Universidad de las Ciencias Informáticas

"Facultad 2"

BASE DE DATOS PARA LA GESTIÓN DEL VUELO DE AVIONES IL-96/300

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

Autor: Yunier Jorge Trujillo Hernández

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

Ciudad de la Habana, julio de 2008
Año 50 de la Revolución

"Los obstáculos son esas cosas que las personas ven cuando dejan de mirar sus metas."

E. Joseph Cossman.

DECLARACION DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 2 de la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio.

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

_____

Firma del Autor

Yunier J. Trujillo Hernández

_____ _____

Firma del Tutor **Firma del Tutor**

Leydis E. Garzón Giro Maidelis Milanés Luque

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: “Base de Datos para la Gestión del Vuelo de Aviones IL-96/300.”

Autor: Yunier J. Trujillo Hernández.

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan:

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de _____.

Firma del Tutor	**Firma del Tutor**
Leydis E. Garzón Giro	Maidelis Milanés Luque

Fecha

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado “Base de Datos para la Gestión del Vuelo de Aviones IL-96/300”, fue realizado en La Universidad de las Ciencias Informáticas. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

□ Totalmente □ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

_____

_____

_____

_____

_____

Como resultado de la implantación de este trabajo se reportará un efecto económico que asciende a _____.

Y para que así conste, se firma la presente a los ____ días del mes de _____ del año _____.

_____	_____
Representante de la entidad	**Cargo**
_____	_____
Firma	**Cuño**

AGRADECIMIENTOS

A mi Mamá y mi Hermana por confiar en mí, guiarme y apoyarme en todo momento.

A mi novia Magdalena, por su apoyo, paciencia y amor.

A mis tutoras Maidelis y Leidy por su apoyo.

A todos mis Amigos.

A Ronny, Jorge, Taimy, Yonelvy por su ayuda incondicional en cada momento.

A los que mencione y a los que se me quedaron por mencionar, muchas gracias a todos.

DEDICATORIA

A mi Mamá y mi Hermana por su amor, comprensión,
por darme fuerzas y apoyo en todo momento,
por ser mis guías y enseñarme el sacrificio,
que es el verdadero camino hacia el futuro añorado.

RESUMEN

Actualmente la Empresa Cubana de Aviación, específicamente la tripulación de los aviones IL-96/300 se encuentra inmersa en la automatización de sus procesos, logrando con esto realizarlos en menor tiempo, con mayor exactitud y productividad.

El objetivo fundamental de esta empresa es la seguridad de sus vuelos, en estos intervienen tres actividades fundamentales, las cuales son: Peso y Balance, Análisis de Pista y Plan de Vuelo, en las cuales se ejecutan una serie de cálculos complicados, que se llevan a cabo por los miembros de la tripulación, en los cuales se pueden cometer errores humanos.

El presente trabajo consiste en la realización de una Base de Datos permitiendo organizar, centralizar y almacenar la información, que interviene en los cálculos para la ejecución de estas actividades, dándole seguridad y confiablidad, de manera tal que se le logre una mejor gestión de vuelo de aviones IL-96/300.

ÍNDICE.

ÍNDICE

ÍNDICE DE FIGURAS.

ÍNDICE DE DIAGRAMAS.

ÍNDICE DE TABLAS.

ÍNDICE DE TABLAS DE LOS ANEXOS.

ÍNDICE

Introducción.

Cuba mediante varios convenios con la República de Rusia en los años 2005-2006 adquirió 3 aviones comerciales IL-96 serie 300, destinados a la transportación de pasajeros y cargas, mejorando así la flota de la Empresa Cubana de Aviación, prestando un mejor servicio.

Cuenta con un peso máximo de despegue y aterrizaje de 250 y 175 toneladas respectivamente, puede transportar hasta 262 pasajeros, su tripulación está compuesta por el Capitán, Piloto, Copiloto, Ingeniero de Vuelos y el Despachador.

Para la correcta operatividad de los aviones IL-96/300 se necesitan hacer una serie de cálculos, los cuales actualmente son realizados de forma manual por el responsable de esta actividad, auxiliado de una serie de tablas hechas por el propio personal; esta situación podría acarrear errores en la realización de estas operaciones que son sumamente importantes y que tienen una gran complejidad debido a la cantidad de variables involucradas en el proceso.

La empresa Cubana de Aviación, por su importancia, ha decidido la informatización de los cálculos referentes al vuelo y operatividad de los nuevos aviones IL-96/300, lo que reduciría en cierta medida algunos errores humanos que puedan resultar fatales para los pasajeros, agregándole seguridad y confiabilidad a las operaciones.

Para la informatización de de este proceso, se ideó el “Sistema Integrador de Gestión de Vuelo para Aviones IL-96/300”, el cual está compuesto por los módulos: “Sistema de Automatización para el Peso y Balance del IL-96-300”, “Sistema Automatizado para la Obtención del Plan de Vuelo del IL-96-300” y “Sistema de Automatización para el Análisis de Pista del IL-96-300”. Este sistema no posee un mecanismo que permita la organización, centralización y almacenamiento de las variables involucradas en las operaciones, de manera que sea posible gestionar la información de forma rápida y segura, constituyendo esta, la situación problémica fundamental de nuestra investigación.

A raíz de lo planteado, se reúnen las condiciones para expresar el **problema científico:** ¿Cómo organizar, centralizar y almacenar la información, de manera que se garantice el proceso de gestión del vuelo para aviones IL-96/300 en la Empresa Cubana de Aviación?

El **objeto de estudio** es la gestión automatizada de la información mediante Base de Datos (BD) relacional, el **campo de acción** se enmarca en organizar, centralizar y almacenar la información.

INTRODUCCIÓN

El **objetivo general** de este trabajo consiste en realizar la BD, logrando organizar, centralizar y almacenar la información necesaria para llevar a cabo la gestión del vuelo de aviones IL-96/300 en la Empresa Cubana de Aviación.

Para dar cumplimiento a los objetivos anteriormente expuestos se deben cumplir las siguientes tareas:

1. Analizar los requisitos funcionales y no funcionales que influyan en la realización de la BD.
2. Entrevista con los analistas del proyecto.
3. Instalación de los software necesarios.
4. Realizar la validación teórica del diseño a través de:
   - ❖ Normalización de la BD.
   - ❖ Análisis de redundancia de información.

La elaboración de la BD permitirá organizar y almacenar la información de manera centralizada, garantizando el proceso de gestión del vuelo para aviones IL-96/300 en la Empresa Cubana de Aviación.

Con este trabajo se realizará una BD en Microsoft SQL Server 2000, logrando con esta la automatización del almacenamiento y control de la información para el Sistema Integrador de Gestión de Vuelo para Aviones IL-96/300.

El presente trabajo esta compuesto por 3 Capítulos, estructurados de la siguiente manera:

Capítulo1: “Fundamentación Teórica”.

Abarca todo lo que respecta a la fundamentación teórica, en la cual se exponen las tendencias, técnica, tecnologías que son utilizadas en la actualidad en los Sistemas de Base de Datos y que son el soporte de la solución propuesta al problema especificado anteriormente, se incluye el estado del arte a nivel nacional e internacional.

Capítulo 2: “Descripción y análisis de la solución propuesta.”

Este está dedicado al análisis, diseño y acceso a datos de la BD. En dicho capítulo se incluyen el diagrama Entidad-Relación y la descripción de las tablas, se detallan los requisitos funcionales y los no funcionales, se describen las clases persistentes y las clases necesarias para el acceso a los datos.

Capítulo 3: “Validación del diseño realizado.”

Se realiza un análisis de la integridad, la redundancia de datos, la seguridad y trazabilidad de la base de datos, se describe el proceso de normalización.

Capítulo I: Fundamentación Teórica.

En este capítulo se presentan los fundamentos teóricos principales en los que se basa la presente investigación, se hace mención a un grupo de conceptos y definiciones, se realiza un análisis de la tecnología a utilizar, haciendo énfasis en los Sistema Gestor de Base de Datos (SGBD), y se analizan metodologías de desarrollo de software.

1.1 Estudio del estado del arte.

En el análisis realizado sobre los sistemas que ejecutan las funciones de: Peso y Balance, Análisis de Pista y Plan de Vuelo, se encuentra, uno a nivel nacional y otro a nivel internacional, de los que se brindarán sus respectivas características.

A nivel nacional:

SAPLAV es un sistema automatizado, desarrollado en la Ciudad Universitaria José Antonio Echeverría (CUJAE), en 1993, que tiene como objetivo fundamental la obtención del Plan de Vuelo por una ruta determinada. Fue diseñado para aviones YAK-40, YAK-42, AN-26 e IL-62.

Este sistema realiza un único proceso como se mencionó anteriormente, además fue diseñado para modelos de aviones en los que no se incluye el IL-96/300.

A nivel Internacional:

De lo que se tiene en el ámbito internacional se conoció a través de la tripulación, que en Rusia país de donde provienen los aviones IL-96/300, existe una calculadora de mano que permite realizar los cálculos referentes a Peso y Balance en conjunto con Análisis de Pista, no contempla dentro de sus opciones la obtención del Plan de Vuelo, dada la confidencialidad del asunto, no se pudo llevar a cabo el estudio de esta herramienta.

Valoración:

Debido a que los sistemas ya existentes, tanto en el país como en el exterior, no cumplen con la realización de los tres procesos fundamentales que se ejecutan para la gestión del vuelo de estos aviones, es que se decide efectuar un sistema que cometa las funciones de: Peso y Balance, Análisis de Pista y Plan de Vuelo.

1.2 Definición de Base de Datos.

Una *BD* es un conjunto de datos que tiene las siguientes propiedades implícitas:

- Representa algún aspecto del mundo real, llamado mini mundo o universo de discurso. Las modificaciones del mini mundo se reflejan en la BD.
- Es un conjunto de datos lógicamente coherentes, con un cierto significado inherente. Una colección aleatoria de datos no puede considerarse propiamente una BD.
- Una BD se diseña, construye y puebla con datos para propósito específico. Está dirigida a un grupo de usuarios y tiene ciertas aplicaciones preconcebidas que interesan a distintos usuarios.

O sea, una BD tiene:

- Una fuente de la cual se derivan los datos.
- Cierto grado de interacción con los hechos del mundo real.
- Un público activamente interesado en el contenido de la BD.
- Su tamaño es variado.
- Debe ser posible buscar, obtener y actualizar los datos siempre que sea necesario. **[1]**

1.3 Tipos de Bases de Datos.

Las bases de datos pueden clasificarse de varias maneras, se agrupan según su variabilidad en cuanto a los datos a almacenar y según el contenido, de acuerdo al criterio elegido:

Según la variabilidad de los datos almacenados:

- **Bases de datos estáticas:**

  Éstas son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

- **Bases de datos dinámicas:**

  Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones

fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub, entre otras.

Según el Contenido:

- **Bases de datos bibliográficas:**
  Solo contienen un representante de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque sino se estaría en presencia de una base de datos a texto completo. Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

- **Bases de datos de texto completo:**
  Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.

- **Directorios:**
  Un ejemplo son las guías telefónicas en formato electrónico.
  Banco de imágenes, audio, video, multimedia, etc.
  Bases de datos o "bibliotecas" de información Biológica.

  Son BD que almacenan diferentes tipos de información proveniente de las ciencias de la vida o médicas. Se pueden considerar en varios subtipos:

- Aquellas que almacenan secuencias de nucleótidos o proteínas.
- BD de estructura, comprende los registros de datos experimentales sobre estructuras 3D de biomoléculas. **[2]**

1.4 Modelos de Bases de Datos.

Además de la clasificación por la función de las bases de datos, éstas también se pueden clasificar de acuerdo a su modelo de administración de datos.

Algunos modelos utilizados en las bases de datos:

- **Bases de datos jerárquicas:**

  Este tipo de BD almacena su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas. Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.

- **Base de datos de red:**

  Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).

- **Base de datos relacional:**

  El Modelo relacional es el más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones"[1]. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas"[2]. Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de

[1] Las relaciones son estructuras formadas por filas y columnas que almacenan los datos referentes a una determinada entidad del mundo real.

[2] Representa una entidad que nosotros queremos memorizar en la base de datos. Las características de cada entidad están definidas por las columnas de las relaciones, que se llaman atributos.

imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es *Structured Query Language* o Lenguaje Estructurado de Consultas (SQL), un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales. Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

Durante los años '80 (1980-1989) la aparición de *dBASE*[3] produjo una revolución en los lenguajes de programación y sistemas de administración de datos. Aunque nunca debe olvidarse que *dBASE* no utilizaba SQL como lenguaje base para su gestión.

- **Bases de datos orientadas a objetos:**

  Este modelo, bastante reciente, y propio de los modelos informáticos orientados a objetos, trata de almacenar en la base de datos los objetos completos (estado y comportamiento).
  Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

  - Encapsulamiento - Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
  - Herencia - Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
  - Polimorfismo - Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

[3] Primer Sistema de Gestión de Base de Datos usado ampliamente para microcomputadoras.

- **Bases de datos documentales:**

  Las bases de datos documentales están concebidas para el procesamiento, captura, almacenamiento, distribución y recuperación de información vinculada con la representación del conocimiento registrado en los documentos. Se construyen con información no estructurada, tipo texto (documentos). Gestionan tipos de datos muy complejos (documentos científicos y técnicos, entre otros) y actividades muy simples como la entrada y salida de documentos (Codina, 1994)

  Poseen un potente sistema de recuperación de información.

- **Base de datos deductivas:**

  Un sistema de base de datos deductiva, es un sistema de base de datos pero con la diferencia de que permite hacer deducciones a través de inferencias. Se basa principalmente en reglas y hechos que son almacenados en la base de datos. También las bases de datos deductivas son llamadas base de datos lógica, a raíz de que se basan en lógica matemática.

- **Bases de datos Objeto-Relacionales:**

  Los modelos de datos relacionales orientados a objetos extienden el modelo de datos relacional proporcionando un sistema de tipos más rico y que se adapte mejor a las nuevas aplicaciones, que requieren guardar un tipo de datos más complejos. Permiten que los atributos de las tuplas tengan tipos complejos. Además, la base de datos es relacional por lo que conserva su rapidez y eficiencia, y permite hacer uso de nuevos elementos, como las relaciones de herencia, que modelan los objetos que serán guardados en dicha BD, por lo que el analista y el diseñador ven un modelo orientado a objetos.

- **Bases de datos distribuidas:**

  La base de datos está almacenada en varias computadoras conectadas en red. Surgen debido a la existencia física de organismos descentralizados. Esto les da la capacidad de unir las bases de datos de cada localidad y acceder así a distintas universidades, sucursales de tiendas, etcétera. **[3]**

1.5 Metodología de diseño de una Base de Datos.

El proceso de diseño de una BD no es algo fácil, es necesario tomar decisiones a distintos niveles. Para solucionar un problema se coge y divide el problema en otros más pequeños, para que sea más fácil la búsqueda de la solución para este, ya que resolver problemas más pequeños es más fácil que

resolver el problema original completo de una sola vez. El diseño de una BD se divide en diseño conceptual, diseño lógico y diseño físico.

1.5.1 Diseño Conceptual.

El primer paso para el diseño de una BD es realizar el diseño conceptual, este nace de las especificaciones de requisitos del usuario y su resultado es el esquema conceptual de la BD. Un esquema conceptual es una descripción de alto nivel de la estructura de una BD, independiente del Sistema de Gestión de Bases de Datos (SGBD) que se vaya a utilizar para su trabajo. El mismo es un lenguaje que se utiliza para la descripción de esquemas conceptuales.

En esta etapa se debe construir un esquema de la información que se usa en la empresa, independientemente de cualquier consideración física. Al mismo se le denomina esquema conceptual. Al construir el esquema, los diseñadores descubren la semántica (significado) de los datos de la empresa: encuentran entidades, atributos y relaciones.

El objetivo es comprender:

- La perspectiva que cada usuario tiene de los datos.
- La naturaleza de los datos, independientemente de su representación física.
- El uso de los datos a través de las áreas de aplicación.

El esquema conceptual se puede utilizar para que el diseñador transmita a la empresa lo que ha entendido sobre la información que ésta maneja. Para ello, ambas partes deben estar familiarizadas con la notación utilizada en el esquema. **[4]**

1.5.2 Diseño Lógico.

El Diseño Lógico parte del esquema conceptual y el resultado es el esquema lógico. Un esquema lógico es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD. El cual es un lenguaje para modelar esquemas lógicos (modelo relacional, modelo de red, etc.). Y a su vez depende del SGBD con que se manipulará la BD.

El diseño lógico es el proceso de construir un esquema de la información que utiliza la empresa, basándose en un modelo de base de datos específico, independiente del SGBD concreto que se vaya a utilizar y de cualquier otra consideración física.

En esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el SGBD que se vaya a utilizar, como puede ser el modelo relacional, el modelo de red, el modelo jerárquico o el modelo orientado a objetos. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos de usuario.

Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Ambos se deben ver como un proceso de aprendizaje en el que el diseñador va comprendiendo el funcionamiento de la empresa y el significado de los datos que maneja. El diseño conceptual y el diseño lógico son etapas clave para conseguir un sistema que funcione correctamente. Si el esquema no es una representación fiel de la empresa, será difícil, sino imposible, definir todas las vistas de usuario (esquemas externos), o mantener la integridad de la base de datos. También puede ser difícil definir la implementación física o el mantener unas prestaciones aceptables del sistema. Además, hay que tener en cuenta que la capacidad de ajustarse a futuros cambios es un sello que identifica a los buenos diseños de bases de datos. Por todo esto, es fundamental dedicar el tiempo y las energías necesarias para producir el mejor esquema que sea posible.

La normalización es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura que las relaciones obtenidas no tienen datos redundantes. Esta técnica se presenta en el capítulo dedicado al diseño lógico de bases de datos.

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos. **[5]**

1.5.3 Diseño Físico.

El Diseño Físico parte del esquema lógico y da como resultado el esquema físico. Un esquema físico es la descripción de la implementación de una BD. A esta altura ya se debe de haber definido el SGBD, ya que el esquema físico se adapta a él.

El diseño físico es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos.

Entre el diseño físico y el diseño lógico hay una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico.

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

- Obtener un conjunto de relaciones y las restricciones que se deben cumplir sobre ellas.
- Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas.
- Diseñar el modelo de seguridad del sistema. **[6]**

1.6 Análisis y Descripción de los Sistemas Gestores de Bases de Datos.

El software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina SGBD.

Existen muchas formas de organizar las BD, pero hay un conjunto de objetivos generales que deben cumplir todos los SGBD, de modo que faciliten el proceso de diseño de aplicaciones y que los tratamientos sean más eficientes y rápidos, dando la mayor flexibilidad posible a los usuarios.

Los objetivos fundamentales de los SGBD son:

- **Abstracción de la información:**
  Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

- **Independencia:**
  La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- **Redundancia mínima:**

  Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

- **Consistencia:**

  En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

- **Seguridad:**

  La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

- **Integridad:**

  Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

- **Respaldo y recuperación:**

  Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

- **Tiempo de respuesta:**

  Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados. **[7]**

1.6.1 Oracle.

Es un sistema de gestión creado por Oracle Corporation. Se considera uno de los SGBD más completos existentes:

Ventajas:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Es multiplataforma.

Desventajas:

- Su mayor inconveniente es su alto precio.
- Ha sido criticado por algunos especialistas es la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años. **[8]**

1.6.2 MySQL.

Es uno de los SGBD más populares desarrollado bajo la política de código abierto, tiene una gran ventaja en la velocidad a la hora de leer los datos pero todo tiene un costo y en este caso es que reduce un conjunto de facilidades que presentan otros SGBD como son: integridad referencial, bloqueo de registros, procedimientos almacenados, entre otros.

Ventajas:

- Diseñado en vistas a la velocidad.
- Consume muy pocos recursos de CPU[4] y memoria. Muy buen rendimiento.
- Tamaño del registro sin límite.
- Buena integración con PHP.
- Buen control de acceso usuarios-tablas-permisos.
- Utilidades de administración (phpMyAdmin).

[4] **Unidad Central de Procesamiento** , **CPU** (por sus siglas del inglés ***C**entral **P**rocessing **U**nit*), o, simplemente, el **procesador**, es el componente en una computadora digital que interpreta las instrucciones y procesa los datos contenidos en los programas de computadora.

Desventajas:

- No soporta subconsultas.
- No soporta vistas.
- No soporta transacciones.

Se hace inestable cuando contiene gran cantidad de datos. **[9]**

1.6.3 Microsoft SQL Server.

Microsoft SQL Server es un SGBD basado en el lenguaje *Transact-SQL (lenguaje de programación del SQL Server)*, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Este incluye interfaces de acceso para la mayoría de las plataformas de desarrollo.

Ventajas:

- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración.
- Permite trabajar en modo cliente-servidor donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- Potencia: Microsoft SQL Server es la mejor base de datos para Windows NT Server.
- Fácil de Usar.

Desventajas:

- Licencia con altos costos.
- Plataforma Windows. **[10]**

1.6.4 Solución Propuesta.

Después de haber abordado las características de los SGBD MySQL, Oracle y Microsoft SQL Server 2000, se ha seleccionado el SGBD Microsoft SQL Server 2000 por su gran capacidad de almacenar, organizar y gestionar la información, posee una gran seguridad. En cuanto a la licencia en la entrevista con el cliente, este expresó no haber ningún problema, pues ellos tenían una empresa que la costeaba y que otros sistemas que ellos tenían estaban montados sobre este gestor y le seria mejor a la hora de darle mantenimiento.

1.8 UML como lenguaje de modelado.

El Lenguaje Unificado del Modelado (UML) es el lenguaje de modelado de sistemas de software más popular y utilizado actualmente. Este es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software.

UML permite modelar sistemas de información, y su objetivo es lograr modelos que, además de describir con cierto grado de formalismo tales sistemas, puedan ser entendidos por los clientes o usuarios de aquello que se modela. Para ello, es muy importante que el idioma en el que estén las palabras y textos que aparezcan en tales modelos sea el propio de estas personas. **[11]**

1.9 Análisis y Descripción de herramientas para el modelado visual.

Existen múltiples herramientas para el modelado visual de un software, los cuales realizan el diseño previo a la implementación mediante diagramas que se desarrollan a medida que avanza el proyecto, utiliza un lenguaje común para todo el equipo de desarrollo asiendo así mas fácil el trabajo y la comunicación entre ellos.

1.9.1 Herramienta de modelado ER/Studio 7.0

Es una herramienta de diseño UML, de modelado de datos fácil de usar y multinivel, para el diseño y construcción de bases de datos a nivel físico y lógico se utiliza para realizar el modelado para el diseño y creación de bases de datos lógicas y físicas. Permite crear y mantener aplicaciones de bases de datos. ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes.

Las capacidades de diseño que contiene, ayudan a crear un diseño lógico que puede transformarse en cualquier número de diseños físicos. Como resultado, se puede mantener un diseño lógico normalizado mientras se no se normalizan los diseños físicos para su desempeño. ER/Studio mantiene ligas entre todos los niveles de su diseño por lo tanto puede mezclar cambios en cualquier dirección entre ellos. Revisa la normalización y la compilación con la sintaxis de la plataforma de la base de datos.

Cuenta con ingeniería de reverso, cuando necesite iniciar un trabajo de una base de datos existente, ER/Studio puede hacer una ingeniería de reverso al esquema completo para cualquier plataforma de

bases de datos. La operación de la ingeniería de reverso extrae eficientemente definiciones de objetos y construye un modelo de datos gráfico. **[12]**

1.9.2 Rational Rose.

Este es una potente herramienta para el modelado visual de un software, permite visualizar, entender, y refinar sus requerimientos y arquitectura antes de enfrentarse al código. Esto le permite evitar esfuerzos desperdiciados en el ciclo de desarrollo. Usar una sola herramienta de modelado a través del ciclo de vida del desarrollo le permite asegurar que usted está construyendo el sistema correcto. El modelo arquitectónico puede ser rastreado hacia el modelo de procesos de negocios y los requerimientos de sistema.

Esta herramienta brinda la posibilidad de construir un modelo de casos de usos bien definido, identificar los objetos y representar cómo interactúan con los diagramas de secuencia y colaboración, organizar los componentes de software y su despliegue para diseminarlos por los distintos nodos de una arquitectura de red, describir la estructura de las clases, generar el código fuente de las clases definidas en el diseño.

También se debe decir que una de sus limitaciones es que, una vez generado el diagrama de clases persistentes a partir del cual se genera la base de datos del sistema, no existe la posibilidad de que el mismo exporte ese modelo hacia algún SGBD. **[13]**

1.10 Metodologías de desarrollo de software.

Teniendo en cuanta que para el desarrollo de cualquier software, que se quiera salga con la calidad requerida es necesario guiarse por una metodología para que al final el producto satisfaga las necesidades de los clientes, una pregunta que nos debemos hacer a la hora de desarrollar un software es qué metodología de desarrollo de software utilizar.

Para definir qué metodología podemos utilizar y cual se adapta más a nuestro medio, mencionaré tres de ellas de las que se considera las más importantes, tal como: El Proceso Unificado Racional (*Rational Unified Process* en inglés, habitualmente resumido como RUP), Extreme Programing (XP) y Microsoft Solution Framework (MSF).

1. 10.1 Extreme Programing.

Es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizadas para proyectos de corto plazo. La metodología consiste en una programación rápida o extrema, cuya

particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

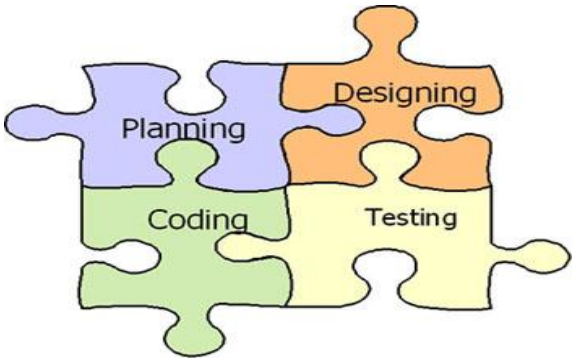

Figura 1 Representación de la metodología XP.

La metodología se basa en:

- **Pruebas Unitarias:**

  Se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos 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. 10.2 Microsoft Solution Framework.

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Figura 2 Representación de la metodología MSF.

Características:

- **Adaptable:**

  Es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.

- **Escalable:**

  Puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.

- **Flexible:**

  Es utilizada en el ambiente de desarrollo de cualquier cliente.

- **Tecnología Agnóstica:**

  Porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología. MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

- **Modelo de Arquitectura del Proyecto:**

  Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.

- **Modelo de Equipo:**

  Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles.

- **Modelo de Proceso:**

  Diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de equipo.

- **Modelo de Gestión del Riesgo:**

  Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.

- **Modelo de Diseño del Proceso:**

  Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.

- **Modelo de Aplicación:**

  Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores.

1. 10.3 Rational Unified Process.

El RUP es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

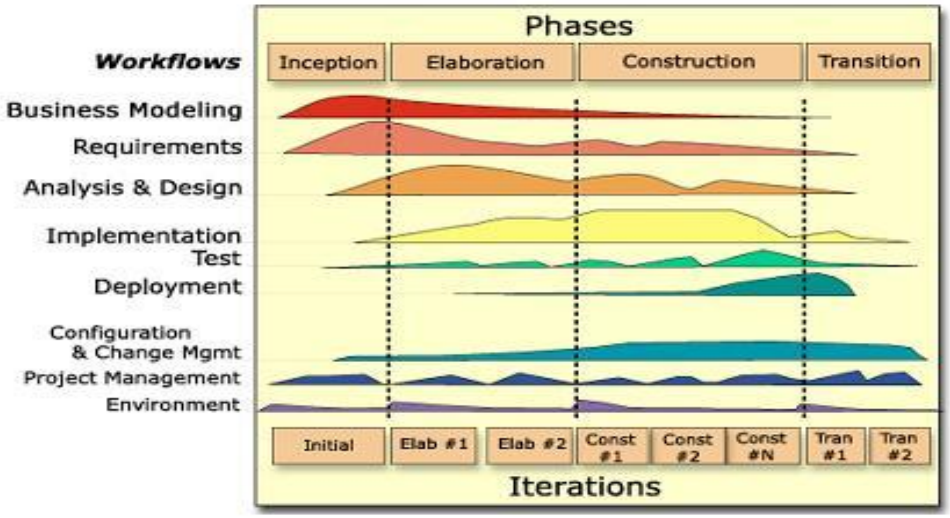

Figura 3 Representación de la metodología RUP.

El ciclo de vida RUP es una implementación del Desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- **Concepción:** se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos.
- **Elaboración:** se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.
- **Construcción:** se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.
- **Transición**: se Instala el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

Principales características:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Desarrollo iterativo e incremental.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios.
- Modelado visual del software.
- Verificación de la calidad del software.

Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso). **[14]**

1. 10.4 Solución Propuesta.

Después de haber visto las características de RUP, XP y MSF que son las metodologías de desarrollo de software más utilizadas actualmente, se ha decidido adoptar la metodología RUP dado al gran acoplamiento con el lenguaje de modelado a utilizar que es el UML, es útil pues es un proceso iterativo e incremental permitiendo rectificar errores antes cometidos, obteniéndose al final un producto de gran calidad.

1.11 Conclusiones.

En este capítulo se analizaron las tecnologías a utilizar para dar cumplimiento a la propuesta de solución, así como algunos conceptos y tendencias en las cuales nos apoyaremos a la hora de la toma de algunas decisiones. El documento hizo gran énfasis en los modelos de datos como base conceptual a la hora del diseño da aplicaciones que hacen un uso agudo de los datos; respecto a este tema nos inclinamos por el Modelo de Datos Relacional como un modelo simple y efectivo para la manipulación de los datos, además de estar sujeto a sólidas bases teóricas y fundamentos planteados con anterioridad. Fue fundamentada la selección de las herramientas y metodologías escogidas para la elaboración de la solución.

Capítulo II: Descripción y Análisis de la Solución Propuesta.

Este capítulo constituye la parte esencial del presente trabajo. En él se exponen los requisitos funcionales y los no funcionales, los diseños lógico y físico, así como los diagramas de clases y la descripción de las clases que los componen, se realizó una descripción y fundamentación de la arquitectura, el análisis de optimización de consultas.

2.1 Descripción de la arquitectura Microsoft SQL Server.

Hay tres características importantes inherentes a los sistemas de bases de datos: la separación entre los programas de aplicación y los datos, el manejo de múltiples vistas por parte de los usuarios y el uso de un catálogo para almacenar el esquema de la base de datos. En 1975, el comité ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee) propuso una arquitectura de tres niveles para los sistemas de bases de datos, que resulta muy útil a la hora de conseguir estas tres características.

El objetivo de la arquitectura de tres niveles es el de separar los programas de aplicación de la base de datos física. Arquitectura donde el esquema de una base de datos se define en tres niveles distintos de abstracción:

- En el **nivel interno** se describe la estructura física de la base de datos mediante un esquema interno. Este esquema se especifica mediante un modelo físico y describe todos los detalles para el almacenamiento de la base de datos, así como los métodos de acceso.
- En el **nivel conceptual** se describe la estructura de toda la base de datos para una comunidad de usuarios (todos los de una empresa u organización), mediante un esquema conceptual. Este esquema oculta los detalles de las estructuras de almacenamiento y se concentra en describir entidades, atributos, relaciones, operaciones de los usuarios y restricciones. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar el esquema.
- En el **nivel externo** se describen varios esquemas externos o vistas de usuario. Cada esquema externo describe la parte de la base de datos que interesa a un grupo de usuario determinado y esconde a ese grupo el resto de la base de datos. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar los esquemas.

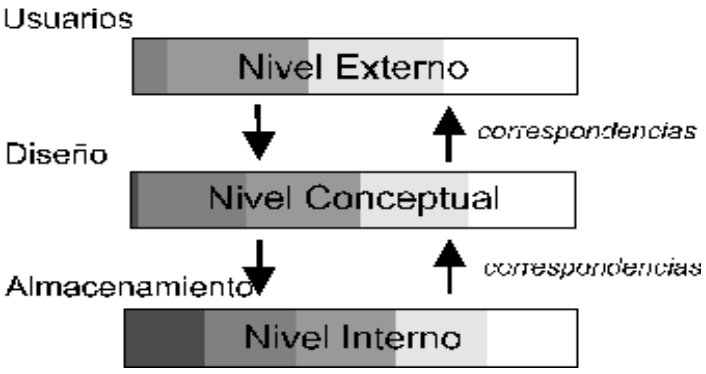

Figura 4 Arquitectura del SGBD Microsoft SQL Server 2000.

Hay que destacar que los tres esquemas no son más que descripciones de los mismos datos pero con distintos niveles de abstracción. Los únicos datos que existen realmente están a nivel físico, almacenados en un dispositivo como puede ser un disco. En un SGBD basado en la arquitectura de tres niveles, cada grupo de usuarios hace referencia exclusivamente a su propio esquema externo. Por lo tanto, el SGBD debe transformar cualquier petición expresada en términos de un esquema externo a una petición expresada en términos del esquema conceptual, y luego, a una petición en el esquema interno, que se procesará sobre la base de datos almacenada.

La arquitectura de tres niveles es útil para explicar el concepto de independencia de datos que podemos definir como la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior. Se pueden definir dos tipos de independencia de datos:

La **independencia lógica** es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la base de datos o para reducirla. Si, por ejemplo, se reduce la base de datos eliminando una entidad, los esquemas externos que no se refieran a ella no deberán verse afectados.

- La **independencia física** es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual (o los externos). Por ejemplo, puede ser necesario reorganizar ciertos ficheros físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización de datos. Dado que la independencia física se refiere sólo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento, es más fácil de conseguir que la independencia lógica.

En los SGBD que tienen la arquitectura de varios niveles es necesario ampliar el catálogo o diccionario, de modo que incluya información sobre cómo establecer la correspondencia entre las peticiones de los usuarios y los datos, entre los diversos niveles. El SGBD utiliza una serie de procedimientos adicionales para realizar estas correspondencias haciendo referencia a la información de correspondencia que se encuentra en el catálogo. La independencia de datos se consigue porque al modificarse el esquema en algún nivel, el esquema del nivel inmediato superior permanece sin cambios, sólo se modifica la correspondencia entre los dos niveles. No es preciso modificar los programas de aplicación que hacen referencia al esquema del nivel superior.

Para plasmar los tres niveles en el enfoque o modelo de datos seleccionado, es necesaria una aplicación que actúe de interfaz entre el usuario, los modelos y el sistema físico.

Esta es la función que desempeñan los SGBD, ya reseñados, y que pueden definirse como un paquete generalizado de software, que se ejecuta en un sistema computacional anfitrión, centralizando los accesos a los datos y actuando de interfaz entre los datos físicos y el usuario.

Las principales funciones que debe cumplir un SGBD se relacionan con la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad.

El SGBD incorpora como herramienta fundamental dos lenguajes, para la definición y la manipulación de los datos. El lenguaje de definición de datos (DDL, Data Definition Language) provee de los medios necesarios para definir los datos con precisión, especificando las distintas estructuras. Acorde con el modelo de arquitectura de tres niveles, habrá un lenguaje de definición de la estructura lógica global, otro para la definición de la estructura interna, y un tercero para la definición de las estructuras externas. **[15]**

2.2 Análisis de Optimización de Querys.

Las *querys* no son más que un lenguaje de consulta de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas, define qué es lo que quiere y cómo conseguirlo.

La complejidad que puede alcanzar una consulta determinada puede ser tal, que se tome para su ejecución un tiempo considerable, obteniendo como resultado no siempre una respuesta óptima. Para mejorar esto se utilizan los índices, estos son una estructura de datos que mejora la velocidad de las

operaciones, permitiendo un rápido acceso a los registros de una tabla. Se suelen usar sobre aquellos campos sobre los cuales se hagan frecuentes búsquedas.

El índice en las bases de datos tiene un funcionamiento similar al índice de un libro, guardando parejas de elementos: el elemento que se desea indexar y su posición en la base de datos. Para buscar un elemento que esté indexado, sólo hay que buscar en el índice dicho elemento para, una vez encontrado, devolver el registro que se encuentre en la posición marcada por el índice.

El espacio en disco requerido para almacenar el índice es típicamente menor que el espacio de almacenamiento de la tabla (puesto que los índices generalmente contienen solamente los campos clave de acuerdo con los que la tabla será ordenada, y excluyen el resto de los detalles de la tabla), lo que da la posibilidad de almacenar en memoria los índices de tablas que no cabrían en ella. En una base de datos relacional un índice es una copia de parte de una tabla.

El uso de índices tiene sus pro y sus contra, como se ha descrito estos son muy buenos para acelerar el tiempo de respuesta de una consulta, por otro lado consumen una memoria adicional, lo cual dependiendo de la BD, uno se complementa el uno con el otro y conlleva a un buen rendimiento. **[16]**

2.3 Requisitos Funcionales y No Funcionales.

2.3.1 Requisitos Funcionales.

RF 1: Autenticar Usuario.

RF 2: Gestionar Usuarios.

- **RF 2.1:** Adicionar Usuario.
- **RF 2.2:** Eliminar Usuario.
- **RF 2.3:** Modificar Usuario.
- **RF 2.4:** Listar Usuarios.
- **RF 2.5:** Cambiar Contraseña.
- **RF 2.6:** Asignar Rol.

RF 3: Gestionar Roles.

RF 3.1: Adicionar Rol.

RF 3.2: Eliminar Rol.

RF 3.3: Modificar Rol.

RF 3.4: Listar Rol.

RF 4: Gestionar historial.

RF 4.1: Limpiar historial.

RF 4.1.1: Limpiar historial completo.

RF 4.1.2: Limpiar historial por fecha.

RF 4.1.3: Limpiar historial entre fechas.

RF 4.1.4: Limpiar historial por usuario.

RF 5: Gestionar Aeropuerto.

RF 5.1: Adicionar Aeropuerto.

RF 5.2: Modificar Aeropuerto.

RF 6: Gestionar Avión.

RF 6.1: Adicionar Avión.

RF 6.2: Modificar Avión.

RF 7: Gestionar Puntos de Control.

RF 7.1: Adicionar Puntos de Control.

RF 7.2: Modificar Puntos de Control.

RF 8: Gestionar Rutas.

RF 8.1: Insertar Rutas.

RF 8.2: Modificar Rutas.

RF 9: Gestionar País.

RF 9.1: Insertar País.

RF 9.2: Modificar País.

RF 10: Gestionar Ciudad.

RF 10.1: Insertar Ciudad.

RF 10.2: Modificar Ciudad.

RF 11: Gestionar Crew.

RF 11.1: Adicionar Crew.

RF 11.2: Modificar Crew.

RF 12: Gestionar Versión.

RF 12.1: Adicionar Versión.

RF 12.2: Modificar Versión.

RF 13: Gestionar Galley.

RF 13.1: Adicionar Galley.

RF 13.2: Modificar Galley.

2.3.2 Requisitos no Funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Para su definición se tienen en cuenta algunas propiedades de factibilidad, usabilidad, confiabilidad, entre otros que el producto debe cumplir.

Requisitos de Usabilidad

- Se debe reducir el tiempo de respuesta a una petición del usuario a la BD.
- La BD poseerá una gran confiabilidad y disponibilidad en cuanto a la información almacenada en ella.

Requisito de Portabilidad

- Es sistema será usado sobre el Sistema Operativo Windows.

Requisito de Confiabilidad

- El sistema será solo accedido por los usuarios que tengan el permiso requerido.

Requisitos de Rendimiento

- Debe permitir la conexión de varios usuarios a la BD.
- Debe darle respuesta a todos los pedidos en corto plazo de tiempo.

Requisitos de Soporte

- Prueba del Sistema, además de un entrenamiento a los futuros usuarios.
- Se trazara una estrategia de mantenimiento de la BD como estrategia de soporte.

Requisitos de Seguridad

- Debe contener varios roles con diferentes niveles de acceso a las distintas áreas.
- Del acceso al sistema estará encargado el Subsistema de Seguridad, el cual se encargara de la gestión de los usuarios y roles.
- Los usuarios que accedan a la aplicación solo tendrán acceso según el rol que le corresponda.
- La información almacenada en la BD será confidencial, por lo que no debe ser consultada por personal no autorizado, ni divulgada.

2.4 Diagrama de Clases Persistentes.

Las clases persistentes son aquellas que sus datos perduran en el tiempo y deben almacenarse en una Base de Datos, al contrario de las clases temporales, las cuales son creadas y manejadas por el sistema en tiempo de ejecución y al terminar este, dejan de existir.

El diagrama de clases persistentes tiene como base el diagrama de clases del diseño, de donde se seleccionan las clases que van a persistir y se confecciona el mismo. A continuación se muestra el diagrama de clases persistentes.

DIAGRAMA DE CLASES PERSISTENTES.

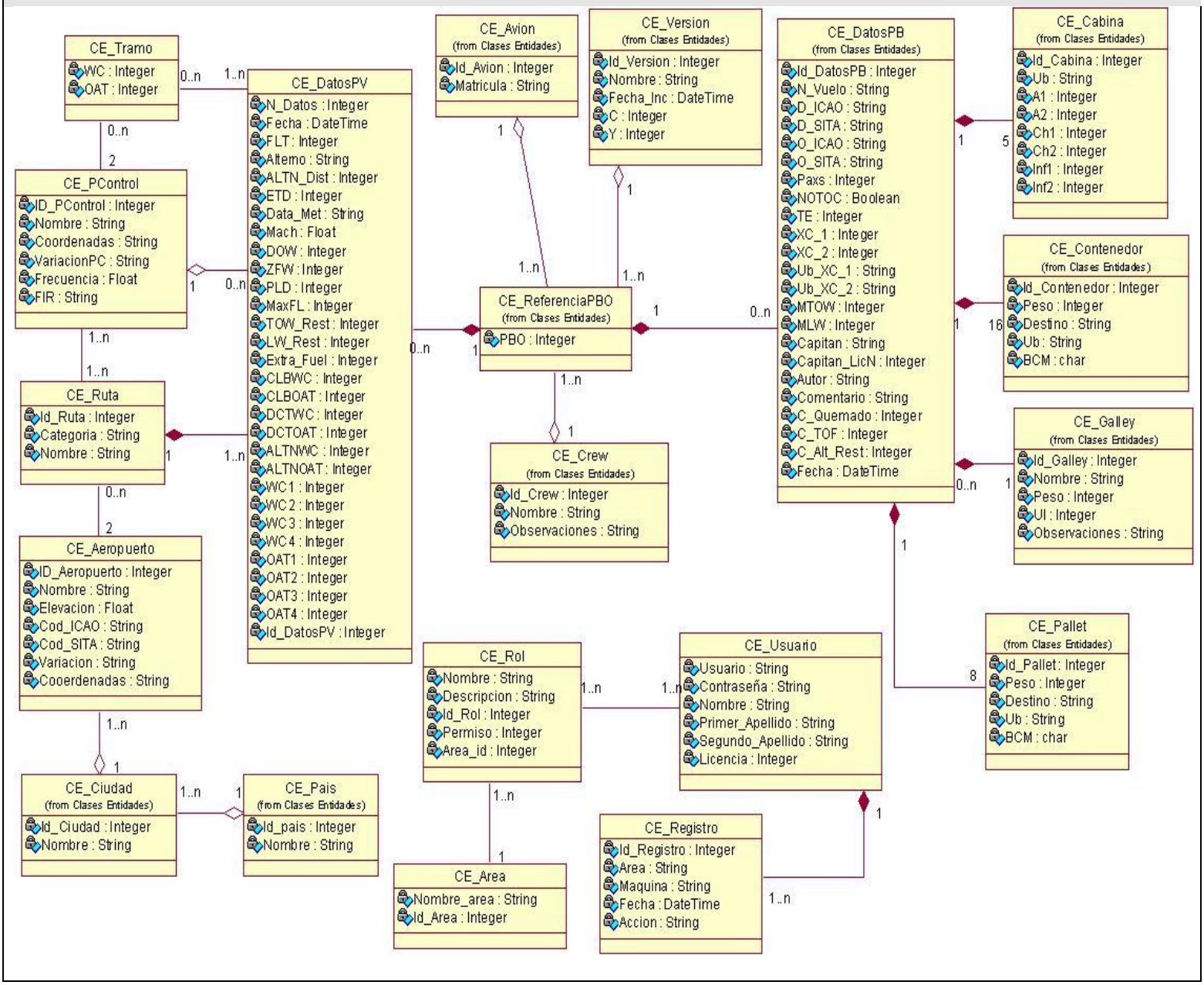

Diagrama 1 Diagrama de clases Persistentes.

2.4.1 Descripción de las Clases.

A continuación se describirán cada una de las clases persistentes, describiendo así sus atributos y métodos.

Nombre: CE_Area	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Nombre_Area	satring
Id_Area	int
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Area()	Constructor vacio, para la creación de objetos de tipo CE_Area.
CE_Area_Dominio(string nombre_area,int id)	Constructor con parámetros, para la creación de objetos de tipo CE_Area.
P_Nombre_Area	Propiedad para la asignación y obtención del atributo nombre_area.
P_Id_Area	Propiedad para la asignación y obtención del atributo Id_Area.

Tabla 1 Descripción de la clase persistente CE_Area.

Nombre: CE_Registro	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Registro	int
Accion	string
Usuario	string
Modulo	string
Area	string
Fecha	DateTime
Maquina	string
Para cada responsabilidad:	
Nombre:	**Descripción:**
Registro()	Constructor vacio, para la creación de objetos de

	tipo CE_Registro
Registro(int id_registro, string maquina, string area, string accion, string usuario, DateTime fecha)	Constructor con parámetros, para la creación de objetos de tipo CE_Registro
P_Id_Registro	Propiedad para la asignación y obtención del atributo Id_Registro
P_Maquina	Propiedad para la asignación y obtención del atributo Maquina
P_ Area	Propiedad para la asignación y obtención del atributo Área.
P_ Accion	Propiedad para la asignación y obtención del atributo Acción.
P_ Usuario	Propiedad para la asignación y obtención del atributo Usuario.
P_ Fecha	Propiedad para la asignación y obtención del atributo Fecha.

Tabla 2 Descripción de la clase persistente CE_Registro.

Nombre: CE_Rol	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Rol	int
Nombre	string
Descripcion	string
Id_Area	int
Permiso	int
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Rol()	Constructor vacio, para la creación de objetos de tipo CE_Rol
CE_Rol (string nombre, string descripcion,int permiso,int id,int area_id)	Constructor con parámetros, para la creación de objetos de tipo CE_Rol
P_ Nombre	Propiedad para la asignación y obtención del

	atributo Nombre
P_ Descripcion	Propiedad para la asignación y obtención del atributo Descripción
P_ Permiso	Propiedad para la asignación y obtención del atributo Permiso
P_ Id_Rol	Propiedad para la asignación y obtención del atributo Id_Rol
P_ Id_Area	Propiedad para la asignación y obtención del atributo Id_Area

Tabla 3 Descripción de la clase persistente CE_Rol.

Nombre: CE_Usuario	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Licencia	int
Usuario	string
Contraseña	string
Nombre	string
Primer_Apellido	string
Segundo_Apellido	string
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Usuario()	Constructor vacio, para la creación de objetos de tipo CE_Usuario
CE_Usuario(string usuario,string contraseña, string nombre, string primer_apellido, string segundo_apellido,int licencia)	Constructor con parámetros, para la creación de objetos de tipo CE_Usuario
P_Usuario	Propiedad para la asignación y obtención del atributo Usuario
P_Contraseña	Propiedad para la asignación y obtención del atributo Contraseña
P_Nombre	Propiedad para la asignación y obtención del atributo Nombre

P_Primer_Apellido	Propiedad para la asignación y obtención del atributo Primer_Apellido
P_Segundo_Apellido	Propiedad para la asignación y obtención del atributo Segundo_Apellido
P_Licencia	Propiedad para la asignación y obtención del atributo Licencia

Tabla 4 Descripción de la clase persistente CE_Usuario.

Para consultar las restantes descripciones de las clases persistentes ver Anexo I.

2.5 Diseño de la base de datos.

2.5.1 Diagrama Entidad Relación de la BD.

La BD cuenta con 23 tablas, de las cuales a continuación se mostrará sus descripciones, describiendo cada atributo, su tipo de dato y significado.

Los pasos seguidos para realizar el Diagrama Entidad Relación (D.E.R) fueron los siguientes:

- Reunirse con los analistas del sistema y dejar claros los parámetros y objetivos del proceso a modelar.
- Identificar los conjuntos de entidades útiles para modelar el proceso.
- Identificar los conjuntos de interrelaciones y determinar su grado y tipo (1:1, 1: n o m: n).
- Trazar un primer D.E.R.
- Identificar atributos y dominios para los conjuntos de entidades e interrelaciones.
- Seleccionar las claves principales para los conjuntos de entidades.
- Verificar que el modelo resultante cumple el planteamiento las peticiones de los clientes que en este caso es lo representado por los analistas.
- Una vez obtenida la versión correcta del modelo traducir el diagrama a un modelo lógico.

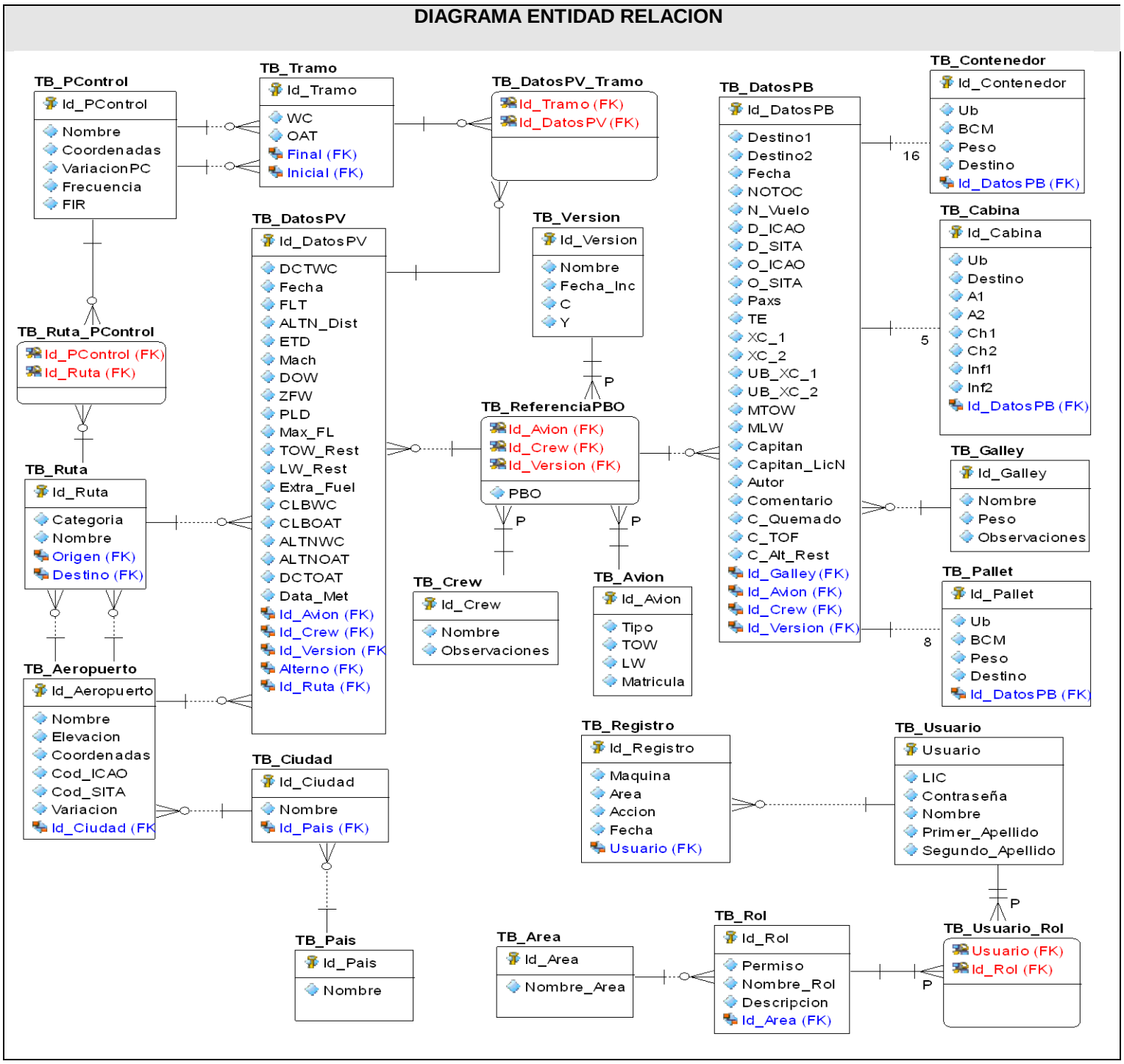

Diagrama 2 Diagrama Entidad Relación.

2.5.1.1 Descripción de las clases.

Nombre: TB_Usuario		
Descripción: Almacena los valores necesarios de los usuarios, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Usuario	varchar	Identificador que usará la persona que accederá al sistema.
Nombre	varchar	Nombre de la persona que accederá al sistema.
Contraseña	varchar	Clave de acceso al sistema del usuario.
Primer Apellido	varchar	Primer Apellido del usuario.
Segundo Apellido	varchar	Segundo Apellido del usuario.
LIC	int	Licencia del Usuario.

Tabla 5 Descripción de la Tabla Usuario.

Nombre: TB_Usuario_Rol		
Descripción: Contiene cada Usuario con su(s) Rol(s) correspondiente(s).		
Atributo	**Tipo**	**Descripción**
Usuario	varchar	Identificador que usará la persona que accederá al sistema.
ID_Rol	int	Este es identificador del Rol, es un auto numérico.

Tabla 6 Descripción de la Tabla Usuario_Rol.

Nombre: TB_Rol		
Descripción: Almacena los datos y descripción de un Rol.		
Atributo	**Tipo**	**Descripción**
ID_Rol	int	Este es identificador del Rol, es un auto numérico.
Nombre_Rol	varchar	Este es el nombre del Rol.
Descripción	varchar	Aquí se realiza toda la descripción sobre el Rol, todo lo que permite.
Permiso	int	Valor entero, 1 para permiso de escritura-lectura y 0

		de lectura.
Id_Area	int	El identificador del Área correspondiente.

Tabla 7 Descripción de la Tabla Rol.

Nombre: TB_Registro		
Descripción: Almacena las acciones realizadas por los usuarios en un área, una fecha y maquina determinada.		
Atributo	**Tipo**	**Descripción**
Id_Registro	int	Este es identificador del Registro, es un auto numérico.
Maquina	varchar	Maquina desde donde se realizó la acción.
Area	varchar	Área donde se realizó la acción.
Accion	varchar	Acción realizada.
Fecha	DateTime	Fecha en que se realizó la acción.

Tabla 8 Descripción de la Tabla Registro.

Nombre: TB_Area		
Descripción: Almacena el nombre de las Arias.		
Atributo	**Tipo**	**Descripción**
Id_Area	int	Identificador del Área, es un auto numérico. .
Nombre_Area	varchar	Nombre del Área.

Tabla 9 Descripción de la Tabla Área.

Para consultar las restantes descripciones de las tablas del Diagrama Entidad Relación ver Anexo II.

2.7 Conclusiones.

En este capítulo se ha expuesto diagrama de clases persistentes con la descripción de cada una de sus clases, el Modelo Entidad Relación con las descripciones de sus tablas, se han descrito las clases de acceso a datos. Logrando obtener finalmente la base de datos propuesta.

Capítulo III: Validación del Diseño Realizado.

Cuando se pretende realizar un buen diseño de BD se debe validar el mismo, o sea, avalar la integridad, realizar la normalización, eliminar redundancia e implementar un mecanismo de seguridad. Se llevó a tercera forma normal la BD, una vez terminado este proceso se realizó un análisis de redundancia para eliminar el riesgo de la presencia de la misma.

3.1 Integridad.

La implementación de la integridad relacional de una BD consiste en establecer las reglas de consistencia correspondientes a los datos requeridos de cada tabla, el chequeo de los valores validos y únicos, así como la integridad referencial de las llaves primarias y foráneas.

Tipos de restricciones en base de datos relacionales.

- **Datos requeridos:** Establece que una columna tenga un valor no nulo. Se define efectuando la declaración de una columna es NOT NULL, cuando la tabla que contiene las columnas se crea por primera vez, como parte de la sentencia CREATE TABLE.
- **Chequeo de Validez:** Cuando se crea una tabla cada columna tiene un tipo de dato y el SGBD chequea que solo los tipo de dato especificado sean insertado en la tabla.
- **Integridad de entidades:** Consiste en que los atributos que componen una llave primaria deben estar definidos y ser únicos para cada fila de la tabla.
- **Integridad Referencial:** Los atributos que hacen referencia a una llave primaria de otra relación tienen que tomar uno de los valores definidos para esa llave o estar indefinidos (relaciones padre/hijo).

3.2 Normalización de la Base de Datos.

La normalización es un proceso mediante el cual se descomponen relaciones que no cumplen con una serie de requisitos en otras más pequeñas que lo hagan. Tiene como objetivo la eliminación de la redundancia y anomalías a la hora de efectuar la inserción, actualización y eliminación de campos en tablas.

Mientras transcurría este proceso se fue comprobando que cada relación (tabla) cumpliera con un conjunto de reglas basadas en la clave primaria y las dependencias funcionales. Se llevó a cabo una serie de pasos donde cada uno corresponde a una forma normal.

3.2.1 Primera Forma Normal (1FN).

Para la obtención de la 1FN se eliminaron los grupos repetitivos. Un grupo repetitivo es el atributo o conjunto de estos que tienen múltiples valores para cada tupla de la relación.

Para su eliminación se puso cada uno en una relación aparte, que heredara de la clave primaria de la relación en que se encontraban. Después de este paso se logró que la BD estuviera en 1FN.

3.2.2 Segunda Forma Normal (2FN).

Una relación esta en 2FN si y solo si esta en 1FN y, además, cada atributo que no esta en la clave primaria es totalmente dependiente de esta.

Para su realización se eliminaron todas las dependencias parciales, se colocaron en una nueva relación con una copia de su determinante (los atributos de la clave primaria de los que depende). Después de este proceso la BD quedo en 2FN, ya que esta en 1FN, que es lo primero que debe cumplir y se logró que cada atributo no primo (atributos que no pertenecen a la llave primaria) fuera totalmente dependiente de la clave primaria.

3.2.3 Tercera Forma Normal (3FN).

Una relación esta en 3FN, si y solo si, esta en 2FN y, además, cada atributo que no pertenece a la llave primaria no depende transitivamente de esta.

Para su realización se eliminaron todas las dependencias transitivas, para así lograr la 3FN, y eliminar las redundancias que aun quedaban de la 2FN. Para esto, se eliminaron los atributos de cada relación que dependían transitivamente y se pusieron en relaciones nuevas con copias de sus determinantes.

Finalmente la BD quedó en 3FN, ya cumple la primera condición, que es estar en 2FN y además cada atributo que no pertenece a la llave primaria no depende transitivamente de la misma.

3.3 Análisis de Redundancia.

Al diseñar una BD se debe tener en cuenta que inevitablemente ocupará espacio en memoria. Es por ello la importancia de evitar tener datos repetidos. A este almacenamiento de información repetida se le conoce como redundancia de la información.

Esto conllevaría a un gasto de memoria innecesario, influye en un mayor coste y mayor tiempo de acceso a datos. Después de llevar la BD a 3FN quedó libre de redundancia.

3.4 Análisis de la Seguridad de la Base de Datos.

En las base de datos un aspecto esencial es la seguridad, para evitar cualquier acceso no autorizado, con la intención de modificar, usar y/o divulgar información almacenada en la misma.

Para la implementación de la seguridad en la BD se confeccionó un sistema de tablas donde se almacena toda la información referida a los usuarios que interactúan con el sistema, el cual, dado a que existen distintos roles los cuales poseen un permiso, que accionan sobre determinadas áreas de trabajo y estas acciones son registradas para que quede constancia de cada una de ellas esta compuesto por cuatro tablas principales las cuales son: TB_Rol, TB_Area, TB_Usuario y TB_Registro.

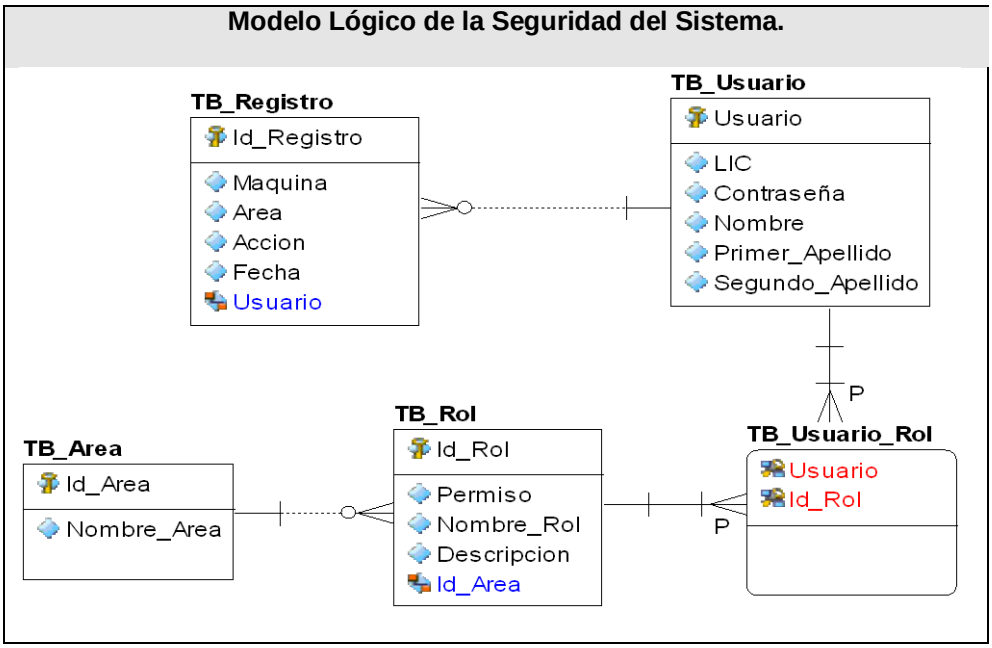

Figura 2. 1 Modelo Lógico de la Seguridad del Sistema.

Interrelaciones:

- Un Área puede ser accedida por ninguno o varios roles.
- Un Rol trabaja sobre una sola Área.
- Un Rol tiene uno o varios usuarios y viceversa.
- Un Usuario tiene ninguno o varios Registros y un Registro tiene un Usuario.

3.5 Trazabilidad de la Transacciones.

En un sistema es importante mantener el control sobre todas las acciones que los usuarios realizan sobre la BD, por ejemplo, si un usuario actualiza un registro, debe quedar constancia de que acción ejecutó, la fecha y hora, la maquina, el área, donde realizó el cambio. Para así de esta forma lograr un seguimiento de todas las acciones ejecutadas, conformando un historial de estas.

Con el objetivo del seguimiento de las trazas, en la BD se creó una tabla llamada TB_Registro en la cual se almacenaran la máquina, fecha y hora, acción y el área por cada acción que ejecute un usuario determinado. Teniendo así archivadas todas las trazas en esta tabla.

3.6 Conclusiones.

En el presente capítulo se realizó la validación teórica de la BD, se tuvo en cuenta la integridad de los datos, se eliminó la redundancia de la información normalizando la BD, garantizando así un diseño óptimo de la misma.

Conclusiones Generales.

Con el establecimiento exitoso del sistema, se dará solución a los problemas actuales existentes en la empresa Cubana de Aviación, específicamente para la tripulación que opera los aviones IL96/300, lo cual implicaría un mejoramiento en las condiciones de trabajo de esta, logrando eliminar errores en los cálculos efectuados y una reducción de costos relacionados con los recursos materiales que se utilizaban.

Se dotó al “Sistema Integrador de Gestión de Vuelo para Aviones IL-96/300” de una Base de Datos garantizando el almacenamiento, la organización y centralización de la información que intervendrá en la correcta realización de los cálculos necesarios para la ejecución de las tres actividades fundamentales que se ejecutan en la operatividad de los aviones.

Recomendaciones.

1) Estudiar la posibilidad de migración hacia software libre según las políticas actuales del país.
2) Instruir al personal que operará con el sistema.
3) Realizarle pruebas al sistema durante un período de tiempo, antes de su puesta en funcionamiento, para comprobar los resultados obtenidos, dada la importancia de la seguridad de los vuelos.
4) Investigar la posibilidad de realizar un sistema que agrupe todos los modelos de aviones existentes en la Empresa Cubana de Aviación, con las mismas funcionalidades del sistema expuesto.
5) Agregarle a la base de datos las tablas relacionadas con el Módulo de Análisis de Pista.

Referencia Bibliográfica.

[1]. Caballero, William Bravo y Trujillo, Rodolfo Ávila. *Diseño de la base de datos para el módulo de IPC de la Oficina Nacional de Estadísticas.* Ciudad de La Habana : s.n., Junio 2007.

[2]. Wikipedia. [En línea] 19 de 11 de 2007. [Citado el: 7 de 12 de 2007.] http://es.wikipedia.org/wiki/Base_de_datos .

[3]. Wikipedia. [En línea] [Citado el: 23 de 11 de 2007.] http://es.wikipedia.org/wiki/Base_de_datos .

[4]. Andrés, María Mercedes Marqués. *Diseño Conceptual*. [En línea] 12 de 02 de 2001. [Citado el: 26 de 11 de 2007.] http://www3.uji.es/~mmarques/f47/apun/node69.html .

[5]. —. *Diseño Lógico*. [Online] 02 11, 2001. [Cited: 11 26, 2007.] http://www3.uji.es/~mmarques/f47/apun/node70.html .

[6]. —. *Diseño Físico.* [En línea] 02 de 12 de 2001. [Citado el: 26 de 11 de 2007.] http://www3.uji.es/~mmarques/f47/apun/node71.html .

[7]. Wikipedia. [En línea] 30 de 11 de 2007. [Citado el: 07 de 12 de 2007.] http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos#Objetivos .

[8]. Wikipedia. [En línea] [Citado el: 25 de 11 de 2007.] http://es.wikipedia.org/wiki/Oracle .

[9]. Wikipedia. [En línea] [Citado el: 23 de 11 de 2007.] http://es.wikipedia.org/wiki/MySQL .

[10]. Wikipedia. [En línea] [Citado el: 23 de 11 de 2007.] http://es.wikipedia.org/wiki/SQLServer .

[11]. Castellanos, Yeneirys Hernández y García., Wendysh Pérez. *Diseño de Bases de Datos para la Intranet 2.* Ciudad de la Habana : s.n., junio 2007.

[12]. Ídem a la Referencia 1.

[13]. Rumbaugh, James, Jacobson, Ivar y Booch, Grady. *El Lenguaje Unificado de Modelado. Manual de Referencia.*

[14]. María A. Mendoza Sanchez. informatizate. [En línea] 07 de 06 de 2004. [Citado el: 07 de 12 de 2007.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

REFERENCIAS

[15]. María Mercedes Marqués Andrés . Arquitectura de los sistemas de bases de datos. [En línea] 12 de 02 de 2001. [Citado el: 02 de 02 de 2008.] http://www3.uji.es/~mmarques/f47/apun/node33.html .

[16]. Wikipedia. [En línea] 26 de 05 de 2008. [Citado el: 24 de 06 de 2008.] http://es.wikipedia.org/wiki/%C3%8Dndice_(base_de_datos) .

Bibliografía

1. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** *El Lenguaje Unificado de Modelado. Manual de Referencia.*

2. **GARCIA, LIC. ROSA MARIA MATO.** *Diseño de Base de Datos.* Octubre 1999.

3. **Castellanos, Yeneirys Hernández y García., Wendysh Pérez.** *Diseño de Bases de Datos para la Intranet 2.* Ciudad de la Habana : s.n., junio 2007.

4. **Caballero, William Bravo y Trujillo, Rodolfo Ávila.** *Diseño de la base de datos para el módulo de IPC de la Oficina Nacional de Estadísticas.* Ciudad de La Habana : s.n., Junio 2007.

5. **Andrés, María Mercedes Marqués.** Diseño Físico. [En línea] 02 de 12 de 2001. [Citado el: 26 de 11 de 2007.] http://www3.uji.es/~mmarques/f47/apun/node71.html .

6. —. Diseño Conceptual. [En línea] 12 de 02 de 2001. [Citado el: 26 de 11 de 2007.] http://www3.uji.es/~mmarques/f47/apun/node69.html .

7. Wikipedia. [En línea] 26 de 05 de 2008. [Citado el: 24 de 06 de 2008.] http://es.wikipedia.org/wiki/%C3%8Dndice_(base_de_datos) .

8. Wikipedia. [En línea] 30 de 11 de 2007. [Citado el: 07 de 12 de 2007.] http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_base_de_datos#Objetivos .

9. Wikipedia. [En línea] 19 de 11 de 2007. [Citado el: 7 de 12 de 2007.] http://es.wikipedia.org/wiki/Base_de_datos .

10. Wikipedia. [En línea] [Citado el: 23 de 11 de 2007.] http://es.wikipedia.org/wiki/SQLServer .

11. Wikipedia. [En línea] [Citado el: 23 de 11 de 2007.] http://es.wikipedia.org/wiki/MySQL .

12. Wikipedia. [En línea] [Citado el: 25 de 11 de 2007.] http://es.wikipedia.org/wiki/Oracle .

13. Wikipedia. [En línea] [Citado el: 23 de 11 de 2007.] http://es.wikipedia.org/wiki/Base_de_datos .

14. **María A. Mendoza Sanchez.** informatizate. [En línea] 07 de 06 de 2004. [Citado el: 07 de 12 de 2007.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

BIBLIOGRAFÍA

15. Diseño Lógico. [En línea] 02 de 11 de 2001. [Citado el: 26 de 11 de 2007.] http://www3.uji.es/~mmarques/f47/apun/node70.html .

16. **María Mercedes Marqués Andrés .** Arquitectura de los sistemas de bases de datos. [En línea] 12 de 02 de 2001. [Citado el: 02 de 02 de 2008.] http://www3.uji.es/~mmarques/f47/apun/node33.html .

17. **D.E.Perry.** *Software Architecture and its relevance for Software Engineering.* 1997.

18. **Estrategia Magazine.** *Año 3- Num. 54- Sección tecnológica*. [En línea] [Citado el: 6 de 05 de 2008.] http://www.e-estrategia.com.ar .

19. **GARCIA, LIC. ROSA MARIA MATO.** *Diseño de Base de Datos.* Octubre 1999.

20. **Elmasri, R.; Navathe.** S.B. *Fundamentos de Sistemas de Bases de Datos. 3ª Edición. Addison-Wesley, Pearson Educación*. 2002.

21. **Elmasri, R.; Navathe.** S.B. *Sistemas de bases de datos. Conceptos fundamentales. 2ª Edición.Addison-Wesley Iberoamerican.* 1997.

22**. Date C. J.** *Introducción a los sistemas de Bases de Datos, primera parte. La Habana, Editorial Félix Varela*. 2003.

23. **CORPORATION, D.** *Modelado de bases de datos*. [En línea] [Citado el: 6 de 06 de 2008.] http://www.danysoft.info/free/model2.pdf .

24. **Ortiz, A.M.** *Bases de Datos. Modelos de Datos*. [En línea] 2000 [Citado el: 20 de 07 de 2008] http://elies.rediris.es/elies9/index-4.htm .

25. —. *El Enfoque Relacional*. [En línea] 2000 [Citado el: 20 de 07 de 2008]: http://elies.rediris.es/elies9/4-2-3.htm .

Anexo I: Descripción de las clases persistentes.

Nombre: CE_Cabina	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Cabina	int
Id_DatosPB	int
Ub	string
A1	int
A2	int
Ch1	int
Ch2	int
Inf1	int
Inf2	int
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Cabina()	Constructor vacio, para la creación de objetos de tipo CE_Cabina.
CE_Cabina(int Id_Cabina, int Id_DatosPB, string Ub, int A1, int A2, int Ch1, int Ch2, int Inf1, int Inf2)	Constructor con parámetros, para la creación de objetos de tipo CE_Cabina.
P_Id_Cabina	Propiedad para la asignación y obtención del atributo Id_Cabina.
P_Id_DatosPB	Propiedad para la asignación y obtención del atributo Id_DatosPB.
P_Ub	Propiedad para la asignación y obtención del atributo Ub.
P_A1	Propiedad para la asignación y obtención del Atributo A1.
P_A2	Propiedad para la asignación y obtención del Atributo A2.
P_Ch1	Propiedad para la asignación y obtención del

	Atributo Ch1.
P_Ch2	Propiedad para la asignación y obtención del atributoCh2
P_Inf1	Propiedad para la asignación y obtención del Atributo Inf1.
P_Inf2	Propiedad para la asignación y obtención del Atributo Inf2.

Anexo: Tabla 1 Descripción de la clase persistente CE_Cabina.

Nombre: CE_Contenedor	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Contenedor	int
Ub	string
Peso	int
Destino	string
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Contenedor()	Constructor vacio, para la creación de objetos de tipo CE_Contenedor.
CE_Contenedor(int Id_Contenedor, string Ub, int Peso, string Destino)	Constructor con parámetros, para la creación de objetos de tipo CE_Contenedor.
P_Id_Contenedor	Propiedad para la asignación y obtención del atributo Id_Contenedor.
P_ Ub	Propiedad para la asignación y obtención del atributo Ub.
P_ Peso	Propiedad para la asignación y obtención del atributo Ub.
P_ Destino	Propiedad para la asignación y obtención del Atributo Destino.

Anexo: Tabla 2 Descripción de la clase persistente CE_Contenedor.

Nombre: CE_DatosPB	
Tipo de clase (Entidad)	
Atributo	**Tipo**
N_Vuelo	string
Id_Avion	Int
Id_Version	Int
Fecha	DateTime
Id_Galley	Int
D_ICAO	string
O_ICAO	string
D_SITA	string
O_SITA	string
TE	int
Paxs	int
NOTOC	bool
Id_Crew	int
XC_1	int
Ub_XC_1	string
XC_2	Int
Ub_XC_2	string
Capitan	string
Capitan_LicN	int
Comentario	string
Autor	string
C_Quemado	int
C_TOF	int
C_Alt_Rest	int
MTOW	int
MLW	int
Destino1	string
Destino2	string
Cabina	List<CE_Cabina>

Contenedor	List<CE_Contenedor>
Pallet	List<CE_Pallet>
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_DatosPB()	Constructor vacio, para la creación de objetos de tipo CE_DatosPB.
CE_DatosPB(DateTime Fecha, bool NOTOC, string N_Vuelo, string D_ICAO, string D_SITA, string O_ICAO, string O_SITA, int Paxs, int TE, int XC_1, int XC_2, string Ub_XC_1, string Ub_XC_2, int MTOW,int MLW, string Capitan, int Capitan_LicN, string Autor, string Comentario, int C_Quemado,int C_TOF, int C_Alt_Rest, int Id_Galley, int Id_Avion, int Id_Crew, int Id_Version, string Destino1, string Destino2)	Constructor con parámetros, para la creación de objetos de tipo CE_DatosPB.
P_N_Vuelo	Propiedad para la asignación y obtención del atributo P_N_Vuelo.
P_Id_Avion	Propiedad para la asignación y obtención del atributo P_Id_Avion.
P_Id_Version	Propiedad para la asignación y obtención del atributo P_Id_Version.
P_Fecha	Propiedad para la asignación y obtención del atributo P_Fecha.
P_Id_Galley	Propiedad para la asignación y obtención del tributo P_Id_Galley.
P_D_ICAO	Propiedad para la asignación y obtención del atributo P_Id_Galley.
P_O_ICAO	Propiedad para la asignación y obtención del atributo P_O_ICAO.
P_D_SITA	Propiedad para la asignación y obtención del atributo P_D_SITA.

P_O_SITA	Propiedad para la asignación y obtención del atributo P_O_SITA.
P_TE	Propiedad para la asignación y obtención del atributo P_TE.
P_Paxs	Propiedad para la asignación y obtención del atributo P_Paxs.
P_NOTOC	Propiedad para la asignación y obtención del atributo P_NOTOC.
P_Id_Crew	Propiedad para la asignación y obtención del atributo P_Id_Crew.
P_XC_1	Propiedad para la asignación y obtención del atributo P_XC_1.
P_Ub_XC_1	Propiedad para la asignación y obtención del atributo P_Ub_XC_1.
P_XC_2	Propiedad para la asignación y obtención del atributo P_XC_2.
P_Ub_XC_2	Propiedad para la asignación y obtención del atributo P_Ub_XC_2.
P_Capitan	Propiedad para la asignación y obtención del atributo P_Capitan.
P_Capitan_LicN	Propiedad para la asignación y obtención del atributo P_Capitan_LicN.
P_Comentario	Propiedad para la asignación y obtención del atributo P_Comentario.
P_Autor	Propiedad para la asignación y obtención del atributo P_Autor.
P_C_Quemado	Propiedad para la asignación y obtención del atributo P_C_Quemado.
P_C_TOF	Propiedad para la asignación y obtención del atributo P_C_TOF.
P_C_Alt_Rest	Propiedad para la asignación y obtención del atributo P_C_Alt_Rest.

P_MTOW	Propiedad para la asignación y obtención del atributo P_MTOW.
P_MLW	Propiedad para la asignación y obtención del atributo P_MLW.
P_Destino1	Propiedad para la asignación y obtención del atributo P_Destino1.
P_Destino2	Propiedad para la asignación y obtención del atributo P_Destino2.
ObtenerCabina(int num)	Método para la obtención del atributo Cabina.
AsignarCabina(CE_Cabina cabina)	Método para la asignación del atributo Cabina.
ObtenerContenedor(int num)	Método para la obtención del atributo Contenedor.
AsignarContenedor(CE_Contenedor contenedor)	Método para la asignación del atributo Contenedor.
ObtenerPallet(int num)	Método para la obtención del atributo Pallet.
AsignarPallet(CE_Pallet pallet)	Método para la asignación del atributo Pallet.
P_Cabina	Propiedad para la asignación y obtención del atributo Cabina.
P_Contenedor	Propiedad para la asignación y obtención del atributo Contenedor.
P_Pallet	Propiedad para la asignación y obtención del atributo Pallet.

Anexo: Tabla 3 Descripción de la clase persistente CE_DatosPB.

Nombre: CE_Galley	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Galley	int
Nombre	atring
Peso	int
Observaciones	string
Para cada responsabilidad:	

Nombre:	Descripción:
CE_Galley()	Constructor vacio, para la creación de objetos de tipo CE_Galley.
CE_Galley(int Id_Galley, string Nombre, int Peso, string Observaciones)	Constructor con parámetros, para la creación de objetos de tipo CE_ Galley.
P_Id_Galley	Propiedad para la asignación y obtención del atributo P_Id_Galley.
P_Nombre	Propiedad para la asignación y obtención del atributo P_Nombre.
P_Peso	Propiedad para la asignación y obtención del atributo P_Peso.
P_Observaciones	Propiedad para la asignación y obtención del atributo P_Observaciones.

Anexo: Tabla 4 Descripción de la clase persistente CE_Galley.

Nombre: CE_Pallet	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Pallet	int
Peso	int
Destino	string
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Pallet()	Constructor vacio, para la creación de objetos de tipo CE_Pallet.
CE_Pallet(int Id_Pallet, int Peso, string Destino)	Constructor con parámetros, para la creación de objetos de tipo CE_Pallet.
P_Id_Pallet	Propiedad para la asignación y obtención del atributo Id_Pallet.
P_Peso	Propiedad para la asignación y obtención del atributo Peso.
P_Destino	Propiedad para la asignación y obtención del

	atributo Destino.

Anexo: Tabla 5 Descripción de la clase persistente CE_Pallet

Nombre: CE_PControl	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_PControl	int
Nombre_PC	string
Coordenadas_PC	string
Variacion_PC	string
FIR_PC	string
Frecuencia_PC	float
Para cada responsabilidad:	
Nombre:	**Descripción:**
PControl()	Constructor vacio, para la creación de objetos de tipo CE_PControl
PControl(string aNombre_PC, string aCoordenadas_PC, string aVariacion_PC, float aFrecuencia_PC, string aFIR_PC)	Constructor con parámetros, para la creación de objetos de tipo CE_PControl, sin el atributo Id_PControl
PControl(int ID_PControl, string aNombre_PC, string aCoordenadas_PC, string aVariacion_PC, float aFrecuencia_PC, string aFIR_PC)	Constructor con parametros, para la creacion de objetos de tipo CE_PControl, con el atributo Id_PControl.
P_Id_PControl	Propiedad para la asignación y obtención del atributo Id_PControl
P_Nombre_PC	Propiedad para la asignación y obtención del atributo Nombre_PC.
P_Coordenadas_PC	Propiedad para la asignación y obtención del atributo Coordenadas_PC.
P_Variacion_PC	Propiedad para la asignación y obtención del atributo Variacion_PC.
P_Frecuencia_PC	Propiedad para la asignación y obtención del

	atributo Frecuencia_PC.
P_FIR_PC	Propiedad para la asignación y obtención del atributo FIR_PC.

Anexo: Tabla 6 Descripción de la clase persistente CE_PControl

Nombre: CE_Ruta	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Ruta	int
Nombre_R	string
Categoria_R	string
Origen	int
Destino	int
Puntos	List<PControl>
Para cada responsabilidad:	
Nombre:	**Descripción:**
Ruta()	Constructor vacio, para la creación de objetos de tipo CE_Ruta
Ruta(string aNombre_R, string aCategoria, int aOrigen, int aDestino, List<PControl> aPuntos)	Constructor con parámetros, para la creación de objetos de tipo CE_PControl, sin el atributo Id_Ruta
Ruta(int aId_Ruta, string aNombre_R, string aCategoria, int aOrigen, int aDestino, List<PControl> aPuntos)	Constructor con parámetros, para la creación de objetos de tipo CE_PControl, con todos los atributos.
P_Id_Ruta	Propiedad para la asignación y obtención del atributo Id_Ruta.
P_Nombre_R	Propiedad para la asignación y obtención del atributo Nombre_R.
P_Categoria_R	Propiedad para la asignación y obtención del atributo Categoria_R.
P_Origen	Propiedad para la asignación y obtención del atributo Origen.

P_Destino	Propiedad para la asignación y obtención del atributo Destino.
P_Puntos	Propiedad para la asignación y obtención del atributo Puntos.

Anexo: Tabla 7 Descripción de la clase persistente CE_Ruta.

Nombre: CE_ Ciudad	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Ciudad	int
Id_Pais	string
Nombre	string
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Ciudad()	Constructor vacio, para la creación de objetos de tipo CE_Ciudad
CE_Ciudad(int aid_pais, string anombre)	Constructor con parámetros, para la creación de objetos de tipo CE_Ciudad.
P_ Id_Ciudad	Propiedad para la asignación y obtención del atributo Id_Ciudad.
P_Id_Pais	Propiedad para la asignación y obtención del atributo Id_Pais.
P_Nombre	Propiedad para la asignación y obtención del atributo Nombre.

Anexo: Tabla 8 Descripción de la clase persistente CE_Ciudad.

Nombre: CE_Crew	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Crew	int
Nombre	string
Observaciones	string
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Crew()	Constructor vacio, para la creación de objetos de tipo CE_Crew.
CE_Crew(int Id_Crew, string Nombre, string Observaciones)	Constructor con parámetros, para la creación de objetos de tipo CE_Crew.
P_Id_Crew	Propiedad para la asignación y obtención del atributo Id_Crew.
P_Nombre	Propiedad para la asignación y obtención del atributo Nombre.
P_Observaciones	Propiedad para la asignación y obtención del atributo Observaciones

Anexo: Tabla 9 Descripción de la clase persistente CE_Crew.

Nombre: CE_Pais	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Pais	int
Nombre	string
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Pais()	Constructor vacio, para la creación de objetos de tipo CE_Pais.
CE_Pais(int aid_pais, string anombre)	Constructor con parámetros, para la creación de objetos de tipo CE_Pais.
P_Id_Pais	Propiedad para la asignación y obtención del atributo Id_Pais.

P_Nombre	Propiedad para la asignación y obtención del atributo Nombre.

Anexo: Tabla 10 Descripción de la clase persistente CE_Pais.

Nombre: CE_Version	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Version	int
Nombre	string
Fecha	DateTime
C	int
Y	int
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Version()	Constructor vacio, para la creación de objetos de tipo CE_Version
CE_Version(int Id_Version, string Nombre, int C, int Y)	Constructor con parámetros, para la creación de objetos de tipo CE_Version.
P_Id_Version	Propiedad para la asignación y obtención del atributo Id_Version.
P_Nombre	Propiedad para la asignación y obtención del atributo Nombre.
P_Fecha	Propiedad para la asignación y obtención del atributo Fecha.
P_C	Propiedad para la asignación y obtención del atributo C.
P_Y	Propiedad para la asignación y obtención del atributo Y.

Anexo: Tabla 11 Descripción de la clase persistente CE_Version.

Nombre: CE_Aeropuerto	
Tipo de clase (Entidad)	
Atributo	**Tipo**
ID_Aeropuerto	int
Ciudad	int
CodIATA	string
CodICAO	string
Coordenadas	string
Fecha	string
Nombre	string
Pais	string
Variacion	string
EleAerop	float
Pistas	List<CE_Pista>
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Aeropuerto()	Constructor vacio, para la creación de objetos de tipo CE_Aeropuerto.
CE_Aeropuerto(string aNombre, int aCiudad, float aEleAerop, string aCodIATA, string aCodICAO, string aVariacion, string aCoordenadas)	Constructor con parámetros, para la creación de objetos de tipo CE_Aeropuerto.
CE_Aeropuerto(int aId_Ciudad, int aId_Aeropuerto,string aCod_SITA,string aCod_ICAO,string aCoordenadas,string aVariacion,float aElevacion,string aNombre)	Constructor con parámetros, para la creación de objetos de tipo CE_Aeropuerto.
P_ID_Aeropuerto	Propiedad para la asignación y obtención del atributo ID_Aeropuerto.
P_Ciudad	Propiedad para la asignación y obtención del atributo Ciudad.
P_Variacion	Propiedad para la asignación y obtención del atributo Variacion.
P_CodIATA	Propiedad para la asignación y obtención del

	atributo CodIATA.
P_CodICAO	Propiedad para la asignación y obtención del atributo CodICAO.
P_Coordenadas	Propiedad para la asignación y obtención del atributo Coordenadas.
P_Fecha	Propiedad para la asignación y obtención del atributo Fecha.
P_Nombre	Propiedad para la asignación y obtención del atributo Nombre.
P_Pais	Propiedad para la asignación y obtención del atributo Pais.
PISTAS	Propiedad para la asignación y obtención del atributo PISTAS.

Anexo: Tabla 12 Descripción de la clase persistente CE_Aeropuerto.

Nombre: CE_Avion	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_Avion	int
Tipo	string
Matricula	string
TOW	int
LW	int
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Avion()	Constructor vacio, para la creación de objetos de tipo CE_Avion.
CE_Avion(int Id_Avion, string Matricula)	Constructor con parámetros, para la creación de objetos de tipo CE_Avion.
CE_Avion(int aId_Avion,string aTipo, string aMatricula, int aTOW, int aLW)	Constructor con parámetros, para la creación de objetos de tipo CE_Avion.
CE_Avion(string aTipo, string aMatricula, int	Constructor con parámetros, para la creación de

aTOW, int aLW)	objetos de tipo CE_Avion.
P_Id_Avion	Propiedad para la asignación y obtención del atributo Id_Avion.
P_Matricula	Propiedad para la asignación y obtención del atributo Matricula.
P_Tipo	Propiedad para la asignación y obtención del atributo Tipo.
P_TOW	Propiedad para la asignación y obtención del atributo TOW.
P_LW	Propiedad para la asignación y obtención del atributo LW.

Anexo: Tabla 13 Descripción de la clase persistente CE_Avion.

Nombre: CE_DatosPV	
Tipo de clase (Entidad)	
Atributo	**Tipo**
Id_DatosPV	int
Fecha	datetime
Id_Origen	int
Id_Destino	int
Id_Avion	int
Id_Crew	int
Id_Version	int
FLT	int
Alterno	int
ALTN_Dist	int
ETD	int
Data_Met	int
Mach	int
DOW	int
ZFW	int
PLD	int

Max_FL	int
TOW_Rest	int
LW_Rest	int
Extra_Fuel	int
CLBWC	int
CLBOAT	int
ALTNWC	int
ALTNOAT	int
DCTWC	int
DCTOAT	int
Id_Ruta	int
Tramos	List<CE_Tramo>
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_DatosPV()	Constructor vacio, para la creación de objetos de tipo.
CE_DatosPV (int aId_DatosPV,DateTime aFecha, int aOrigen, int aDestino, int aAvion, int aCrew, int aVersion,int aFLT, int aAlterno, int aAltn_dist, int aETD, DateTime aDataMet, float aMach,int aDOW, int aZFW, int aPLD, int aMax_FL, int aTOW_Rest, int aLW_Rest, int aExtra_Fuel, int aCLBWC,int aCLBOAT, int aAltnWC, int aAltnOAT, int aDCTWC, int aDCTOAT, List<CE_Tramo> aTramos)	Constructor con parámetros, para la creación de objetos de tipo CE_DatosPV.
CE_DatosPV(DateTime aFecha, int aOrigen, int aDestino, int aAvion, int aCrew, int aVersion, int aFLT, int aAlterno, int aAltn_dist, int aETD, DateTime aDataMet, float aMach, int aDOW, int aZFW, int aPLD, int aMax_FL, int aTOW_Rest, int aLW_Rest, int aExtra_Fuel, int aCLBWC, int aCLBOAT, int aAltnWC, int	Constructor con parámetros, para la creación de objetos de tipo CE_DatosPV.

aAltnOAT, int aDCTWC, int aDCTOAT, List<CE_Tramo> aTramos)	
P_Id_Origen	Propiedad para la asignación y obtención del atributo Id_Origen.
P_Id_Destino	Propiedad para la asignación y obtención del atributo Id_Destino.
P_Id_Avion	Propiedad para la asignación y obtención del atributo Id_Avion.
P_Id_Crew	Propiedad para la asignación y obtención del atributo Id_Crew.
P_Id_Version_Avion	Propiedad para la asignación y obtención del atributo Id_Version_Avion.
P_FLT	Propiedad para la asignación y obtención del atributo FLT.
P_Alterno	Propiedad para la asignación y obtención del atributo Alterno.
P_Altn_dist	Propiedad para la asignación y obtención del atributo Altn_dist.
P_ETD	Propiedad para la asignación y obtención del atributo EDT.
P_DataMet	Propiedad para la asignación y obtención del atributo DataMet.
P_Mach	Propiedad para la asignación y obtención del atributo Mach.
P_DOW	Propiedad para la asignación y obtención del atributo DOW.
P_ZFW	Propiedad para la asignación y obtención del atributo ZFW.
P_PLD	Propiedad para la asignación y obtención del atributo PLD.
P_Max_FL	Propiedad para la asignación y obtención del atributo Max_FL.

P_TOW_Rest	Propiedad para la asignación y obtención del atributo TOW_Rest.
P_LW_Rest	Propiedad para la asignación y obtención del atributo LW_Rest.
P_Extra_Fuel	Propiedad para la asignación y obtención del atributo Extra_Fuel.
P_CLBWC	Propiedad para la asignación y obtención del atributo CLBWC.
P_CLBOAT	Propiedad para la asignación y obtención del atributo CLBOAT.
P_AltnWC	Propiedad para la asignación y obtención del atributo AltnWC.
P_AltnOAT	Propiedad para la asignación y obtención del atributo AltnOAT.
P_DCTWC	Propiedad para la asignación y obtención del atributo DCTWC.
P_DCTOAT	Propiedad para la asignación y obtención del atributo DCTWC.
P_Tramos	Propiedad para la asignación y obtención del atributo Tramos.
P_Fecha	Propiedad para la asignación y obtención del atributo Fecha.
P_Id_Ruta	Propiedad para la asignación y obtención del atributo Id_Ruta.

Anexo: Tabla 14 Descripción de la clase persistente CE_DatosPV.

Nombre: CE_Tramo	
Tipo de clase (Entidad)	
Atributo	**Tipo**
ID_Tramo	int
ID_PControl_Inicial	int
ID_PControl_Final	int
WC	int
OAT	int
Para cada responsabilidad:	
Nombre:	**Descripción:**
CE_Tramo()	Constructor vacio, para la creación de objetos de tipo CE_Tramo.
CE_Tramo(int inicial,int final,int aWC,int aOAT)	Constructor con parámetros, para la creación de objetos de tipo CE_Tramo.
CE_Tramo(int aID, int inicial,int final,int aWC,int aOAT)	Constructor con parámetros, para la creación de objetos de tipo CE_Tramo.
P_ID_PControl_Inicial	Propiedad para la asignación y obtención del atributo ID_PControl_Inicial.
P_ID_PControl_Final	Propiedad para la asignación y obtención del atributo ID_PControl_Final.
P_WC	Propiedad para la asignación y obtención del atributo WC.
P_OAT	Propiedad para la asignación y obtención del atributo OAT.
P_ID_Tramo	Propiedad para la asignación y obtención del atributo ID_Tramo.

Anexo: Tabla 15 Descripción de la clase persistente CE_Tramo.

Anexo II: Descripción de las Tablas del Diagrama Entidad Relación.

Nombre: TB_Aeropuerto		
Descripción: Contiene toda la información necesaria referente al aeropuerto.		
Atributo	**Tipo**	**Descripción**
ID_Aeropuerto	int	Este es el atributo llave de esta tabla, es un valor auto numérico.
Ciudad	int	Llave foránea, que indica la Ciudad del Aeropuerto.
CodIATA	varchar	Es un código de tres letras que designa a cada aeropuerto.
CodICAO	varchar	Identifica a cada aeropuerto.
Coordenadas	varchar	Coordenadas de este.
Nombre	varchar	Nombre de este.
Variacion	varchar	
Elevacion	varchar	Elevación con respecto al nivel del mar.

Anexo: Tabla 16 Descripción de la Tabla Aeropuerto.

Nombre: TB_Avion		
Descripción: Datos de un avión.		
Atributo	**Tipo**	**Descripción**
ID_Avion	int	Este es el atributo llave de esta tabla, es un valor auto numérico.
Tipo	varchar	Tipo de Avión.
TOW	int	El peso total del avión con combustible.
LW	int	Landing Weit por sus siglas en ingles, peso de aterrizaje.
Matricula	varchar	Matricula del Avión.

Anexo: Tabla 17 Descripción de la Tabla Avión.

Nombre: TB_Cabina		
Descripción: Almacena los valores necesarios de la Cabina, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Cabina	int	Identificador de la Cabina, es un auto numérico.
Id_DatosPB	int	Identificador de DatosPB.
A1	int	Adulto destio1.
A2	int	Adulto destio 2.
Ch1	int	Niños destio 1.
Ch2	int	Niños destio 2.
Inf1	int	Infante destio 1.
Inf2	int	Infante destio 2.

Anexo: Tabla 18 Descripción de la Tabla Cabina.

Nombre: TB_Contenedor		
Descripción: Almacena los valores necesarios del Contenedor, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Contenedor	int	Identificador del contenedor.
Peso	int	Peso del contenedor.
Destino	varchar	Destino del contenedor.

Anexo: Tabla 19 Descripción de la Tabla Contenedor.

Nombre: TB_DatosPB		
Descripción: Almacena los valores necesarios de DatosPB, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
N_Vuelo	varchar	Numero del Vuelo.
Id_Avion	Int	Identificador del avión.
Id_Version	Int	Identificador de la versión.
Fecha	DateTime	Fecha.
Id_Galley	Int	Identificador del Galley.
D_ICAO	varchar	Identificador del aeropuerto destino.
O_ICAO	varchar	Identificador del aeropuerto origen.
D_SITA	varchar	Codigo del aeropuerto destino.
O_SITA	varchar	Codigo del aeropuerto origen..
TE	int	Carga que se coloca en la cola del avión.
Paxs	int	Peso permisible por cada pasajero según el vuelo.
NOTOC	bool	Mercancías peligrosas que se llevará en el avión.
Id_Crew	int	Identificador del Crew.
XC_1	int	Personal que no pertenece a la tripulación.
Ub_XC_1	varchar	Ubicación XC1.
XC_2	Int	Personal que no pertenece a la tripulación.
Ub_XC_2	varchar	Ubicación XC2.
Capitan	varchar	Capitán de la tripulación.
Capitan_LicN	int	Número de licencia del capitán.
Autor	varchar	El que realiza el peso y balance.
C_Quemado	int	Combustible quemado.
C_TOF	int	Representa la cantidad de combustible que se le suministrará al avión.
C_Alt_Rest	int	Resultado de la resta entre MTOW y MLW.
MTOW	int	Máximo Peso de Despegue del avión.
MLW	int	Máximo Peso de Aterrizaje que permite el avión.
Destino1	varchar	Destino 1.
Destino2	varchar	Destino 2.

Anexo: Tabla 20 Descripción de la Tabla DatosPB.

Nombre: TB_Galley		
Descripción: Almacena los valores necesarios del Galley, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Galley	int	Identificador del este.
Nombre	varchar	Nombre de este.
Peso	int	Peso de este.
Observaciones	varchar	Observaciones sobre este.

Anexo: Tabla 21 Descripción de la Tabla Galley.

Nombre: TB_Pallet		
Descripción: Almacena los valores necesarios del Pallet, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Pallet	int	Identificador de este.
Peso	int	Su peso.
Destino	varchar	Su destino.
Id_DatosPB	int	Identificador de DatosPB al que pertenece.

Anexo: Tabla 22 Descripción de la Tabla Pallet.

Nombre: TB_PControl		
Descripción: Almacena los valores necesarios del PControl, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_PControl	int	Identificador de este.
Nombre_PC	varchar	Nombre de este.
Coordenadas_PC	varchar	Sus coordenadas.
Variacion_PC	varchar	
FIR_PC	varchar	Indica que el punto de control es frontera.
Frecuencia_PC	float	Frecuencia de radio para comunicarse en ese punto.

Anexo: Tabla 23 Descripción de la Tabla PControl.

Nombre: TB_Ruta		
Descripción: Almacena los valores necesarios de la Ruta, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Ruta	int	Identificador de la ruta.
Nombre	varchar	Su nombre.
Categoria	varchar	Su categoría.
Origen	int	Su origen.
Destino	int	Su destino.

Anexo: Tabla 24 Descripción de la Tabla Ruta.

Nombre: TB_Ciudad		
Descripción: Almacena los valores necesarios de la Ciudad, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Ciudad	int	Identificador de la ciudad.
Id_Pais	varchar	Identificador del país al que pertenece.
Nombre	varchar	Nombre de la ciudad.

Anexo: Tabla 25 Descripción de la Tabla Ciudad.

Nombre: TB_Crew		
Descripción: Almacena los valores necesarios del Crew, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Crew	int	Identificador del crew.
Nombre	varchar	Su nombre.
Observaciones	varchar	Observaciones hacerca de este.

Anexo: Tabla 26 Descripción de la Tabla Crew.

Nombre: TB_Pais		
Descripción: Almacena los valores necesarios del Pais, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Pais	int	Identificador del país.
Nombre	varchar	Su nombre.

Anexo: Tabla 27 Descripción de la Tabla País.

Nombre: TB_Version		
Descripción: Almacena los valores necesarios de la Version, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Version	int	Identificador de la versión.
Nombre	varchar	Su nombre.
Fecha_Inc	DateTime	Su fecha.
C	int	Clase alta, parte de los pasajeros primera clase.
Y	int	Clase, comercial.

Anexo: Tabla 28 Descripción de la Tabla Versión.

Nombre: TB_DatosPV		
Descripción: Almacena los valores necesarios de DatosPV, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_DatosPV	int	Identificador de DatosPV.
Fecha	datetime	Fecha del vuelo.
Id_Avion	int	Identificador del avión.
Id_Crew	int	Identificador del crew.
Id_Version	int	Identificador de la versión.
FLT	int	Número del vuelo.
Alterno	int	Identificador del aeropuerto alterno.
ALTN_Dist	int	Distancia al alterno.
ETD	int	Hora estimada de salida.
Data_Met	int	Fecha de los datos meteorológicos.
Mach	int	Velocidad del avión respecto a la velocidad del sonido.
DOW	int	Peso correspondiente al PBO.
ZFW	int	Peso sin combustible.
PLD	int	Peso de los pasajeros, carga, equipaje y correo.
Max_FL	int	Altura máxima del vuelo.
TOW_Rest	int	Peso de despegue.
LW_Rest	int	Peso de aterrisaje.
Extra_Fuel	int	Combustible extra.
CLBWC	int	Componente del viento en el ascenso.
CLBOAT	int	Temperatura exterior en el ascenso.
ALTNWC	int	Componente del viento en el alterno.
ALTNOAT	int	Temperatura del aire exterior en el alterno.
DCTWC	int	Componente del viento del descenso.
DCTOAT	int	Temperatura del aire exterior descenso.
Id_Ruta	int	Identificador de la ruta.

Anexo: Tabla 29 Descripción de la Tabla DatosPV.

Nombre: TB_Tramo		
Descripción: Almacena los valores necesarios de DatosPV, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Tramo	int	Identificador del tramo.
WC	varchar	Componente del viento.
OAT	varchar	Temperatura ambiental.
Inicila	int	Identificador del aeropuerto inicial.
Final	int	Identificador del aeropuerto final.

Anexo: Tabla 30 Descripción de la Tabla Tramo.

Nombre: TB_DatosPB_Tramo		
Descripción: Almacena los valores necesarios de DatosPV, para interactuar con el sistema.		
Atributo	**Tipo**	**Descripción**
Id_Tramo	int	Identificador del Tramo.
Id_DatosPB	int	Identificador de DatosPB.

Anexo: Tabla 31 Descripción de la Tabla DatosPB_Tramo.