y finaliza el caso de uso.
Flujos Alternos	
	4- Se emite un mensaje para que llene los campos obligatorios.
	5- Si la ruta no es válida o existe se muestra un mensaje informativo y finaliza el caso de uso.
Sección “Modificar”	
Acción del Actor	**Respuesta del Sistema**
3- El usuario selecciona si va a modificar una ruta.	4- El sistema lee los datos de la Base de Datos y llena los campos de la interfaz de usuario.

5- El usuario realiza las modificaciones deseadas.	6- Se verifica que los campos obligatorios estén llenos.
	7- Se actualiza la información y finaliza el caso de uso.
Flujos Alternos	
	7- Se emite un mensaje para que llene los campos obligatorios.
Poscondiciones:	
Prioridad:	Crítico.
Especificaciones Complementaria:	

Descripción del Caso de Uso: **Generar Plan de Vuelo**

Caso de Uso:	**Generar Plan de Vuelo**
Actores:	**Tripulante/PV**
Resumen:	Permite construir el Plan de Vuelo
Referencia:	R8, R8.1
CU asociados:	
Precondiciones:	El usuario debe estar registrado y autenticado. El sistema debe tener almacenadas las rutas, el avión y los aeropuertos de origen y destino.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario selecciona dentro de la interfaz de Plan de Vuelo la opción de Generar Plan de Vuelo.	2- El sistema carga los datos necesarios de la base de datos.
	3- El sistema llena los campos necesarios.
4- El usuario selecciona los datos y llena los restantes campos.	5- El sistema valida los datos.

	6- El sistema realiza los cálculos y genera el Plan de Vuelo.
Flujos Alternos	
	5- El sistema encuentra algún error, e informa que no se pudo realizar el Plan de Vuelo.
Poscondiciones:	
Prioridad:	Crítico.
Especificaciones Complementaria:	

Descripción del Caso de Uso: **Gestionar Avión**

Caso de Uso:	**Gestionar Avión**
Actores:	**Tripulante/PV**
Resumen:	Permite la introducción de un nuevo avión al sistema, la persona encargada por parte de la tripulación de realizar la operación puede modificar la información referente a un avión determinado.
Referencia:	R4, R4.1, R4.2
CU asociados:	
Precondiciones:	El usuario debe estar registrado y autenticado. Si lo que se desea es modificar, el avión sobre el que se va a trabajar debe estar creado con anterioridad.
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario selecciona dentro de la interfaz de Plan de Vuelo la opción de Gestionar Avión.	2- El sistema ejecuta alguna de las siguientes acciones: • Si decide adicionar un Avión, ir a la sección “Adicionar”. • Si decide modificar los datos de un avión, ir a la sección "Modificar".
Sección “Adicionar”	

Acción del Actor	Respuesta del Sistema
2- El Administrador del Sistema introduce los datos del avión que desea adicionar.	3- El sistema verifica que todos los campos estén llenos.
	4- El sistema verifica que este avión sea valido y no exista.
	5- El avión se almacena en el sistema.
	6- Se muestra un mensaje informándole al usuario que la operación ha sido realizada satisfactoriamente y finaliza el caso de uso.
Flujos Alternos	
	4- Se emite un mensaje para que llene los campos obligatorios.
	5- Si el avión no es valido o existe se muestra un mensaje informativo y finaliza el caso de uso.
Sección "Modificar"	
Acción del Actor	**Respuesta del Sistema**
3- El usuario selecciona si va a modificar un avión.	4- El sistema lee los datos de la Base de Datos y llena los campos de la interfaz de usuario.
5- El usuario realiza las modificaciones deseadas.	6- Se verifica que los campos obligatorios estén llenos.
	7- Se actualiza la información y finaliza el caso de uso.
Flujos Alternos	
	7- Se emite un mensaje para que llene los campos obligatorios.
Poscondiciones:	
Prioridad:	Crítico.
Especificaciones Complementaria:	

Descripción del Caso de Uso: **Gestionar Aeropuerto**

Caso de Uso:	**Gestionar Aeropuerto**
Actores:	**Tripulante/PV**

Resumen:	Permite la introducción de un nuevo aeropuerto al sistema, la persona encargada por parte de la tripulación de realizar la operación puede modificar la información referente a un aeropuerto determinado.
Referencia:	R1, R1.1, R1.2
CU asociados:	
Precondiciones:	El usuario debe estar registrado y autenticado. Si lo que se desea es modificar, el aeropuerto sobre el que se va a trabajar debe estar creado con anterioridad.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El usuario selecciona dentro de la interfaz de Plan de Vuelo la opción de Gestionar Avión.	2- El sistema ejecuta alguna de las siguientes acciones: • Si decide adicionar un Aeropuerto, ir a la sección “Adicionar”. • Si decide modificar los datos de un aeropuerto, ir a la sección "Modificar".
Sección “Adicionar”	
Acción del Actor	**Respuesta del Sistema**
2- El Administrador del Sistema introduce los datos del aeropuerto que desea adicionar.	3- El sistema verifica que todos los campos estén llenos.
	4- El sistema verifica que este aeropuerto sea valido y no exista.
	5- El aeropuerto se almacena en el sistema.
	6- Se muestra un mensaje informándole al usuario que la operación ha sido realizada satisfactoriamente y finaliza el caso de uso.
Flujos Alternos	
	4- Se emite un mensaje para que llene los campos obligatorios.
	5- Si el aeropuerto no es valido o existe se muestra un mensaje informativo y finaliza el caso de uso.
Sección “Modificar”	
Acción del Actor	**Respuesta del Sistema**

3- El usuario selecciona si va a modificar un aeropuerto.	4- El sistema lee los datos de la Base de Datos y llena los campos de la interfaz de usuario.
5- El usuario realiza las modificaciones deseadas.	6- Se verifica que los campos obligatorios estén llenos.
	7- Se actualiza la información y finaliza el caso de uso.
Flujos Alternos	
	7- Se emite un mensaje para que llene los campos obligatorios.
Poscondiciones:	
Prioridad:	Crítico.
Especificaciones Complementaria:	

Conclusiones

En este capítulo se abordaron temas de suma importancia para lograr un mayor entendimiento del negocio actual pues se explicaron detalladamente el flujo de procesos que lo componen, además de aspectos referentes a la propuesta del sistema a construir, enumerándose los requerimientos del sistema y las funcionalidades que debe brindar.

Capítulo 3: Análisis y Diseño del Sistema.

Introducción

En este capítulo se hace el análisis y diseño de la propuesta de solución, modelándose los artefactos que intervienen en la comprensión de los pasos para implementar dicha propuesta mediante el uso del UML. El análisis permite adentrarse en el problema a resolver, en él se representa una vista interna del sistema en la que, usando el lenguaje de los desarrolladores se analizan con mayor profundidad los requisitos, refinándolos y estructurándolos en base a clases y paquetes, proporcionando una visión general del sistema. Por otro lado el diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales. Se realizarán los diferentes artefactos que se generan en cada uno de estos flujos de trabajo como son: los diagramas de clases e interacción del Análisis y los diagramas de clases e interacción del Diseño.

3.1 Análisis

Diagramas de Clases del Análisis.

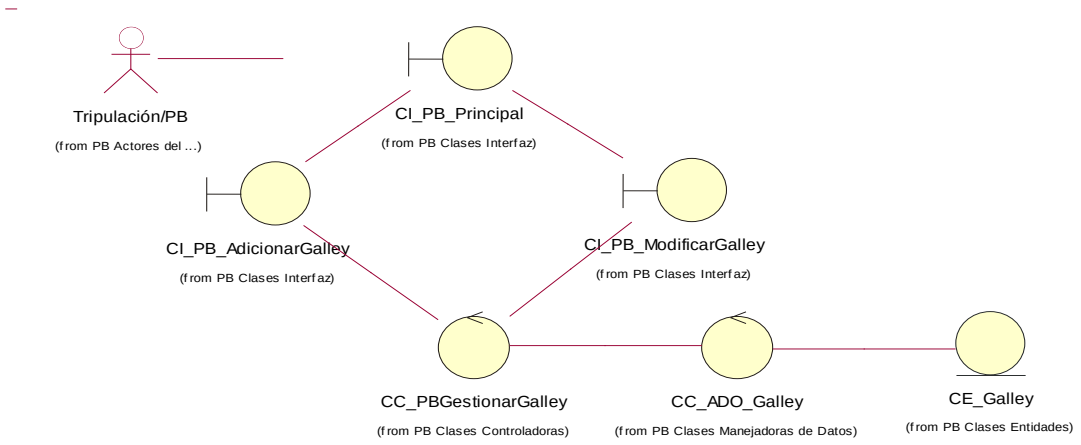


Fig. 15: Diagrama de Clases del Análisis CU Gestionar Galley.

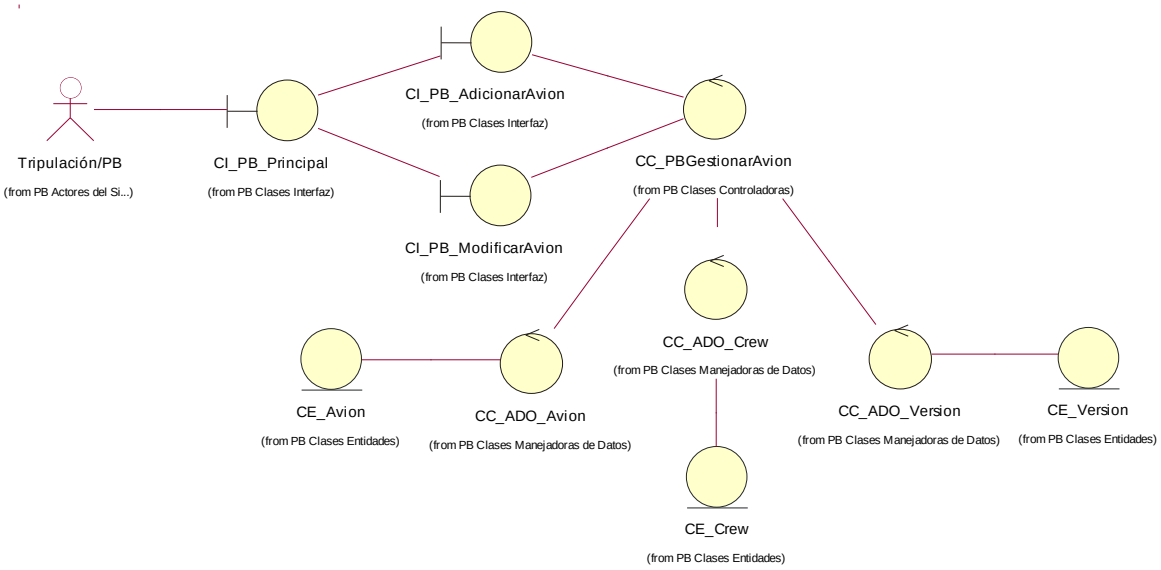


Fig. 16: Diagrama de Clases del Análisis CU Gestionar Avión.

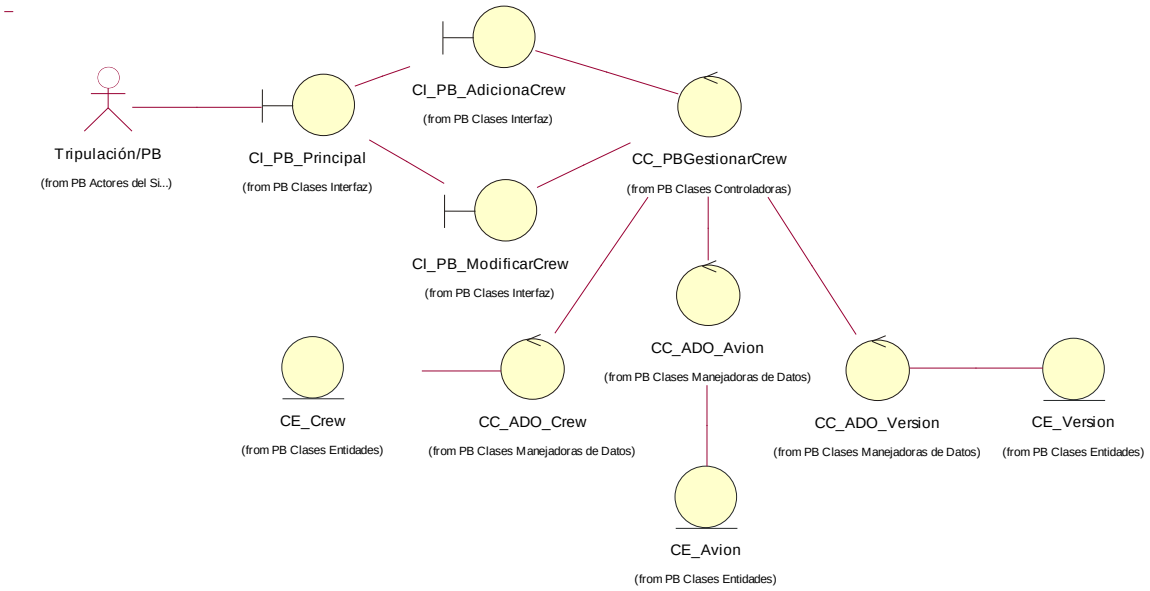


Fig. 17: Diagrama de Clases del Análisis CU Gestionar Crew.

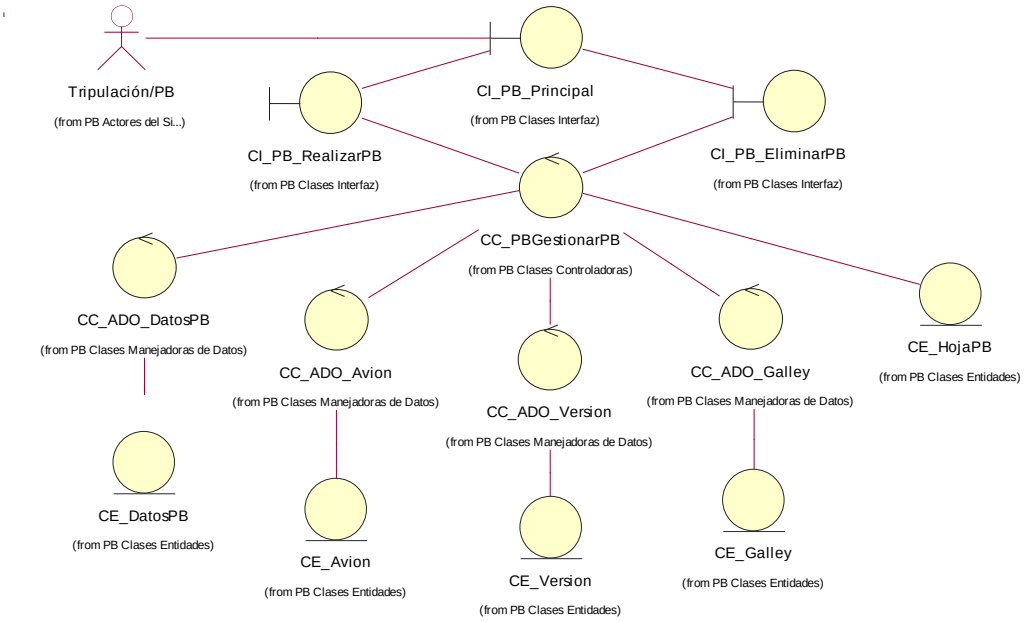


Fig. 18: Diagrama de Clases del Análisis CU Gestionar Hoja de Peso y Balance.

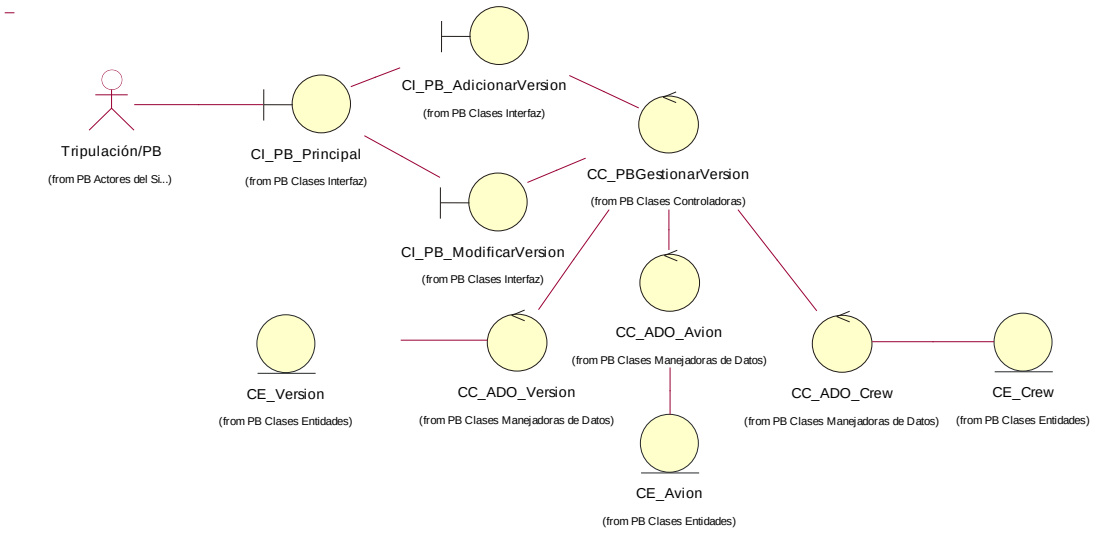


Fig. 19: Diagrama de Clases del Análisis CU Gestionar Versión.

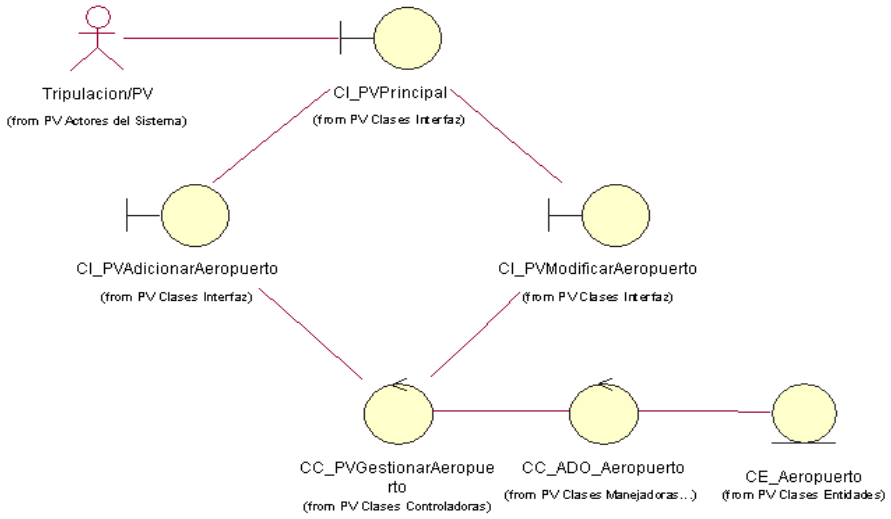


Fig. 20: Diagrama de Clases del Análisis CU Gestionar Aeropuerto

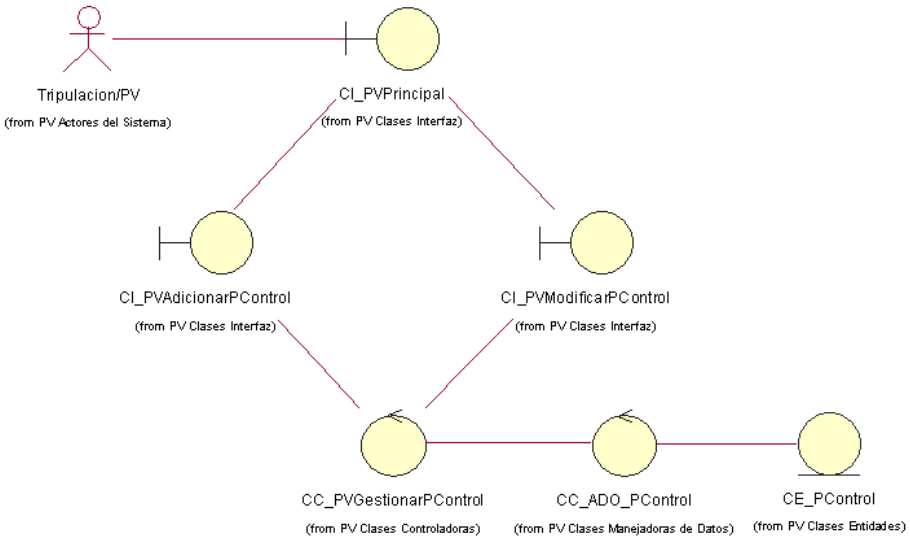


Fig. 21: Diagrama de Clases del Análisis CU Gestionar Punto de Control.

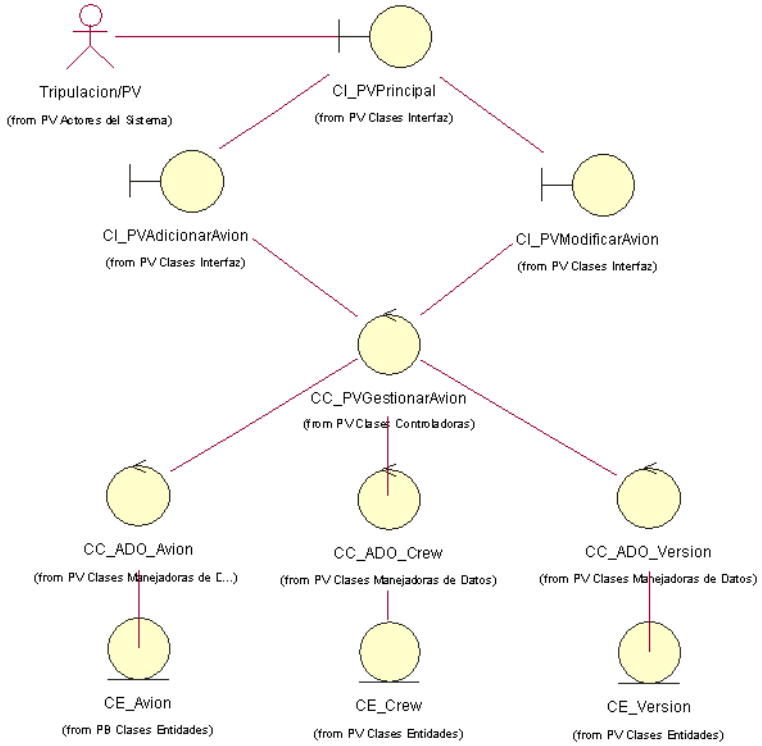


Fig. 22: Diagrama de Clases del Análisis CU Gestionar Avión.

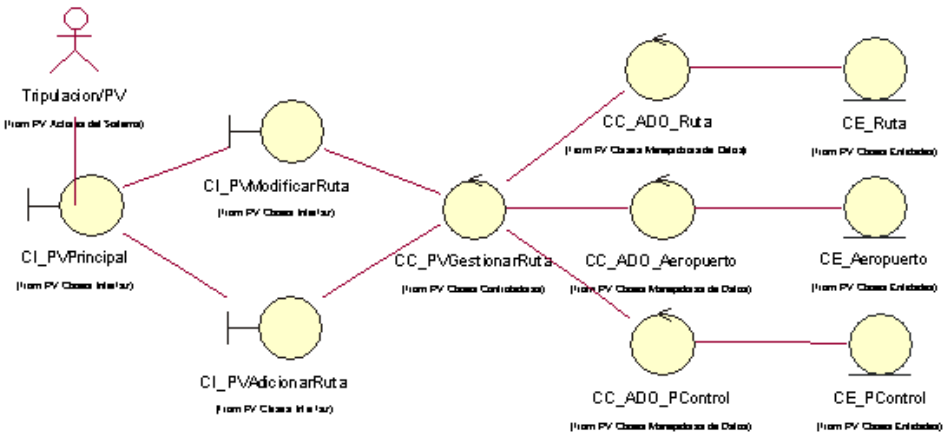


Fig. 23: Diagrama de Clases del Análisis CU Gestionar Rutas.

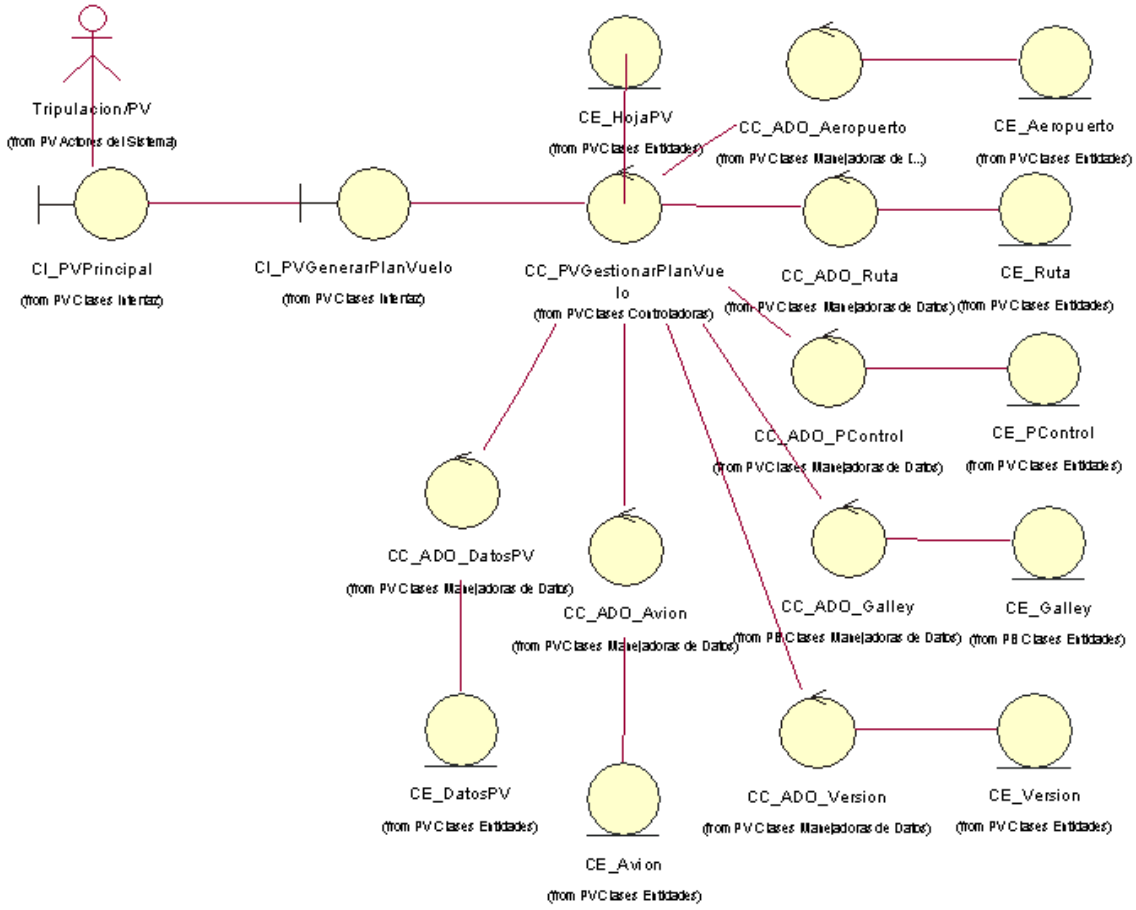


Fig. 24: Diagrama de Clases del Análisis CU Gestionar Plan de Vuelo.

3.2 Diseño

Diagramas de Clases del Diseño

Peso y Balance

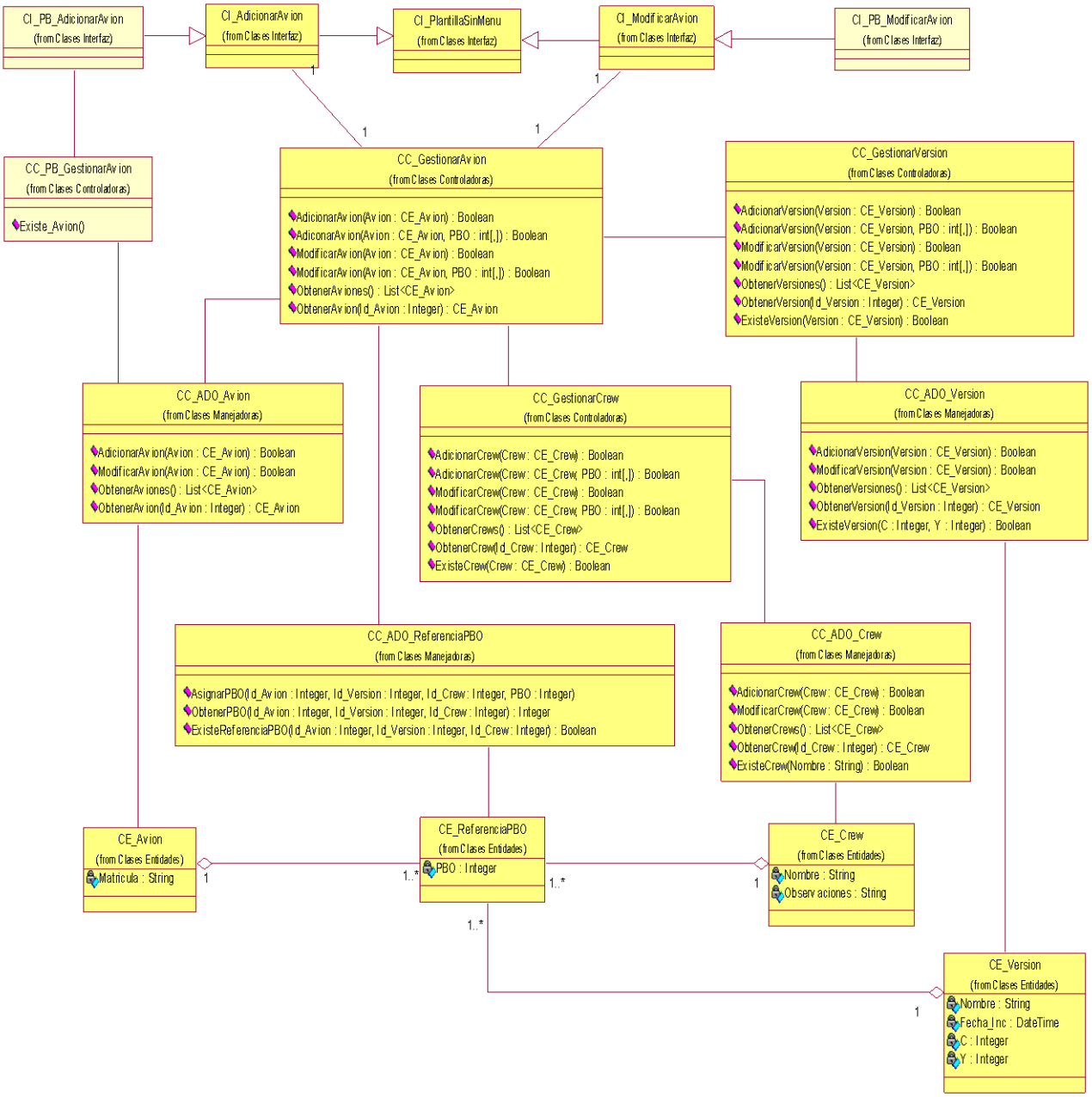


Fig. 25: Diagrama de Clases del Diseño CU Gestionar Avión.

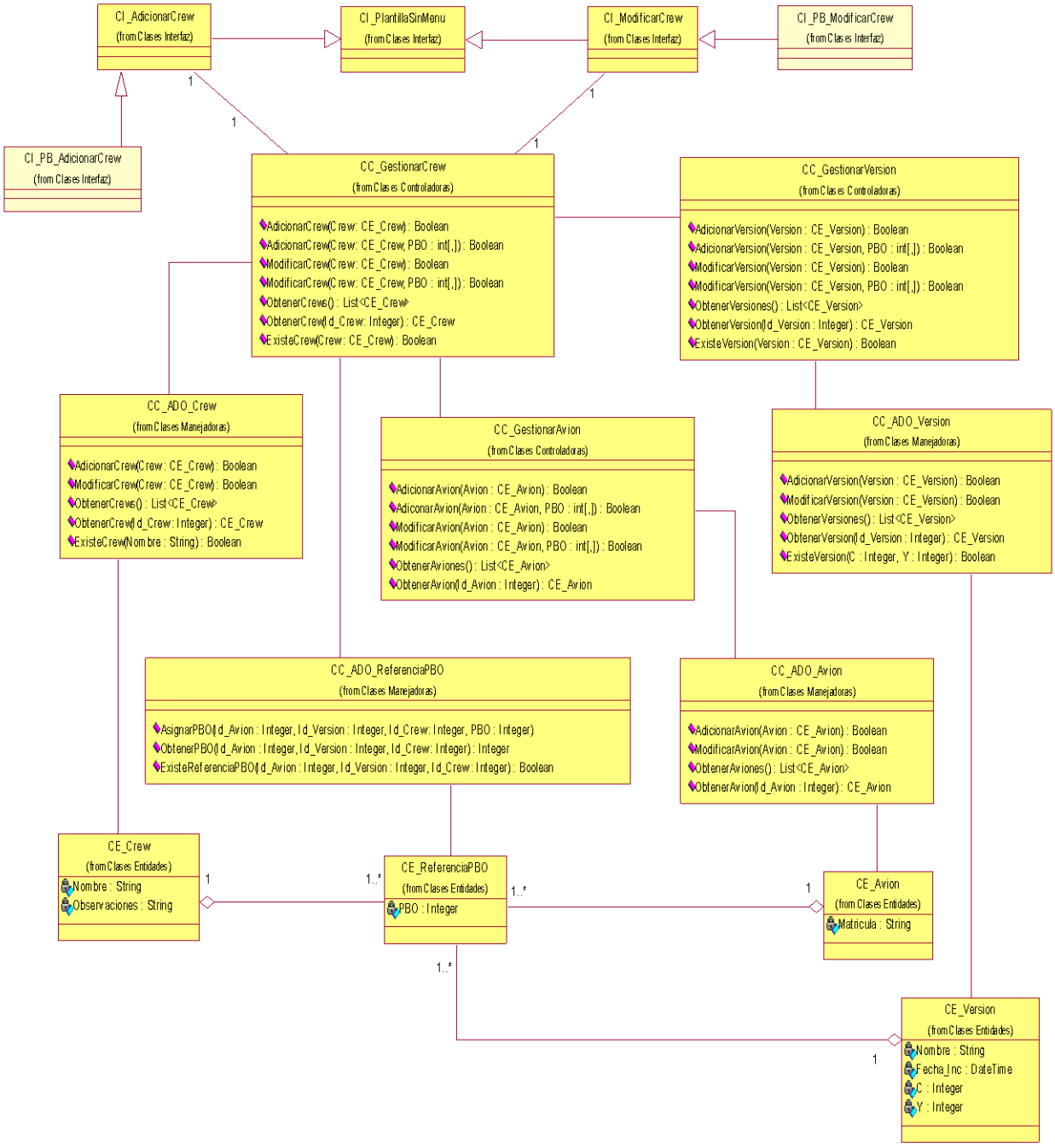


Fig. 26: Diagrama de Clases del Diseño CU Gestionar Crew.

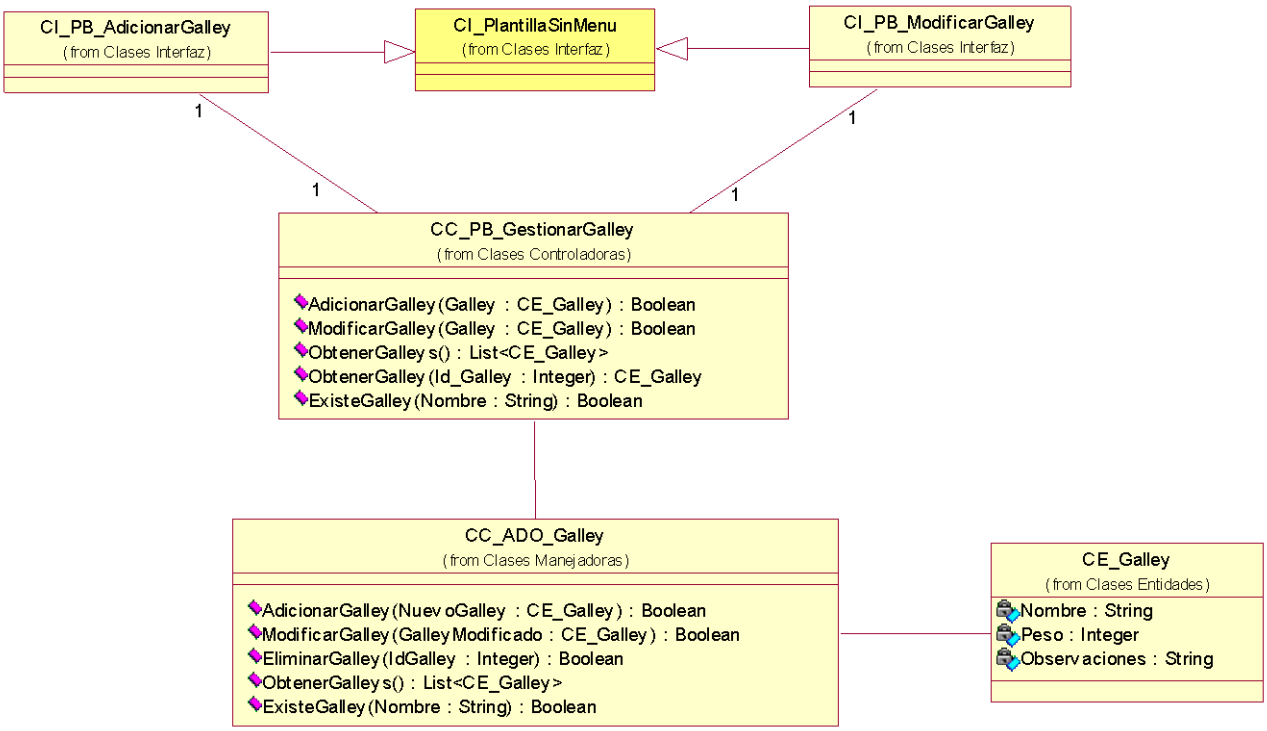


Fig. 27: Diagrama de Clases del Diseño CU Gestionar Galley

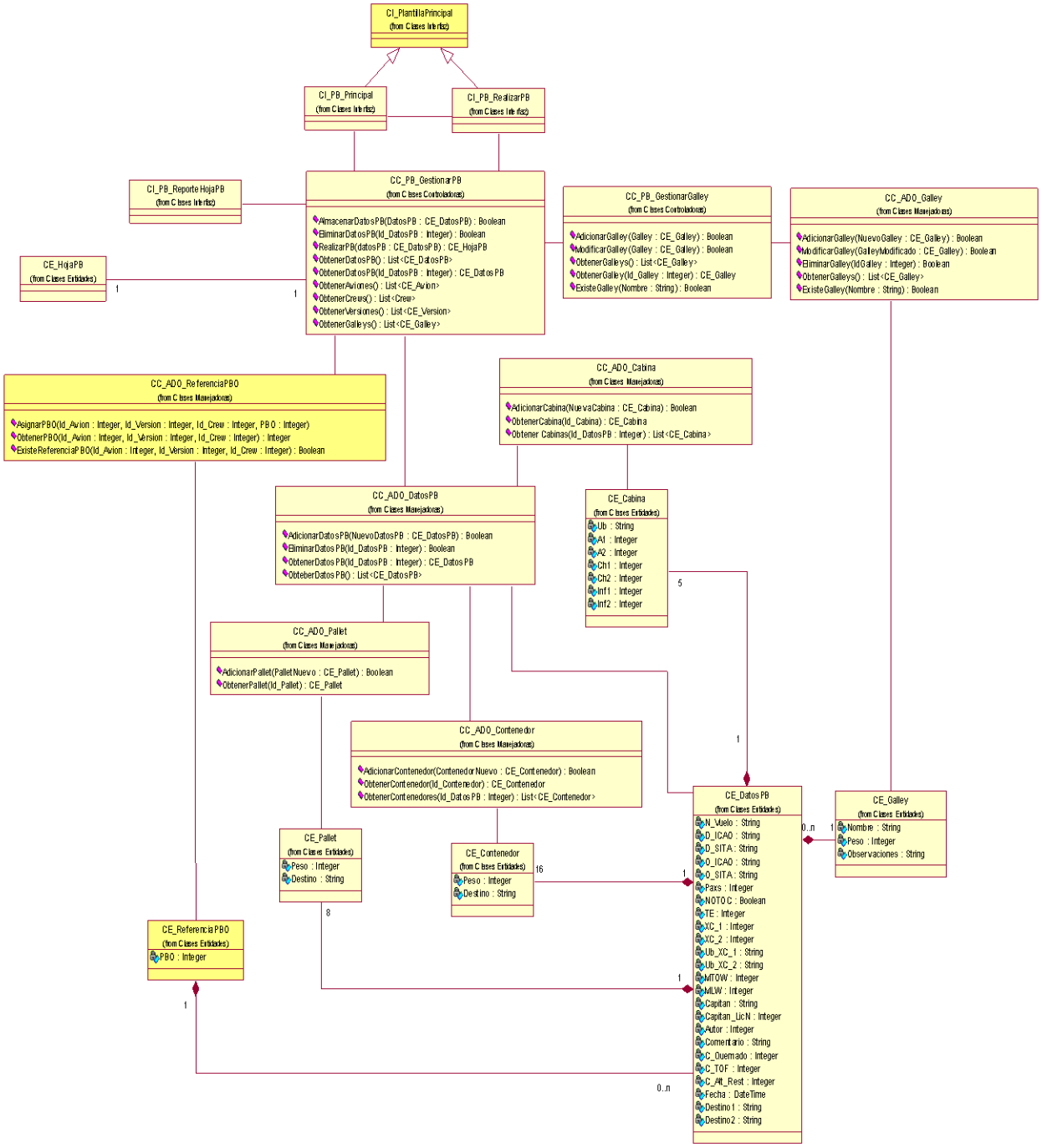


Fig. 28: Diagrama de Clases del Diseño CU Gestionar Hoja Peso y Balance

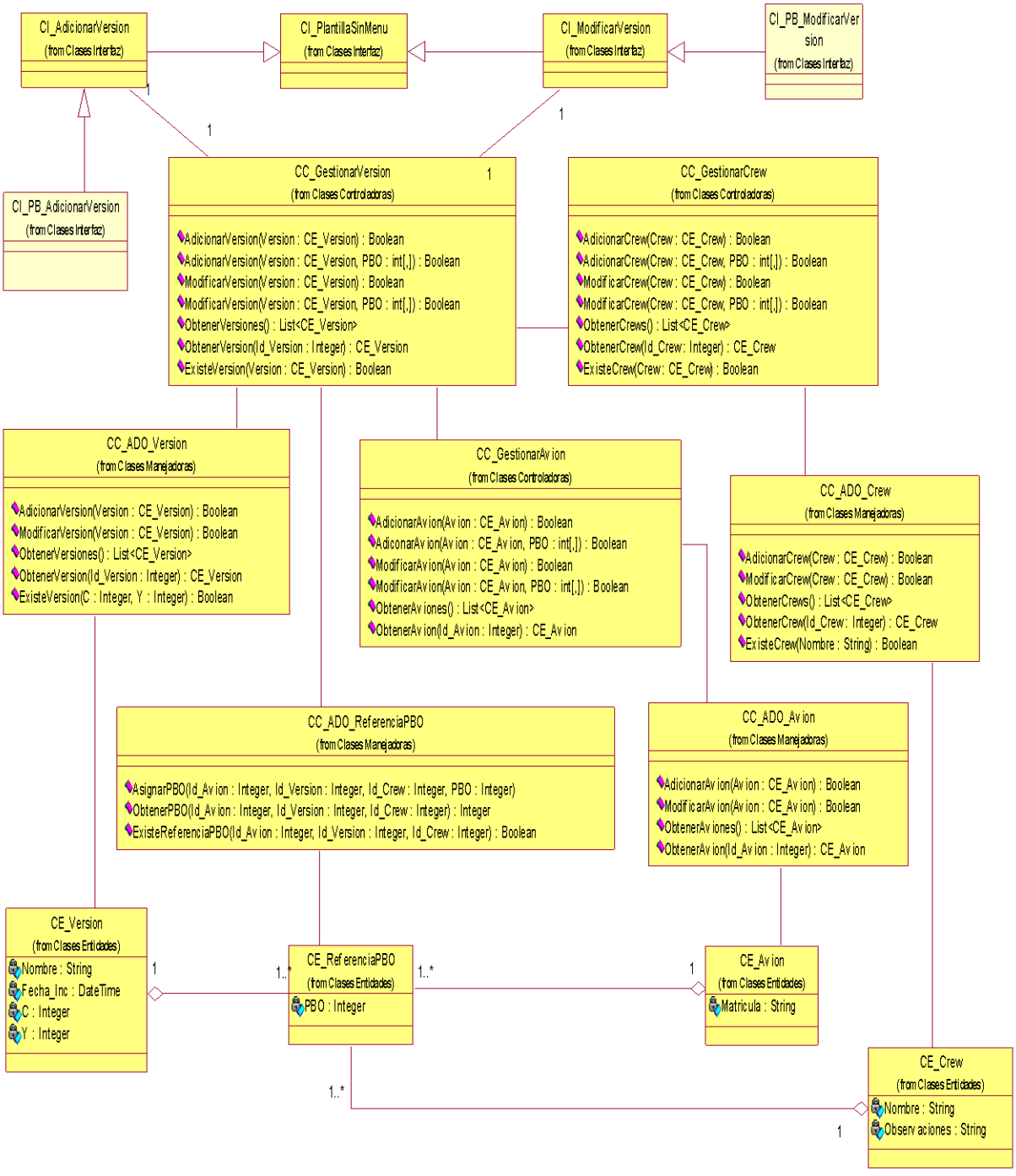


Fig. 29: Diagrama de Clases del Diseño CU Gestionar Versión

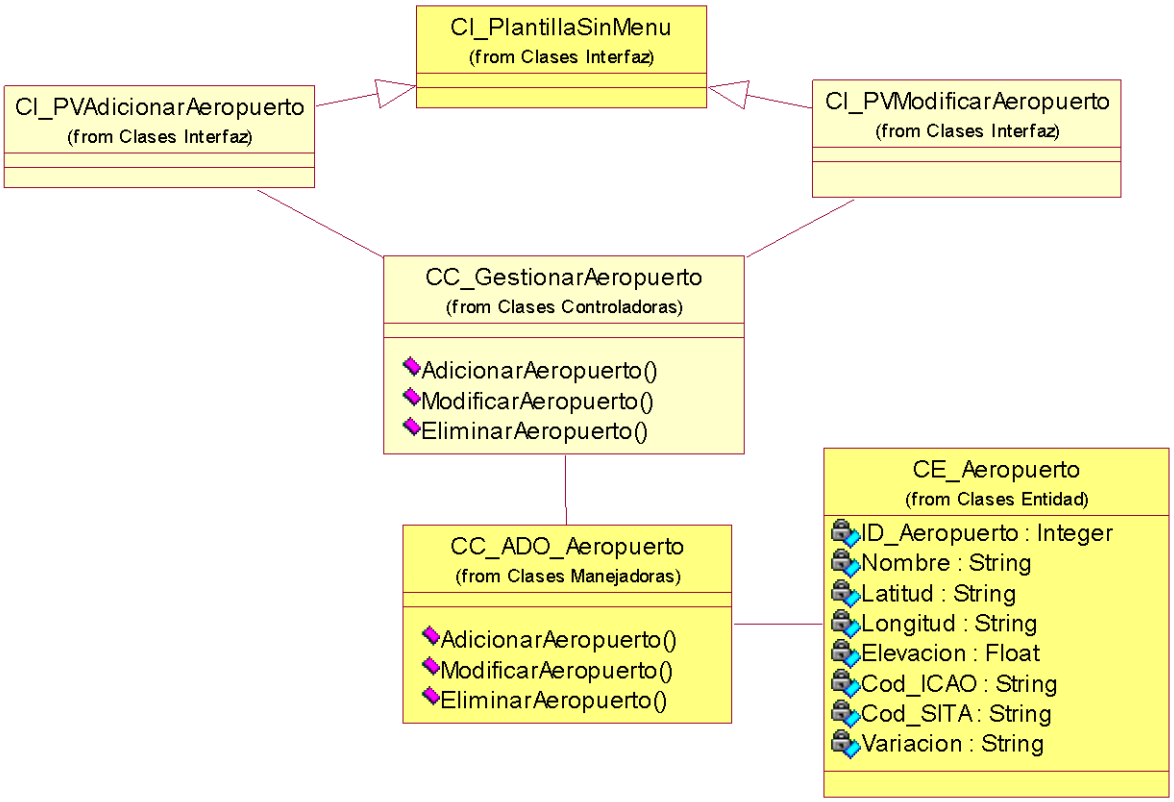


Fig. 30: Diagrama de Clases del Diseño CU Gestionar Aeropuerto

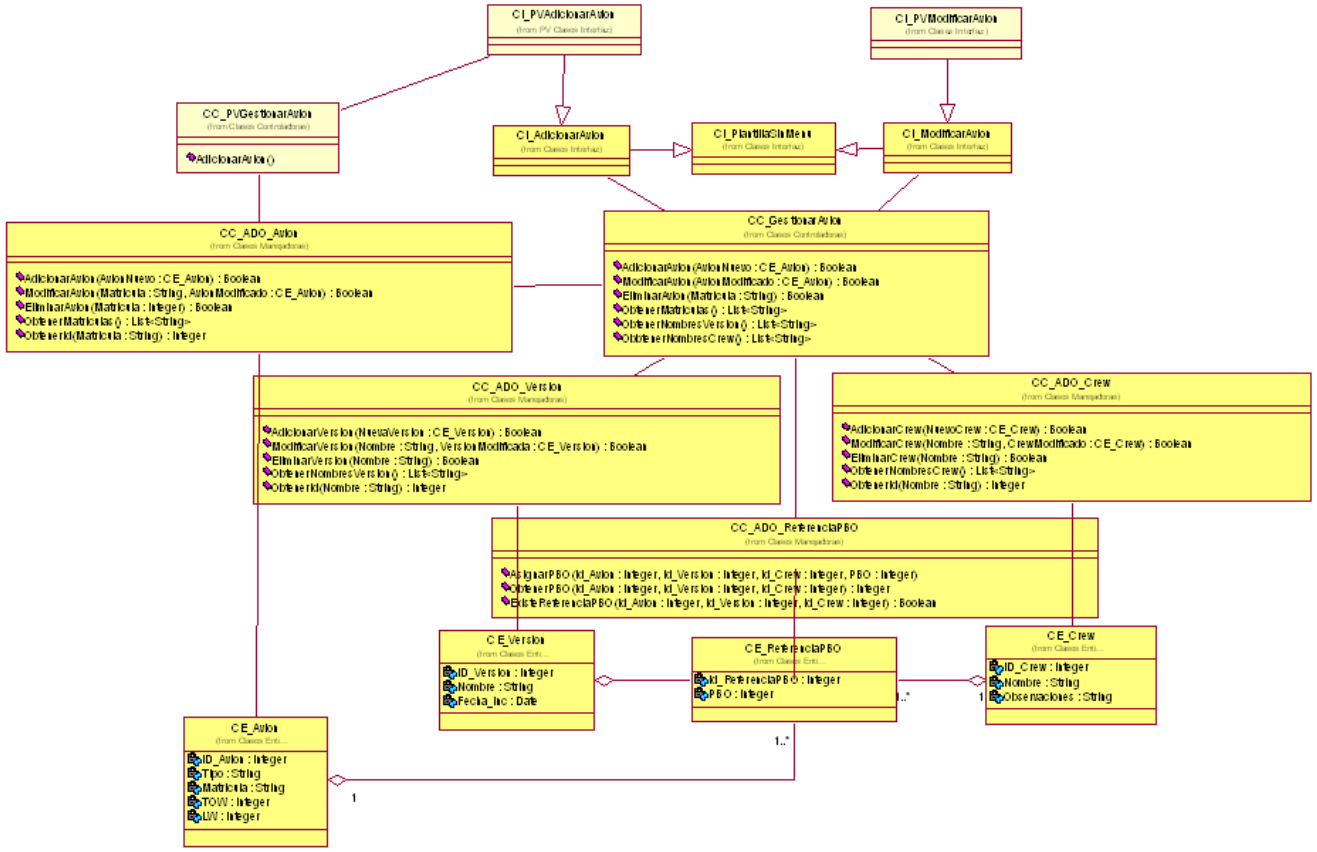


Fig. 31: Diagrama de Clases del Diseño CU Gestionar Avión

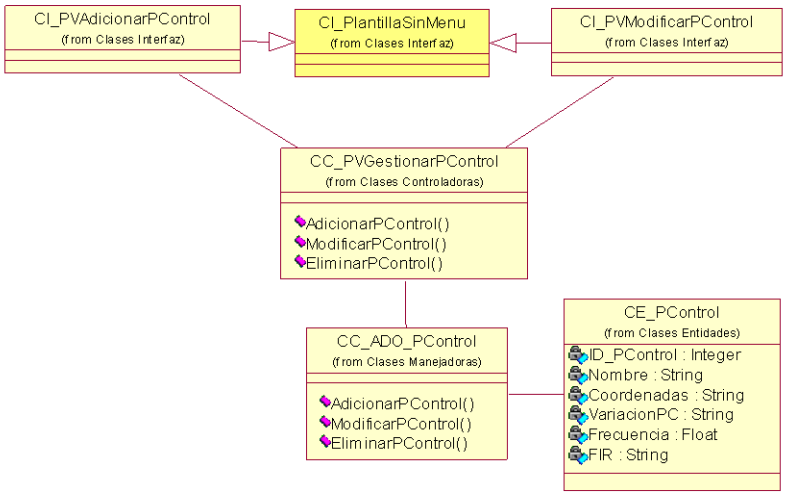


Fig. 32: Diagrama de Clases del Diseño CU Gestionar Punto de Control

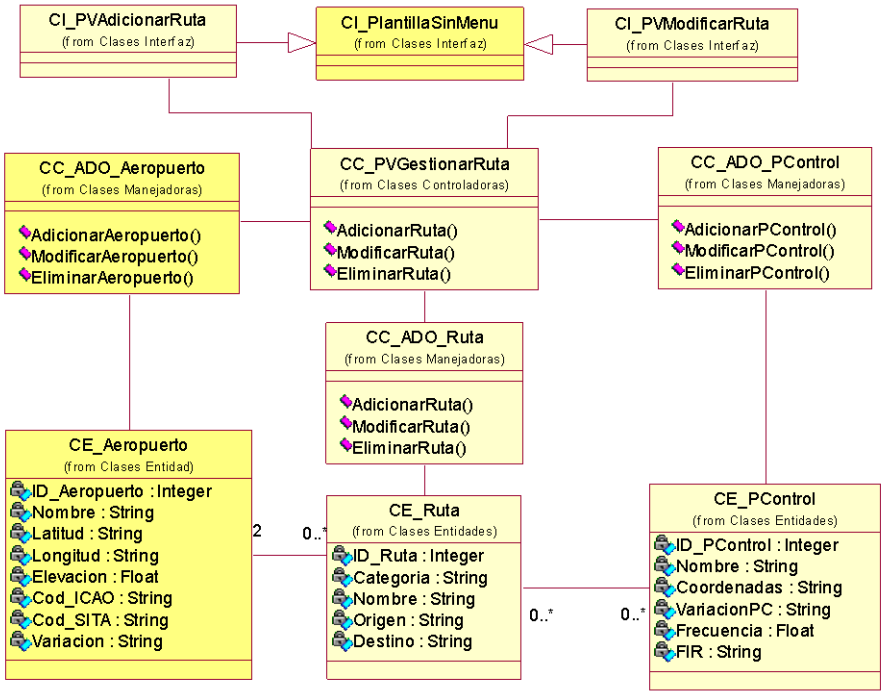


Fig. 33: Diagrama de Clases del Diseño CU Gestionar Ruta.

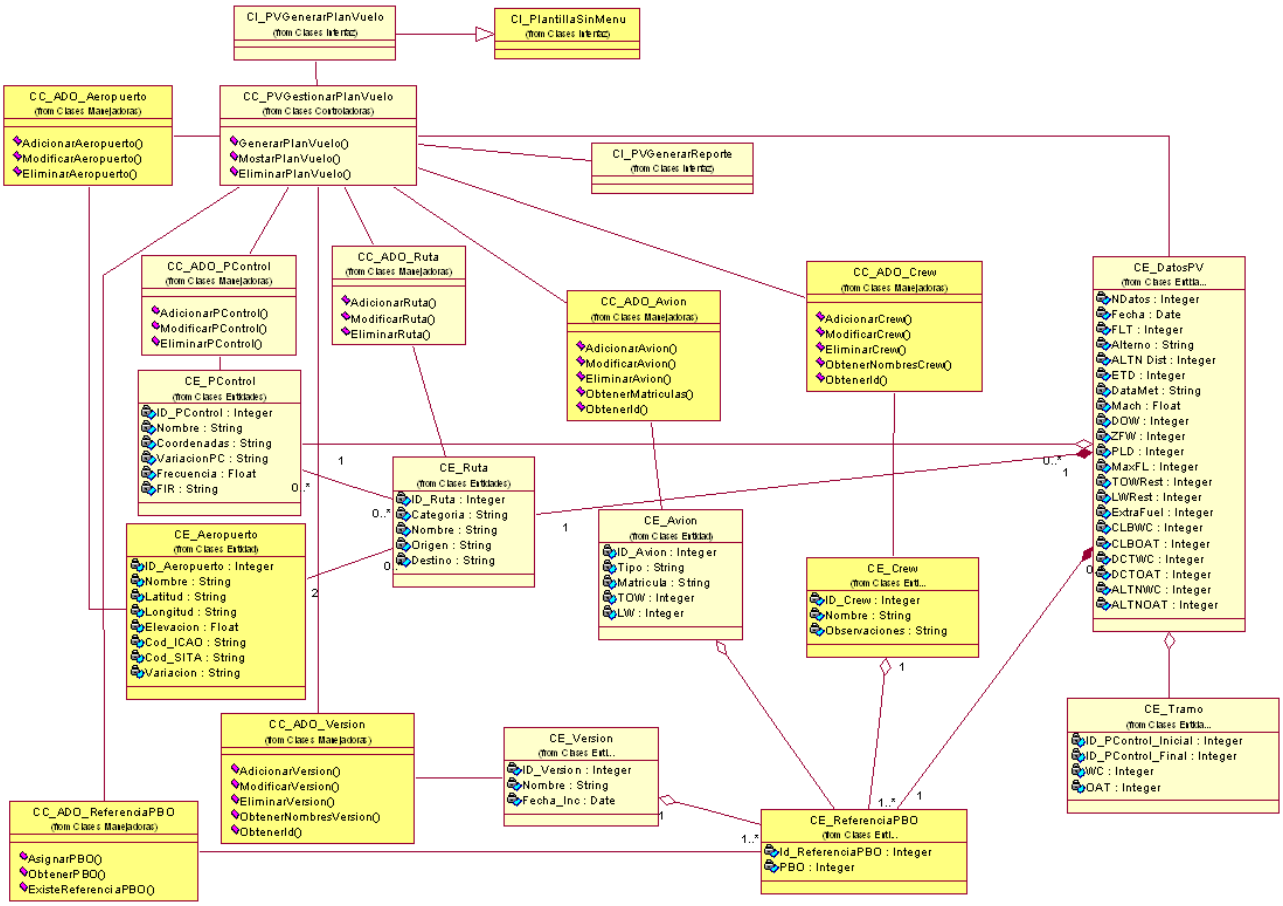


Fig. 34 : Diagrama de Clases del Diseño CU Generar Plan de Vuelo

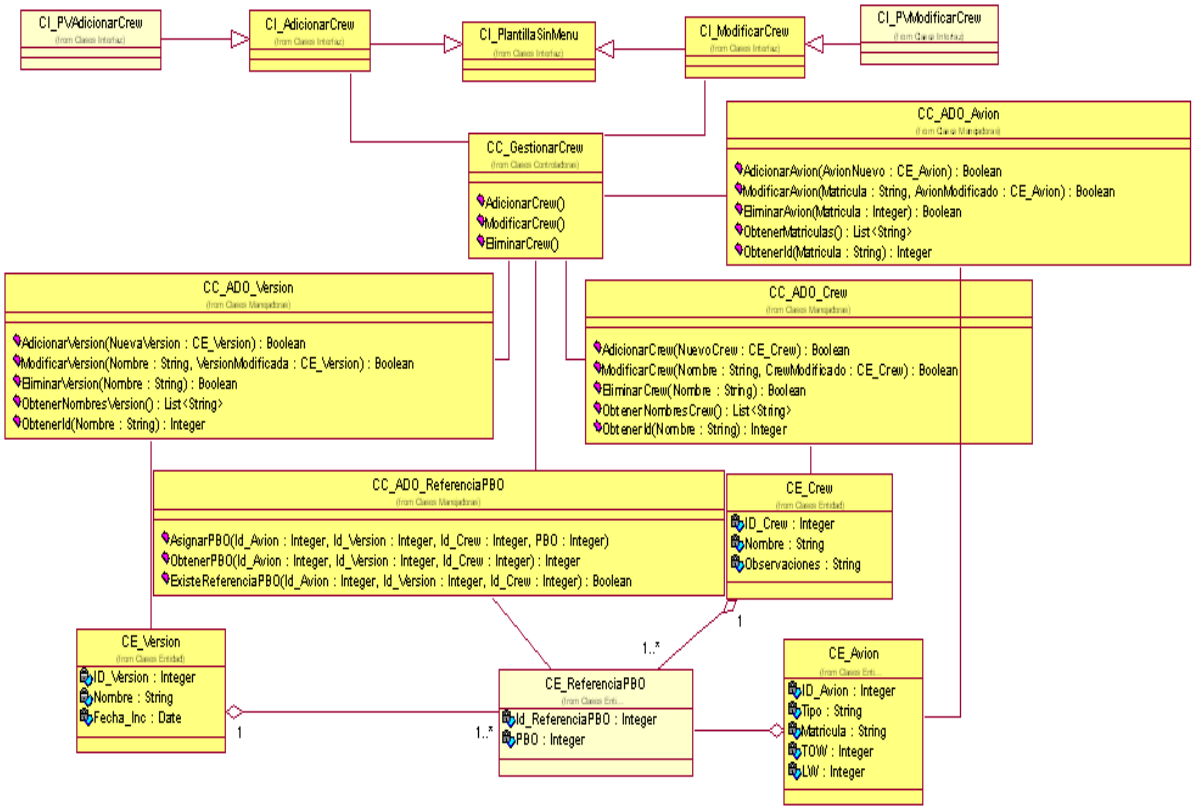


Fig. 35: Diagrama de Clases del Diseño CU Gestionar Crew

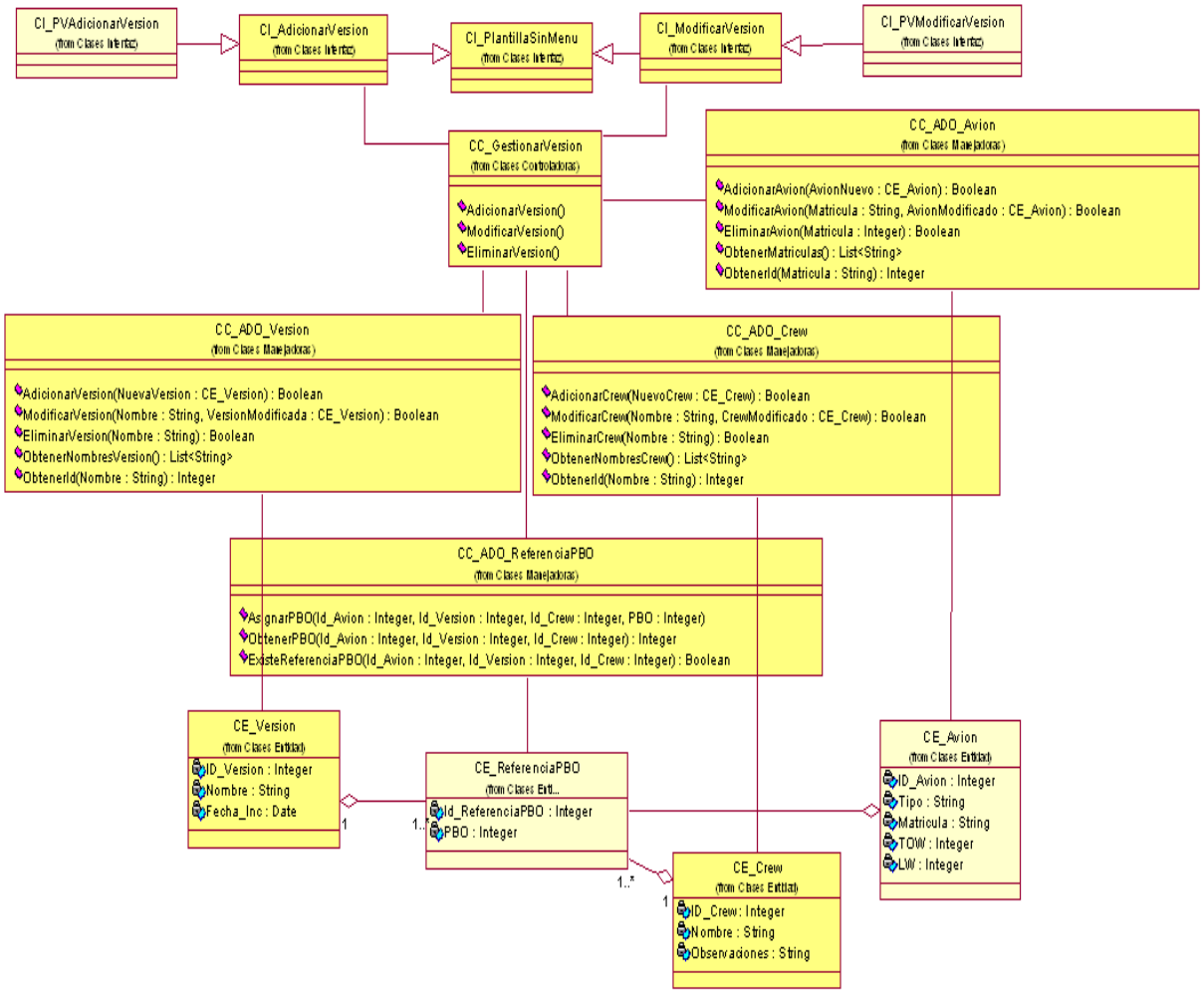


Fig. 36: Diagrama de Clases del Diseño CU Gestionar Versión

Diagramas de Interacción.

Ver Anexo 1.

Conclusiones

En este capítulo se detalló a mayor profundidad la propuesta del sistema a través del análisis y el diseño. Se detallaron los requerimientos de la aplicación mediante los artefactos del análisis, con el objetivo de que el sistema cumpla esos requerimientos mediante los artefactos del diseño ya que refinan el análisis pues toman en cuenta los requisitos no funcionales.

Capítulo 4: Implementación.

Introducción

Mediante los diagramas de despliegue y componentes que se realizarán en este capítulo, se consolidarán las estructuras modeladas en el diseño, permitiendo de esta forma la construcción de un sistema robusto para la Gestión de Vuelo del IL-96-300. Se realiza un análisis de la organización física de la implementación, tanto desde el punto de vista de los dispositivos necesarios para lograr implantar el Sistema, como de la organización que presenta a nivel de ficheros.

4.1. Modelo de Despliegue

El diagrama de despliegue contiene los nodos (procesador o dispositivo) que forman la topología de hardware sobre la que se ejecuta el sistema.

El sistema se diseña sobre la base del siguiente diagrama de despliegue, contiene 1 nodo y 1 dispositivo que representan la ubicación física del Sistema. En el nodo, estará ubicado todo el sistema con el que interactúa la tripulación directamente, el dispositivo que se utiliza permitirá para imprimir los reportes obtenidos del Sistema.

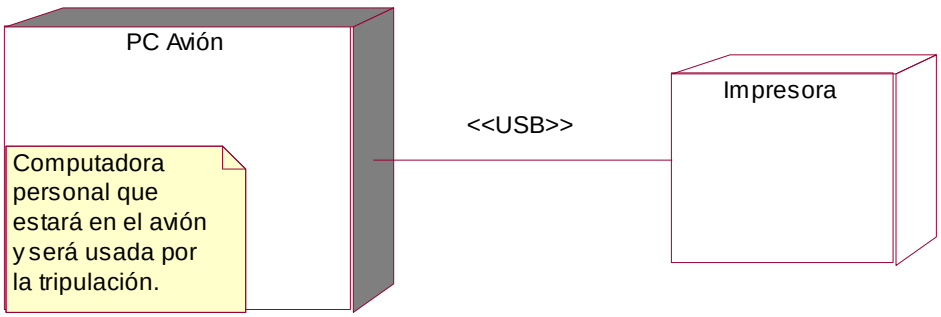


Fig. 37: Modelo de Despliegue

4.2. Diagramas de Componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. A continuación se muestran los diagramas correspondientes al Sistema.

Peso y Balance

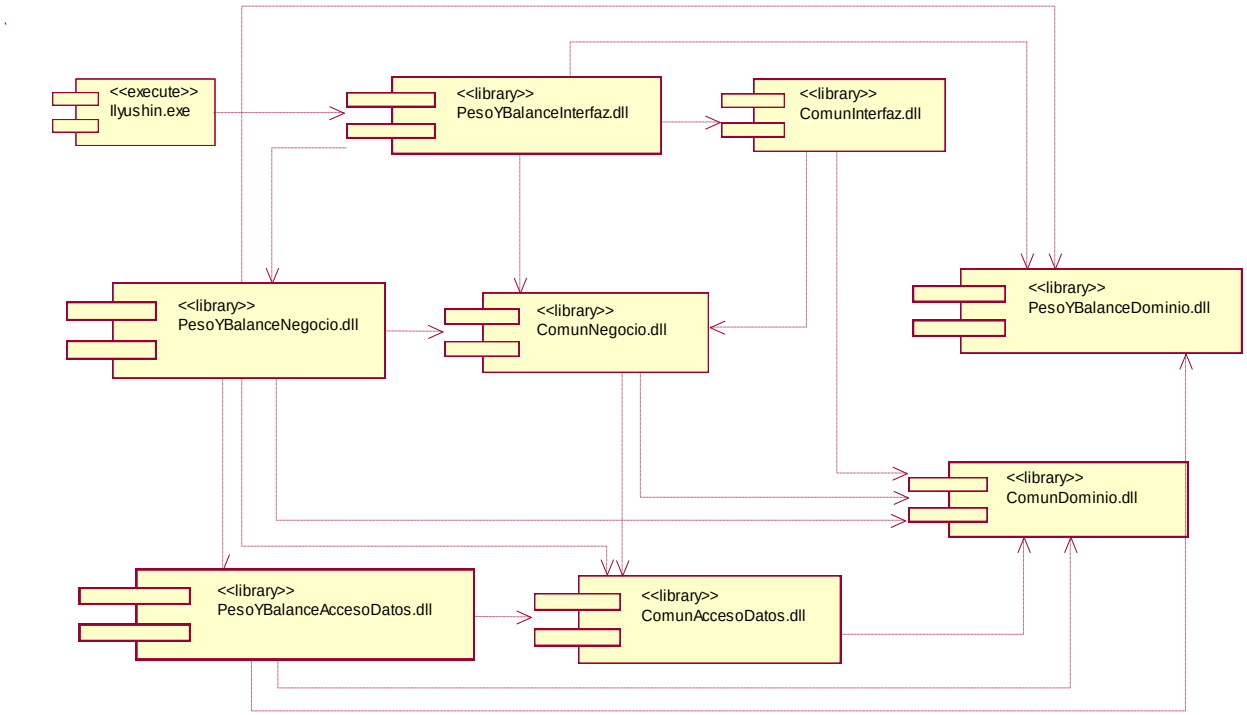


Fig. 38: Diagrama de Componentes. Peso y Balance

Plan de Vuelo

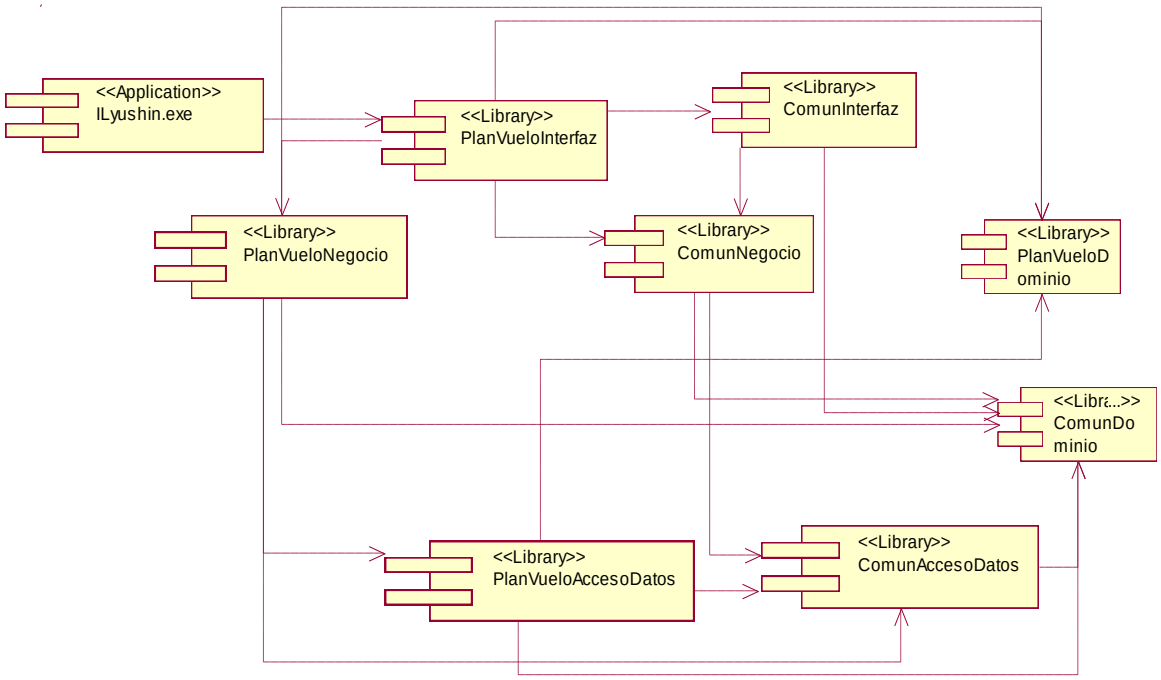


Fig. 39: Diagrama de Componentes. Plan de Vuelo

Conclusiones

Con el desarrollo de este capítulo y basados en la modelación correspondiente para definir el diagrama de componente, el diagrama de despliegue y demás elementos necesarios para la implementación, se logró una visión de la aplicación como entidad física y la definición de los componentes que la conforman, teniendo así una base sólida para un desarrollo eficiente de la solución propuesta.

Capítulo 5: Estudio de Factibilidad.

Introducción

Para una correcta realización de un proyecto es de gran importancia que se analice el costo y lo que reportará para así realizar una comparación con respecto a si es válida o no la idea de continuar desarrollándolo. A continuación se mostrará el estudio de la estimación de costo del sistema propuesto.

5.1 Planificación basada en casos de uso.

a) Cálculo de los Puntos de casos de uso desajustados.

UUCP =UAW +UUCW

UUCP: Puntos de casos de uso sin ajustar.

UAW: Factor de peso de los actores sin ajustar.

UUCW: Factor de peso de los casos de uso sin ajustar.

Factor de Peso de los Actores sin ajustar.

Tipo de Actor	Descripción	Factor de Peso	Actores	Total
Simple	Sistema con sistema a través de interfaz de programación.	1		
Medio	Sistema con sistema a través de interfaz basada en texto.	2		
Complejo	Persona que interactúa con el sistema mediante interfaz Grafica	3	5	15
Total				15

UAW = $\sum$cant.actores * peso

UAW = 5 *3 = 15

Factor de peso de los casos de uso sin ajustar.

Tipo de CU	Descripción	Peso	Cantidad	Total
Simple	El caso de uso tiene de 1 a 3 transacciones.	5	9	45
Medio	El caso de uso tiene de 4 a 7 transacciones.	10	2	20
Complejo	El caso de uso tiene más de 8 transacciones.	15		
Total			11	65

UUCW = $\sum$cant.CU * Peso

UUCW =65

UUCP =65+15

UUCP =80

b) Cálculo de los Puntos de casos de uso ajustados.

UCP =UUCP*TCF * EF

UCP: Puntos de casos de uso ajustados.

UUCP: Puntos de casos de uso sin ajustar.

TCF: Factor de complejidad técnica.

EF: Factor de ambiente.

Calculándose el Factor de complejidad técnica (TCF) mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada factor se pondera con un valor desde 0 (aporte no considerable) hasta 5 (aporte muy apreciable).

Significado de los valores.

0: No presente o sin influencia.

1: Influencia incidental o presencia incidental.

2: Influencia moderada o presencia moderada.

3: Influencia media o presencia media.

4: Influencia significativa o presencia significativa.

5: Fuerte influencia o fuerte presencia.

Factor de complejidad técnica

Factor	Descripción	Peso	Valor	Total
T1	Sistema distribuido	2	0	
T2	Objetivos de performance tiempo de respuesta	1	4	4
T3	Eficiencia del usuario final	1	4	4
T4	Procesamiento interno complejo	1	2	2
T5	El código debe ser reutilizable	1	5	5
T6	Facilidad de instalación	0.5	5	2.5
T7	Facilidad de uso	0.5	5	2.5
T8	Portabilidad	2	5	10
T9	Facilidad de cambio	1	5	5
T10	Concurrencia	1	1	1
T11	Incluye objetivos especiales de seguridad	1	5	5
T12	Provee acceso directo a terceras partes	1	0	0
T13	Se requieren facilidades especiales de entrenamiento a los usuarios.	1	0	0
Total				36

TCF = 0.6+0.01 * Σ (peso * valor asignado)

TCF= 0.6+0.01*36

TCF= 0.96

Factor de ambiente (EF) se relaciona directamente con las habilidades y entrenamiento del grupo que realiza el sistema. Cada factor se pondera con un valor desde 0 (aporte no considerable) hasta 5 (aporte muy apreciable).

EF = 1.4 - 0.03 * Σ **(Peso$_i$ * Valor$_i$)**

Factor	Descripción	Peso	Valor	Total
E1	Familiaridad con el modelo de proyecto utilizado.	1.5	0	0
E2	Experiencia en la aplicación.	0.5	1	0.5
E3	Experiencia en la orientación a objetos.	1	4	4
E4	Capacidad de analista líder.	0.5	3	1.5
E5	Motivación.	1	3	3
E6	Estabilidad de requerimientos.	2	5	10
E7	Personal Part-Time	-1	2	-2
E8	Dificultad del lenguaje de programación.	-1	2	-2
				15

$EF = 1.4 - 0.03 * \Sigma (Peso_i * Valor_i)$

EF = 1.4 - 0.03 * 15

EF= 0.95

UCP= 80 * 0.95 * 0.96

UCP= 72.96

c) Estimación de esfuerzo a través de los puntos de casos de uso.

E=UCP*CF

E: Esfuerzo estimado en horas hombres.

UCP: Punto de casos de usos ajustados.

CF: Factor de conversión.

Factor de convención (CF) se cuentan cuantos valores del factor ambiente están por debajo de la media (3) de E1 a E6, y cuántos están por encima de la media en E7 y E8. Si el total es 2 o menos se utiliza el factor de convención 20 HH/Puntos de CU. Si el total es 3 o 4 se utiliza 28 HH/Puntos de CU. Si el total es mayor a 5 se recomienda efectuar cambios en el proyecto ya que tiende a perecer en corto tiempo. En este caso se puede afirmar que:

CF= 20HH/Puntos de CU.

E= 72.96*20

E= 1459.2 Horas/Hombre.

Calcular esfuerzo de todo el proyecto.

Actividad	Porcentaje(%)	Horas Hombre
Análisis	15%	486.4 Horas/Hombre
Diseño	30%	972.8 Horas/Hombre
Implementación	45%	1459.2 Horas/Hombre
Pruebas	0%	0 Horas/Hombre
Sobrecarga(Otras Actividades)	10%	324 Horas/Hombre
Total	100%	3242.6 Horas/Hombre

Si E_T = 3242.6 Horas/ hombre y se estima que cada mes promediado 192 horas laborables, quedaría E_T= 16.88 Mes/Hombre.

Costo del Proyecto

En el caso del salario mensual es de $150.00

CH: Cantidad de hombres

Tiempo: Tiempo total del Proyecto.

CH= 2 hombres.

CHM= 2 hombres * Salario.

CHM= 300 $/mes

Costo= CHM * E_T

Costo= 300 * 16.88 /2

Costo= $ 2532

Tiempo total del Proyecto

Tiempo = E_T / CH

Tiempo= 16.88 meses /2 hombres

Tiempo= 8.44 meses

Del resultado obtenido se interpreta que con 2 hombres el proyecto tiene un tiempo de duración de 8.44 meses y el costo total que se estima es de $ 2532.

5.2 Beneficios tangibles e intangibles

Esta aplicación tiene como beneficio fundamental que se agilice el proceso de realización de los cálculos necesarios para realizar los vuelos del avión IL_96 300 y obtención de los reportes de Plan de Vuelo, Peso y Balance, erradicando así las posibles inclusiones de errores humanos, ya que estos cálculos son complicados y en ocasiones se realizan en situaciones de peligro.

Además se puede apreciar que:

- Disminuye considerablemente la pérdida de tiempo, pues estas operaciones se realizaban de forma manual.
- Posee una interfaz agradable a la vista, organizada y sencilla, que permitirá una rápida y fácil utilización. Facilita la gestión de los datos necesarios y un fácil procedimiento para la obtención de los reportes.
- Permite la disminución de los gastos en materiales de oficina.
- Se crean mayor cantidad de reportes en un corto período de tiempo.
- Agilizará y optimizará la búsqueda de reportes hechos con antelación que el sistema tendrá guardados.

5.3 Análisis de costos y beneficios.

El desarrollo de todo producto de software requiere de una inversión de capital, la que se dividen en gastos de personal, y de materiales necesarios para su elaboración, por lo que la viabilidad del proyecto depende de la medida en que los beneficios del mismo puedan saldar lo invertido en su desarrollo, ya sea, social o económicamente. En la realización de este proyecto se usará software propietario para el cual el cliente garantizó la licencia del producto por lo que este gasto no está reflejado en el estudio de factibilidad.

Al implantar el sistema se tendrá como resultado el aumento de la eficiencia de los procesos relacionados con el Peso y Balance y Plan de Vuelo para el avión IL-96-300, disminuyendo el tiempo

de realización de estos, logrando una mayor exactitud en dichos cálculos. Además se valora la solución para Cuba, puesto que no existe un software que realice los cálculos de Peso y Balance y Plan de Vuelo que cubra las necesidades de la tripulación del avión IL-96-300. Al mismo tiempo la puesta en práctica de este sistema ahorraría una suma considerable de dinero utilizado para el pago de estos servicios a SITA. A través del análisis del costo del proyecto y los numerosos beneficios que este reporta, se puede concluir que su implementación es realmente factible.

Conclusiones

En este capítulo se realizó el estudio de factibilidad y validación del sistema. Se obtuvo como resultado que con las dos personas que desarrollarán el sistema se requiere un tiempo de 8 meses y 13 días para la confección total del mismo, con un costo total del proyecto de $ 2532. Se expusieron los beneficios del sistema y se efectuó un análisis entre estos y los costos, el cual brindó elementos para concluir que es factible el desarrollo de esta aplicación. Se comprobó que la herramienta propuesta reportará beneficios significativos e importantes para la tripulación de los aviones IL-96-300.

Conclusiones Generales

Con la implantación exitosa del sistema, se dará solución a los problemas actuales existentes en la empresa Cubana de Aviación para la tripulación que opera los aviones IL-96-300. Esto implicará un mejoramiento en las condiciones de trabajo de los usuarios logrando eliminar los errores en los resultados obtenidos y una reducción de costos relacionados con los servicios contratados a SITA que se utilizan. Como resultado de la selección de las herramientas se determinó utilizar para la implementación del software el lenguaje de programación C#, la plataforma .NET, el SQL Server 2000 como servidor de base de datos, y el Rational Rose para el modelado visual del sistema.

Con los beneficios tangibles e intangibles expuestos durante el desarrollo del trabajo y el análisis de costos efectuado, resulta factible el desarrollo de la aplicación, siendo incuestionables las mejoras que se producirán al implantar este servicio en la Empresa Cubana de Aviación.

Recomendaciones

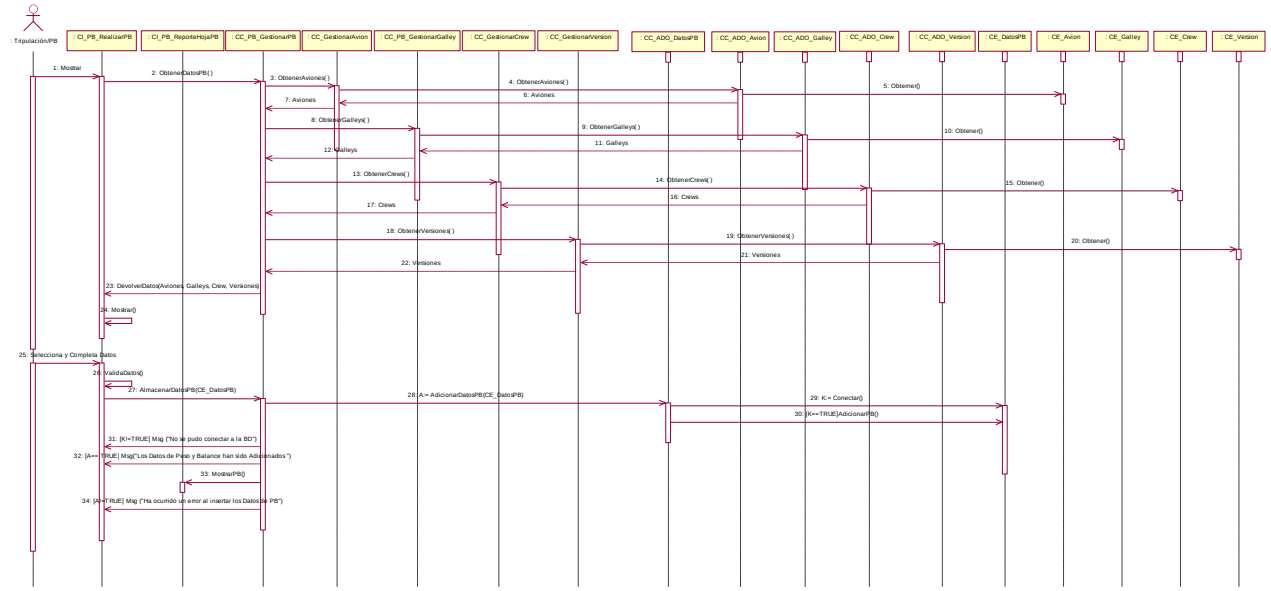
El software creado tiene implementadas las funcionalidades que abarcan los requisitos funcionales y no funcionales solicitados por los clientes. Teniendo en cuenta el logro de un mejor funcionamiento del mismo se reflejan las siguientes recomendaciones:

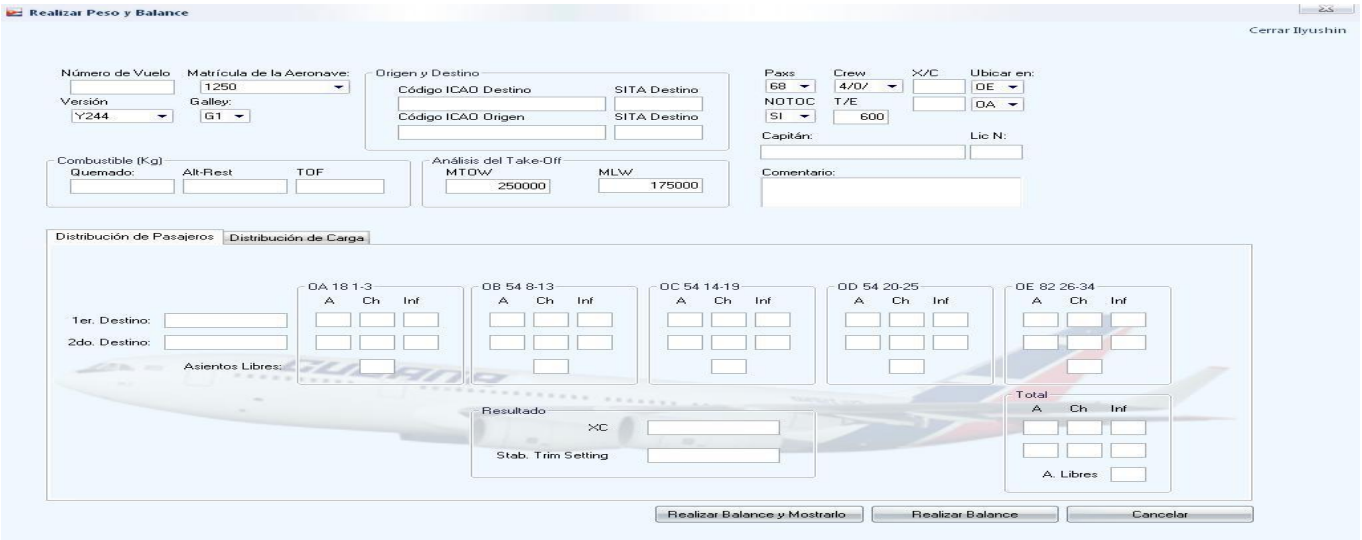
- Realizar las investigaciones que permitan modelar e implementar los procesos que se llevan a cabo para la realización del Análisis de Pista del IL-96-300.
- Investigar la posibilidad de realizar una versión de esta aplicación con herramientas de desarrollo con licencias de Software Libre.
- Investigar la posibilidad de realizar una versión de esta aplicación que gestione los vuelos de todos los modelos de aviones que utiliza la Empresa Cubana de Aviación.
- Desarrollar la ayuda del sistema.
- Poner a prueba el sistema durante un período de tiempo significativo para comprobar los resultados obtenidos pues es un factor importante para mantener la seguridad en los vuelos.
- Mantener sobre el sistema un estricto cumplimiento del proceso de mantenimiento y actualización periódica, logrando así que se mantenga la fiabilidad y funcionamiento óptimo del sistema.

Anexos

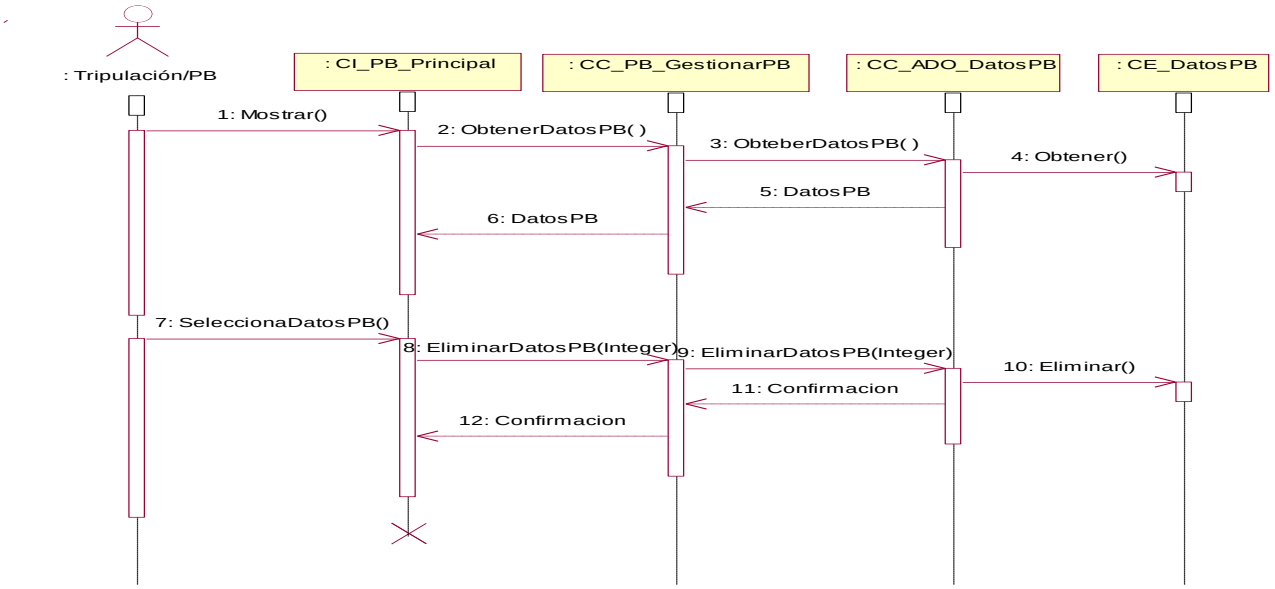
Anexo 1: Diagramas de Interacción por realización de CU

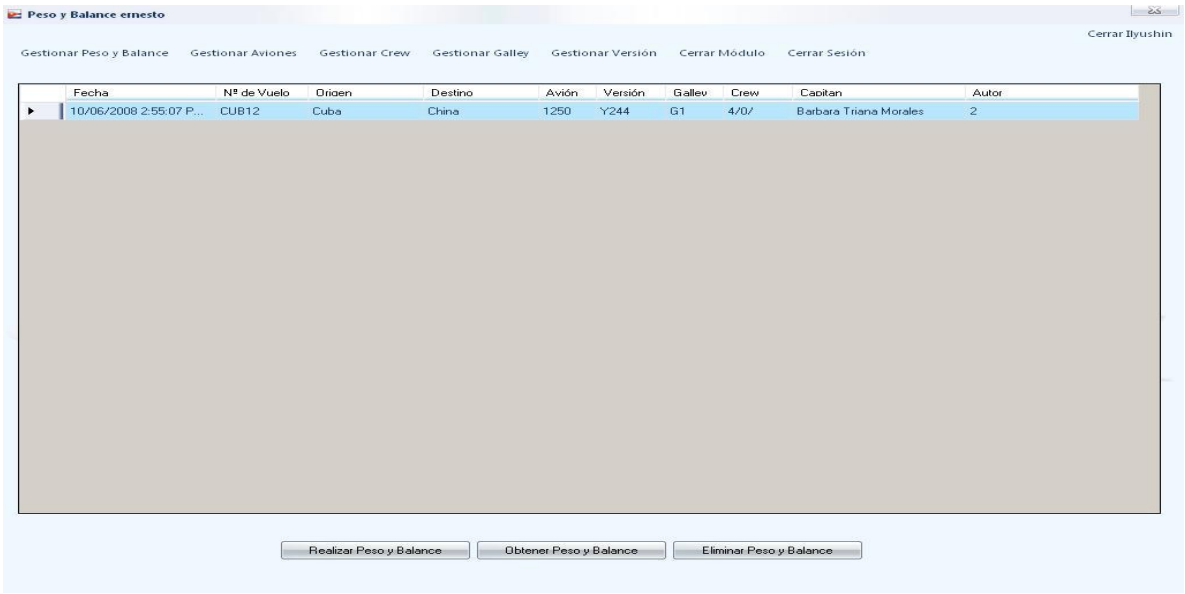
Realizar Peso y Balance



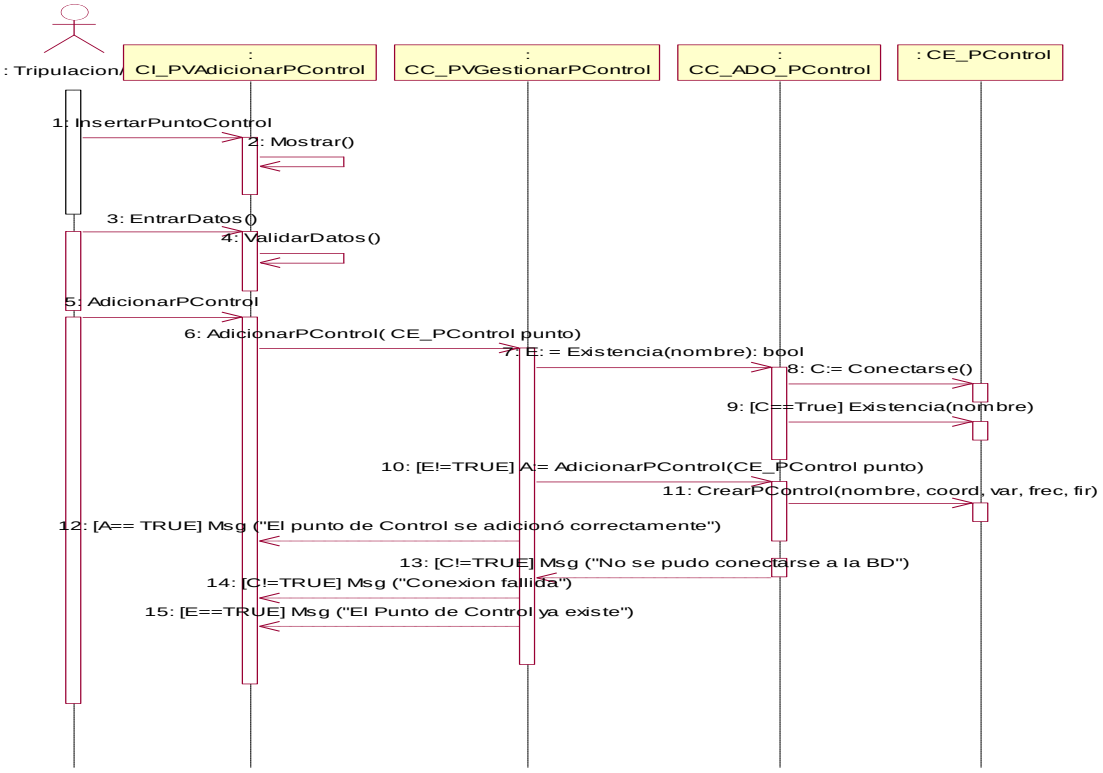


Eliminar Peso y Balance

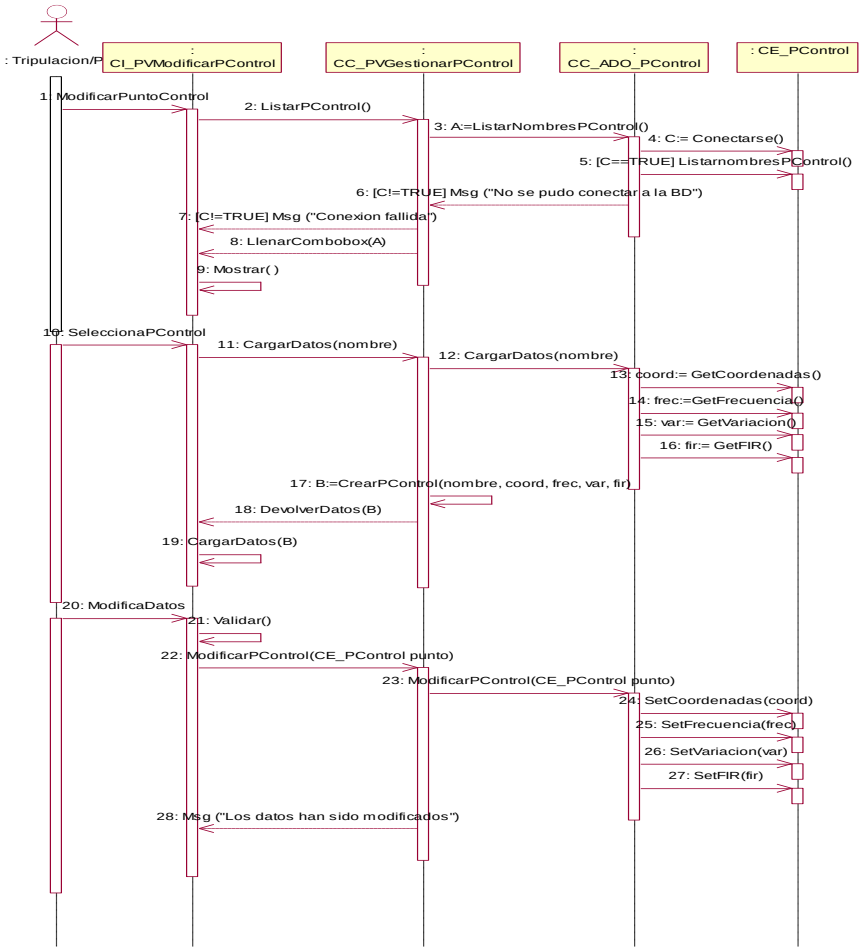




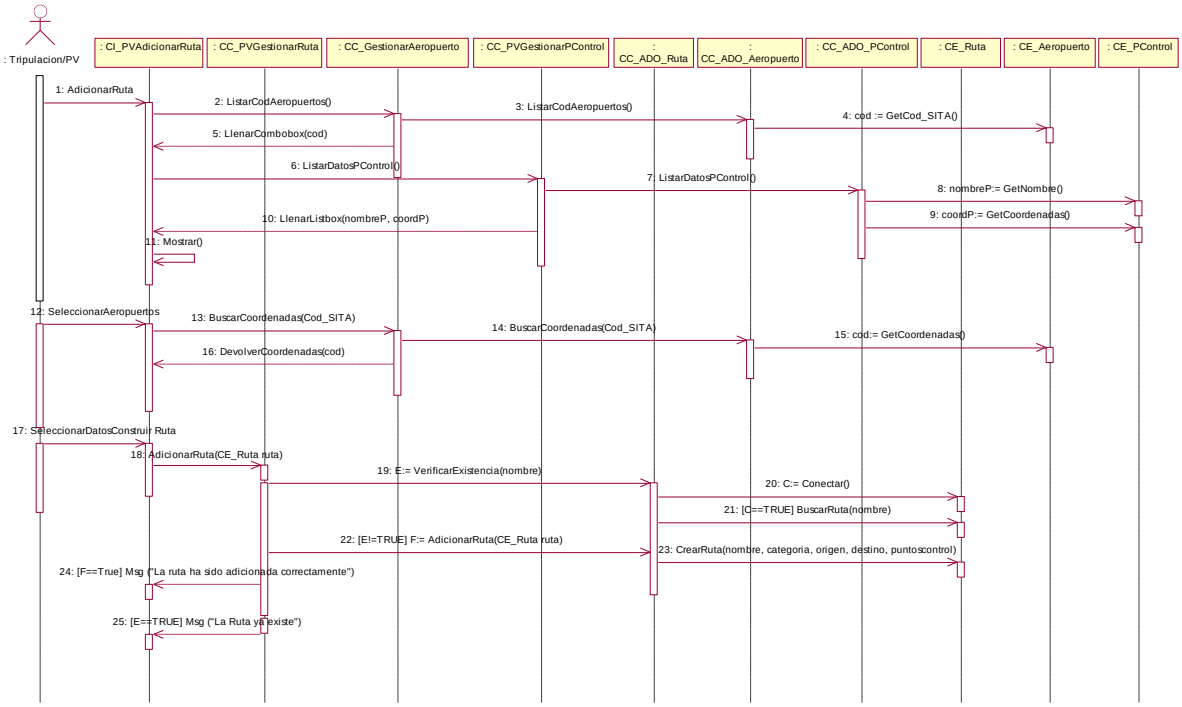
Adicionar Punto de Control

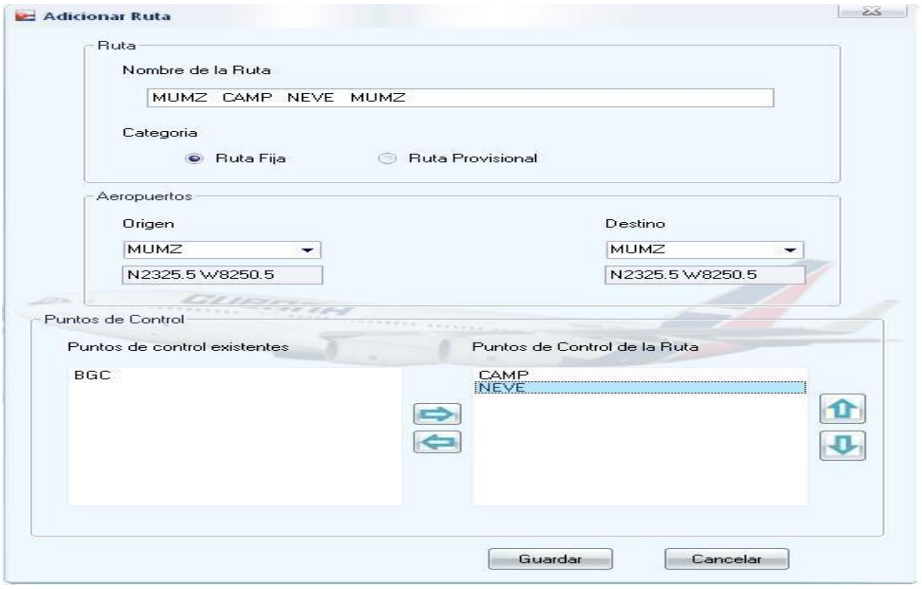


Modificar Puntos de Control

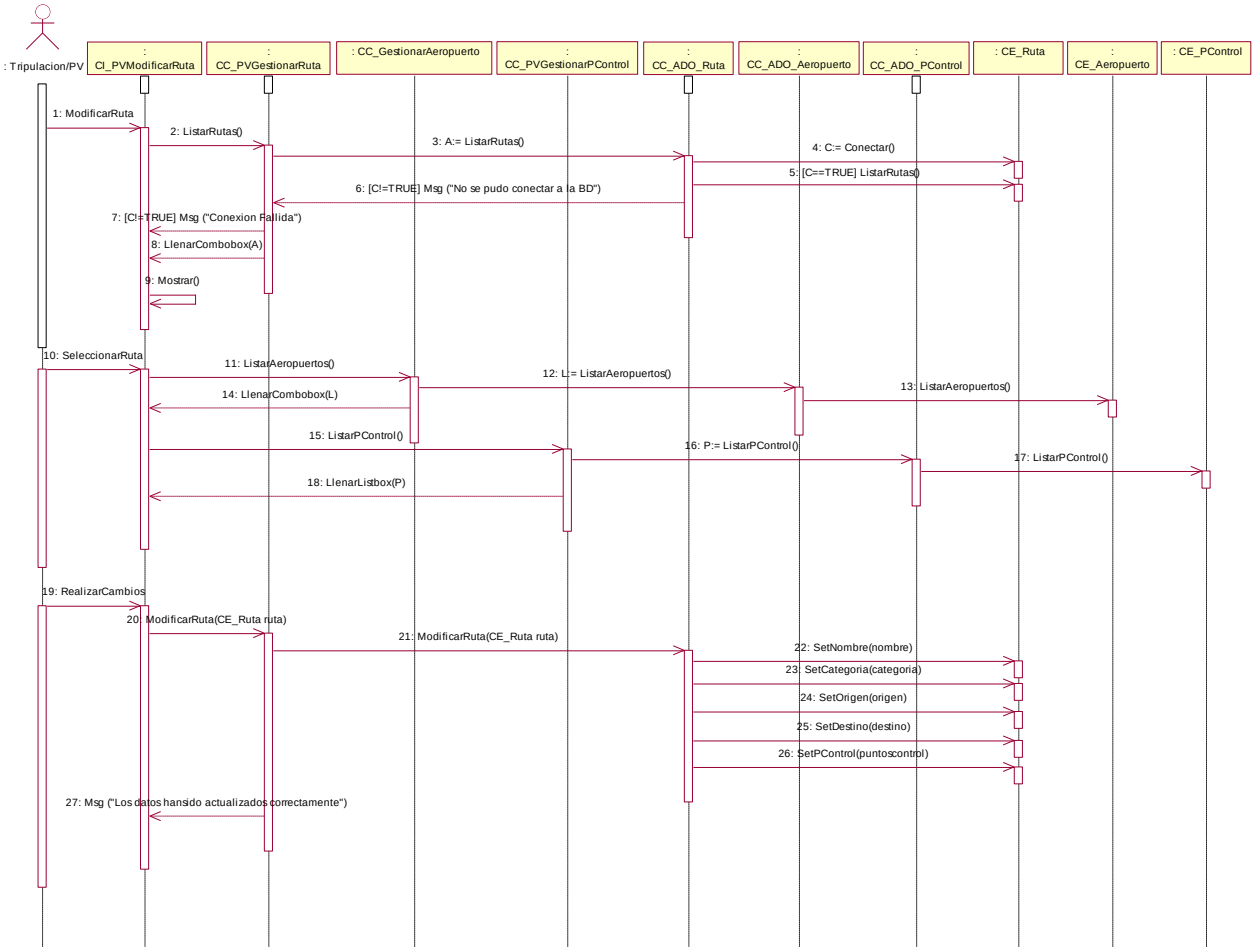


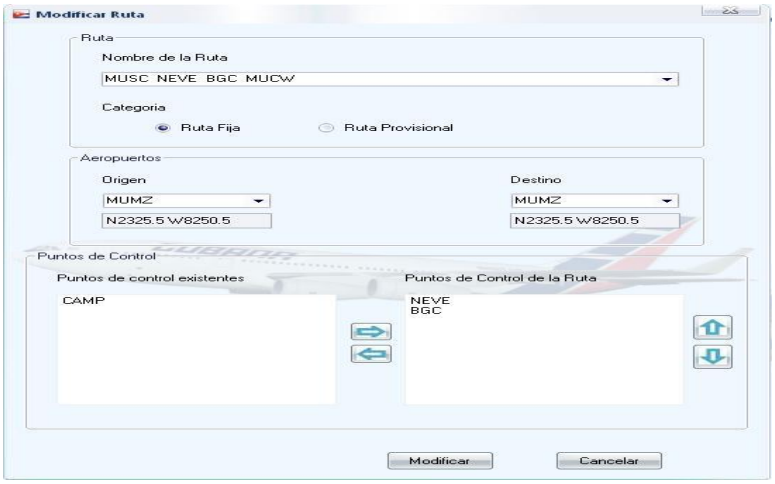
Adicionar Ruta



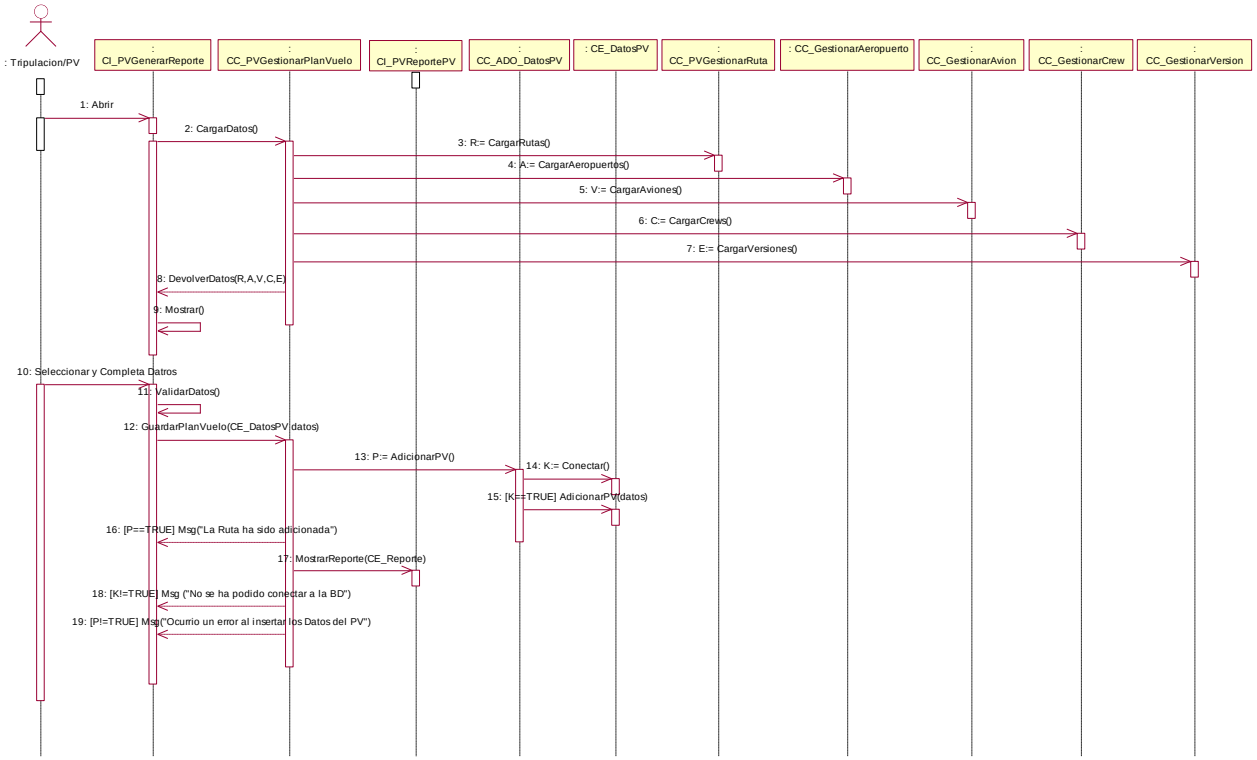


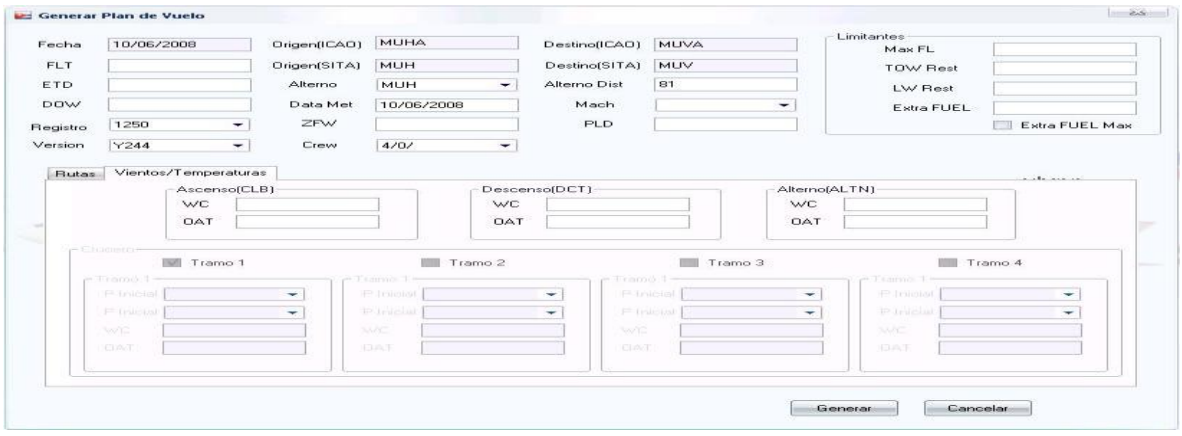
Modificar Ruta



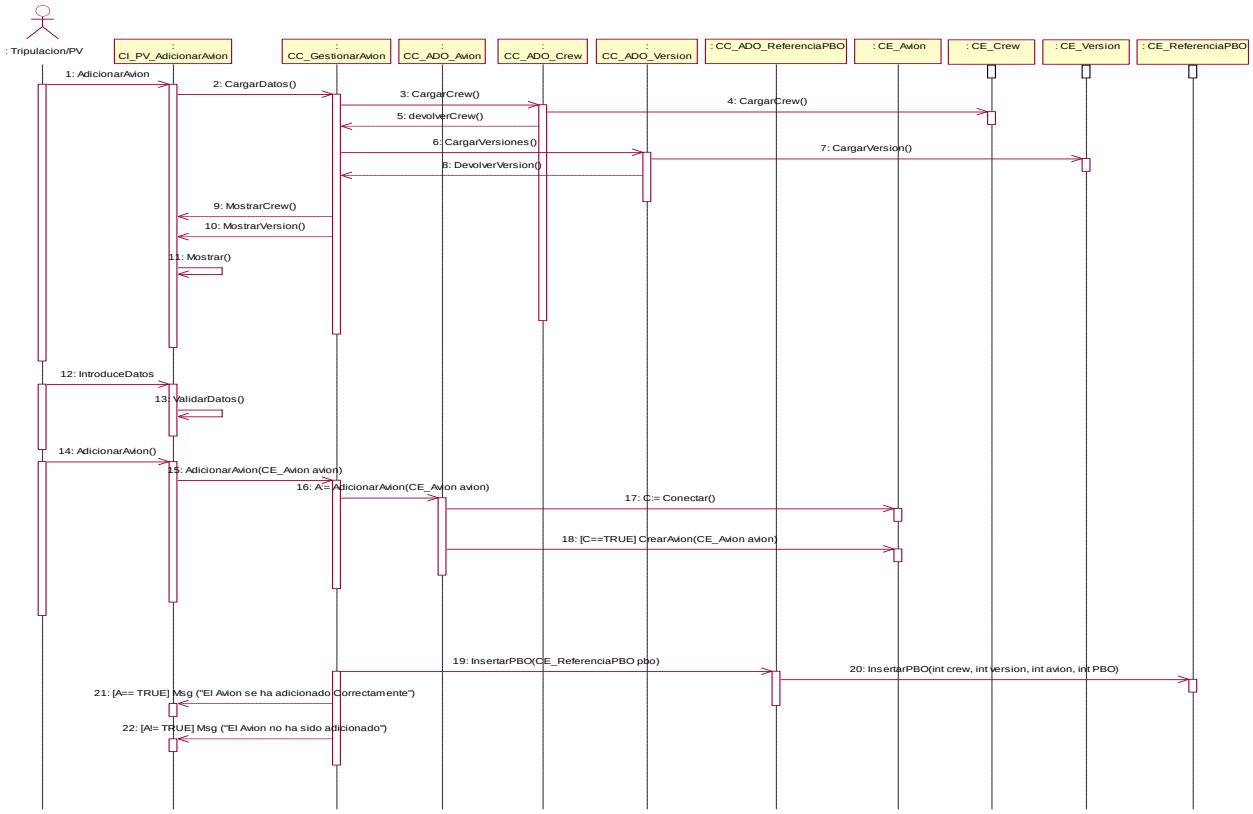


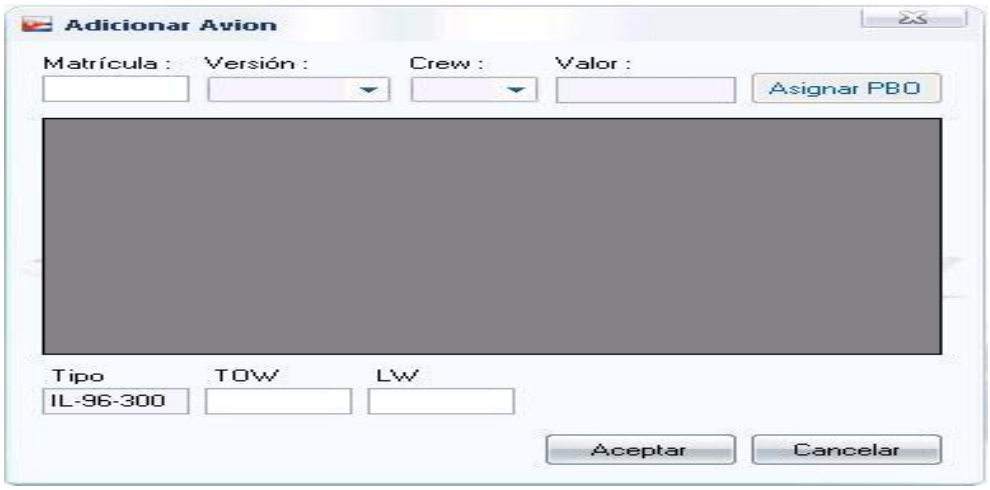
Generar Plan de Vuelo



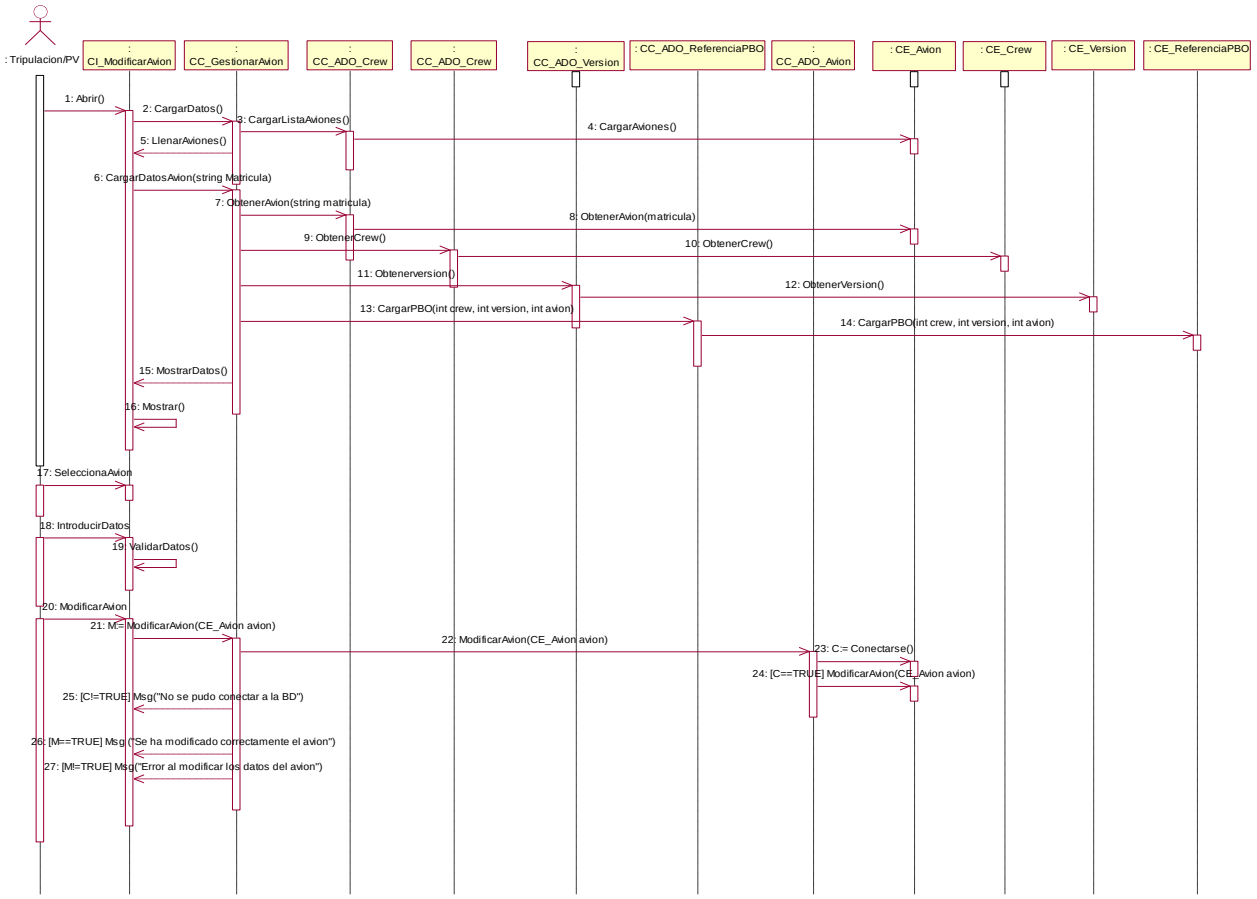


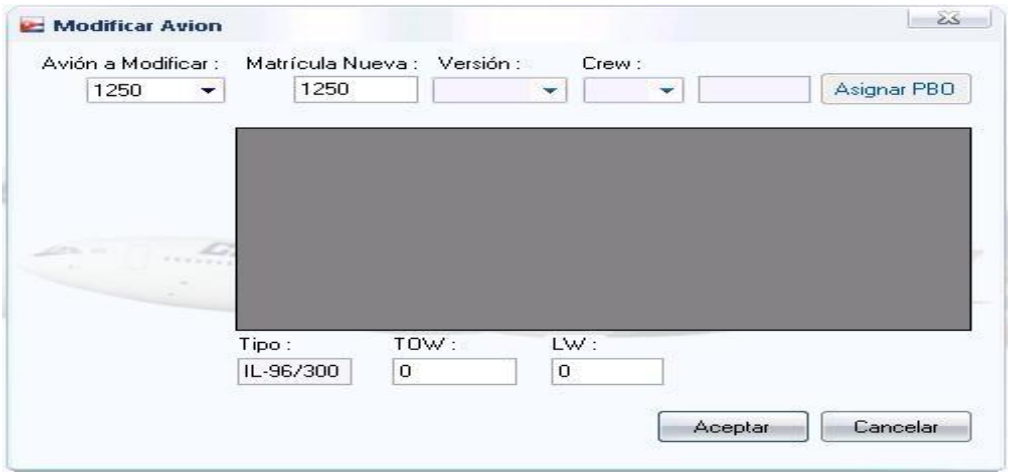
Adicionar Avión



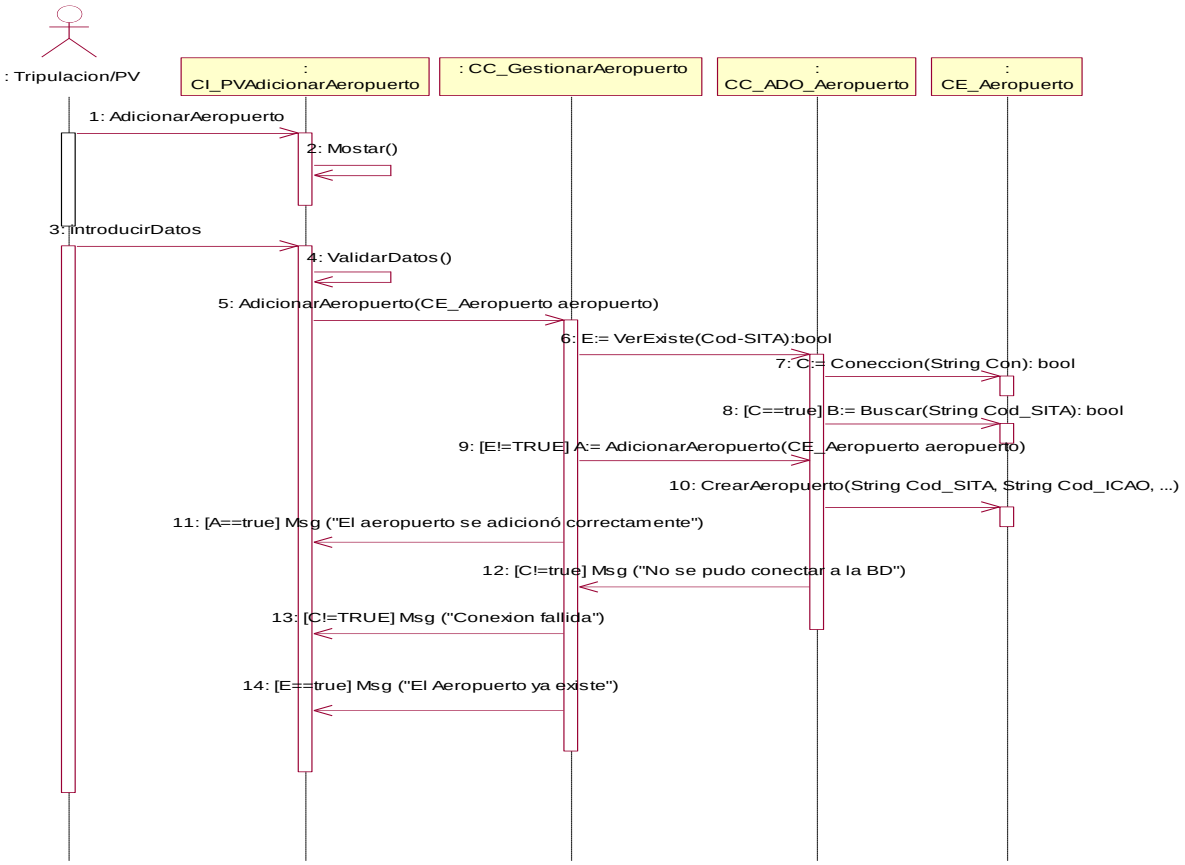


Modificar Avión

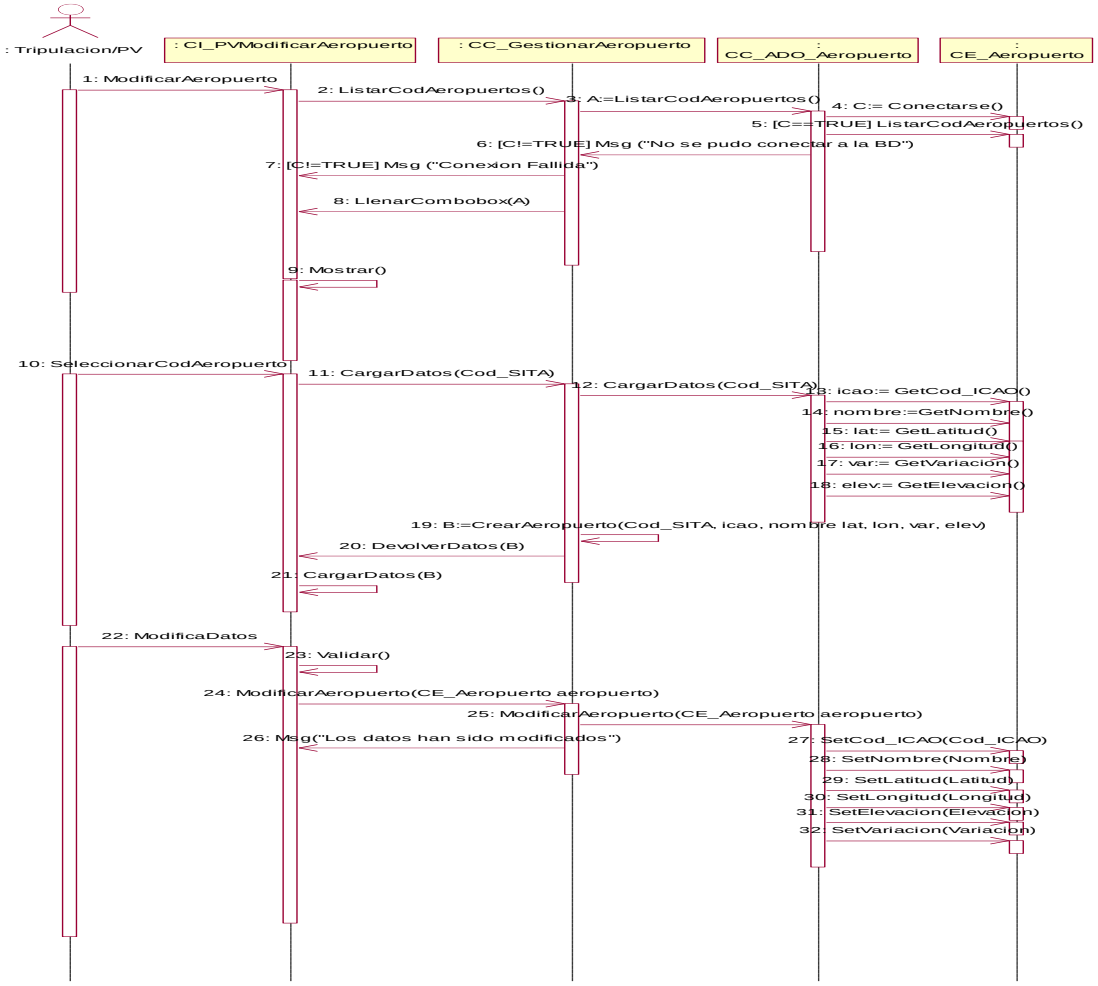




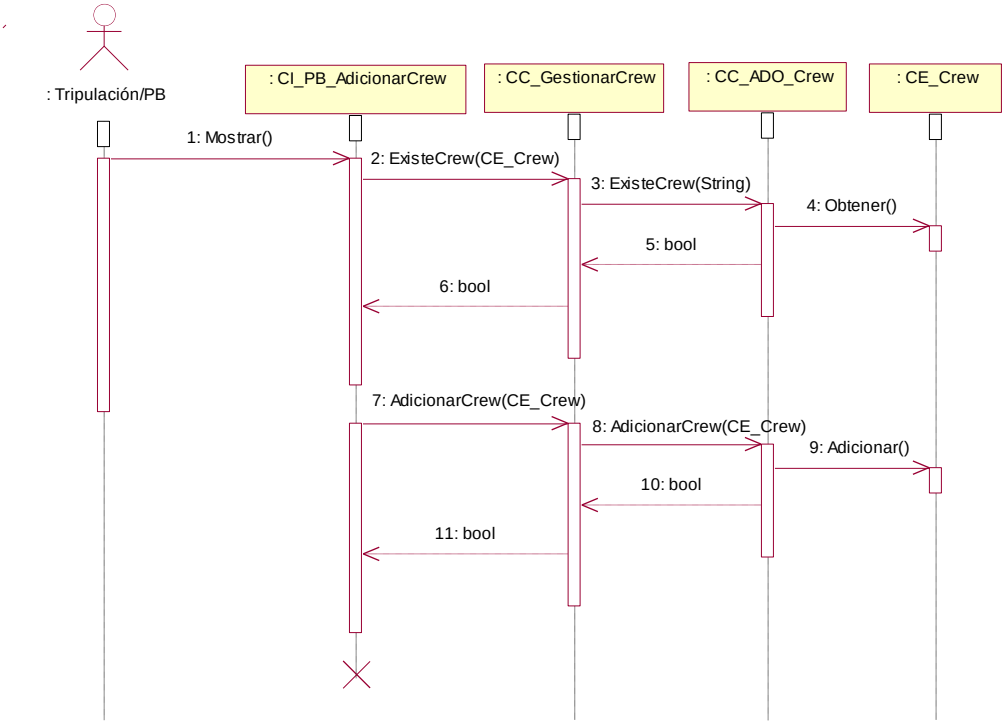
Adicionar Aeropuerto

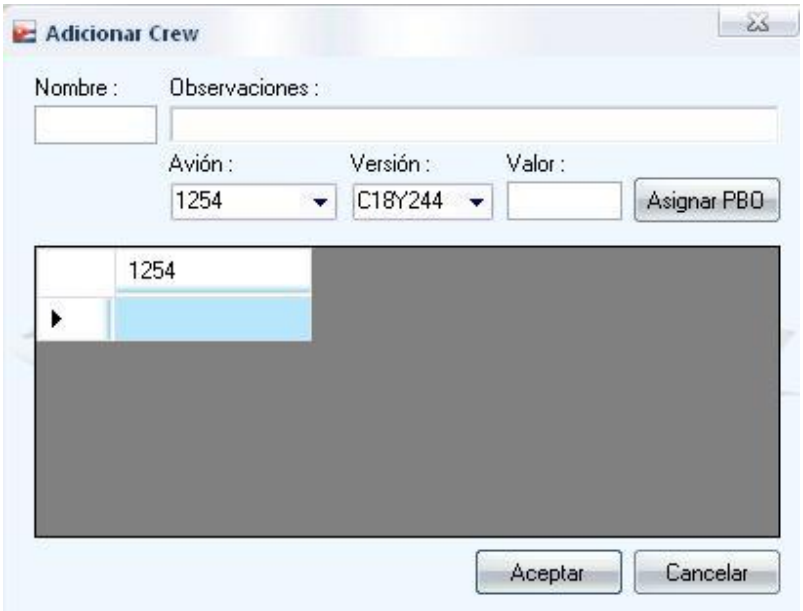


Modificar Aeropuerto

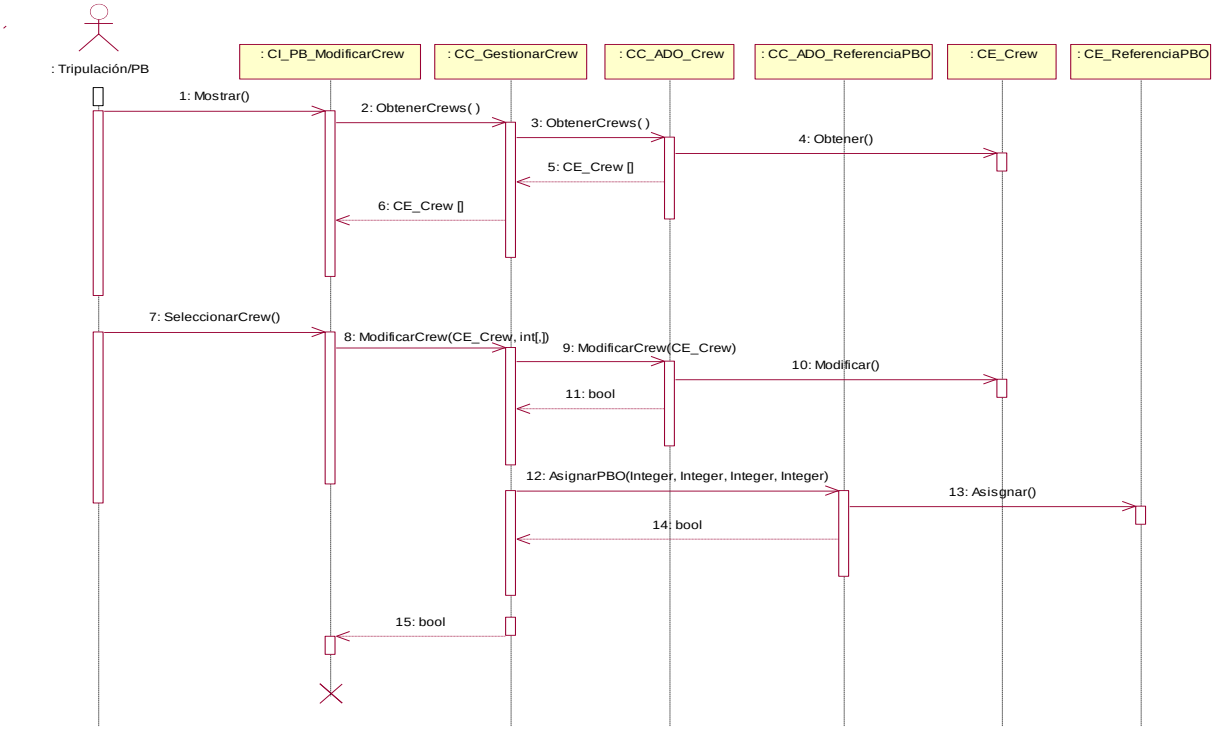


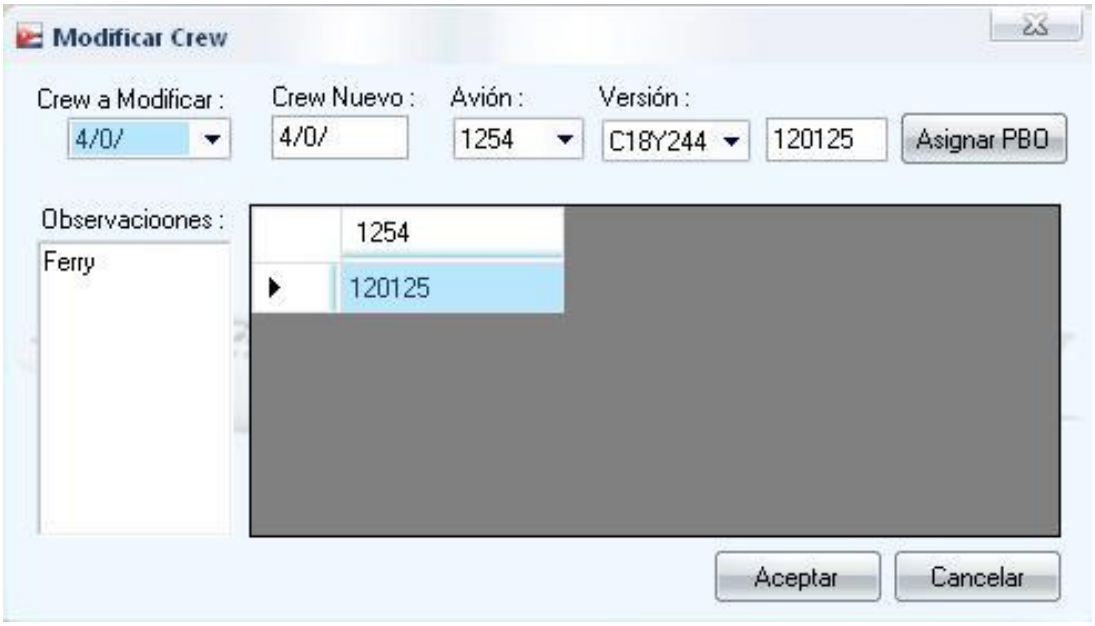
Adicionar Crew



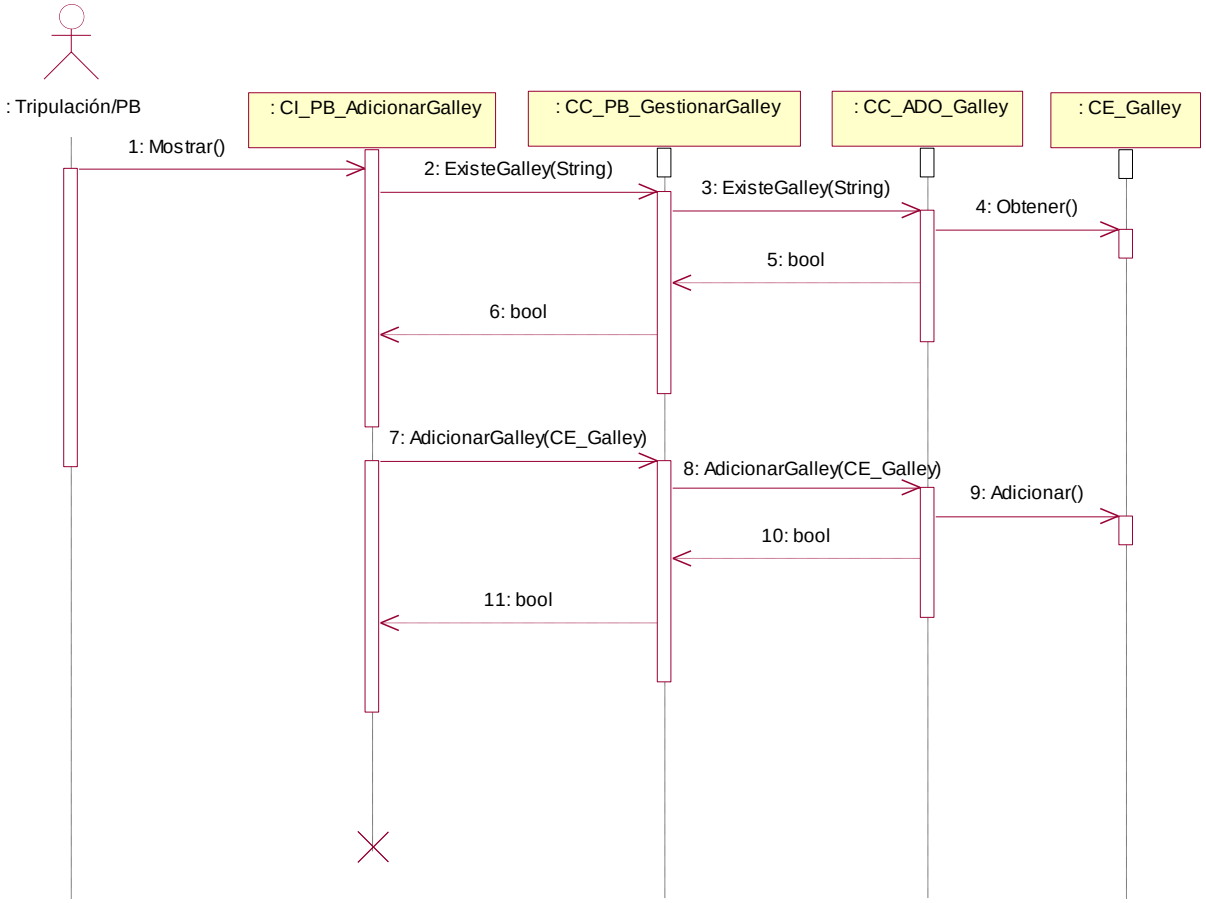


Modificar Crew

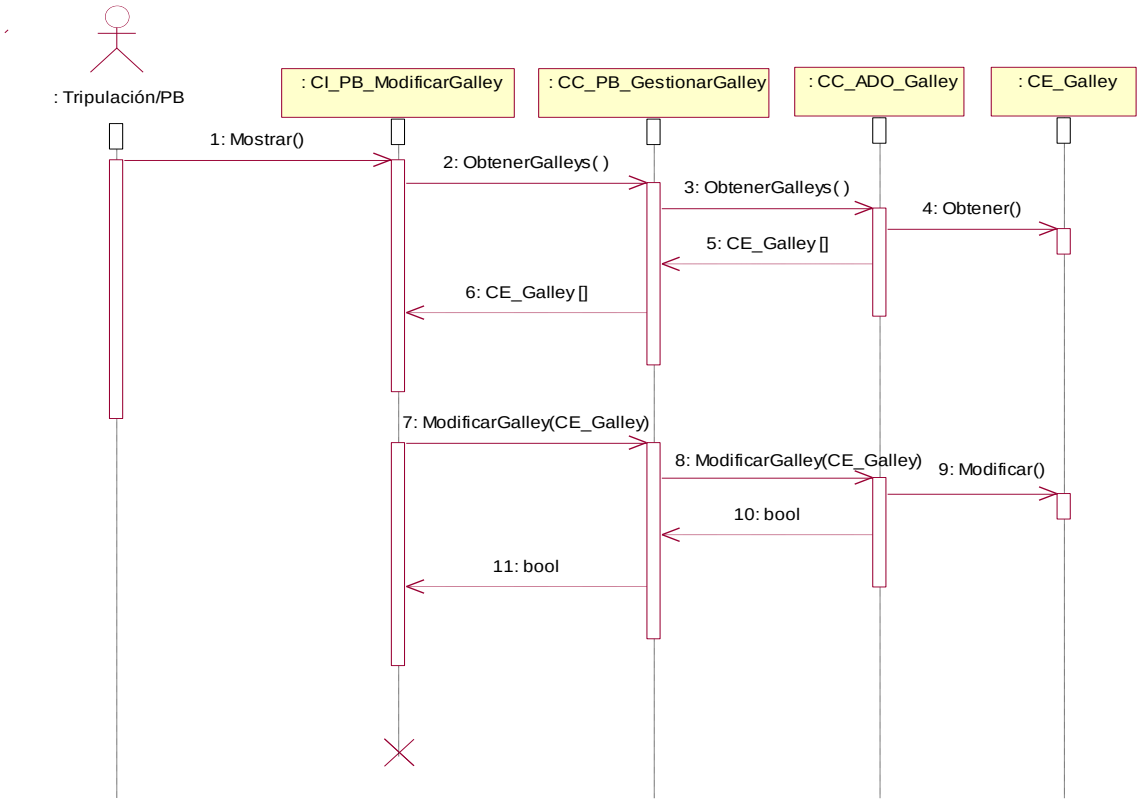


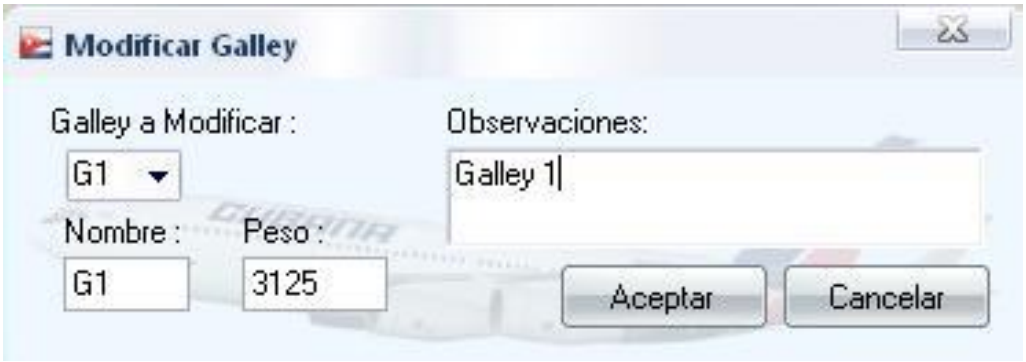


Adicionar Galley



Modificar Galley





Bibliografía

1. El lenguaje C# y la plataforma .NET. en: *PROGRAMACIÓN C#*. 20. p.
2. IBM_RATIONAL_SOFTWARE. Rational Rose Enterprise, http://www-306.ibm.com/software/awdtools/developer/rose/enterprise/ , 2008.
3. LARMAN, C. *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. España, 1998. 195 a 250 p. 0-13-748880-7
4. JACOBSON, I. B., GRADY Y RUMBAUGH, JAMES. El Proceso Unificado de Desarrollo de Software. Editorial Félix Varela ed. La Habana: 2000.
5. MacDonald, M. (2002). *biblioteca.uci.cu.* Retrieved 2 15, 2008, from http://bibliodoc.uci.cu/pdf/reg03057.pdf
6. *msdn*. (2008). Retrieved 1 16, 2008, from http://msdn.microsoft.com/library/spa/default.asp?url=/library/spa/vsintro7/html/vsstartpage.asp
7. MOLPECERES, A. *Procesos de desarrollo: RUP, XP y FDD*, http://www.willydev.net/descargas/articulos/general/cualxpfddrup.PDF , 2003.
8. PRESSMAN, R. S. *Ingeniería de software. Un enfoque practico*. Mc Graw Hill, 1998. 614 p.
9. Seco, J. A. *El lenguaje de programación C#.*
10. *wikipedia.org*. (n.d.). Retrieved 1 15, 2008, from http://es.wikipedia.org/wiki/Multiplataforma
11. *willydev.net*. (2003). Retrieved 1 20, 2008, from http://www.willydev.net/CrystalDesde0/
12. *www.Devjoker.com* . (n.d.). Retrieved 1 15, 2008, from http://www.devjoker.com/asp/impresion_contenido.aspx?co_contenido=89
13. ZAPATA, D. E. V. *Introducción a la programación Multicapas*, Retrieved 12 -12- 2007, 2004. From http://www.elguille.info/colabora/puntoNET/jevergara_Multitier.htm

Glosario de Términos

1. **PBO**: Peso Básico Operacional del avión.
2. **Velocidad de crucero**: La velocidad de crucero es la velocidad estable a la que se mueve el avión.
3. Tripulación: La tripulación son aquellas personas que se encargan del manejo del avión y de la atención a los pasajeros. En este caso esta compuestas por el Capitán, Piloto, Copiloto, Ingeniero de vuelo, Navegante, Despachador y Aeromozas.
4. **Framework.NET**: La arquitectura .NET (.NET Framework) es el modelo de programación de la plataforma .NET para construir y ejecutar los servicios .NET.
5. **CLR (Common Language Runtime):** es el motor de la plataforma. NET, encargado de gestionar la ejecución de las aplicaciones .NET, a las cuales ofrece numerosos servicios para simplificar su desarrollo, favoreciendo con ello su fiabilidad y seguridad, sus principales características y servicios.
6. **Performance**: Desempeño con respecto al rendimiento, es decir, medida o cuantificación de los
7. componentes del avión.
8. **LEMAC**: Distancia en metros desde el comienzo de las coordenadas hasta el comienzo de la MAC con valor de 23,291 m.
9. **MAC**: Cuerda Media Aerodinámica. Constante con un valor de 6.636 m.
10. **C:** Constante para la conversión magnitudes de los momentos de índices con un valor de 2000.
11. **K**: Constante para obtener solo magnitudes de los momentos de índices con un valor de 100.
12. **TTL Carga**: Peso de la carga total que llevará el avión, no se incluye el peso de los pasajeros.
13. **Address**: Representa la dirección ICAO de la estación de destino.
14. **From**: Representa el código que establece SITA de la estación de origen.
15. **Originator**: Representa la dirección ICAO de la estación de origen.
16. **LDM**: Representa el código que establece SITA de la estación de destino.
17. **Flight #:** Representa el número del vuelo que se va a realizar.
18. **A\C Reg**: Matrícula del avión.
19. **CREW**: Representa el tipo de vuelo que se realizará, ya sea comercial.
20. **X\C**: Personal que no pertenece a la tripulación, pueden ser mecánico, persona de seguridad del estado, etc.
21. **Galley**: Representa el destino que tomará el avión, lo cual limita al avión según los pesos estructurales.

22. **T\E**: Carga que se coloca en la cola del avión, este peso es la suma de gomas y piezas de repuesto que lleva el avión IL_96 300 en cada vuelo.
23. **Paxs**: Peso permisible por cada pasajero según el vuelo, es decir, el peso que tomaría cada pasajero según el lugar al cual se dirige el avión, que puede ser 68, 75 y 80.
24. **Burn**: Representa la cantidad de combustible que el avión IL_96 300 quemará en el vuelo.
25. **TOF**: Representa la cantidad de combustible que se le suministrará al avión.
26. **FUEL**: Operaciones establecidas que se realizaran con el combustible.
27. **MTW**: Peso Máximo que resiste el avión en Rampa y Rodaje. Que en este caso del IL_96 300 permite 251 000 Kg.
28. **MFW**: Peso máximo de combustible que permite el avión IL_96 300.
29. **MTOW**: Máximo Peso de Despegue del avión, limitado estructuralmente y por las normas de pilotaje.
30. **MLW**: Máximo Peso de Aterrizaje que permite el avión limitado estructuralmente y por las normas de pilotaje.
31. **Ballet**: Plancha sobre la cual va generalmente el correo y la carga empaquetada.
32. **Contenedores**: Lugar en el cual se deposita la carga comercial.
33. **ZFW**: Es el peso total del avión sin combustible.
34. **TOW**: El peso total del avión con combustible.
35. **Basic Weight**: Peso básico del avión, es decir, lo que pesa la carrocería del avión, alfombras, cableado, etc.
36. **DOW**: Peso del avión mas el avituallamiento principal, adicional y del miembro adicional de la tripulación con su equipaje.
37. **Operating weight**: Es el peso de funcionamiento del avión, que no es más que la suma del peso básico de los aviones más los pasajeros, carga, CREW, PANTRY, Bulk, Pallet, conten y el combustible destinado para el vuelo.
38. **MZFW**: Es el máximo Peso Permisible sin combustible limitado estructurado.
39. **NOTOC**: Mercancías peligrosas que se llevará en el avión.
40. **ICAO**: Es un código alfanumérico de 4 dígitos que identifica a cada aeropuerto alrededor del mundo. Estos códigos son definidos por la organización internacional de la aviación civil (ICAO). Estos códigos son únicos. La primera letra es asignada generalmente pro el continente y representa un país o un grupo de países dentro de ese continente. La segunda letra representa un país dentro de esa región, y las dos restantes se utilizan para identificar cada aeropuerto.