# INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA
Y ELÉCTRICA
UNIDAD CULHUACAN

## "SISTEMA DE GESTIÓN PARA AEROLINEAS DE BAJO COSTO"

# TESIS

QUE PARA OBTENER EL TÍTULO DE :

INGENIERO EN COMPUTACIÓN

PRESENTA:

**LINO ALVAREZ FRANCISCO JAVIER**

ASESORES

DR. ROSARIO DEL PILAR GIBERT DELGADO

ING. ROBERTO OSORNIO SOTO

MÉXICO, D.F. JUNIO DE 2013

# INSTITUTO POLITÉCNICO NACIONAL
# ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
# UNIDAD CULHUACAN

## TESIS INDIVIDUAL

Que como prueba escrita de su Examen Profesional para obtener el Título de Ingeniero en Computación, deberá desarrollar el C.:

**FRANCISCO JAVIER LINO ALVAREZ**

**"SISTEMA DE GESTION PARA AEROLINEAS DE BAJO COSTO"**

**Las Aerolíneas de bajo costo, basan su plan de trabajo, en una cantidad reducida de empleados, los cuales deben desarrollar múltiples funciones como la documentación de pasajeros, gestión del equipaje y servicios como el catering. A diferencia de sus competidores, las Aerolíneas Internacionales, las cuales cuentan con un significativo número de empleados, además de sistemas de información desarrollados a medida, cuyos costos en ocasiones llegan a los cinco o seis millones de pesos, cantidad que no es justificable para una Aerolínea de bajo costo. De esta forma, es técnicamente posible, desarrollar una aplicación web, que permita realizar el proceso de Gestión Aeroportuaria de una Aerolínea de bajo costo, basada específicamente en los procesos que esta desarrolla, con un costo mínimo utilizando software Open Source de alta calidad para el desarrollo.**

**CAPITULADO**



**México D. F., a 25 de junio del 2013**

**PRIMER ASESOR:**

**DRA. ROSARIO DEL PILAR GIBERT DELGADO**

**SEGUNDO ASESOR:**

**ING. ROBERTO OSORIO SOTO**

**Vo. Bo.**

**DR. JOSE VELAZQUEZ LOPEZ**
**JEFE DE LA CARRERA DE I.C.**

**APROBADO**

**M. en C. HECTOR BECERRIL MENDOZA**
**SUBDIRECTOR ACADÉMICO**

# Agradecimientos


## A mi madre:

Por toda la paciencia que me has tenido, por hacerme auto suficiente, independiente, por creer siempre en mí a pesar de la opinión de los demás.

Discúlpame por todas las lágrimas y todas las preocupaciones que te he causado, olvida por favor todas las ideas erróneas que has tenido, nunca me dejaste solo, siempre estuviste ahí para cuidarme, para enseñarme, para mostrarme la realidad de las cosas, gracias a ti he llegado hasta donde estoy y soy lo que soy.

## A mis maestros:

Martha Hernández Cuellar, Luis Carlos Castro Madrid, Roberto Osornio Soto, Antonio Castañeda, Daniel Cruz Perez, Rosario del Pilar Gibert, por todas las veces que estuvieron ahí para apoyarme, orientarme, evitar que me metiera en problemas y sobre todo por todas las veces que me demostraron que me equivocaba.

## A mis amigos:

Leyte, Isael, Alejandro Bonilla, Javier Serratos, por hacerme sonreír, por tolerar mis ataques de neurosis, porque sin ustedes los cuatro años de la carrera hubieran sido realmente aburridos.

## A Perla:

Muñeca hermosa, gracias por hacerme ver que realmente no estoy solo, por hacerme reír con tus burradas, por soportar mi pésimo sentido del humor, por pararme de frente cuando empiezo a enloquecer, simplemente gracias por darle sentido a mi vida y demostrarme que no siempre tengo razón, te amo.

## A mis hijas:

Gracias por obligarme a buscar una forma simple de explicar las cosas, por todas esas horas de X box y todas las veces que han creído ciegamente en mi aun cuando saben que les digo las cosas para espantarlas, por recordarme que no siempre el mundo es como yo creo, por impulsarme siempre a ser mejor

# Índice General

# Índice de Figuras y Tablas

## Índice de figuras

# Índice de Tablas

# Resumen

Esta tesis propone el desarrollo de un sistema de gestión, enfocado en las necesidades de una empresa aeroportuaria de bajo costo.

Para realizar este sistema de gestión, es necesario comprender los principios fundamentales mediante los cuales fueron desarrolladas las aplicaciones web, sus orígenes y hacia donde se dirigen años después de su aparición. Una vez que los fundamentos de las aplicaciones web han sido cubiertos, es posible determinar que la aplicación, al ser un sistema de información, requerirá de la implementación de una base de datos y dependerá de un proceso de ingeniería especifico, denominado ingeniería web, el cual determinara cada uno de los detalles y componentes que conformaran el sistema de gestión.

El Gestor de Base de Datos, almacenara, toda la información generada por el sistema y se desarrollara en base a estándares y métricas internacionales, específicamente, los que se encuentran orientados a bases de datos relacionales como son las 12 reglas de Codd y las reglas de normalización. Para esta implementación se ha recurrido a Gestor de Bases de Datos, PostgreSQL, el cual en conjunto con Oracle, es uno de los líderes del mercado y se diferencia de su competidor, no por sus prestaciones, ni por su calidad, ya que ambos son similares en este aspecto, si no por el costo de su licenciamiento, siendo PostgreSQL un Gestor de Bases de Datos Open Source.

El siguiente paso se enfocara en de proceso de Ingeniería Web, el cual brindara las normas y requisitos necesarios a fin de desarrollar una aplicación web de calidad, fácil de actualizar, mantener y eficiente.

Mediante las etapas de este proceso: análisis de contexto, diseño y gestión del proyecto y con la integración de la información obtenida mediante el análisis de procesos de la gestión aeroportuaria, se desarrollara una aplicación web, la cual interactuara con los diferentes procesos de la gestión aeroportuaria, para brindar de esta manera un sistema de información, con la capacidad de acelerar, simplificar y optimizar los procesos de Gestión Aeroportuaria.

Finalmente el sistema de gestión, será capaz de disminuir a mediano plazo los costos del proceso de gestión Aeroportuaria.

# Introducción

El transporte aéreo constituye una de las industrias más dinámicas que existen a nivel mundial, y se encuentra en constante crecimiento. Debido al incremento del mercado, se requieren líneas que realicen vuelos nacionales e internacionales, líneas troncales, regionales y de bajo costo.

Las aerolíneas de bajo costo cuentan con un plan de trabajo el cual se basa en tener menos empleados y obtener mayor productividad de los recursos que invierte. Para poder obtener lo anterior, dentro de la Administración Aeroportuaria, existen un conjunto de agentes que operan en un aeropuerto ejerciendo un amplio abanico de actividades, muchas veces son realizadas por parte de empresas y de diversas agencias de la Administración, ajenas al operador aeroportuario.

El operador aeroportuario es el organizador y regulador del intercambio modal (aéreo, terrestre) y de los diversos flujos físicos: aviones, pasajeros, personal, maletas, vehículos de *handling*, taxis, por mencionar algunos.

Uno de los principales problemas a los que se enfrenta la Administración Aeroportuaria es la insuficiencia de elementos metodológicos al alcance del personal, que permitan realizar las actividades fundamentales, de una manera sencilla, segura y accesible.



Actividades que actualmente tres empleados se encargan de llevar a cabo manualmente, tal como la documentación de pasajeros, el peso y el balance de la aeronave, que bien es posible manejarlos de una forma en la que la variación del tiempo así como los costos se vean reducidos, y a su vez minimizar el margen de error.

Debido a ésta problemática surge la necesidad de desarrollar un sistema basado en aplicaciones Web, que cuente con una Base de Datos, administrando el flujo de información generada dentro de la Administración Aeroportuaria por medio de una Intranet. Basando dicho sistema en una disciplina metodológica como lo es la Ingeniería Web, la cual se basa en los principios de funcionalidad del sistema para desarrollar y mantener una aplicación Web de alta calidad.

Se define la tecnología de desarrollo del sistema, la cual consiste en lo siguiente:

- Arquitectura: cliente-servidor.
- Redes locales en oficinas corporativas (nacional y regional) con servidor Apache.
- Manejador de bases de datos: Postgre SQL

- Herramienta de desarrollo: PHP

Las Aplicaciones web se encuentran estructuradas como una aplicación de tres capas. Estas capas son: el navegador web que es la primer capa, un motor capaz de usar alguna tecnología web dinámica que constituye la capa de en medio, y por último una base de datos que es la tercera capa.

# I. Estado del Arte y Planteamiento del problema de Investigación

Esta investigación versa sobre el desarrollo de un sistema de gestión, que permitirá desarrollar el proceso de gestión aeroportuaria, específicamente de aerolíneas de bajo costo.

Las Aerolíneas de bajo costo, basan su plan de trabajo, en una cantidad reducida de empleado, los cuales deben de desarrollar múltiples funciones como la documentación de pasajeros, gestión del equipaje y servicios como el catering. A diferencia de sus competidores, las Aerolíneas Internacionales, las cuales cuentan con un significativo número de empleados, además de sistemas de información desarrollados a medida, cuyos costos en ocasiones llegan a los cinco o seis millones de pesos, cantidad que no es justificable para una Aerolínea de bajo costo.

## 1.1 Problema de Investigación



De esta forma, es técnicamente posible, desarrollar una aplicación web, que permita realizar el proceso de Gestión Aeroportuaria de una Aerolínea de bajo costo, basada específicamente en los procesos que esta desarrolla, con un costo mínimo utilizando software Open Source de alta calidad para el desarrollo

## 1.2 Objetivo General

Investigar, diseñar y desarrollar una aplicación web, que cumpla con las funciones de un Sistema de Gestión que permita realizar los procesos y procedimientos de una línea de transporte aéreo de bajo costo.

## 1.3 Objetivos Específicos

- Fundamentar los principios sobre los cuales se basan las aplicaciones web, su funcionamiento y componentes.

- Desarrollar un esquema de base de datos relacional, basado en métricas y normas que permitan estandarizar y normalizar la estructura de la base de datos así como la información que se almacenara en la misma.

- Desarrollar una aplicación web, basada en estándares y normas específicas de la Ingeniería Web, lo cual se verá reflejado en una aplicación web de calidad.

- Integrar cada uno de los componentes desarrollados, basados en las necesidades de la gestión aeroportuaria, para de esta formar, obtener un sistema de información, de calidad capaz de solventar las necesidades específicas de dicho proceso de gestión.

# II. Marco Teórico y Metodología

## 2.1. Marco Teórico de la Investigación

Desde el anuncio de su liberación al público en 1993 hasta la fecha actual, 2013, la WWW se ha convertido en un servicio fundamental para la sociedad, siendo ya bastante lejanos los tiempos en los que esta era un simple servicio de páginas web estáticas que servían como carta de presentación a empresas y personas.

Con el paso del tiempo, la web ha ido creciendo exponencialmente y las aplicaciones basadas en web ahora ofrecen una compleja serie de funcionalidades y contenido, a un número ingente de usuarios. Actualmente la web se ha transformado en una plataforma de todo tipo de aplicaciones, con infinidad de propósitos, como las tiendas virtuales, servicio de mensajería electrónica y en fechas recientes, el surgimiento del concepto de redes sociales, situación que deriva en la complejidad de las interacciones entre el sistema web y los sistemas de información que se encuentran en su trasfondo.

Debido a la dependencia cada vez mayor de los sistemas web y sus aplicaciones, el rendimiento, fiabilidad y calidad de los sitios se ha convertido en un asunto de gran importancia, aumentando significativamente las necesidades y expectativas de estos sistemas, teniendo como resultado que el desarrollo y mantenimiento de los sistemas web adquieran un nivel de complejidad más elevado.



Sin embargo, debido a la existencia de cantidades masivas de desarrollo Web y mantenimiento, se encuentran sistemas y aplicaciones pobres en cuanto a funcionalidad. Los problemas tales como la información anticuada o impertinente, las dificultades de uso del sitio Web, el manejo inapropiado de la información, la respuesta llega a ser lenta, y además, las intrusiones en cuanto a seguridad, son encontradas comúnmente. Estos tipos de problemas se encuentran debido a que, los desarrolladores Web no llegan a tomar en cuenta las necesidades de los usuarios y la escalabilidad de las aplicaciones Web.

Muchos desarrolladores Web parecen pensar que el desarrollo de aplicación Web es la creación de páginas Web simples utilizando HTML o software de desarrollo Web tal como Dreamweaver o Front Page y llega a ser personalizado con algunas imágenes e hipervinculando páginas de documentos. Sin embargo ciertas aplicaciones requieren una presentación con la cual obtengan un punto de satisfacción ya que son requeridas para cubrir un conjunto de necesidades que llegan a ser desafiantes, pues todo está en constante cambio y evolución. Todo ello supone planear, arquitecturas Web y diseño de sistemas, con una actualización continua y un mantenimiento adecuado de los sistemas.

La diversidad de problemas, tiene como consecuencia, sistemas que no son lo que el usuario desea, no llegan a ser sustentables ni escalables, y por lo tanto, tienen una vida útil muy corta. Además, los niveles de seguridad y ejecución no son los esperados, convirtiendo así una aplicación Web en un sistema atrasado y poco funcional.

Muchas empresas y organizaciones no pueden afrontar tener sistemas Web imperfectos ni tiempo muerto o inconsistente, ya que los problemas en un sistema Web son muy notorios, ocasionando la frustración de los usuarios y que estos abandonen el sitio, acarreando pérdidas financieras y de reputación.

Para construir sistemas y aplicaciones útiles, los desarrolladores Web necesitan adoptar un desarrollo disciplinado y una metodología, usando herramientas mejores. La disciplina de la ingeniería Web dirige estas necesidades al desarrollo exitoso de los sistemas y aplicaciones con base en la Web, y a su vez aboga por un acercamiento disciplinado al desarrollo Web.

La ingeniería Web se enfoca, a los principios de manejo del sistema para desarrollar, desplegar, y mantener con buen resultado sistemas y aplicaciones Web de alta calidad. Ello pretende el desarrollo de sistema con base en la Web controlados, con riesgos mínimos y mejorar la calidad, mantenimiento, y escalabilidad de las aplicaciones Web.

En 1998, Roger Pressman moderó una mesa redonda virtual con representantes la ingeniería software tradicional y del desarrollo software basado exclusivamente en Internet. Bajo el argumento principal de Pressman: *"Me parece que cualquier producto o sistema importante es merecedor de recibir una ingeniería. Antes de comenzar a construirlas, lo mejor es entender el problema, diseñar una solución viable, implementarla de una manera sólida y comprobarla en profundidad. Probablemente también se deberían controlar los cambios a medida que el trabajo vaya avanzando, y disponer de mecanismos para asegurar la calidad del resultado final. Muchos de los que desarrollan Webs no dicen lo mismo; ellos piensan que su mundo es realmente diferente, y que simplemente no se van a aplicar los enfoques de ingeniería del software convencionales."* [Pressman, (2002)]



El debate principalmente se centró en discutir si valía la pena aplicar un proceso de ingeniería a las aplicaciones con base en internet, o qué características tenían éstas que justificaran el no utilizarlo. La conclusión general fue que aplicar un proceso de ingeniería nunca es una mala idea pero que éste debería adaptarse a los requerimientos de cambio continuo y rapidez siempre presentes en el proceso de desarrollo Web. De iniciativas como ésta y de otras como la organización de congresos y talleres especializados en el desarrollo para la Web, surge el nacimiento de una nueva disciplina denominada Ingeniería Web.

Murugesan et al, promotores iniciales del establecimiento de la Ingeniería Web como nueva disciplina, dan una definición que escuetamente se puede traducir como: "*El*

*proceso utilizado para crear, implantar y mantener aplicaciones y sistemas Web de alta calidad*".

Bajo este concepto, es posible determinar que un sistema web, deberá de cumplir con ciertos atributos y características o requisitos, para ser denominado un sistema de calidad.

De entre los atributos de un sistema web, destacan los siguientes, debido a que invariablemente deberán encontrarse en todas y cada una de dichas aplicaciones.

- *Residente en red*. Una aplicación web reside en una red, y debe dar servicio a una comunidad diversa de clientes.

- *Inmediatez*. Se refiere al corto tiempo que normalmente tienen los proyectos web para terminar, o por lo menos, lanzar una versión inicial.

- *Evolución continúa*. A diferencia del software de aplicaciones convencional, que evoluciona a través de versiones planeadas y cronológicamente espaciadas, las aplicaciones web están en constante evolución, y se actualizan gradualmente.

- *Seguridad*. Dado que no controlamos con certeza quién puede acceder a nuestra aplicación; la seguridad y confidencialidad de la información requieren un énfasis especial.



- *Estética*. Una parte innegable del atractivo de una aplicación web es su apariencia e interacción. Cuando se ha diseñado una aplicación con el fin de comercializarse o vender productos o ideas, la estética puede tener mucho que ver con el éxito del diseño técnico.

- *Medible*. Mediante la cuantificación de resultados, podemos conocer la cantidad de usuarios que tenemos, así como sus patrones de comportamiento.

A diferencia de los atributos de un sistema web, definir los requisitos de calidad presenta una dificultad más alta, ya que el concepto de calidad varía en función de las expectativas de la empresa o de las personas que visita el sitio. Algunos usuarios disfrutan de los gráficos llamativos, otros prefieren el texto sencillo, bastante información o bien presentaciones abreviadas. Sin embargo es posible definir los requisitos de calidad de un sistema web en base a las características más relevantes planteadas por Olsina [Pressman, (2002)]: usabilidad, fiabilidad, eficiencia y capacidad de mantenimiento. Esta definición proporciona una base para evaluar la calidad de los sistemas basados en web.

La Figura 2.1 presenta el árbol de requisitos de calidad para aplicaciones web propuesto por Olsina en 1999. [Nolasco, (2013)]

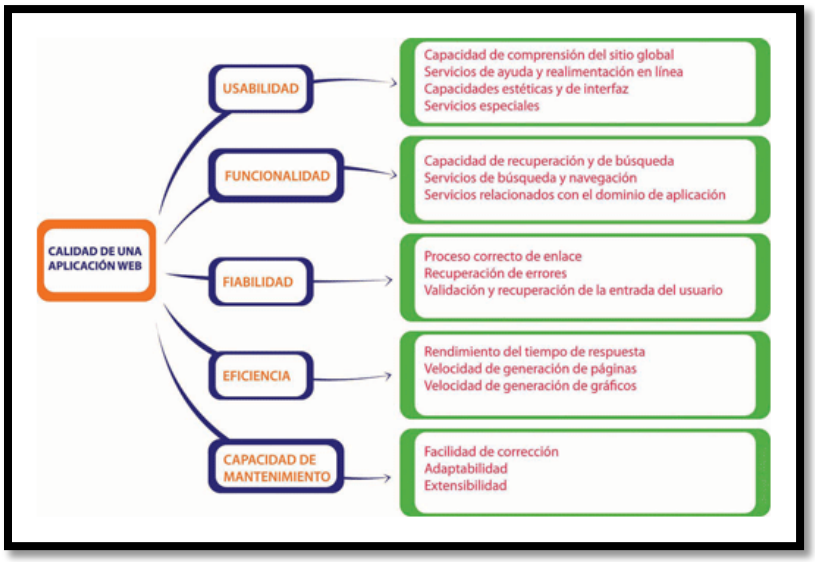



**Figura 2.1.1 Requisitos de calidad para aplicaciones Web**

Adicionalmente, el diseño y la implementación de sistemas web incorporan tres tecnologías importantes, el desarrollo basado en componentes, la seguridad y los estándares de internet.

- Desarrollo basado en componentes: se basa en tres estándares para el desarrollo de la infraestructura, estos son: CORBA; COM/DCOM y Java Beans, estos estándares proporcionan una infraestructura que permite diseñar y personalizar componentes de terceros, comunicando de esta manera componentes propios con externos, así como con servicios a nivel de sistema.

- Seguridad: debido a que la aplicación web se encuentra disponible a todo aquel que tenga acceso a internet, el concepto de seguridad cobra importancia al evitar que el sistema sea víctima de un ataque o intrusión por parte de una persona mal intencionada, la cual, podría encontrarse incluso dentro del personal interno de la empresa o institución.

- Estándares de internet: con la imparable evolución de los sistemas web, surgen también dificultades al momento de la interacción entre sistemas, los cuales han sido desarrollados en diferentes tecnologías, ocasionando comportamientos erráticos en el despliegue de los sitios web, el uso de estándares, permite que las aplicaciones web se comporten de la misma manera independientemente de la plataforma en que se ejecuten o sean accedidas.

Siendo estos componentes, requisitos indispensables para el ingeniero a cargo del desarrollo de una aplicación web de calidad.

## 2.2. Metodología a seguir en la investigación

La metodología utilizada para el desarrollo de este documento se basa en el proceso de investigación teórica y se resume en las siguientes fases:

*Etapa 1:* Recopilación de información referente a la WWW, sus mecanismos, componentes y métodos

*Etapa 2:* Recopilación de información referente a los Sistemas de Bases de datos, específicamente los sistemas de relacionales, así como su normatividad y estándares.

*Etapa 3:* Recopilación de información referente al proceso de Ingeniería Web, así como sus estándares y procedimientos

*Etapa 4:* Recopilación de información referente al proceso de Gestión Aeroportuaria, sus componentes, procedimientos y métodos.

*Etapa 5:* Desarrollo del sistema de información, basado en los estándares, procedimientos, procesos y necesidades recabados en los puntos anteriores



Una vez haya sido concluida cada una de las etapas, se definirá como finalizado el sistema de información y será puesto a disposición del público en general para su implementación, análisis y/o mejoras.

# III. Desarrollo de la Investigación

Hasta el año 2013 el término Internet es utilizado para describir una gran magnitud de servicios que van desde el correo electrónico, redes sociales, sistemas de mensajería e incluso actividades recreativas, sin embargo, detrás de este concepto de uso tan ambiguo se esconde una red descentralizada de computadoras interconectadas entre sí, cada una de las cuales brinda una utilidad diferente a este concepto que ha evolucionado a una velocidad exorbitante.

A mediados de los noventa se inició el boom de Internet. En esa época el número de proveedores de acceso privado se disparó, permitiendo a millones de personas acceder a Internet, que a partir de ese momento se empezó a conocer como la Red, desbancado a las demás redes de comunicación existentes (*Compuserve, FidoNet/BBS*).

El punto de inflexión vino marcado por la aparición de implementaciones del Protocolo TCP/IP (*Transmission Control Protocol / Internet Protocol*, Protocolo de Control de Transmisión / Protocolo de Internet) gratuitas, así como por la popularización y abaratamiento de medios de acceso cada vez más rápidos (módems de mayor velocidad, cable, satélite, fibra óptica y microondas). El efecto de todos estos cambios fue de "*bola de nieve*": a medida que se conectaban más usuarios, los costes se reducían, aparecían más proveedores e Internet se hacía más atractivo y económico.



Disponer de una dirección de correo electrónico de acceso a la Web, ha dejado de ser una novedad para convertirse en algo normal en muchos países del mundo. Por eso las empresas, instituciones y administraciones están migrando rápidamente todos sus servicios, aplicaciones, tiendas, a un entorno Web que permita a sus clientes y usuarios acceder por Internet.

Es la costumbre, la que obliga a pensar en internet como un solo ente capaz de brindar múltiples funcionalidades a las personas, por el contrario, como se explicó anteriormente, internet es un conjunto de recursos, de entre los cuales destaca por su popularidad la World Wide Web, la cual facilita la existencia de las denominadas páginas Web, concepto fundamental sobre el que se encuentra cimentado este documento.

## 3.1 World Wide Web

La WWW (*World Wide Web*), es en realidad un armazón arquitectónico diseñado para acceder a documentos distribuidos en miles de computadoras en toda Internet; Su enorme popularidad se debe a que cuenta con una interfaz gráfica atractiva, fácil de usar y proporciona acceso a una enorme cantidad de información sobre casi cualquier tema.

Sus comienzos se remontan a 1989 en el CERN, (*Conseil Européen pour la Recherche Nucléaire,* Centro Europeo de Investigación Nuclear) en Ginebra, donde surgió como una necesidad de lograr que grandes grupos de investigadores, dispersos internacionalmente, colaboraran en proyectos de investigación de partículas, intercambiando informes, planos, dibujos, fotos y otros documentos. [Tanenbaum (2003)]

La propuesta inicial de una red de documentos vinculados, surgió del físico Tim Berners – Lee en Marzo de 1989, 18 meses después fue demostrado públicamente en la conferencia Hypertext 91 en San Antonio Texas.

En marzo de 1991, Berners-Lee escribió el primer programa visualizador para un servidor y cliente, que se convirtió en el origen de la World Wide Web. Este sistema duró hasta 1993, cuando la ISO (*International Standards Organization*, Organización Internacional de Estándares) estandarizó el lenguaje HTML (*Hypertext Markup Language*, Lenguaje de Marcado de Hipertexto). Hasta entonces, los documentos se editaban mediante TeX y PostScript, pero estos lenguajes eran demasiado complicados teniendo en cuenta que debían ser leídos por todo tipo de computadoras. [Coulouris, (2001)]



Por este motivo se desarrolló un método eficiente y rápido para intercambiar datos entre la comunidad científica. Se combinaron dos tecnologías ya existentes el hipertexto y el protocolo de comunicaciones de Internet, creando un nuevo modelo de acceso a la información intuitivo e igualitario: la Web que hace posible que cualquiera pueda utilizar Internet.

En marzo de 1993, la Web supone el 0,1% del tráfico total de Internet y el CERN declaraba a la WWW como una tecnología de acceso libre. En septiembre del mismo año, el uso de la web ya alcanzaba el 1% del tráfico de Internet. En octubre, había unos 500 servidores web activos. A partir de ahí, su crecimiento es milagroso: a finales de 1994 existían ya más de 10.000 servidores y 10 millones de usuarios. Y en 1997, la cifra superaba los 650.000 servidores.

Durante 1994, el CERN y el MIT (*Massachusetts Institute of Technology*, Instituto Tecnológico de Massachusetts) firman un acuerdo para establecer el W3C (*World Wide Web Consortium*), una organización dedicada al desarrollo de la Web, estandarización

de protocolos y el fomento de interoperabilidad entre los sitios, siendo Tim Berners – Lee, designado como director de la misma. [Tanenbaum, (2003)]

A partir de entonces el diseño, así como la estructura de las páginas web ha variado de manera significativa. Por un lado, las interfaces gráficas se han hecho completamente multimedia y la hipermedia ha desbancado al hipertexto en la Web.

Hasta hoy, la Web no ha cesado en su desarrollo, evolucionando con nuevas capacidades, teniendo interfaces gráficas que se han hecho completamente multimedia y mientras que el hipertexto ha ido perdiendo terreno, mejorando este sistema digital, en red que es el Internet.

El éxito de la Web se basa en dos puntos fundamentales: el protocolo HTTP y el lenguaje HTML. El primero permite una implementación simple y sencilla de un sistema de comunicaciones para enviar cualquier tipo de ficheros de una forma fácil, simplificando el funcionamiento del servidor y permitiendo que servidores poco potentes atiendan miles de peticiones y reduzcan los costes de despliegue. Mientras que el segundo, proporciona un mecanismo de composición de páginas enlazadas simple y fácil, altamente eficiente y de uso muy simple.

Desde el punto de vista de usuario, Web consiste en un enorme conjunto de documentos a nivel mundial, generalmente llamados páginas Web. Cada página puede contener vínculos o enlaces a otras páginas relacionadas en cualquier lugar del mundo. Los usuarios pueden seguir un vínculo haciendo clic en él, lo que los llevara a la página apuntada.



Las páginas Web pueden ser visualizadas mediante un programa llamado navegador; como puede ser Internet Explorer, Firefox, Chrome, etc. El navegador obtiene la página solicitada, interpreta el texto y los comandos de formateo que contiene, y despliega la página en la pantalla. En la figura 3.1 se observa e modelo básico del funcionamiento de Web.

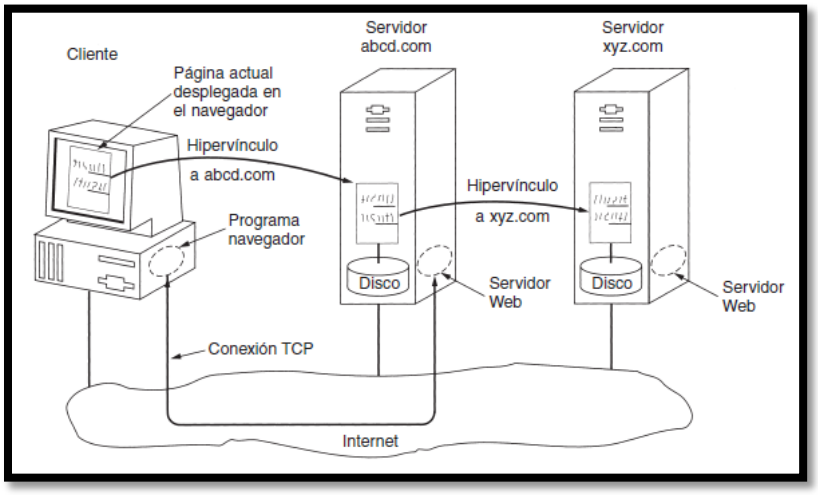



**Figura 3.1.1. Modelo web**

### 3.1.1 Cliente

En esencia, un navegador es un programa que cuenta con la capacidad de desplegar una página Web y determinar los clics que se hacen en los elementos de la página desplegada. Cuando se selecciona un elemento, el navegador sigue el hipervínculo y obtiene la página seleccionada. Las páginas Web se nombran utilizando URL (*Uniform Resource Locator*, Localizadores Uniformes de Recursos).

Un URL típico tiene la forma: http://www.abcd.com/productos.html

Cuando el usuario hace clic en un hipervínculo, el navegador lleva acabo un procedimiento específico para obtener la página a la que se intenta acceder, siendo este procedimiento: [Lujan, (2002)]

1) El navegador determina el URL
2) El navegador solicita al DNS la dirección IP del destino
3) El DNS responde con la dirección IP del destino
4) EL navegador realiza una conexión TCP al puerto 80 de la dirección IP destino
5) EL navegador envía un mensaje solicitando el archivo Index.html
6) El servidor responde a este petición enviando el archivo
7) Se libera la conexión TCP
8) El navegador despliega todo el texto del archivo solicitado
9) El navegador obtiene y despliega todas las imágenes del archivo



Para poder mostrar cualquier página Web, es necesario que el navegador entienda el formato de la página, motivo por el cual, todas las páginas Web son escritas en un lenguaje estandarizado llamado HTML. [Tanenbaum, (2003)]

### 3.1.2 Servidor

Al hablar de páginas Web, el concepto de servidor, automáticamente se enfoca en los servidores Web, los cuales son programas que procesan una aplicación del lado del servidor, realizando conexión bidireccionales y/o unidireccionales con el cliente, generando o cediendo una respuesta.

El servidor Web se ejecuta en una computadora manteniéndose a la espera de peticiones por parte de algún cliente, y responde a estas peticiones mediante una página Web que se mostrara en el navegador o bien mostrando el respectivo mensaje si se detectó algún error.

El procedimiento de respuesta a estas peticiones es el siguiente: [Gourley, (2002)]

1) Acepta una conexión TCP de un cliente
2) Obtiene el nombre del archivo solicitado
3) Obtiene el archivo del disco duro
4) Regresa el archivo al cliente
5) Libera la conexión TCP

Los servidores Web modernos hacen más que solo aceptar y regresar nombres de archivos, debido a que el procesamiento real de cada solicitud puede ser muy complicado.

### 3.1.3 URL



Las conexiones en la WWW se basan en el protocolo TCP, lo cual asigna una dirección IP a cada una de las páginas Web, de forma que cuando se accede a la dirección www.google.com, en realidad el navegador realiza todo el proceso de conexión a la dirección 74.125.227.84.

El URL es una cadena de caracteres alfanuméricos con la cual se asigna una dirección única a cada uno de los recursos de información disponibles en Internet, existe un URL única para cada página de cada uno de los documentos de la WWW, y permite simplificar la búsqueda de información. El URL asigna también un nombre entendible para las personas a cada uno de estos recursos, lo cual simplifica drásticamente la dificultad de recordar cada una de las direcciones IP, al transformarlas en direcciones como: www.google.com.

Los URL se están conformados por tres partes: el protocolo o esquema, el nombre DNS de la computadora donde se encuentra la página y un nombre local que indica de manera única la página especifica o nombre del archivo. [Coulouris, (2001)] [Gourley, (2002)]

El esquema de URL es abierto, en el sentido de que es directo hacer que los navegadores utilicen múltiples protocolos para obtener diferentes tipos de recursos, la Tabla 3.1.3.1 muestra las formas simplificadas de las URL más comunes.

**Tabla 3.1.3.1. URL comunes**

| Protocolo | Uso | Ejemplo |
|---|---|---|
| **http** | Hipertexto | http://www.cs.vu.nl/~ast/ |
| **ftp** | Transferencia de archivos | ftp://ftp.cs.vu.nl/pub/minix/README |
| **file** | Archivo local | file:///usr/suzanne/prog.c |
| **news** | Grupo de noticias | news:comp.os.minix |
| **news** | Articulo | news:AA0134223112@cs.utah.edu |
| **gopher** | Gopher | gopher://gopher.tc.umn.edu/11/Libraries |
| **mailto** | Envió de correo electrónico | mailto:JohnUser@acm.org |
| **telnet** | Sesión remota | telnet://www.w3.org:80 |

### 3.1.4 El protocolo HTTP

El protocolo HTTP (*Hyper Text Transfer Protocol*, Protocolo de Transferencia de Hipertexto), es el protocolo base de la WWW. Especifica cuales mensajes pueden enviar los clientes a los servidores y que respuestas obtienen. Cada interacción consiste en una solicitud ASCII (*American Standard Code for Information Interchange*, Código Estándar Estadounidense para el Intercambio de Información), seguida por una respuesta tipo MIME (*Multipurpose Internet Mail Extensions*, Extensiones Multipropósito de Correo de Internet). Todos los clientes y servidores se deben de regir por este protocolo. [w3.org. (2013)]



El protocolo HTTP se define en el RFC 2616, que especifica la versión 1.1 de HTTP, la cual expone la sintaxis y semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse. [Group, N. W. (2013)]

Se trata de un protocolo orientado a transacciones y sigue el esquema petición – respuesta entre un cliente y un servidor

El protocolo no mantiene estado, es decir, cada transferencia de datos es una conexión independiente de la anterior, sin relación alguna entre ellas, hasta el punto de que para transferir una página Web tenemos que enviar el código HTML del texto, así como las imágenes que la componen, pues en la especificación inicial de HTTP, se abrían y usaban tantas conexiones como componentes tenía la página, transfiriéndose por cada conexión un componente (el texto de la página o cada una de las imágenes). [Lujan Mora, S. (2002)]

Existe una variante de HTTP llamada HTTPS (S por Secure) que utiliza el protocolo de seguridad SSL (*Secure Socket Layer*, capa de conexión segura) para cifrar y autenticar el tráfico entre cliente y servidor, siendo ésta muy usada por los servidores web de comercio electrónico, así como por aquellos que contienen información personal o confidencial.

En la Figura 3.1.4.1 se muestra de manera esquemática el funcionamiento de HTTP.


**Figura 3.1.4.1. Petición HTTP**

## 3.1.5 Conexiones



El proceso estándar en que un navegador contacta con un servidor web es estableciendo una conexión TCP (*Transmission Control Protocol*, Protocolo de Control de Transmisión) con el puerto 80 de la computadora servidor, la ventaja de utilizar TCP radica en que ni los servidores o los navegadores, deberán de preocuparse por los mensajes largos, perdidos, duplicados o por las confirmaciones de recepción, ya que la implementación de TCP se encarga de manejar todos estos aspectos.

En HTTP 1.0, una vez establecida la conexión, se enviaba una solicitud, se obtenía una respuesta y después se liberaba dicha conexión. Este método resultaba adecuado en una época en la que una página Web consistía por completo en texto HTML. Sin embargo con el pasar del tiempo, las mejoras tecnológicas y el uso de multimedia, una página Web promedio contenía una gran cantidad de iconos, imágenes e incluso en ocasiones música, por lo que establecer una conexión TCP para transportar únicamente una imagen o un icono era un método demasiado costoso.

Considerando lo anterior, se diseñó HTTP 1.1, el cual cuenta con la capacidad de realizar conexiones persistentes, con ellas es posible establecer una conexión TCP, enviar una solicitud y obtener su respectiva respuesta y después enviar solicitudes adicionales. Esta cualidad permite reducir la sobre carga de TCP e incluso enviar las solicitudes adicionales antes de que la respuesta a la solicitud inicial haya sido recibida. [Gourley, (2002)]

### 3.1.6 Métodos

HTTP define ocho métodos que indican la acción que se desea efectuar en el sitio web identificado. Lo que este recurso representa o si los datos se generan de forma dinámica, dependen de la aplicación del servidor. A menudo, el recurso corresponde a un archivo o la salida de un ejecutable que reside en el servidor. [Marshall, (2013)]

En la tabla 3.1.6.1 se observan los métodos HTTP.

**Tabla 3.1.6.1. Métodos de Solicitud HTTP**

| Método | Descripción |
|---|---|
| **GET** | Solicita la lectura de una página web |
| **HEAD** | Solicita la lectura del encabezado de una página web |
| **PUT** | Solicita el almacenamiento de una página web |
| **POST** | Inserta algo en un recurso con nombre |
| **DELETE** | Elimina una página web |
| **TRACE** | Repite la solicitud entrante |
| **CONNECT** | Reservado para uso futuro |
| **OPTIONS** | Consulta cierta opciones |



Cada solicitud obtiene una respuesta que consiste en una línea de estado y posiblemente de información adicional. La línea de estado contiene un código de estado de tres dígitos que indica si la solicitud fue atendida y en caso contrario el motivo por el cual fue rechazada. En la práctica, los códigos 1XX no son utilizados con frecuencia, los códigos 2XX significan que la solicitud se realizó exitosamente y que se regresa el contenido en caso de existir. Los códigos 3XX indican al cliente que busque en otro lado, ya sea en una URL diferente o en su propio cache, los códigos 4XX significan que la solicitud fallo debido a un error del cliente, como pudiera ser una solicitud no valida o una página no existente y por último los códigos 5XX significan que existe un problema en el servidor, posiblemente un error de codificación o una sobre carga temporal. En la Tabla 3.1.6.2 se muestran los grupos de respuesta del código de estado. [Gourley, (2002)]

**Tabla 2.1.6.2. Grupos de respuesta del código de estado**

| Código | Significado | Ejemplo |
|---|---|---|
| **1xx** | Información | 111 = Conexión rechazada |
| **2xx** | Éxito | 200 = Solicitud exitosa |
| **3xx** | Redirección | 301 = Pagina movida |
| **4xx** | Error del cliente | 404 = Pagina no encontrada |
| **5xx** | Error del servidor | 500 = Error interno del servidor |

### 3.1.7 Aplicaciones web

Inicialmente la Web era simplemente una colección de páginas estáticas, y documentos, que podían consultarse o descargarse. Comenzó su evolución con la inclusión de un método para confeccionar páginas dinámicas que permitiesen que lo mostrado fuese dinámico (generado o calculado a partir de los datos de la petición).

Dicho método fue conocido como CGI (Common Gateway Interface) y definía un mecanismo mediante el cual podíamos pasar información entre el servidor HTTP y programas externos. Los CGI siguen siendo muy utilizados, puesto que la mayoría de los servidores Web los soportan debido a su sencillez. Además, proporcionan total libertad a la hora de escoger el lenguaje de programación para desarrollarlos.

El esquema de funcionamiento de los CGI tenía un punto débil: cada vez que se recibía una petición, el servidor Web lanzaba un proceso que ejecutaba el programa CGI.

Por otro lado, la mayoría de CGI's eran escritos en algún lenguaje interpretado (Perl, Python) o en algún lenguaje que requería entorno de ejecución (Visual Basic, Java), esto implicaba una gran carga para la máquina del servidor. Además, si la Web tenía muchos accesos al CGI, esto suponía problemas graves.



Una aplicación web está comúnmente estructurada como una aplicación de tres-capas. En su forma más común, el navegador web es la primera capa, un motor usando alguna tecnología web dinámica (ejemplo: CGI, PHP, Java Servlets o ASP) es la capa de en medio, y una base de datos como última capa. El navegador web manda peticiones a la capa media, que la entrega valiéndose de consultas y actualizaciones a la base de datos generando una interfaz de usuario. Debido a dicha estructura, se presentan dentro de las aplicaciones multinivel. [Coulouris, (2001)]

Una ventaja significativa en la construcción de aplicaciones web que soporten las características de los navegadores estándar es que deberían de funcionar igual independientemente de la versión del Sistema Operativo instalado en el cliente.

Las Aplicaciones Web tienen sus ventajas, por ejemplo, permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados, permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar.

Los servicios Web se encuentran conformados por los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento. Al apoyarse en HTTP, pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.

Una razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más acusada.

En la Figura 3.1.7.1 se presenta el esquema de esta arquitectura, compuesta por: interfaz o capa de presentación, lógica de la aplicación o capa de negociación y los datos.


**Figura 3.1.7.1. Arquitectura web de tres capas**

La capa intermedia es el código que el usuario invoca para recuperar los datos deseados. La capa de presentación recibe los datos y los formatea para mostrarlos adecuadamente. Esta división entre la capa de presentación y la de la lógica permite una gran flexibilidad a la hora de construir aplicaciones, ya que se pueden tener múltiples interfaces sin cambiar la lógica de la aplicación. La tercera capa consiste en los datos que gestiona la aplicación.

Estos datos pueden ser cualquier fuente de información como una base de datos o documentos XML. El funcionamiento de la arquitectura de tres niveles, se puede entender mejor, tomando en cuenta que una aplicación Web típica recoge datos del usuario (primer nivel), los envía al servidor, que ejecuta un programa (segundo y tercer nivel) y cuyo resultado es formateado y presentado al usuario en el navegador (primer nivel otra vez).

Una aplicación web con bases de datos está diseñada para ayudar al usuario a realizar una tarea. Puede ser una aplicación simple que muestre información en la ventana del explorador o puede ser un programa complicado con funcionalidad extendida. Y consta de una aplicación Web y una base de datos –solo dos partes-:

- *Base de datos*. La base de datos es la memoria de largo plazo de la aplicación para la Web. La aplicación no puede cumplir su propósito sin la base de datos. No obstante, la base de datos por sí misma no es suficiente.

- *Aplicación*. La parte de la aplicación es el programa o grupo de programas que realiza las tareas. Los programas crean la presentación visual que el usuario ve en la ventana del explorador; hacen que la aplicación sea interactiva al aceptar y procesar la información que el usuario digita en la ventana del explorador; almacenan información en la base de datos y extraen información de la base de datos. (La base de datos es inútil a menos que se puedan insertar y extraer datos de ella).

Sin embargo los componentes fundamentales de toda aplicación Web compleja serán siempre: Lenguaje de programación del lado del servidor y un Servidor Web.

### 3.1.8 HTML y XHTML

Actualmente las páginas Web son escritas en un lenguaje denominado HTML, el cual permite a los usuarios producir páginas Web que incluyan texto, graficas e hipervínculos a otras páginas Web. HTML es un lenguaje de marcado que define cual es el formato de los documentos Web.



Al integrar todos los comando de marcado dentro de cada archivo HTML y estandarizarlos, es posible que cualquier navegador Web les y despliegue correctamente cualquier página Web. Esta capacidad es crucial, debido que ha quedado atrás la época en la que solo era posible acceder a una página Web mediante una computadora. Actualmente es posible realizarlo desde dispositivos electrónicos como celulares, Tablet y laptops, los cuales difieren totalmente entre los tamaños de la pantalla en la que será visualizada la página Web.

HTML surge en 1990 como un subconjunto de SGML (*Standard Generalized Markup Language* o, Estándar de Lenguaje de Marcado Generalizado) de la mano de Tim Berners – Lee, con el paso del tiempo, el estándar ha sido redefinido en varias ocasiones, sin embargo los más destacables son HTML 4.0, el cual se publicó en 1997 y se mantendría hasta 2004 cuando el W3C reabrió el debate para la evolución del lenguaje y se dan a conocer las bases para HTML 5, no obstante este trabajo fue rechazado por los miembros del W3C, dando preferencia al desarrollo de XML y la aparición de XHTML. Finalmente, el W3C retomaría el desarrollo de HTML en 2006. [Tanenbaum, (2003)] [Lujan, (2002)]

A partir de HTML 2.0 se integró una de las características que más ha revolucionado las páginas Web, los formularios. Estos contienen cuadros o botones que permiten al usuario proporcionar información o tomar decisiones, y después enviar dicha información al dueño de la página.

Los formularios utilizan la etiqueta <input> para realizar sus funciones, esta etiqueta, tiene una gran variedad de parámetros para determinar el tamaño, naturaleza y uso del cuadro presentado. Los formularios más comunes constan de campos en blanco para aceptar texto del usuario, casillas de verificación que pueden marcarse, mapas activos y botones de envió.

Debido a las limitaciones de HTML, en el año 2000, surge XHTML (*eXtensible HyperText Markup Language*, Lenguaje de Marcado de Hipertexto Extensible) el cual es una versión más estricta y limpia de HTML, extiende la especificación de HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML el cual fue diseñado para describir los datos.

Surge como un lenguaje de etiquetado más estricto que HTML, lo cual permite una correcta interpretación de la información independientemente del dispositivo desde el que se acceda a ella. XHTML es capaz de incluir otros lenguajes como MathML, SMIL o SVG al contrario que su predecesor.

XHTML está orientado al uso de un etiquetado correcto, por lo que exige cumplir con una serie de requisitos básicos en lo que a codificación se refiere. Entre estos requisitos se puede mencionar una estructuración coherente dentro del documento donde se incluirán elementos correctamente anidadas, etiquetas en minúsculas, elementos cerrados correctamente, atributos de valores entrecomillados, etc. [w3.org. (2013)]



### 3.1.9 Documentos Web Dinámicos

Hablar únicamente de HTML o XHTML, significa hablar de páginas Web con contenido estático, es decir, información almacenada en disco duro y todos los usuarios son capaces de ver única y exclusivamente la misma información.

Las páginas Web dinámicas surgen como respuesta a la necesidad de los usuarios de obtener información personalizada, que sea generada en base a una solicitud específica, la creación de este tipo de páginas Web se divide en dos puntos de generación del contenido: páginas Web dinámicas en el servidor y páginas Web dinámicas en el cliente.

Debido a los riesgos que representan y a que su estudio está fuera del alcance de este documento, las páginas Web dinámicas en el cliente no serán analizadas.

Básicamente, el contenido dinámico del lado del servidor se genera mediante un lenguaje de programación de lado el servidor, como puede ser JSP, PHP, ASP, etc. Este lenguaje de programación procesara la información solicitada por el usuario, y la enviara al navegador como respuesta en un formato que pueda ser entendido por el mismo, es decir HTML o XHTML. [Tanenbaum, (2003)] [Coulouris, (2001)]

Dependiendo del tipo de arquitectura en la que se encuentre desarrollada la página Web, será el alcance y la capacidad de información que será entregada al usuarios, por ejemplo, en una arquitectura de tres capas, el servidor Web recibirá mediante un formulario la petición del usuario, para esto verificara si la información solicitada se encuentra en la base de datos, se comunicara con esta capa y solicitara dicha información.

La base de datos buscara la información dentro de su sistema y en caso de encontrarla la enviara de regreso al servidor Web, el cual le dará formato y finalmente la enviara al navegador para ser visualizada por el usuario.

La Figura 3.1.9.1 ilustra el proceso de información de un formulario HTML

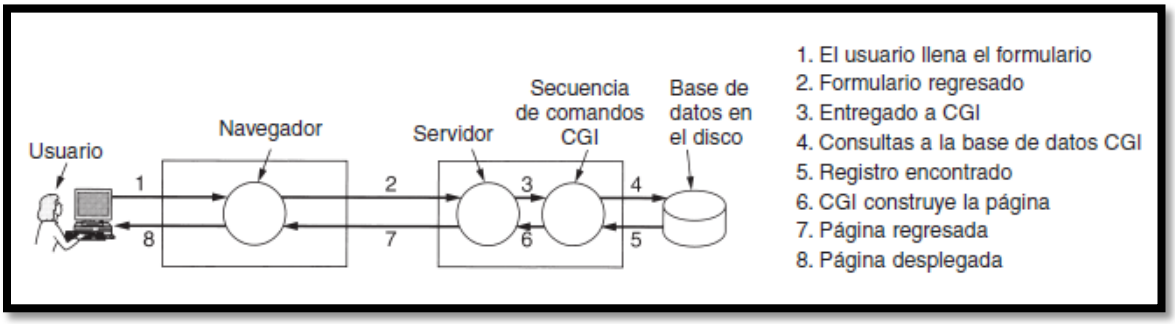



**Figura 3.1.9.1. Procesamiento de un formulario HTML**



## 3.1.10 PHP

PHP (*Hypertext Preprocessor*, Pre Procesador de Hipertexto) es un lenguaje de programación, relativamente nuevo, concebido principalmente como herramienta para el desarrollo de aplicaciones Web. PHP permite diseñar paginas dinámicas de servidor, es decir, generar paginas bajo petición capaces de responder de manera inteligente a las demandas del cliente y que permiten la automatización de gran cantidad de tareas. PHP es un lenguaje interpretado de alto nivel embebido en HTML y ejecutado en el servidor.

En junio de 1998 se liberó oficialmente PHP 3.0 anunciado como sucesor oficial de PHP/FI 2.0. El siguiente paso en la evolución de PHP consistió en la reescritura de su núcleo, dando lugar a un nuevo motor denominado Zeed (acrónimo de los apellidos Zeev y Andi). PHP 4.0, basado en este motor, y acoplado con un gran rango de nuevas características adicionales, fue oficialmente liberado en mayo de 2000.

La última y actual versión de PHP, liberada en Julio de 2004, es la 5.0, está basada en el nuevo motor Zend 2¸el cual ha vuelto a ser reescrito por completo. Entre sus características y novedades más resaltables está el completo soporte para la programación orientada a objetos. [Group, T. P. (2013)] [PHP. (2013)]

También incorpora la gestión de excepciones, una nueva librería de XML, soporte nativo para el sistema gestor de base de datos SQLite, y mejoras en la gestión de las cadenas de caracteres. PHP 5.0 soporta también MySQLi, una nueva ampliación de MySQL, la cual, además de la interfaz habitual, encierra una interfaz basada en objetos.

En la Figura 3.1.10.1 es posible observar que PHP se ha convertido en uno de los lenguajes del lado del servidor, más ampliamente utilizados para el desarrollo de páginas dinámicas junto con ASP, JSP, ColdFusion y Perl. [Netcraft. (2013)]

Según encuestas realizadas por la compañía inglesa Netcraft, que entre otros servicios se encarga de estar analizando constantemente el estado de los sitios Web de internet. Las estadísticas abarcan de Enero del año 2002 a Enero del año 2012.

En torno al aumento del desarrollo de páginas web, los desarrolladores deseaban hacer más que páginas estáticas, querían interactuar con los visitantes, recopilar información de los usuarios y brindar páginas web que estuvieran hechas a la medida de cada individuo.

Existen lenguajes de scripting del lado del servidor, utilizados para la web, como lo es el caso de PHP, que puede crear dinámicamente el código HTML, que genera la página web, lo cual permite a los usuarios individuales ver páginas web personalizadas. Los visitantes de la página web ven el output de los scripts, pero no los scripts mismos



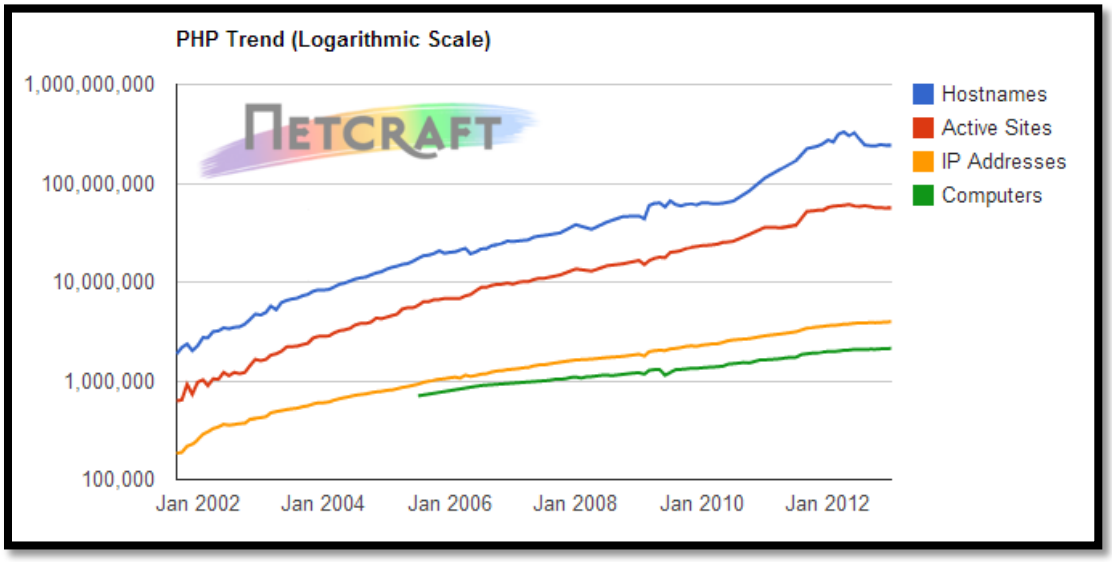



**Figura 3.1.10.1. Estadística de uso de PHP**

PHP tiene muchas características diseñadas específicamente para su uso en los sitios web, incluyendo los siguientes:

- Interactúa con formularios HTML. PHP puede mostrar un formulario HTML y procesar la información que el usuario digita.

- Se comunica con bases de datos. PHP puede interactuar con bases de datos para almacenar información del usuario o recuperar información que se le enseña al usuario.

- Genera páginas web seguras. PHP permite al desarrollador crear páginas web seguras que obligan a los usuarios a digitar un nombre de usuario y una contraseña validos antes de ver el contenido de la página web.

Cada sitio web requiere de un servidor web. PHP funciona en sociedad con el servidor web. El servidor web es el software que entrega las páginas web al mundo.

Cuando se instala PHP, el servidor web se configura para buscar código PHP intercalado en los archivos con extensiones específicas.

Cuando el servidor web recibe una solicitud para un archivo con la extensión designada .php, envía los enunciados HTML tal y como están, pero los enunciados PHP son procesados por el software PHP antes de ser enviados a quien los solicito, como se puede observar en la Figura 3.1.10.2, en el que se muestra el funcionamiento de las páginas PHP.

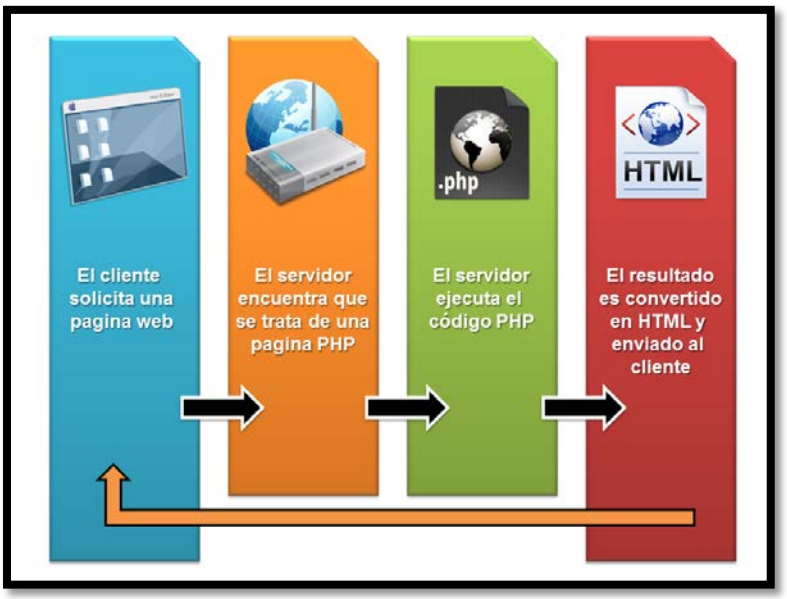



**Figura 3.1.10.2. Procesamiento de petición a un sitio web con PHP**



Cuando los enunciados en lenguaje PHP se procesan, el output consiste en enunciados HTML. Los enunciados en el lenguaje PHP no se incluyen en el HTML enviado al explorador, de manera que el código PHP es seguro y transparente para el usuario.

PHP no está integrado con todos los servidores web pero funciona con muchos de los más populares.

Es desarrollado como un proyecto de Apache Software Fundation y, en consecuencia, funciona mejor con Apache, pero también funciona con Microsoft IIS/PWS, iPlanet (anteriormente Netscape Enterprise Server) por mencionar algunos.

Algunas de las ventajas que han hecho que PHP sea más utilizado en la Web son:

- Rápido. Como esta empotrado en condigo HTML, el tiempo de respuesta es muy corto.
- Fácil de usar. PHP contiene muchas características y funciones especiales necesarias para crear páginas web dinámicas. El lenguaje PHP está diseñado para incluirse con facilidad en archivos en HTML.
- Seguro. El usuario no ve el código PHP.
- Diseñado para mantener bases de datos. La funcionalidad de PHP fue diseñado para interactuar con bases de datos específicas.
- Soporte para múltiples sistemas operativos. Unix (entre otras, Linux, HP-UX, Solaris y OpenBSD). Microsoft Windows, Mac OS X, RISC OS.
- Soporte para múltiples servidores web. Apache, Microsoft Internet Information Server, Personal Web Server, iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros.



### 3.1.11 Apache

El protocolo que emplean el servidor y el cliente para comunicarse es el HTTP o protocolo para la transmisión de hipertexto. Este es un protocolo orientado a caracteres del tipo solicitud/respuesta. Por lo general los servidores Web escuchan las solicitudes de los clientes a través del puerto 80 y es a este a donde se van a dirigir los clientes por defecto para hacer sus solicitudes.

Apache es un servidor de páginas web que nace a partir del servidor http de la NCSA. Convirtiéndose automáticamente en rival de los servidores http de Unix utilizados hasta la fecha por su eficiencia, funcionalidad y rapidez. Se desarrolla de forma estable y segura gracias a la cooperación y los esfuerzos de un grupo de personas conocidas como grupo Apache (*Apache Group*), los cuales se dedican a perfeccionar el servidor y su documentación regidos por la ASF (*Apache Software Foundation*). [Foundation, A. S. (2013)]

El servidor Apache es un software que está estructurado en módulos. La configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Los módulos del Apache se pueden clasificar en tres categorías: [Apache. (2013)]

- Módulos Base. Módulo con las funciones básicas del Apache

- Módulos Multiproceso. Son los responsables de la unión con los puertos de la máquina, aceptando las peticiones y enviando a los hijos a atender a las peticiones

- Módulos Adicionales. Cualquier otro módulo que le añada una funcionalidad al servidor.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código.

El resto de funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software.

Apache es usado primariamente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.



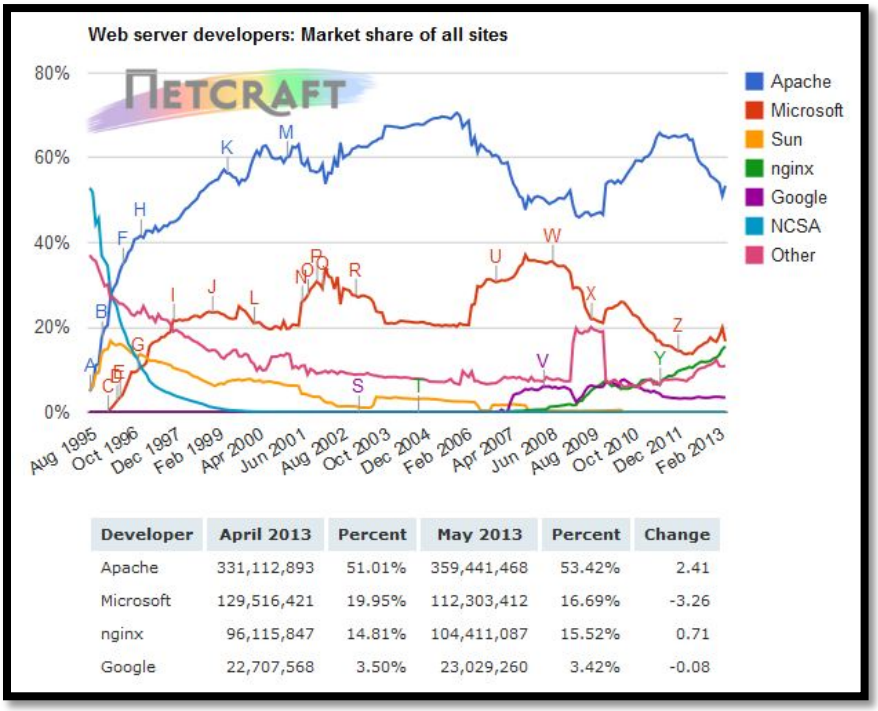
Apache es el componente de servidor web en la popular plataforma de aplicaciones LAMP, junto a MySQL y los lenguajes de programación PHP/Perl/Python (y ahora también Ruby).

Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Un ejemplo es al momento de compartir archivos desde una computadora personal hacia Internet. Un usuario que tiene Apache instalado en su escritorio puede colocar arbitrariamente archivos en la raíz de documentos de Apache, desde donde pueden ser compartidos. [Niq. (2013)]

Microsoft Internet Information Services (IIS) es el principal competidor de Apache, así como Sun Java System Web Server de Sun Microsystems y un anfitrión de otras aplicaciones como Zeus Web Server. Algunos de los más grandes sitios web del mundo están ejecutándose sobre Apache. Apache tiene amplia aceptación en la red: desde 1996, es el servidor HTTP más usado, según análisis de la compañía Netcraft, que deja en claro el dominio del servidor web Apache.

Gracias a las estadísticas de Netcraft siempre se ha sabido que la mayoría de los sitios del mundo utilizan el servidor web de código abierto Apache. En la Figura 3.1.11.1 se muestra la interpretación donde se considera sólo los sitios webs de alto tráfico, quitando gran parte del ruido estadístico. [Netcraft. (2013)]

Para esta medición se tomaron en cuenta un millón de sitios, en donde por relevancia quedaron eliminados los blogs de mala muerte y sitios que están de capa caída. Quedando así la cuota de mercado para los servidores superiores a través de todos los dominios de Agosto de 1995 a Febrero de 2013.




| Developer | April 2013 | Percent | May 2013 | Percent | Change |
|---|---|---|---|---|---|
| Apache | 331,112,893 | 51.01% | 359,441,468 | 53.42% | 2.41 |
| Microsoft | 129,516,421 | 19.95% | 112,303,412 | 16.69% | -3.26 |
| nginx | 96,115,847 | 14.81% | 104,411,087 | 15.52% | 0.71 |
| Google | 22,707,568 | 3.50% | 23,029,260 | 3.42% | -0.08 |



**Figura 3.1.11.1. Cuota de mercado de los sitios activos en todos los dominios**

Entre las características principales de Apache se encuentran: que es un servidor Web potente, flexible, altamente configurable y extensible; puede ser configurado a través de la definición de módulos empleando su propia API (*Aplication Programming Interface*, Interface de Programación de Aplicaciones); se distribuye para diversas plataformas como es: Windows, Macintosh, Novell NetWare, OS/2, Linux y la mayoría de los Unix existentes.

## 3.2 Bases de Datos

Uno de los objetivos fundamentales de un sistema de información es contar con los mecanismos necesarios para poder encontrar y recuperar información. De esta forma, las bases de datos se han convertido en un elemento indispensable no sólo para el funcionamiento de los grandes motores de búsqueda y la recuperación de información a lo largo y ancho de la Web, sino también para la creación de sedes web, Intranets y otros sistemas en los que se precisa manejar grandes o pequeños volúmenes de información.

La creación de una base de datos a la que puedan acudir los usuarios para hacer consultas y acceder a la información que les interese es, pues, una herramienta imprescindible de cualquier sistema informativo sea en red o fuera de ella. Una base de datos se puede definir como una colección de datos organizados y estructurados según un determinado modelo de información que refleja no sólo los datos en sí mismos, sino también las relaciones que existen entre ellos.

Se diseña con un propósito específico y debe ser organizada con una lógica coherente. Los datos podrán ser compartidos por distintos usuarios y aplicaciones, pero deben conservar su integridad y seguridad al margen de las interacciones de ambos



Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

En una base de datos, las entidades y atributos del mundo real, se convierten en registros y campos. Estas entidades pueden ser tanto objetos materiales como libros o fotografías, pero también personas e, incluso, conceptos e ideas abstractas. Las entidades poseen atributos y mantienen relaciones entre ellas.

Los modelos clásicos de tratamiento de los datos son: [Campos, (2005)]

- *Jerárquico*. Puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos. Este modelo tiene forma de árbol invertido en el que una rama puede tener varios hijos, pero cada hijo sólo puede tener un padre.

- *En red.* Este modelo permite la representación de muchos a muchos, de tal forma que cualquier registro dentro de la base de datos puede tener varias ocurrencias superiores a él. El modelo de red evita redundancia en la información, a través de la incorporación de un tipo de registro denominado el conector. En el modelo en red se representa el mundo real mediante registros

lógicos que representan a una entidad y que se relacionan entre sí por medio de flechas.

- *Relacional*. En este modelo se representa el mundo real mediante tablas relacionadas entre sí por columnas comunes. Las bases de datos que pertenecen a esta categoría se basan en el modelo de relaciones, cuya estructura principal es la relación, es decir una tabla bidimensional compuesta por líneas y columnas. Cada línea, que en terminología relacional se llama tupla, representa una entidad que se quiere memorizar en la base de datos. Las características de cada entidad están definidas por las columnas de las relaciones, que se llaman atributos.

La Figura 3.2.1 muestra que existen entidades con características comunes, es decir descritas por el mismo conjunto de atributos, que forman parte de la misma relación.

Dentro de las Bases de Datos destacan las distribuidas ya que cada vez es más corriente el uso de arquitecturas de cliente-servidor y trabajo en grupo. Los principales problemas que se generan por el uso de la tecnología de bases de datos distribuidas se refieren a la duplicidad de datos y a su integridad al momento de realizar actualizaciones a los mismos.

Además, el control de la información puede constituir una desventaja, debido a que se encuentra diseminada en diferentes localizaciones geográficas. [Azundris. (2013)]



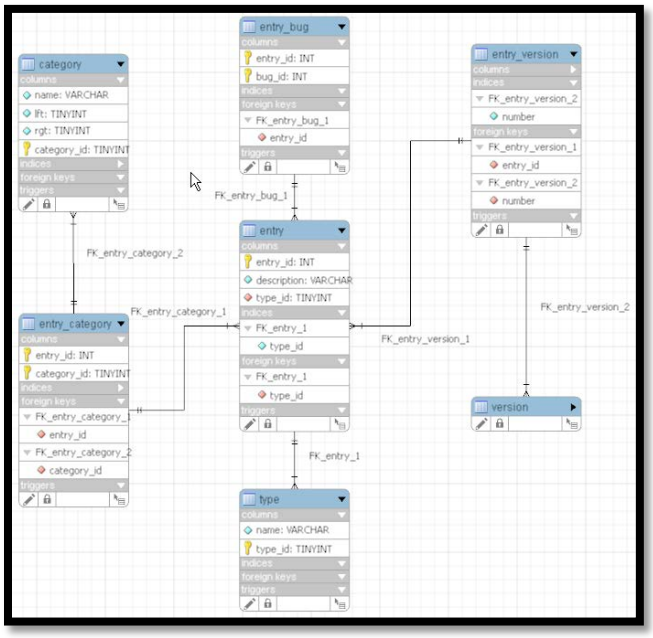



**Figura 3.2.1. Ejemplo de esquema base de datos relacional**

A principios de los 90 hacen su aparición los modelos de bases de datos orientadas a objetos. En estos, el esquema de la base de datos está representado por un conjunto

de clases que definen las características y el comportamiento de los objetos que conformarán la base de datos. La diferencia principal respecto a los modelos anteriores es la no positividad de los datos.

Esto es, con una base de datos tradicional, las operaciones que se tienen que efectuar en los datos se les piden a las aplicaciones que los usan. Con una base de datos orientada a objetos sucede lo contrario, los objetos memorizados en la base de datos contienen tanto los datos como las operaciones posibles con tales datos. En cierto sentido, se podrá pensar en los objetos como en datos a los que se les ha dotado de "*cierto nivel de inteligencia*" que les permite saber cómo comportarse, sin tener que apoyarse en aplicaciones externas.

La arquitectura de un sistema de base de datos se basa en tres niveles distintos, como se explica en la Tabla 3.2.1.

Este modelo de arquitectura permite establecer el principio de independencia de los datos, ya sea que se trate de una independencia lógica o física. La primera significa que los cambios en el esquema lógico no deben afectar a los esquemas externos que no utilicen los datos modificados; la independencia física significa que el esquema lógico no se va a ver afectado por los cambios realizados en el esquema interno, correspondientes a modos de acceso. [Silberschatz, (2004)]

**Tabla 3.2.1. Niveles de un Sistema de Bases de Datos**

| Nivel | Descripción |
|---|---|
| **Nivel físico** | Nivel más bajo de abstracción y el nivel real de los datos almacenados. Define cómo se almacenan los datos en el soporte físico, así como los métodos de acceso. |
| **Nivel Conceptual** | Representación de los datos realizada por la organización, que recoge los datos parciales de los requerimientos de los diferentes usuarios y aplicaciones parciales. Incluye la definición de los datos y las relaciones entre ellos. |
| **Nivel de visión** | Es parte del esquema conceptual. Presenta toda la base de datos, mientras que los usuarios, por lo general, sólo tienen acceso a pequeñas parcelas de ésta. El nivel de visión es el encargado de dividir estas parcelas. |



Es posible referirse también a estos niveles como nivel interno, nivel conceptual y nivel externo. Para plasmar los tres niveles en el enfoque o modelo de datos seleccionado, es necesario un programa o aplicación que actúe como interfaz entre el usuario, los modelos y el sistema físico. Esta es la función que desempeñan los Sistemas de Gestión de Bases de Datos.

Un DBMS (*Data Base Management System*, Sistema de Gestión de Bases de Datos) no es más que un paquete de software, que se ejecuta en una computadora principal

(host) que es quien centraliza los accesos a los datos y actúa de interfaz entre los datos físicos y los usuarios. Este sistema es capaz de llevar a cabo funciones como la creación y gestión de la base de datos misma, el control de accesos y la manipulación de datos de acuerdo a las necesidades de cada usuario.

Así pues, las bases de datos pueden ser creadas, mantenidas y gestionadas por una serie de aplicaciones denominadas DBMS. El DBMS está formado por personas, máquinas, programas y datos. Estos sistemas de gestión abarcan el conjunto de rutinas de software interrelacionadas cada una de las cuales es responsable de una determinada tarea. [Silberschatz, (2004)]

Los DBMS tienen dos funciones principales que son:

- La definición de las estructuras para almacenar los datos.
- La manipulación de los datos.

Algunos de los componentes con los que debe contar un sistema de gestión de bases de datos ideal son: lenguaje de definición de esquema conceptual, un sistema de diccionario de datos, lenguaje de especificación de paquetes de entrada/salida, lenguaje de definición de esquemas de base de datos, una estructura simétrica de almacenamiento de datos, módulo de transformación lógica a física, generador de programas de aplicación, generador de programas de informes, lenguaje de consulta de propósito general, entre otros.



Además, los DBMS deben incorporar como herramienta fundamental dos tipos de lenguajes: uno para la definición de los datos, y otro para la manipulación de los mismos.

El primero se denomina DLL (*Data Definition Language*, Lenguaje de Definición de Datos) y es el que provee de los medios necesarios para definir los datos con precisión, especificando las distintas estructuras. [Silberschatz, (2004)]

El segundo se conoce como DML (*Data Manipulation Language*, Lenguaje de Manipulación de Datos) y es el facilita a los usuarios el acceso y manipulación de los datos.

Tradicionalmente se ha hecho una distinción clara entre dos tipos de bases de datos:

- *Bases de datos referenciales.* Aquellas bases de datos que ofrecen registros que a su vez son representaciones de documentos primarios, ya sean bibliográficas o de directorio.
- *Bases de datos fuente.* Son las que ofrecen el documento completo, no una representación del mismo, entre ellas existen las numéricas, textuales y mixtas que en estas últimas se combinan los dos tipos de información anteriores.

Sin embargo, el desarrollo de las aplicaciones multimedia ha dado un vuelco al concepto tradicional de base de datos, que sólo integraba elementos de información textual y numérica. Con el multimedia, han hecho su aparición otro tipo de objetos: gráficos, sonoros y audiovisuales que comparten el mismo entorno que los datos textuales y numéricos.

La aparición del CD-ROM y otros soportes ópticos como el DVD, han hecho posible el desarrollo de las bases de datos multimedia. A la vez, se han ido estandarizando poco a poco los formatos de archivo gráficos, de audio y de vídeo, y se han perfeccionado los métodos de compresión de este tipo de datos, ya que ocupan grandes cantidades de memoria.

Hasta épocas recientes, las bases de datos eran productos comerciales desarrollados y mantenidos por ciertas empresas que las comercializaban bien en formato CD-ROM o bien las distribuían para su consulta, previo pago, en línea vía telnet. La mayoría eran bases de datos bibliográficas o de legislación. Las organizaciones también contaban con sus propias bases de datos construidas sobre los sistemas de gestión más conocidos para crear y mantener bases de datos como FileMaker, Knosys, Access, etc.

Hoy todos estos programas se han visto obligados a ser compatibles con la Web y a ofrecer la posibilidad de acceder, buscar y recuperar los datos en línea vía protocolo http. De esta forma, se han desarrollado y comercializado una serie de herramientas y aplicaciones, comúnmente denominadas pasarelas web, que permiten consultar las viejas -o nuevas- bases de datos creadas con estos sistemas de gestión mediante el navegador web, pero también, la existencia de estas herramientas ha favorecido el hecho de que cualquier persona pueda hoy publicar su propia base de datos en su página web, para que pueda ser consultada por cualquier usuario de la red.



Estas pasarelas no son más que herramientas de software que permiten la comunicación entre el servidor web y la base de datos. Por medio de estas herramientas de acceso (CGI, DAO, ODBC, JSP, ASP, PHP) y desde entornos de desarrollo distintos es posible construir una base de datos utilizando aplicaciones y sistemas de gestión de bases de datos como Microsoft Access, Oracle, Sybase, MySQL, MSQL, SQL Server o PostgreSQL.

Para poder acceder a los datos que se encuentran en una base de datos, es necesario tomar en cuenta las interfaces de programación, donde se denota el proceso de acceso y manipulación de los datos, partiendo de la aplicación [Elmasri, (1997)]. El siguiente esquema muestra cuatro niveles o interfaces (Figura 3.2.2):

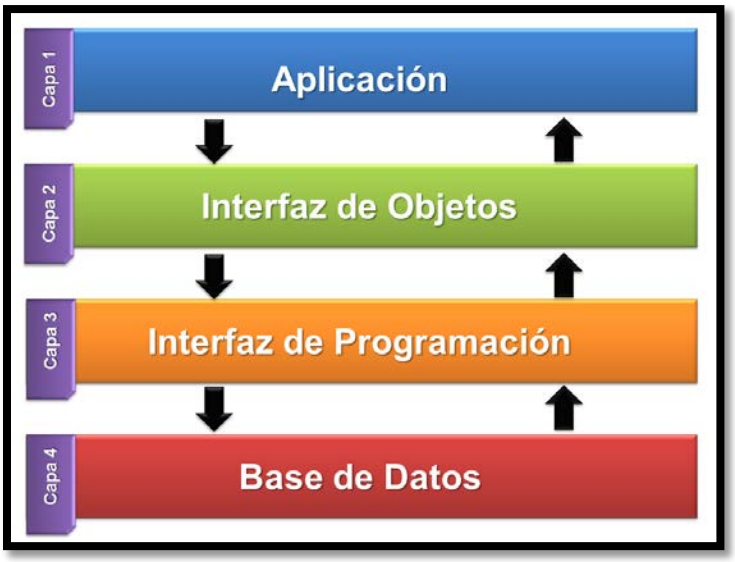



**Figura 3.2.2. Interfaces de programación para el acceso a datos**

La primera interfaz corresponde a la de Aplicación, la cual abarca y/o corresponde a cada uno de los programas clientes. La Interfaz de Objetos de Acceso a Datos, se encuentra como punto medio entre las aplicaciones y las API's que llegan a ser necesarias para el acceso a las bases de datos.



Entre las tecnologías que pertenecen a la Interfaz de Objetos de Acceso de Datos encontramos: DAO (Data Access Objects), ADO (ActiveX Data Objects), RDO (Remote Data Object), RDS (Remote Data Service) y MIDAS (Middle-tier Distributed Application Service). Su función es encapsular los componentes que se encuentran en la interfaz que corresponde a la de API's, con la finalidad de reducir el desarrollo de la aplicación y los costos de mantenimiento y deben situarse en todos los equipos que ejecuten la aplicación, ya que se encuentran casi de manera conjunta con la aplicación.

Por su parte, la Interfaz de Programación de Aplicaciones (Application Programming Interface, API), se encarga de mantener el diálogo con la base de datos, para poder llevar a cabo el acceso y manipulación de los datos. Algunos de los componentes que forman parte de esta interfaz son los siguientes: OLE DB, ODBC (Open Database Connectivity), JDBC (Java Data Base Connectivity), ISAPI (Internet Server Application Programming Interface), CGI (Common Gateway Interface) y ADO (Active X Data Objects).

La función que tienen las API's, es la de ser una interfaz entre las aplicaciones y las bases de datos, llevando ésta tarea unas veces a través de los clientes y otros a través del servidor de base de datos. Esto quiere decir, que puede darse el caso de que el cliente conste de las tres primeras interfaces o niveles, o que se encuentren las dos últimas en el servidor. La interfaz correspondiente a la base de datos, es donde se encontrará el servidor y toda la información depositada en él.

Las tecnologías que se emplea para la conectividad entre los datos y la aplicación, se ha convertido en un factor muy importante a la hora de desarrollar un proyecto web que cuente con funcionalidad de acceso a datos. A continuación en la Tabla 3.2.2 se muestra una comparación de las dos tecnologías más importantes en este sentido: ActiveX Data Objects (ADO) y Java Data Base Connectivity (JDBC). [Lamarca, (2013)]

**Tabla 3.2.2. Comparativa entre ADO y JDBC**

| ADO | JDBC |
|---|---|
| Tecnología elaborada por Microsoft | Tecnología desarrollada por Sun Microsystems |
| Su función principal es realizar a solicitud de los datos a la base de datos | Su función principal se enfoca en ser un gestor para la aplicación con respecto a la BD |
| Realizara esta solicitud mediante la tecnología OLE DB, misma que estará en contacto de manera directa con la BD | Su primera implementación necesito del uso de ODBC como intermediario con la BD |
| OLE DB solo será empleado cuando el DBMS sea propietario de Microsoft, como es SQL Server | En el modelo cliente – servidor, se encontrara trabajando en el cliente accediendo directamente a la BD |
| ADO encapsulara objetos de OLE DB para realizar la conexión con la BD | En el modelo de tres capas, se encontrara en una capa intermedia, la cual cruzaran todos los usuarios para acceder a la BD |
| ADO utilizara objetos de la tecnología Remote Data Objects (RDO) para gestionar el acceso a bases de datos heterogéneas | Existen módulos de JDBC propios de los fabricantes de DBMS |
| RDO dependerá de los ODBC's para efectuar la conexión a la BD | No se encuentra ligado a ninguna tecnología especifica debido a su desarrollo con la finalidad de ser portable |
| Mediante mecanismos de introducción de instrucciones, ADO podrá trabajar en conjunto con código HTML | Mediante JavaScript, JDBC colabora junto con HTML |
| Los objetos que conformar ADO solo son compatibles con Lenguajes de Programación de Microsoft | Se desarrolló con la finalidad de ser portable e independiente de un fabricante de software |



Por último, es necesario destacar una tecnología llamada Web DB utilizada por algunos servidores de bases de datos, con la cual, un usuario puede solicitar la información que requiera y visualizarla a modo de respuesta en una página Web, que será creada y elaborada por el propio servidor de base de datos.

El proceso que comprende desde la solicitud a la visualización de la información, puede ser representado como se muestra en la Figura 3.2.3: [Sánchez, (2013)]

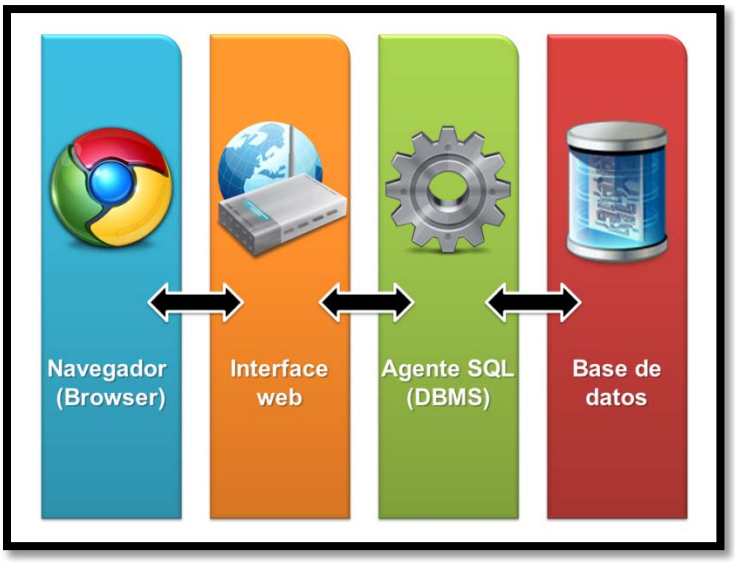



**Figura 3.2.3. Proceso de solicitud de información**

El esquema anterior comprende el navegador que es la aplicación mediante la cual, se tiene acceso libre a los servicios de Internet, y el medio que permite al usuario introducir la solicitud para visualizar la información; la interfaz Web que genera como salida el código HTML, en lugar de leer simplemente un archivo estático de texto.

Con ésta interfaz se podrán crear las páginas Web de forma dinámica y/o utilizar la implementación de formularios HTML. Esta interfaz permite tecnologías como los CGI's o aquellas otras que son propias del servidor de base de datos; el Agente PL/SQL que es el eslabón final del proceso entre un navegador cliente y el servidor de base de datos. El agente ejecutará una llamada a un procedimiento almacenado en el servidor.



Este procedimiento creará una página HTML dinámica como salida, y el agente devolverá dicha salida al cliente a través del navegador empleando de igual manera la Interfaz de Web; y la Base de Datos (BD), en ella se mantendrá almacenada la información; se encarga de proporcionar los datos que hayan sido solicitados previamente, al momento de la ejecución de un procedimiento por parte del Agente PL/SQL.

Desde el punto de vista de la organización lógica existen diferentes tipos de Bases de Datos como son: Jerárquicas (Progress), Relacionales (Oracle, Access, Sybase), desde el punto de vista de número de usuarios: Monousuario (dBase, Access, Paradox), Multiusuario cliente/servidor (Oracle, Sybase)
Una base de datos permite la introducción de datos por parte de los usuarios (o programadores), salida de datos, almacenamiento de datos, protección de datos (seguridad), y elaboración de datos.

Básicamente, la comunicación del usuario-programador con la base de datos se hace a través de un lenguaje denominado SQL: Structured Query Language (Lenguaje estructurado de consultas).

### 3.2.1 Diseño de bases de datos

El primer paso para diseñar una base de datos, es decidir los aspectos importantes del sistema e incluirlos en el modelo. El modelo de datos es una cuestión fundamental a la hora de diseñar bases de datos: Los modelos fundamentales se explican en la Tabla 3.6.

Los objetivos del modelo de datos son, por un lado formalizar y definir las estructuras permitidas para representar los datos, y por otro, diseñar la base de datos.

En el diseño de una base de datos, hay que tener en cuenta distintas consideraciones, entre las que destacan:

- Velocidad de acceso
- Tamaño de la información
- Tipo de información
- Facilidad de acceso a la información
- Facilidad para extraer la información requerida
- Comportamiento del sistema de gestión de bases de datos con cada tipo de información.



En el diseño de una base de datos, el tamaño de la misma es una cuestión fundamental, puesto que éste afecta tanto a la eficiencia en el almacenamiento, como a la agilidad en la búsqueda y recuperación.

Como los datos pueden estar en cualquier morfología (texto, imagen, audio, etc.), en algunos casos se deberán utilizar técnicas de compresión de datos con el fin de disminuir el espacio y tamaño de la base, pero estas técnicas de compresión deberán ir acompañadas de las correspondientes técnicas de indización que hagan posible la recuperación de dichos datos.

**Tabla 3.2.1.1. Modelos de Bases de Datos**

| Modelo | Características |
|---|---|
| Modelos Lógicos basados en objetos | • Los dos más extendidos son el modelo entidad-relación y el orientado a objetos.<br>• El modelo entidad – relación se basa en una percepción del mundo compuesta por objetos, llamados entidades y relaciones entre ellos.<br>• El orientado a objetos también se basa en valores y métodos, entendidos como ordenes que actúan sobre los valores en niveles de anidamiento. |
| Modelos Lógicos basados en registros | • El más extendido es el relacional, mientras que los otros dos existentes: jerárquico y en red, se encuentran en retroceso.<br>• El modelo relacional representa los datos y sus relaciones mediante tablas bidimensionales, que contienen datos tomados de los dominios correspondientes. |
| Modelos físicos de datos | • Muy poco utilizados, se dividen en modelo unificador y modelo de memoria de elementos |

Una Base de Datos depende de atributos específicos que permiten el funcionamiento de la misma, dichos atributos son necesarios para que la información que se desea incluir en la base de datos, tenga una relación e interactué entre sí. Estos atributos son: las llaves primarias que se usan para incorporar la idea de un identificador único en las tablas puede haber más de una llave primaria dependiendo del diseño; las llaves Foránea, que son columnas o un grupo de columnas en una tabla que corresponden o se refieren a una llave primaria o a otra tabla en la base de datos. [Campos Paré, R. (2005)]



Una llave foránea no tiene que ser única, sino que debe identificar únicamente la columna en la tabla con la que se va a relacionar; los índices, los cuales pueden resultar un ahorro de tiempo. Un índice en una tabla ayuda a crear catálogos en tablas donde el contenido habitualmente no cambia, esto con el fin de referenciar los datos; Integridad de datos.

Las bases de datos pueden experimentar distintas clases de integridad y un número de problemas que puedan afectar integridad. Existen tres tipos de integridad: de entidad, de dominio, y de referencia. La integridad de entidad de cada tabla en una base de datos corresponde a una entidad en el mundo verdadero. Que la entidad puede ser física o conceptual, pero en un cierto sentido, la existencia de la entidad es independiente de la base de datos.

Una tabla tiene integridad de la entidad si la tabla es enteramente constante con la entidad que modela.

Para tener integridad de la entidad, una tabla debe tener una llave primaria. La llave primaria identifica únicamente cada fila en la tabla. Sin una llave primaria, no existe seguridad de que la fila recuperada sea la que se necesita. En la Integridad del dominio no es posible garantizar que un registro de datos particular en una base de datos está correcto, pero es posible determinar si un registro de datos es válido. [Campos, (2005)]

Muchos registros de datos tienen un número limitado de valores posibles. En la Integridad de referencia si cada tabla en el sistema tiene integridad de la entidad e integridad del dominio, puede surgir un problema debido a inconsistencias. En la mayoría de las bases de datos bien diseñadas, cada tabla contiene por lo menos una columna que refiera a una columna en otra tabla en la base de datos. Estas referencias son importantes para mantener la integridad total de la base de datos.

Para el diseño de la base de datos es necesario tomar en cuenta la Normalización de la misma. La Normalización es un proceso que clasifica relaciones, objetos, formas de relación y elementos en grupos, en base a las características que cada uno posee. Si se identifican ciertas reglas. Estas reglas, se denominan Niveles de Normalización.

### 3.2.2 Normalización de Bases de Datos

El proceso de normalización de bases de datos consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad – relación al modelo relacional.



Las bases de datos relacionales deben de ser normalizadas con la finalidad de:

- Evitar la redundancia de datos
- Evitar problemas de actualización de los datos en las tablas
- Proteger la integridad de los datos

El modelo relacional fue desarrollado por Edgar Frank Codd para IBM a finales de los años 60, buscando, mantener la independencia de la estructura lógica de la base de datos respecto al modo de almacenamiento y otras características físicas.

El modelo relacional persigue, al igual que la mayoría de los modelos de datos los siguientes objetivos:

1. Independencia física de los datos: El modelo de almacenamiento de los datos no debe influir en su manipulación lógica.
2. Independencia lógica de los datos: Los cambios que se realicen en los objetos de la base de datos sin alterar los datos almacenados previamente, no deben repercutir en los programas y los usuarios que acceden a la misma.

3. Flexibilidad: Para representar a los usuarios los datos de la forma más adecuada a la aplicación que utilicen.

4. Uniformidad: En la presentación de las estructuras lógicas de los datos, que son tablas, lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.

5. Sencillez: Pues las características anteriores, así como unos lenguajes de usuario sencillos, hacen que este modelo sea sencillo de comprender y utilizar por el usuario.

Codd introdujo el concepto de relación o tabla como estructura básica del modelo. Todos los datos de una BD se representan en forma de relaciones cuyo contenido varia con el tiempo. El modelo relacional se basa en dos ramas de las matemáticas: la teoría de conjuntos y la lógica de predicados. [Sánchez, (2004)]

La tabla 3.2.2.1 muestra un ejemplo de una relación con cuatro campos y dos tuplas

**Tabla 3.2.2.1. Relación con cuatro campos y dos tuplas**

| Numero_Empleado | Nombre_Completo | Numero_Depto | Fecha_Alta |
|---|---|---|---|
| 165483 | Daniel Cruz | 32 | 15/04/2000 |
| 197543 | Carolina Ruiz | 4 | 31/12/2013 |



Durante los años 80, múltiples DBMS que se anunciaban como relacionales surgieron en el mercado, sin embargo estos sistemas carecían de muchas características que se consideran importantes en un sistema relacional, perdiendo muchas ventajas de este sistema.

Debido a esto, en 1984, Codd publico 12 reglas que un verdadero sistema relacional deberá cumplir, en la práctica, algunas de estas reglas son bastante difíciles de realizar. Sin embargo mientras más reglas sean aplicadas al sistema, este se acercara más al concepto y beneficios del concepto relacional.

A continuación se describen estas reglas y sus enunciados. [Codd, (1970)]

*Regla 0* Para que un sistema, se denomine sistema de gestión de bases de datos relacionales, este sistema debe usar exclusivamente sus capacidades relacionales para gestionar la base de datos.

*Regla 1 Regla de la información:* Toda la información en una base de datos relacional se representa explícitamente en el nivel lógico exactamente de una manera: con valores en tablas.

*Regla 2 Regla del acceso garantizado:* Para todos y cada uno de los datos de una BDR se garantiza que son accesibles a nivel lógico utilizando una combinación de nombre de tabla, valor de clave primaria y nombre de columna.

*Regla 3 Tratamiento sistemático de valores nulos:* Los valores nulos se soportan en las DBMS totalmente relacionales para representar información desconocida o no aplicable de manera sistemática, independientemente del tipo de datos.

*Regla 4 Catálogo dinámico en línea basado en el modelo relacional:* La descripción de la base de datos se representa a nivel lógico de la misma manera que los datos normales, de modo que los usuarios autorizados pueden aplicar el mismo lenguaje relacional a su consulta, así como a los datos normales.

*Regla 5 Regla del sub lenguaje de datos completo:* Un sistema relacional debe soportar varios lenguajes y varios modos de uso de terminal. Sin embargo, debe existir al menos un lenguaje cuyas sentencias sean expresables, mediante una sintaxis bien definida, como cadenas de caracteres y que sea completo.

*Regla 6 Regla de actualización de vistas:* Todas las vistas que son teóricamente actualizables se pueden actualizar por el sistema.

*Regla 7 Inserción, actualización y borrado de alto nivel:* La capacidad de manejar una relación base o derivada como un solo operando se aplica no sólo a la recuperación de los datos, sino también a la inserción, actualización y borrado de datos.



*Regla 8 Independencia física de datos:* Los programas de aplicación y actividades del terminal permanecen inalterados a nivel lógico aunque se realicen cambios en las representaciones de almacenamiento o métodos de acceso.

*Regla 9 Independencia lógica de datos:* Los programas de aplicación y actividades del terminal permanecen inalterados a nivel lógico aunque se realicen cambios a las tablas base que almacenan la información.

*Regla 10 Independencia de integridad:* Las restricciones de integridad específicas para una determinada base de datos relacional deben poder ser definidas en el sub lenguaje de datos relacional, y almacenables en el catálogo, no en los programas de aplicación.

*Regla 11 Independencia de distribución:* Una BDR tiene independencia de distribución, como por ejemplo las mismas órdenes y programas se ejecutan igual en una BD centralizada que en una distribuida. Las BDR son fácilmente distribuibles, sin embargo se complica más la gestión interna de la integridad.

*Regla 12 Regla de la no subversión:* Si un sistema relacional tiene un lenguaje de bajo nivel que no puede ser usado para saltarse las reglas de integridad y las restricciones expresadas en los lenguajes relacionales de más alto nivel.

Como bien lo menciona Codd, implementar las 12 reglas además de ser lo ideal, es bastante complicado, sin embargo el simple hecho de implementar la mayor cantidad posible, acercara la base de datos al concepto relacional, mejorando su integridad y confiabilidad.

### 3.2.3 Lenguaje Estructurado de Consulta

Las aplicaciones en red son cada día más numerosas y versátiles. En muchos casos, el esquema básico de operación es una serie de scripts que rigen el comportamiento de una base de datos.

Debido a la diversidad de lenguajes y de bases de datos existentes, la manera de comunicar entre unos y otras sería realmente complicada a gestionar de no ser por la existencia de estándares que permiten el realizar operaciones básicas de una forma universal.

Es de eso de lo que trata el Structured Query Language (SQL) que no es más que un lenguaje estándar de comunicación con bases de datos, que permite la comunicación con el sistema gestor de base de datos. Se habla por tanto de un lenguaje normalizado que permite trabajar con cualquier tipo de lenguaje (ASP o PHP) en combinación con cualquier tipo de base de datos (MS Access, SQL Server, MySQL). Diseñado por IBM desarrollado en los años 70, alrededor del 1979. Oracle corp. Presentó la primera implementación comercial de SQL. [Silberschatz, (2004)]



Entre las principales características de SQL se puede destacar que es un lenguaje para todo tipo de usuarios: administradores, desarrolladores y usuarios finales. El usuario que emplea SQL especifica que quiere, no donde ni como, y le permite hacer cualquier consulta de datos.

El hecho de que sea estándar no quiere decir que sea idéntico para cada base de datos. En efecto, determinadas bases de datos implementan funciones específicas que no tienen necesariamente que funcionar en otras.

Aparte de esta universalidad, el SQL posee otras dos características muy apreciadas. Por una parte, presenta una potencia y versatilidad notables que contrasta, por otra, con su accesibilidad de aprendizaje.

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

Existen dos tipos de comandos SQL:

- *Sentencias DDL* (*Data Description Language*). Se trata del lenguaje con el que se crea y mantiene la estructura de la base de datos. Y sirve para realizar las tareas como: crear un objeto de la base de datos (tablas, vistas, procedimientos). Eliminar un objeto de la base de datos. Modificar un objeto e la base de datos. Conceder privilegios sobre un objeto de la base de datos. Retirar privilegios sobre un objeto de la base de datos. [Campos, (2005)]

- *Sentencias DML* (*Data Manipulation Language*). Este lenguaje está formado por un conjunto de sentencias que sirven para manipular los datos contenidos en la base de datos y permite realizar la siguientes operaciones: Insert: insertar filas de datos en una tabla. Update: actualizar filas de datos de una tabla. Delete: eliminar filas de datos de una tabla. Select: recuperar filas de datos de una tabla. [Campos, (2005)]

Para definir los datos que se desean seleccionar o manipular, se utilizan las cláusulas que son condiciones de modificación.

### 3.2.4 PL/PgSQL



Como se mencionó anteriormente, cada desarrollador de bases de datos crea implementaciones personalizadas del estándar SQL, agregándole funcionalidades únicas que brindan originalidad a sus aplicaciones, de esta manera surge PL/PgSQL (*Procedural Language/PostgreSQL Structured Query Language*, Lenguaje Procedural/Lenguaje Estructurado de Consultas de PostgreSQL), el lenguaje SQL del DBMS PostgreSQL. [Obe, (2012)]

Se trata de un lenguaje procedural, que se encuentra entre las funcionalidades de PostgreSQL, y puede ser cargado o activado en el sistema de base de datos. Fue diseñado con el objetivo de crear un lenguaje que pueda ser utilizado para crear funciones y triggers (disparadores), agregando estructuras de control al lenguaje, además de realizar cálculos complejos y heredando todos los tipos de datos definidos por el usuario, así como funciones y operadores.

Entre sus principales ventajas se encuentra la capacidad de ejecutar programas en el servidor de base de datos, teniendo como resultado la posibilidad de que las consultas y el resultado de las mismas, no necesitan ser transportadas entre el cliente y el servidor, ya que los datos residen en el propio servidor. Además, es posible planificar optimizaciones en la ejecución de la búsqueda y actualización de datos. [Obe, (2012)]

El gestor de llamadas PL/PgSQL analiza el texto de las funciones y produce un árbol de instrucciones binarias interno la primera vez que la función es invocada por una aplicación.

El bytecode producido es identificado por el manejador de llamadas mediante el ID de la función. Esto asegura que el cambio de una función por parte de una secuencia DROP/CREATE tendrá efecto sin tener que establecer una nueva conexión con la base de datos.

Para todas y las expresiones y sentencias SQL usadas en la función, el intérprete de bytecode de PL/PgSQL crea un plan de ejecución preparado usando los gestores de SPI, funciones SPI_prepare() y SPI_saveplan(). Esto se hace la primera vez que las sentencias individuales se procesan en la función PL/PgSQL. Así, una función con código condicional que contenga varias sentencias que puedan ser ejecutadas, solo preparará y almacenará las opciones que realmente se usarán durante el ámbito de la conexión con la base de datos.

Excepto en el caso de funciones de conversión de entrada/salida y de cálculo para tipos definidos, cualquier cosa que pueda definirse en funciones de lenguaje C puede ser hecho con PL/PgSQL. Es posible crear funciones complejas de cálculo y después usarlas para definir operadores o usarlas en índices funcionales.

Debido a que SQL es el lenguaje de la mayoría de las bases de datos relacionales, es portátil y fácil de aprender, sin embargo cada sentencia SQL debe ser ejecutada de forma individual por el servidor de bases de datos. Eso significa que la aplicación cliente debe enviar cada consulta al servidor de bases de datos, esperar a que sea procesada, recibir y procesar los resultados, hacer algunos cálculos y a continuación enviar nuevas consultas al servidor.



Todo esto afecta la comunicación entre procesos y también se reflejara en la sobrecarga de la red, siempre que el cliente se encuentre en un equipo diferente que el servidor de bases de datos, situación que en una aplicación web ocurre todo el tiempo.

Con PL/PgSQL, es posible agrupar un bloque de cálculos y una serie de consultas en el servidor de bases de datos, teniendo las ventajas de un lenguaje de procedimientos y la facilidad de SQL pero con un considerable ahorro de comunicación entre cliente y servidor, aumentando drásticamente el rendimiento de la aplicación en comparación con otros lenguajes SQL. [Obe, (2012)]

Las funciones escritas en este lenguaje aceptan argumentos y pueden devolver valores de tipo básico o tipo complejo como pueden ser: registros, vectores, conjuntos, o incluso tablas. Permitiendo de esta manera la tipificación polimórfica para funciones abstractas o genéricas.

El lenguaje PL/PgSQL no es sensible a las mayúsculas, todas las palabras claves e identificadores pueden utilizarse en cualquier mezcla de mayúsculas y minúsculas, sin embargo por cuestiones de buenas prácticas de programación, es recomendable escribirlos en mayúsculas para evitar confusiones y clarificar el código.

Como es un lenguaje orientado a bloques, un bloque se define como: [Group, P. G. (2013)]

```
[<<label>>]
[DECLARE
     Declaración de variables]
BEGIN
     Enunciados de la consulta
END;
```

Por lo cual es posible que exista cualquier número de sub bloques en la sección de sentencia de un bloque. Los sub bloques pueden ser utilizados para ocultar variables a otros bloques de sentencias. Las variables declaradas en la sección de declaraciones se inicializan a su valor por defecto cada vez que se inicia el bloque y no cada vez que se realiza la llamada a la función.

Es importante considerar que las funciones y triggers no pueden iniciar o realizar transacciones y que PostgreSQL no soporta transacciones anidadas. [Obe, (2012)]

### 3.2.5 PostgreSQL

PostgreSQL es un sistema gestor de bases de datos relacionales de código abierto que se originó como un proyecto de la Universidad de Berkeley en California. Originalmente fue desarrollado bajo la licencia BSD, sin embargo con el tiempo ha migrado a un esquema de licencia propia llamado PostgreSQL (TPL). [Obe, (2012)]



El concepto actual de PostgreSQL, comenzó en 1996, sin embargo las bases de este sistema remontan sus orígenes hasta la década de los 70, es en esta época cuando Michael Stonebreaker, después de analizar algunos artículos publicados sobre el proyecto "*System R*" de IBM, decide utilizar los fondos destinados al proyecto Ingres, para desarrollar sus ideas sobre bases de datos relacionales.

A principios de los 80, Ingres comienza su competencia contra Oracle por el liderazgo en el mundo de las bases de datos, esta competencia tuvo como consecuencia, la aparición de otros sistemas de bases de datos relacionales como son Informix, NonStop SQL, Sybase, misma que da origen a Microsoft SQL Server.

Años más tarde, en 1985, Stonebreaker se convirtió en el líder de un nuevo proyecto, denominado Postgres, que significa "*Después de Ingres*", patrocinado por la DARPA (*Defense Advanced Research Projects Agency*), ARO (*Army Research Office*), NSF (*National Science Foundation*) y ESL Inc.

Este proyecto se basó en muchas de las ideas de Ingres, sin embargo, no utilizo el mismo código fuente, lo cual lo convierte en un proyecto totalmente nuevo, dependiente únicamente de conceptos e ideas. Finalmente, la última versión de Postgres en este proyecto fue la 4.2.

Durante 1994, Andrew Yu y Jolly Chen, estudiantes de Berkeley comenzaron a trabajar con el código fuente de Postgres 4.2, limpiando el código, corrigiendo errores e implementando mejoras, creando de esta manera la nueva versión del sistema, Postgres95, debido a que vio la luz pública hasta ese año. Ya que el código fuente fue publicado y liberado bajo la licencia BSD cada vez más personas comenzaron a colaborar en el desarrollo del proyecto, situación similar a lo ocurrido con Linux.

En 1996, usuarios habituales de las listas de correo del proyecto decidieron hacerse cargo del mismo y crearon el llamado "*PostgreSQL Global Development Team*". El nombre del proyecto fue cambiado de Postgres95 a PostgreSQL y lanzaron la versión 6.0 en enero de 1997. Existen empresas que colaboran con dinero y/ó con tiempo/personas en mejorar PostgreSQL. Muchos desarrolladores y nuevas características están muchas veces patrocinadas por empresas privadas.

El Proyecto PostgreSQL tiene como objetivo mantener y soportar cada versión de PostgreSQL durante 5 años desde el momento de su lanzamiento.

La Tabla 3.2.5.1 muestra el resumen del ciclo de vida de las diferentes versiones de PostgreSQL. [Group, T. P. (2013)]

**Tabla 3.2.5.1. Ciclo de Vida de las versiones de PostgreSQL**

| Versión | Versión menor | Soportada | Lanzamiento | Soporte |
|---|---|---|---|---|
| **9.2** | 9.2.0 | Si | Sep 2012 | Sep 2017 |
| **9.1** | 9.1.5 | Si | Sep 2011 | Sep 2016 |
| **9.0** | 9.0.9 | Si | Sep 2010 | Sep 2015 |
| **8.4** | 8.4.13 | Si | Jul 2009 | Jul 2014 |
| **8.3** | 8.3.20 | Si | Feb 2008 | Feb 2013 |
| **8.2** | 8.2.23 | No | Dic 2006 | Dic 2011 |
| **8.1** | 8.1.23 | No | Nov 2005 | Nov 2010 |
| **8.0** | 8.0.26 | No | Ene 2005 | Oct 2010 |
| **7.4** | 7.4.30 | No | Nov 2003 | Oct 2010 |
| **7.3** | 7.3.21 | No | Nov 2002 | Nov 2007 |
| **7.2** | 7.2.8 | No | Feb 2002 | Feb 2007 |
| **7.1** | 7.1.3 | No | Abr 2001 | Abr 2006 |
| **7.0** | 7.0.3 | No | May 2000 | May 2005 |
| **6.5** | 6.5.3 | No | Jun 1999 | Jun 2004 |
| **6.4** | 6.4.2 | No | Oct 1998 | Oct 2003 |
| **6.3** | 6.3.2 | No | Mar 1998 | Mar 2003 |



PostgreSQL permite escribir procedimientos almacenados y funciones en varios lenguajes de programación, como pueden ser: PL/PgSQL, PL/Perl, PL/Phyton, PL/Java

y PL/R, este soporte permite resolver problemas desde diferentes perspectivas, creando una combinación bastante más poderosa que la ofrecida por cualquier lenguaje SQL simple. La Figura 3.2.5.1 muestra los componentes fundamentales del sistema PostgreSQL. [Group, T. P. (2013)]

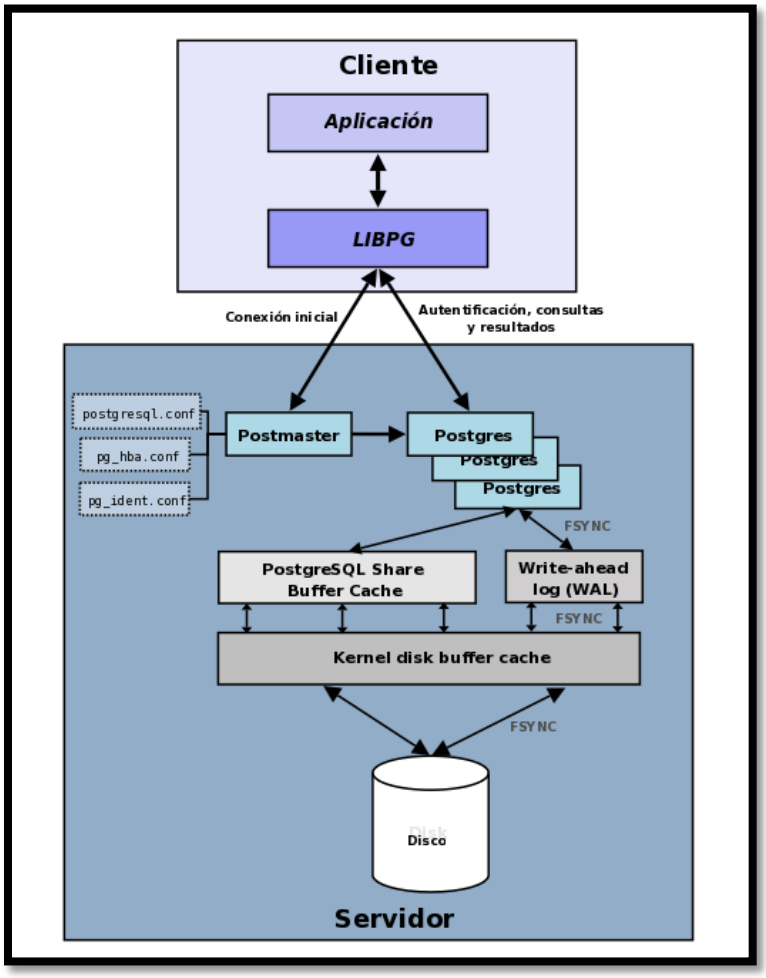



**Figura 3.2.5.1. Componentes fundamentales de PostgreSQL**

La última serie de producción es la 9.2. Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

Adicionalmente a PostgreSQL, existen herramientas que permiten la interacción entre los usuarios y la base de datos, estas aplicaciones son llamadas DBMS y PostgreSQL cuenta con al menos dos aplicaciones destacables, PgAdmin III y phpPgAdmin.

### 3.2.6 PgAdmin III

Un Sistema de Gestión de Bases de Datos (DBMS) es 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. Tiene como objetivo fundamental suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, de forma que no le sea necesario conocer el modo de almacenamiento de los datos en la computadora, ni el método de acceso empleado.

Los programas de aplicación operan sobre los datos almacenados en la base utilizando las facilidades que brindan los DBMS, los que, en la mayoría de los casos, poseen lenguajes especiales de manipulación de la información que facilitan el trabajo de los usuarios.

Los DBMS brindan facilidad a la hora de elaborar tablas y establecer relaciones entre las informaciones contenidas en ellas. Pueden mantener la integridad de una base de datos permitiéndole a más de un usuario actualizar un registro al mismo tiempo y también puede impedir registros duplicados en una BD.

Algunas características de los DBMS son:



- Permiten crear y gestionar base de datos de forma fácil, cómoda y rápida.
- Ofrecen una gran flexibilidad para el trabajo con base de datos relacionales.
- Ofrecen un ambiente agradable dado por su interfaz gráfica.

PgAdmin III, es una aplicación de diseño y manejo de bases de datos para su uso con PostgreSQL. La aplicación se puede utilizar para manejar a partir de PostgreSQL 7.3 y superiores y funciona sobre casi todos los sistemas operativos. [pgAdmin. (2013)]

Fue diseñado para responder a las necesidades de todos los usuarios, desde la escritura de simples consultas SQL a la elaboración de bases de datos complejas. La interfaz gráfica es compatible con todas las características de PostgreSQL y facilita la administración. La aplicación también incluye un editor de la sintaxis SQL, un editor de código del lado del servidor, un agente para la programación de tareas SQL/batch/shell, soporte para el motor de replicación Slony-I y mucho más.

La conexión del servidor se puede realizar mediante TCP/IP o Unix Domain Sockets (en plataformas Unix/Linux), y puede ser cifrado mediante SSL por seguridad. No se requieren controladores adicionales para comunicarse con la base de datos del servidor.

PgAdmin II cuenta con las siguientes características: [Ecured. (2013)] [Obe, R., y Hsu, L. (2012)]

- Explicación grafica de las consultas
- Panel de interacción con la base de datos mediante SQL
- Edición y configuración directa de archivos de sistema como *postgresql.conf* y *pg_hba.conf*
- Exportación de datos en formatos como CSV y HTML
- Ayudante para la restauración de respaldos
- Ayudante de asignación de permisos
- Arquitectura modular que permite instalar y deshabilitar *plugins* rápidamente

Un detalle importante que se debe resaltar es que en cada sistema operativo, PgAdmin es una aplicación nativa, por lo tanto, se ejecuta directamente a nivel binario, no en una máquina virtual, lo que ofrece un excelente desempeño. [Martínez, (2013)]

La Figura 3.2.6.1 muestra la interface de PgAdmin III bajo el sistema operativo Linux.



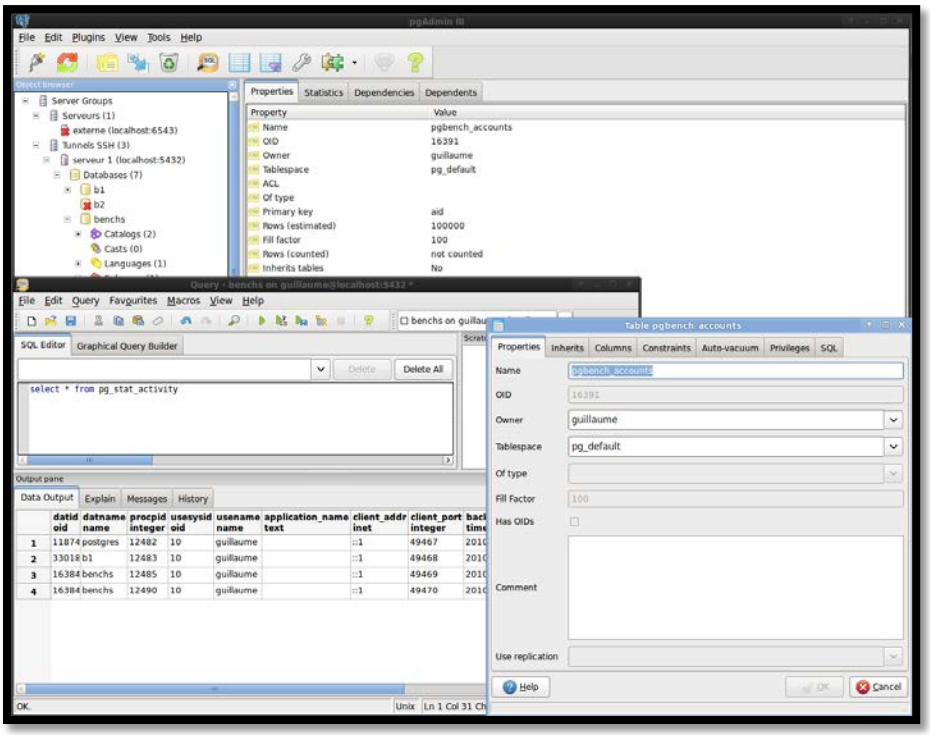



**Figura 3.2.6.1. Interface del DBMS PgAdmin III**

### 3.2.7 phpPgAdmin

A finales del 2002 Christopher Reyes-Lynne dirigió el proyecto phpPgAdmin en la creación de la tercera versión del sistema, esta versión fue reescrita totalmente para apoyar algunas de las características más importantes de PostgreSQL 7.3 como son los esquemas.

phpPgAdmin es una aplicación web, escrita en PHP, para administrar bases de datos PostgreSQL, con capacidades similares a las ofrecidas por PgAdmin III, sin embargo en esta ocasión de trata de una aplicación web, por lo cual, la interacción con el sistema se realizara mediante un navegador.

phpPgAdmin provee una manera conveniente a los usuarios para crear bases de datos, tablas, alterarlas y consultar sus datos usando el lenguaje estándar SQL. Estuvo basado en phpMyAdmin, pero hoy día ya no comparte código con él; incluso provee las mismas funcionalidades y más a los usuarios del servidor de base de datos PostgreSQL. [DataPrix. (2013)]

Características:

- Administración de múltiples servidores
- Soporte para PostgreSQL 8.4, 9.0, 9.1, 9.2
- Administración de todos los aspectos de: usuarios, grupos, bases de datos, esquemas, tablas, índices, restricciones, *triggers*, reglas y privilegios, tablas, secuencias y funciones, objetos avanzados y reportes.
- Manipulación sencilla de datos: búsqueda en tablas, vistas y reportes, ejecución de SQL arbitrario y comandos como *select insert, update* y *delete*
- Descargar tablas en formatos: SQL, COPY, XML, XHTML y CSV
- Importar datos desde archivos SQL, XML y CSV
- Soporte del motor de replicación maestro – esclavo *Slony*
- Excelente soporte de idiomas: disponible en 27 idiomas, sin conflictos de codificación lo que permite incluso editar datos en ruso utilizando una interface en japonés.

## 3.3 Ingeniería Web

Así como los seres vivos, las aplicaciones web evolucionan. En muchas aplicaciones web, no es posible especificar totalmente al inicio de su desarrollo sus requisitos o el contenido de estos sistemas, debido a que su estructura y funcionalidad deberá cambiar constantemente con el tiempo. Además, la información contenida y presentada en un sitio web con frecuencia cambia, en algunas aplicaciones este cambio se realiza cada cierta cantidad de tiempo o bien un par de veces al día. En consecuencia la habilidad de mantener la información actualizada y escalar la estructura del sitio web y las funciones que provee, es un punto importante a considerar en el desarrollo de aplicaciones web.

Con el pasar de los años, la evolución de las aplicaciones web ha pasado de utilizar recursos estáticos al uso de entornos de aplicación dinámicos controlados por el usuario, esta evolución ha orillado a los creadores de sitios web a aplicar procedimientos de gestión solidos además de principios de ingeniería, con la finalidad de desarrollar una aplicación web de calidad.

Alcanzar esta meta es posible, mediante el desarrollo de un marco de trabajo de ingeniería web que se encuentre acompañado de un modelo de proceso eficiente, definido por las actividades del marco de trabajo mismo así como las tareas de ingeniería. Este modelo de proceso engloba dentro de si las siguientes fases: formulación, planificación, análisis, ingeniería, diseño del contenido, producción, generación de página y pruebas, tal como se muestra en la Figura 3.3.1: [Nolasco, (2013)]



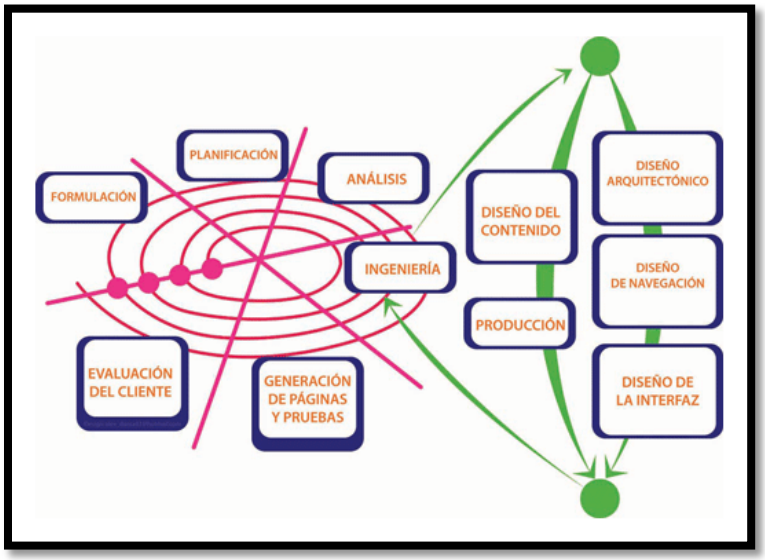



**Figura 3.3.1. Modelo de proceso de Ingeniería Web**

Dado este entorno web, se aprecia que el único enfoque viable para el desarrollo sustentable de una aplicación web se basa en un modelo de desarrollo evolutivo, en el cual los cambios son vistos como pautas, obligando a adoptar un proceso disciplinado para un desarrollo web exitoso.

Un proceso de desarrollo web delimita los múltiples pasos y actividades en el desarrollo de sistemas basados en web. Este deberá definir claramente un conjunto de pasos que el desarrollador pueda seguir y sea cuantificable. [M. Brandon, (2008)]

Las características de una aplicación web que dificultan su desarrollo incluyen la interacción en tiempo real, complejidad y el deseo de proveer información personalizada. Adicionalmente, es difícil estimar con exactitud el esfuerzo y tiempo requerido para diseñar y desarrollar una aplicación web.

Debido a que se trata de un modelo de desarrollo evolutivo, en el cual, se comienza con el desarrollo de cada una de sus fases, una vez que la anterior ha sido terminada, una vez que todas las fases han sido completadas, se reinicia el proceso. [Pressman, (2002)]

Dichas fases son descritas a continuación:

- *Formulación*: en esta actividad de identifican las metas y los objetivos de la aplicación web y establece el ámbito del primer incremento.



- *Planificación*: estima el coste global del proyecto, evalúa los riesgos asociados con el esfuerzo del desarrollo y define una planificación del desarrollo bien granulada para el incremento final de la aplicación.

- *Análisis*: establece los requisitos técnicos de la aplicación e identifica los elementos del contenido que serán incorporados. En este punto también se deben definir los aspectos estéticos de la aplicación web.

- *Ingeniería*: incorpora dos actividades paralelas, el diseño del contenido y la producción, estas tareas deben ser llevadas a cabo por personas no técnicas del equipo de ingeniería web. El objetivo de estas tareas es diseñar, producir o adquirir todo el contenido de texto, gráficos y video que serán integrados a la aplicación web, al mismo tiempo que se realizan un conjunto de tareas de diseño que serán descritas posteriormente.

- *Generación de páginas*: es una actividad de construcción que hace uso de las herramientas automatizadas para la creación de la aplicación web. En este punto el contenido definido en la fase de ingeniería es fusionado con los diseños arquitectónicos, de navegación y la interface para elaborar páginas web en HTML, XML y otros lenguajes orientados a procesos. Durante esta actividad también se lleva a cabo la integración con el software intermedio de componentes como CORBA, DCOM o Java Beans.

- *Pruebas*: ejercitan la navegación, intentan descubrir los errores de las aplicaciones, guiones, formularios y ayuda a asegurar que la aplicación web funcionara correctamente en diferentes entornos, como pueden ser los diferentes navegadores web.

Cada una de estas fases o incrementos producidos como parte del proceso de ingeniería web, será revisado durante la fase de *evaluación del cliente*, siendo en este punto en donde se solicitan cambios y estos serán integrados en la siguiente iteración del proceso. Abstrayendo el modelo de proceso adoptado, es posible identificar las siguientes etapas: [M. Brandon, (2008)]

1. Entender el entorno, requisitos del cliente y de las partes interesadas.

2. Entender los problemas principales específicos de las partes interesadas que deberán ser direccionados.

3. Desarrollar una arquitectura para el sitio web en su totalidad, así como soluciones a problemas no técnicos.

4. Desarrollar sub procesos o sub proyectos para implementar la arquitectura y las soluciones a problemas no técnicos.

5. Implementar estos sub proyectos.



Si los sub proyectos son demasiado complejos, se deberá aplicar el mismo proceso hasta obtener una lista de tareas manejables. La Figura 3.3.2 muestra el proceso de desarrollo en su totalidad

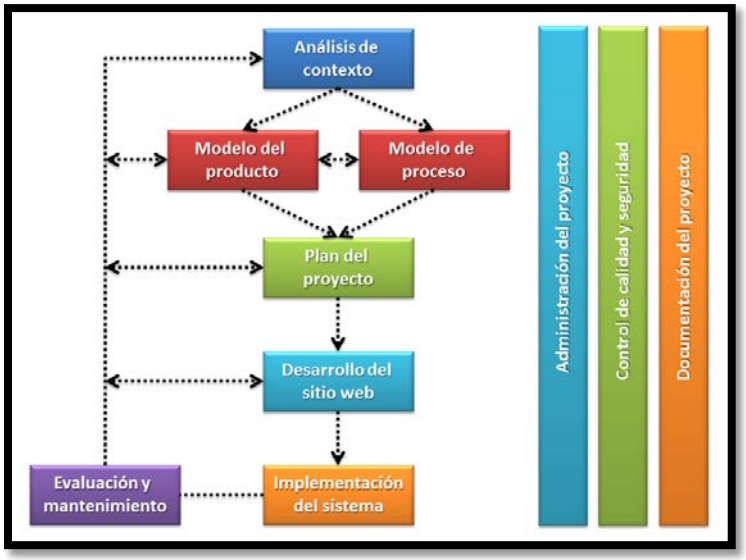



**Figura 3.3.2. Proceso de desarrollo Web**

Los primeros dos pasos que se llevan a cabo para comprender las cuestiones más generales forman parte del análisis de contexto. Este será el primer paso que cualquier proyecto de desarrollo web necesita realizar. Una vez que las cuestiones generales han sido entendidas, es posible iniciar la integración de tecnologías orientadas a la resolución de problemas tecnológicos. El modelo de producto, representa en conjunto la arquitectura del sitio web que será desarrollado.

Una vez finalizado este proceso, es posible definir los sub procesos requeridos para implementar la arquitectura y hacer frente a los problemas no técnicos. Esto es conocido como el modelo de procesos.

Estos sub procesos pueden ser convertidos en un plan de proyecto mediante la asignación de recursos y programando tiempos. Basado en el plan de proyecto, las actividades de desarrollo pueden ser realizadas y al ser finalizadas serán añadidas a la fase de mantenimiento.

Un aspecto importante es que de vez en cuando el entorno, los requisitos corporativos, partes interesadas o incluso la información puede cambiar. Cuando esto sucede, dependiendo de la naturaleza y magnitud del cambio, será necesario llevar a cabo nuevamente las secciones pertinentes de algunos de los sub procesos relevantes.

### 3.3.1 Análisis de contexto



En el desarrollo de un sistema basado en la web, el primer paso es fundamental, este paso es conocido como el análisis de contexto, en el que es necesario obtener y comprender los principales requisitos y objetivos del sistema, así como las necesidades de los usuarios habituales y la organización que requiere el sistema. En esta etapa es sumamente importante considerar que los requisitos cambian y evolucionan, e incluso es posible apreciar estos cambios durante el desarrollo del sistema y/o después de su implementación.

De igual manera es fundamental analizar brevemente el motivo por el cual, la aplicación web será desarrollada y las posibles consecuencias de a integración del nuevo sistema dentro de la organización. Este estudio deberá incluir como será desarrollada y gestionada la información que se encontrara disponible en el sitio web, las políticas organizacionales referentes a la propiedad y control de la información, planes actuales y a futuro, objetivos de negocio, posible impacto en la producción, cambios resultantes en los negocios y procesos de negocio, así como las tendencias emergentes en el sector de la organización para la cual se desarrolla la aplicación web. [M. Brandon, (2008)]

Dado que las aplicación web evolucionan continuamente y deben modificarse para atender a nuevas exigencias, algunas de las cuales surgen de los cambios o mejoras en los procesos de negocio como consecuencia de la implementación de un nuevo sistema, la comprensión de un panorama general sobre la organización, políticas y

prácticas de gestión de la información es un requisito previo para el existo de diseño, desarrollo y despliegue de una aplicación basada en web.

Por lo tanto, antes de iniciar el desarrollo, es necesario obtener y comprender los principales objetivos y requisitos del sistema, recopilar información sobre el entorno operativo e identificar e perfil de los usuarios característicos del sistema.

Además de los requisitos funcionales, solicitudes potenciales referentes a la escalabilidad, mantenimiento, disponibilidad y rendimiento del sistema, durante los primeros pasos del proceso de desarrollo es necesario ser bastante especifico y entendible en la interacción con los desarrolladores ya que basándose en esta información, los desarrolladores podrán determinar las necesidades funcionales, técnicas y no técnicas del sistema, lo cual a su vez influye en el diseño arquitectónico del sistema.

Por ejemplo, si el contenido y las funciones del sistema van a evolucionar considerablemente, como en la mayoría de los sistemas de comercio electrónico, el sistema tiene que ser diseñado orientando hacia la escalabilidad. Por otro lado si la información cambia con frecuencia, como en los informes meteorológicos, las ofertas en las ventas especiales, ofertas de trabajo, listas de precios de productos, folletos, y noticias o anuncios, para mantener la información actualizada y consistente, el sistema deberá ser diseñado orientándolo a facilitar el mantenimiento de la información.



Por otra parte, cuando el sistema exige un alto nivel de disponibilidad y la capacidad de atender picos elevados o inciertos de solicitudes, es posible que el sistema requiera ejecutarse en múltiples servidores web con balanceo de carga y otros mecanismos que mejoren su rendimiento. Ejemplos de esta categoría de aplicaciones son el comercio electrónico de acciones, banca en línea y sitios web de live streaming de deportes y entretenimiento, como podría ser actualmente el sistema Netflix, el cual permite ver películas online en tiempo real a nivel internacional.

Por lo tanto es fundamenta integrar a la arquitectura de sistema, los requisitos de escalabilidad, facilidad de mantenimiento y las necesidades de rendimiento especificas del sistema. Ya que integrarlos al finalizar el desarrollo seria básicamente imposible. Para ilustrar esto, considere un sitio web de comercio electrónico que ofrece información de productos, tales como el precio y la disponibilidad, que aparece en la mayoría de las secciones del sitio web y los cambios se realizan con frecuencias diferentes en base a la popularidad del producto. Si el sitio web se encuentra desarrollado en base a páginas web estáticas, cada vez que sea necesario modificar la información de un producto, habrá que realizar el cambio en todas y cada una las páginas web en las que se encuentra promocionado dicho producto. Debido a que esta tarea será sumamente laboriosa se tendrá como consecuencia que la información que aparece referente al mismo producto sea incongruente. [M. Brandon, (2008)]

Un enfoque óptimo mediante el cual es posible garantizar la coherencia de la información dentro del sitio web, se basa en obtener esta información cuando y donde sea necesario, a partir de una única fuente de información. Si la información se almacena en una sola base de datos central, es posible extraerla y crear dinámicamente varias páginas de sitio web. En este enfoque, solo es necesario actualizar la información dentro de la base de datos, para que este cambio se vea reflejad en su totalidad dentro del sitio web.

Un sitio web con base de datos, requiere una arquitectura totalmente diferente a la desarrollada para un sitio web de contenido estático. Por lo tanto es necesario seleccionar la arquitectura adecuada durante los pasos iniciales del desarrollo. En base a esto, es posible conceptualizar los objetivos del análisis de contexto de la siguiente manera:

- Identificar los grupos de interés, sus necesidades y experiencias.
- Identificar las funciones que el sitio web tiene que ofrecer (inmediatamente, y en el corto, mediano y largo plazo).
- Establecer la información que deberá estar en el sitio Web, la forma de obtener esta y la frecuencia con que esta información puede cambiar.
- Identificar los requerimientos corporativos en relación con la apariencia, rendimiento, seguridad y gobernabilidad.
- Tener una idea de la cantidad de usuarios y sus diferentes frecuencias durante el transcurso del día, así como las demandas de servicio previstas en el sistema.
- Estudio de sitios web similares (competencia) para ganar una comprensión de sus funcionalidades, ventajas y limitaciones.



Adicionalmente el análisis de contexto permite identificar problema son técnicos que es necesario abordar para realizar una implementación exitosa, estos problemas no técnicos pueden incluir la reingeniería de procesos del negocio, políticas organizativas y de gestión, capacitación del personal y aspectos legales, culturales y sociales que pueden influir en el desarrollo del sistema.

El análisis de contexto puede ser dividido en dos sub procesos o fases principales, estos son: formulación y análisis.

La formulación y análisis de sistemas y aplicaciones basados en web representan una sucesión de actividades de ingeniería web que comienza con la identificación de metas globales para la aplicación web y termina con el desarrollo de un modelo de análisis o especificación de los requisitos del sistema.

La formulación permite que el cliente o diseñador establezca un conjunto común de metas y objetivos para la construcción de la aplicación web. También identifica e ámbito de esfuerzo en el desarrollo y proporciona un medio para determinar un resultado satisfactorio. El análisis es una actividad técnica que identifica los datos y requisitos funcionales y de comportamiento de la aplicación web. [Pressman, (2002)]

*Formulación*

Para desarrollar el proceso de formulación, se ha sugerido, realizar una serie de preguntas que deberán formularse y responderse al comienzo de la etapa de formulación. Estas preguntas son:

- ¿Cuál es la motivación principal para la aplicación web?
- ¿Por qué es necesaria la aplicación web?
- ¿Quién va a utilizar la aplicación web?

La respuesta a estas preguntas deberá ser lo más concreta posible e implicar metas específicas para la aplicación web, este tipo de metas se dividen en dos categorías:

- *Metas informativas*: indican la intención de proporcionar el contenido y/o información específica para el usuario final.
- *Metas aplicables*: indican la habilidad de realizar algunas metas dentro de la aplicación web.



Una vez que todas las metas han sido identificadas, es posible desarrollar el perfil del cliente, el cual recoge las características relevantes de los usuarios potenciales incluyendo antecedentes, conocimiento, preferencias, etc. Finalizado esto, la actividad de formulación se centra en la afirmación del ámbito para la aplicación web.

En la mayoría de los casos, las metas ya desarrolladas se integran en la afirmación del ámbito. Además es útil indicar el grado de integración que se espera para la aplicación web. En este punto también es necesario considerar los temas de conectividad e integración con otros sistemas.

*Análisis*

Los conceptos y principios que se trataron para el análisis de requisitos, se aplican sin revisión en la actividad de análisis de ingeniería web. Para crear un modelo de análisis completo para la aplicación web se elabora el ámbito definido durante la actividad de formulación. [Pressman, (2002)]

Durante la ingeniería web se realizan cuatro tipos de análisis diferentes

- *Análisis de contenido*: se trata de la identificación de espectro completo de contenido que se va a proporcionar. En el contenido se incluyen datos de texto,

gráficos, imágenes, video y sonido. Para identificar y describir cada uno de los objetos de datos que se van a utilizar dentro de la aplicación web se puede utilizar el modelado de datos.

- *Análisis de la interacción*: se trata de la descripción detallada de la interacción del usuario y la aplicación web. Para proporcionar descripciones detalladas de esta interacción se pueden desarrollar casos prácticos.

- *Análisis funcional*: los casos de uso creados como parte del análisis de interacción definen las operaciones que se aplicaran en el contenido de la aplicación web e implicaran otras funciones de procesamiento. Aquí se realiza una descripción detallada de todas las funciones y operaciones.

- *Análisis de la configuración*: se efectúa una descripción detallada del entorno y de la infraestructura en donde reside la aplicación web. La aplicación web puede residir en internet, en una intranet o en una extranet. Además, se deberá identificar la infraestructura de los componentes y el grado de utilización de la base de datos para generar contenido.

Aún cuando se recomienda una especificación detallada de los requisitos para aplicaciones web grandes y complejas, tales documentos no son os usuales. Se puede decir que la continua evolución de los requisitos de la aplicación web puede hacer que cualquier documento se quede obsoleto antes de finalizarse. Aunque se puede decir que esto sucede de verdad, es necesario definir un modelo de análisis que pueda funcionar como fundamento de a siguiente actividad de diseño. Como mínimo, la información recogida durante las cuatro tareas de análisis anteriores deberá ser revisada, modificada a petición y organizada para formar un documento que pueda pasarse a los diseñadores de aplicaciones web.



## 3.3.2 Diseño de aplicaciones web

Uno de los problemas principales del desarrollo de aplicaciones web deriva de la presión por evolucionar constantemente la aplicación, obligando al ingeniero encargado del proyecto, a resolver el problema comercial inmediato, al mismo tiempo que debe definir una arquitectura de la aplicación, que cuente con la habilidad de evolucionar rápidamente con el tiempo. Sin embargo resolver este tipo de problemas rápidamente puede dar como resultado compromisos que afectan la capacidad de la aplicación de realizar esta evolución con forme el paso del tiempo. [M. Brandon, (2008)]

En esta etapa de diseño se define:

- Una arquitectura general del sistema que describe cómo interactúan la red y los distintos servidores (servidores web, servidores de aplicaciones y servidores de bases de datos);

- Una arquitectura de aplicación que representan diversos módulos de información y las funciones que apoyan

- Una arquitectura de software identificando diversos módulos de software y bases de datos necesarias para implementar la arquitectura de la aplicación.

En la Tabla 3.3.2.1 se muestran los medios para cumplir algunos de los requisitos de las aplicaciones basadas en la Web. [M. Brandon, (2008)]

**Tabla 3.3.2.1. Medios para cubrir requisitos en una aplicación Web**

| Requisito | Forma de cumplirlo |
|---|---|
| Diseño uniforme en todo el sitio web que pueda ser modificado fácilmente | Creación de páginas Web utilizando plantillas y hojas de estilo |
| Consistencia en la información que pudiera aparecer en diferentes secciones del sitio web | Almacenamiento de información en un solo lugar y la capacidad de recuperarla donde y cuando sea necesario |
| La facilidad de actualización de la información y de mantenimiento | Proveer al sistema de un trasfondo que permita editar la información, el cual podría contar con una interfaz Web de fácil acceso desde cualquier lugar |
| Posibilidad de añadir nuevas páginas fácilmente | Generación dinámica de enlaces de navegación, en lugar de enlaces estáticos |
| Sistema de administración descentralizado | Proveer un sistema de trasfondo de acceso a múltiples usuarios e incluir un sistema de administración de usuarios que defina funciones específicas a cada uno |
| Mecanismos para el control de calidad y la evaluación de la relevancia de la información | Inclusión de metadatos en las páginas web, uso de un robot web capaz de analizar la información saliente y realizar acciones para asegurar la relevancia y calidad de dicha información |
| Incrementar la probabilidad de encontrar el sitio web a través de un buscador | Uso de etiquetas meta y el registro en los motores de búsqueda |



Con el objeto de realizar un diseño eficaz para una aplicación web, se debe de trabajar utilizando cuatro elementos técnicos: [Pressman, (2002)]

- *Principios y métodos de diseño*: mediante la modularidad, elaboración paso a paso del proyecto y cualquier otra heurística de diseño de software, es posible conducir el proyecto a niveles que destacan por su sencillez a la hora de realizar adaptaciones, mejoras, realizar pruebas y utilizar el sistema.

  Durante el proceso de ingeniería web, es posible utilizar los métodos de diseño que se implementan en los sistemas orientados a objetos. La hipermedia define

objetos que interactúan mediante un protocolo de comunicación similar a la mensajería. Adicionalmente la notación de diagramas propuesta por UML puede ser adaptada y utilizada durante las actividades de diseño de las aplicaciones web.

- *Configuraciones de diseño*: se trata de un enfoque genérico utilizado para resolver pequeños problemas, mismas que pueden ser utilizadas para solventar una amplia variedad de problemas específicos. En el contexto de las aplicaciones web, las configuraciones de diseño, es posible aplicarlas no solo a los elementos funcionales de una aplicación, sino también a los documentos, gráficos y estética general del sitio.

- *Plantillas*: las plantillas pueden ser utilizadas para proporcionar un marco de trabajo esquemático de cualquier configuración de diseño o documento a utilizar dentro de la aplicación.

*Diseño arquitectónico*

En los sistemas y aplicaciones web, se basa en la definición de la estructura global hipermedia para la aplicación y en la implementación de las configuraciones de diseño y plantillas constructivas para popularizar la estructura. Adicionalmente existe una actividad paralela, denominada diseño del contenido, que deriva en la estructura y el formato detallado del contenido de la información que se presenta como parte de la aplicación web.



*Estructura de las aplicaciones web*

La estructura arquitectónica global va unida a las metas establecidas para una aplicación web, el contenido que será presentado, los usuarios y la filosofía de navegación que dicta que todo contenido debe de ser alcanzado con la menor cantidad de clics posible. Cuando el encargado de la arquitectura realiza el diseño de una aplicación web, debe de elegir entre cuatro posibles opciones:

- Estructuras lineales: aparecen cuando es común la sucesión predecible de interacciones, como por ejemplo un manual de usuario, en el que las páginas de información se presentan con gráficos relacionados, videos cortos o sonido solo después de haber presentado un requisito previo.

  A medida que el contenido y el procesamiento crecen en complicación, el flujo puramente lineal da como resultado estructuras lineales más complejas en las que es posible invocar contenido alternativo o en tendrá lugar una desviación para adquirir un contenido complementario.

  En la Figura 3.3.2.1 se puede observar un ejemplo de estructura lineal.

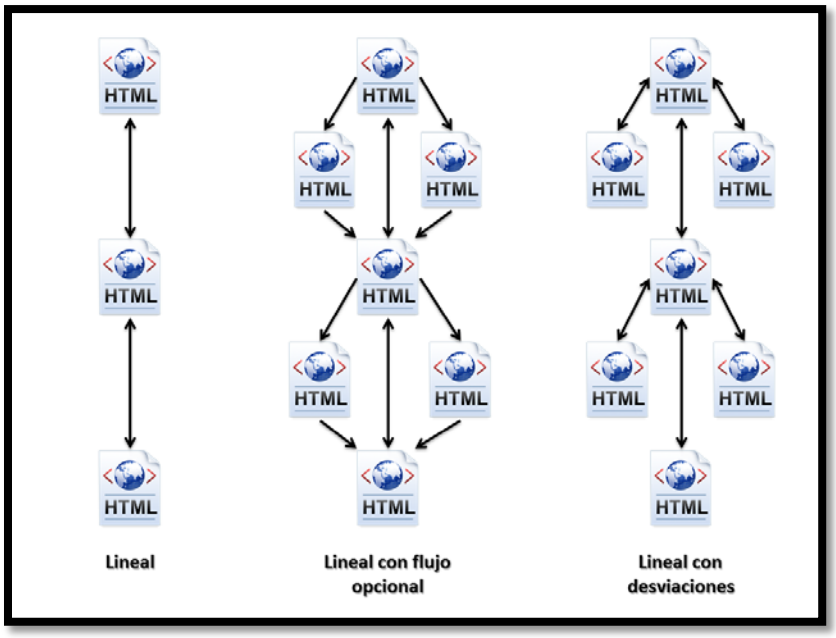



**Figura 3.3.2.1. Estructuras Lineales**

- Estructuras reticulares: se trata de una opción arquitectónica que puede ser utilizada cuando el contenido de la aplicación web puede ser organizado categóricamente en dos o más dimensiones. La dimensión vertical podría representar una oferta, desde ella el usuario podría navegar por la retícula horizontal para encontrar la columna de algún accesorio y nuevamente recorrer la columna de los accesorios para examinar las ofertas que presentan. Esta arquitectura solo es utilizada cuando se encuentra contenido muy regular.

  En la Figura 3.3.2.2 se puede observar un ejemplo de estructura reticular.

- Estructuras jerárquicas: se trata de la estructura más común dentro del desarrollo de aplicación web y posibilita el flujo de control en horizontal atravesando las ramas verticales de la estructura. Por lo tanto el contenido presentado en la rama del extremo izquierdo de la jerarquía puede tener enlaces de hipertexto que lleven al contenido que existe en medio de la rama derecha de la estructura. Sin embargo es necesario destacar que aún que dicha rama permita una navegación rápida por el contenido de la aplicación web, puede originar confusión por parte del usuario.

  En la Figura 3.3.2.3 se puede observar un ejemplo de estructura jerárquica

- Estructura en red: se asemeja en muchos aspectos a la arquitectura evolutiva para los sistemas orientados a objetos. Los componentes arquitectónicos se diseñan de forma que pueden pasar el control a otros componentes del sistema. Este enfoque brinda a la aplicación flexibilidad considerable, sin embargo es posible que resulte confuso para el usuario.

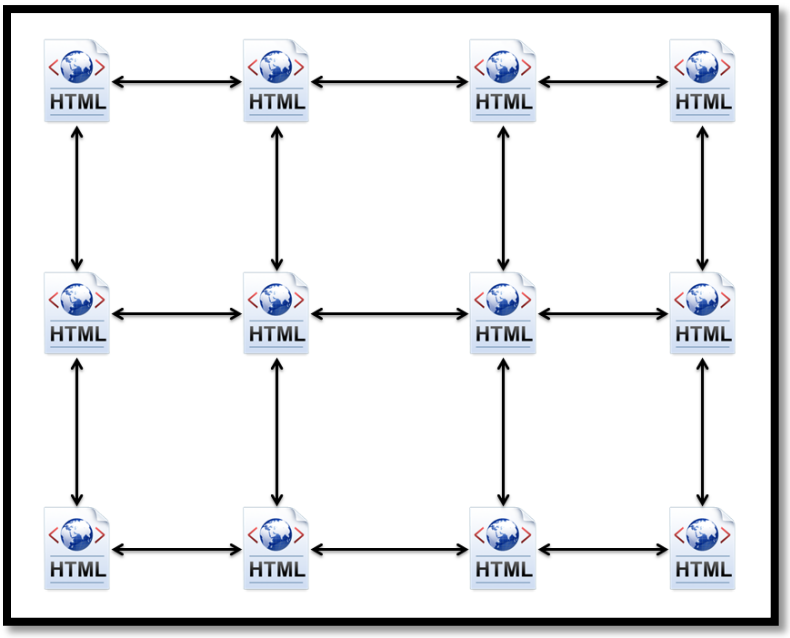



**Figura 3.3.2.2. Estructura Reticular**

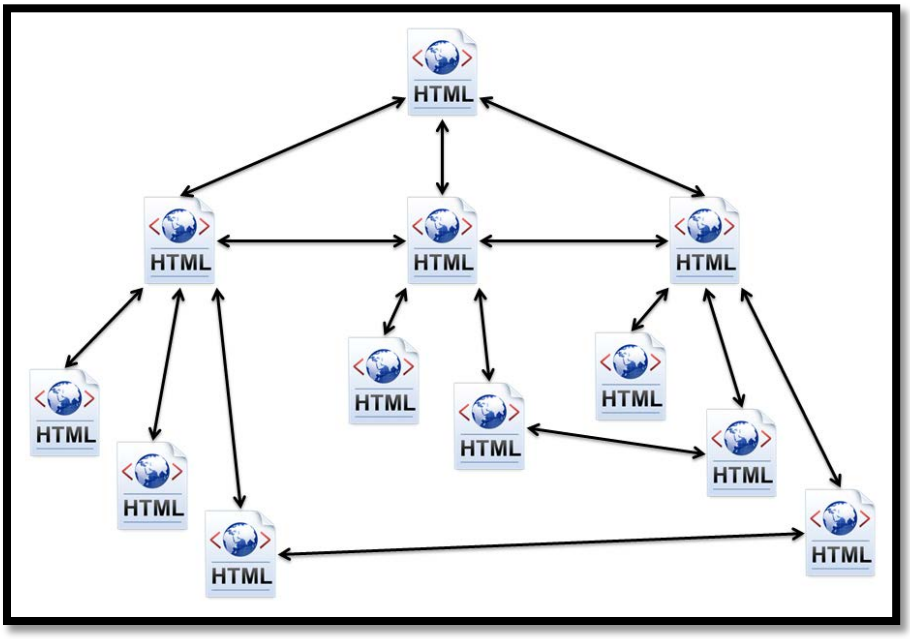



**Figura 3.3.2.3. Estructuras Jerárquicas**

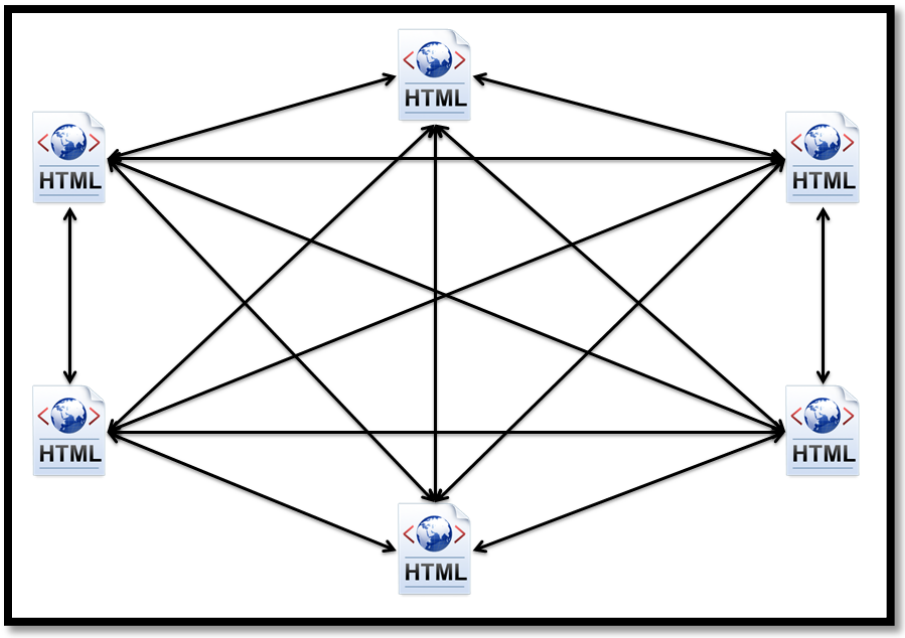



**Figura 3.3.2.4. Estructura en red**

En la Figura 3.3.2.4 se puede observar un ejemplo de estructura en red

Todas estas estructuras, pueden ser combinadas para formar una estructura compuesta. La arquitectura global de una aplicación web puede ser jerárquica, sin embargo parte de dicha estructura podrá presentar características lineales, mientras que otra parte podría comportarse como una estructura de red. La meta del diseñador arquitectónico es hacer corresponder la estructura de la aplicación con el contenido que será presentado y el procesamiento que se llevara a cabo.



*Diseño de navegación*

Una vez definida la arquitectura de la aplicación web y los componentes de la arquitectura, el diseñador deberá definir las rutas de navegación que permitirán al usuario acceder al contenido y a los servicios de la aplicación web. Para que el diseñador pueda realizar esta tarea deberá identificar la semántica de la navegación para los diferentes tipos de usuario del sitio y definir la mecánica para realizar la navegación.

Generalmente una aplicación web de tamaño considerable tendrá una variedad de roles de usuarios diferentes, por ejemplo visitantes, clientes registrados o clientes privilegiados, así como administradores y/o gestores del sistema. Cada uno de estos roles será asociado a diferentes niveles de acceso al contenido y servicios distintos. Donde la semántica de cada uno de estos roles será diferente.

*Diseño de páginas Web*

Diseño de páginas web es una actividad importante, ya que determina la información que se presenta y cómo se presenta a los usuarios. Un prototipo por lo general contiene una serie de páginas de muestra para evaluar el diseño de página, presentación y

navegación. Con base en la retroalimentación de las partes interesadas, el diseño de la página será convenientemente modificado. Este proceso puede ir a través de unas pocas iteraciones hasta que las partes interesadas y los diseñadores están satisfechos con el diseño de la página, la presentación y la estructura de navegación.

El desarrollo de contenido de la página web debe tener en cuenta los requisitos, de los usuarios de los grupos de interés, cuestiones técnicas y consideraciones, los problemas no técnicos, experiencias anteriores de los desarrolladores y usuarios, y las lecciones aprendidas de las aplicaciones web similares. [M. Brandon, (2008)]

En la siguiente figura se muestran los componentes del diseño de páginas web

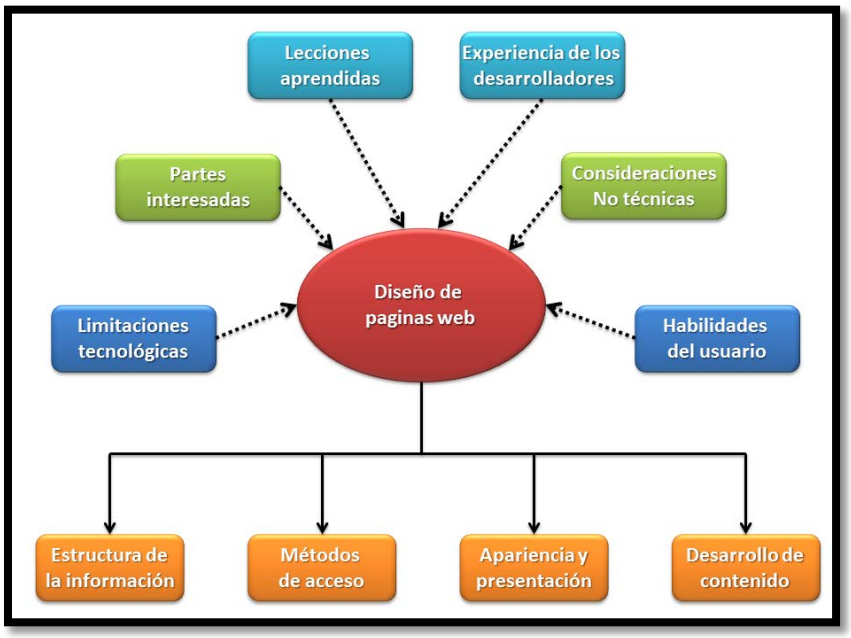



**Figura 3.3.2.5. Diseño de Páginas Web**

Si el contenido del sistema web ha sido desarrollado desde una perspectiva global, deberá ser responsable con la sensibilidad cultural, en el lenguaje extranjero, con el adecuado uso de colores, presentación y animación.

## 3.3.3 Gestión del proyecto

La finalidad de la gestión de proyectos, es asegurar que todos los procesos y actividades clave trabajan en armonía. Crear aplicaciones web exitosas, requiere de una estrecha coordinación entre los diferentes grupos involucrados en el ciclo de desarrollo.

Sin embargo, existen estudios que demuestran que una gestión pobre de proyectos es la principal causa de fallas durante el desarrollo y posteriormente en la fase de

explotación del sistema, una pobre gestión del proyecto puede anular fácilmente una buena ingeniería, mientras que una buena gestión es la clave del éxito. Gestionar correctamente una aplicación web a gran escala es una tarea que representa un gran reto, que requiere de habilidades multi disciplinarias y se diferencia en muchas formas a un proyecto de TI tradicional.

El control de calidad, la documentación y un análisis exhaustivo son actividades importantes durante la gestión del proyecto, sin embargo, son a menudo descuidados, mientras que en realidad estas tareas deberían de extenderse a través de todo el ciclo de vida del desarrollo web.

El desarrollo exitoso de sistemas Web implica múltiples pasos interactivos que se influyen mutuamente: [M. Brandon, (2008)]

1. Comprender la función general del sistema y el medio entorno operativo, incluyendo los objetivos de negocio y los requisitos, cultura de la organización y gestión de la información política.

2. Identificar claramente los grupos de interés, es decir, principales usuarios del sistema y sus perfiles, la organización que tiene la sistema, y los trasfondos del desarrollo.

3. Obtener o especificar la funcionalidad inicial, requisitos técnicos y no técnicos de los grupos de interés y el sistema en general. Además, reconocer que estos requisitos no pueden permanecer siendo los mismos, sino que se encuentran obligados a evolucionar con el tiempo durante el desarrollo del sistema.

4. Desarrollar una arquitectura general del sistema que cumpla con los requisitos técnicos y no técnicos.

5. Identificar los sub proyectos o subprocesos a implementar en la arquitectura del sistema. Si los sub proyectos son demasiado complejos para gestionar, será necesario dividirlos hasta que se convierten en un conjunto de tareas manejables.

6. Desarrollar e implementar los sub proyectos.

7. Incorporar mecanismos eficaces para gestionar la evolución, el cambio y mantenimiento del sistema Web. A medida que el sistema evolucione, será necesario repetir el proceso general o algunas partes del mismo, según se requiera.

8. Abordar las cuestiones no técnicas, como los procesos de negocio, organizativas y las políticas de gestión, los recursos humanos y los aspectos legales, culturales, y sociales.

9. Medir el rendimiento del sistema, analizar el uso de la aplicación web desde los registros del sistema, revisar y direccionar las sugerencias de los usuarios.

10. Perfeccionar y actualizar el sistema.

Internet es una plataforma abierta que ofrece oportunidades sin precedentes. Sin embargo, es básicamente imposible ejercer control sobre el volumen de visitantes de un sitio web o como y cuando acceder a una página web.

Esto hace que el desarrollo de aplicaciones web, que presenten un rendimiento satisfactorio incluso ante un aumento repentino de usuarios, sea una tarea nebulosa y complicada.

Satisfacer las expectativas y necesidades de los diferentes tipos de usuarios no es sencillo, cuando se encuentran con un sitio hostil, confuso o que presenta demasiada información, experimentaran un sentimiento de frustración o peor aún, estos usuarios frustrados comenzaran a difundir una mala opinión. Los factores de usabilidad de un sitio web incluyen buen uso de los colores, contenido de la información y una navegación sencilla entre otros factores. [M. Brandon, (2008)]

Un sistema basado en la Web también tiene que satisfacer a diferentes grupos de interés, además de la amplia gama de usuarios, incluyendo: las personas que mantienen el sistema, la organización que necesita el sistema, y los que financian el desarrollo del mismo. Esto puede plantear algunos desafíos adicionales al diseño de sitios web y desarrollo del sistema. Dado que las aplicaciones Web se están convirtiendo en una necesidad crítica, hay una mayor demanda de mejora, fiabilidad, rendimiento y seguridad.



El mal diseño de la aplicación o la infraestructura, ha causado que muchas páginas web sean incapaces de solventar las solicitudes de los usuarios. Mutiles sitios web han sufrido accidentes, fallas en el desempeño, brechas de seguridad y cortes de energía, los cual se refleja en la molestia de los clientes, pérdida de ingresos, devaluación de las acciones de la empresa, mala publicidad, pérdida de confianza por parte de los clientes, perdidas permanentes de clientes y finalmente demandas judiciales.

Estos fracasos y su impacto en las empresas han servido como un recordatorio contundente de la necesidad de una planificación de la capacidad, y la mejora de rendimiento, calidad y fiabilidad. La implementación exitosa de un sitio web requiere disponibilidad consistente de la aplicación, un mejor entendimiento del rendimiento, escalabilidad y balanceo de carga.

El diseño de sistemas Web a gran escala es una actividad compleja y difícil, ya que es necesario considerar múltiples aspectos y requisitos, algunos los cuales pueden tener necesidades que generen un conflicto entre ellas mismas.

Por lo tanto, el desafío consiste en diseñar y desarrollar sistemas web sostenibles para una mejor: [M. Brandon, (2008)]

- Usabilidad – diseño de interface y navegación
- Comprensión
- Rendimiento
- Seguridad e integridad del sistema
- Evolución, crecimiento y mantenimiento
- Capacidad de prueba.

La ingeniería Web es una actividad técnica complicada. Existen muchas personas implicadas, y a menudo trabajando en paralelo. La combinación de tareas técnicas y no técnicas para producir una aplicación web de alta calidad representa un reto para cualquier grupo de profesionales. Con la finalidad de evitar cualquier confusión, frustración y fallo, se deberá poner en acción una planificación, tener en cuenta los riesgos, establecer y rastrear una planificación temporal y definir los controles. Estas son las actividades clave que constituyen lo que se conoce como gestión de proyectos.



La creación de una aplicación Web exige un amplio abanico de conocimientos. Y dado las exigencias asociadas a los proyectos importantes de desarrollo de aplicaciones web, el conjunto de conocimientos diversos necesario podrá distribuirse al crear un equipo de ingeniería web.

Entre los conocimientos que deberán encontrarse dentro de los miembros del equipo se encuentran: ingeniería del software basada en componentes, realización de redes, diseño arquitectónico y de navegación, lenguajes y estándares de Internet, diseño de interfaces, diseño gráfico, disposición del contenido y pruebas. Dentro de cada equipo se deberán asignar los siguientes roles: [Pressman, (2002)]

- *Desarrolladores y proveedores de contenido***:** deben centrarse en la generación y/o recolección del contenido el cual abarca un amplio abanico de objetos de datos como información de productos e imágenes gráficas, composiciones gráficas y contenidos estéticos, contenido basado en texto. Además, existe la posibilidad de necesitar personal de investigación que encuentre y dé formato al contenido externo y lo ubique y referencie dentro de la aplicación.

- *Editores de Web***:** el contenido generado por los desarrolladores y proveedores deberá organizarse e incluirse dentro de la aplicación. Además de actuar como conexión entre el personal técnico, los diseñadores y proveedores del contenido. El editor de Web, deberá entender la tecnología tanto del contenido como de la

aplicación, lo que incluye el lenguaje HTML, la funcionalidad de la base de datos y guiones, y la navegación global del sitio Web.

- *Ingeniero de Web:* el ingeniero Web se verá involucrado en las actividades del desarrollo de una aplicación web como la obtención de requisitos, modelado de análisis, diseño arquitectónico, de navegación y de interfaces, implementación de la aplicación y pruebas. Adicionalmente deberá conocer a fondo las tecnologías de componentes, las arquitecturas cliente/servidor, HTML/XML, y las tecnologías de bases de datos, conocimiento del trabajo con conceptos de multimedia, plataformas de hardware/software, seguridad de redes y temas de soporte a los sitios Web.

- *Especialistas de soporte:* Se asigna a la persona que tiene la responsabilidad de continuar dando soporte a la aplicación. Dado que esta se encuentra en constante evolución, el especialista de soporte es el responsable de las correcciones, adaptaciones y mejoras del sitio Web, donde se incluyen actualizaciones del contenido, implementación de productos y formularios nuevos, y cambios del patrón de navegación.

- *Administrador.* Se suele llamar *Webmaster,* y es el responsable del funcionamiento diario de la aplicación, lo que incluye: el desarrollo e implementación de normas para el funcionamiento de la aplicación; establecimiento de los procedimientos de soporte y realimentación, Derechos de acceso y seguridad de la implementación; medición y análisis del tráfico del sitio Web, coordinación de los procedimientos de control de cambios, coordinación con especialistas de soporte.



El administrador también puede estar involucrado en las actividades técnicas realizadas por los ingenieros de Web y por los especialistas de soporte.

Aún cuando sea necesario recurrir a la subcontratación para el desarrollo de una aplicación web, toda organización deberá realizar una serie de tareas antes de buscar el proveedor de subcontratación para hacer el trabajo. [Pressman, (2002)]

1. Identificar el público de la aplicación; realizar un listado de aquellos con intereses internos en la aplicación; definir y revisas las metas globales de la aplicación; especificar tanto la información como los servicios que se van a proporcionar en la aplicación; destacar los sitios Web rivales, y definir las medidas cuantitativas y cualitativas para obtener una aplicación de calidad. Esta información deberá de documentarse en una especificación del producto.

2. Elaborar internamente un diseño aproximado de la aplicación, identificando el aspecto e interacción general de la aplicación para el proveedor de subcontratación, este diseño deberá incluir la indicación del proceso interactivo que se va a llevar a cabo como son: formularios, entrada de pedidos, etc.

3. Desarrollar una planificación temporal aproximada del proyecto, incluyendo no solo las fechas finales de entrega, sino también las fechas significativas. Se deberá identificar el grado de supervisión e interacción del contratista con el proveedor. Deberá incluirse el nombre del contacto del proveedor y la identificación de la responsabilidad y autoridad del contacto, la definición de los puntos de revisión de calidad a medida que el desarrollo va avanzando, y la responsabilidad de los proveedores en relación con la comunicación entre organizaciones.

Toda la información desarrollada en los pasos anteriores deberá de organizarse en una solicitud de opciones que se transmite a los proveedores candidatos.

Finalmente, dado que es muy probable que el ámbito cambie a medida que avanza el proyecto de la aplicación, el modelo de proceso deberá ser incremental. Esto permite que el equipo de desarrollo “congele” el ámbito para poder crear una nueva versión operativa de la aplicación. El incremento siguiente puede afrontar los cambios de ámbito sugeridos por una revisión del incremento precedente, pero una vez que comienza el incremento, el ámbito se congela de nuevo “temporalmente”. Este enfoque hace posible que el equipo de la aplicación trabaje sin tener que aceptar una corriente continua de cambios, pero reconociendo la característica de evolución continúa de la mayoría de las aplicaciones. [Pressman, (2002)]

## 3.4 Gestión y Administración Aeroportuaria

Existe un conjunto de agentes que operan en un aeropuerto ejerciendo un amplio abanico de actividades, la mayor parte de estas son realizadas por parte de empresas y de diversas agencias de la Administración, y no por parte del operador aeroportuario. El operador aeroportuario es el organizador y regulador del intercambio modal (aéreo, terrestre) y de los diversos flujos físicos: aviones, pasajeros, personal, maletas, vehículos de handling, taxis, etc. Asimismo, es el propietario o explotador de los terrenos aeroportuarios y espacios susceptibles de desarrollar actividades económicas.

**Tabla 3.4.1. Agentes que operan en un Aeropuerto**

| Función | Actividades |
|---|---|
| Operador Aeroportuario | Gestión operaciones aire y tierra, propiedad de pistas, terminal y terrenos afectados |
| Proveedor de servicios en aeronaves | Handling rampa, catering, combustible |
| Proveedor de servicios a pasajeros | Handling pasajeros, tiendas, restauración, estacionamientos, servicios complementarios |
| Navegación Aérea | Servicios de control del tráfico |
| Regulación Administrativa | Aprobación precios y tasas, autorizaciones administrativas |
| Otros | Policía / Seguridad, aduanas, transporte público |



### 3.4.1 Gestión Aeroportuaria

Los modelos de gestión de los aeropuertos son consecuencia de la historia y de la arquitectura político administrativa de cada país. No obstante, hay una evolución clara, detectable prácticamente en todo el mundo, desde modelos de gestión directa por parte de la Administración hacia modelos de gestión mediante sociedades mercantiles con grados variables de propiedad y control privado. Se detecta claramente cómo en los países menos desarrollados, los aeropuertos, así como la navegación aérea, son gestionados directamente por la administración de aviación civil, o incluso militar en algunos casos. En este contexto, el aeropuerto se concibe fundamentalmente como un proveedor de infraestructura y la gestión está en gran medida orientada a los aspectos operacionales de la actividad aeroportuaria. Esta situación coexiste en muchos de los países donde se da con la presencia de una compañía aérea “de bandera” como

operador dominante en el aeropuerto y, por ello, el operador aeroportuario no percibe como prioridad ni el marketing ni la captación de otras compañías o destinos que incluso podrían competir con su compañía nacional. En este modelo, los equipos directivos tienen un perfil esencialmente técnico y administrativo (cuando no político), están poco familiarizados con una gestión comercial y reciben pocos incentivos para maximizar los ingresos aeronáuticos y no aeronáuticos.

Este modelo está también generalizado en Estados Unidos, donde la mayoría de los aeropuertos comerciales son de titularidad de las administraciones locales, aunque en muchos casos las terminales están concesionadas a las compañías aéreas que operan desde instalaciones propias. En Estados Unidos, las inversiones en los aeropuertos se financian con emisiones de bonos de los municipios, mientras que en muchos otros lugares del mundo la venta de acciones al sector privado se utiliza a menudo para financiar ampliaciones de los aeropuertos.

Un segundo estadio es la gestión por parte de organismos públicos de tipo empresarial. Éste es el caso, por ejemplo, de Aena en España y México, de la ONDA de Marruecos y, hasta 2005, de los aeropuertos de París (ADP). En este modelo se exige a los gestores aeroportuarios que, sin perder la titularidad y gestión públicas, apliquen criterios de optimización de resultados económicos y operativos. Se puede exigir al organismo público, como es el caso de Aena, que se autofinancie al 100% sin recurrir a financiación presupuestaria. Este modelo permite más agilidad y demuestra un mayor desarrollo de la cultura de gestión pública de un país.



Un tercer nivel de desarrollo, es la gestión de los aeropuertos mediante sociedades mercantiles. Este modelo aleja la gestión de los aeropuertos de la esfera administrativa y permite una mayor eficiencia, menores interferencias políticas y una relación más proactiva con las compañías aéreas que pasan de ser “*usuarias*” a ser “*clientes*”.

En el modelo societario, la unidad de negocio natural es el aeropuerto individual con sus propios recursos, objetivos y cuentas de resultados. En este caso, los sistemas multi aeropuertos se consolidan normalmente en sociedades holding con diversas filiales.

Un cuarto estadio sería propiamente la “privatización” del aeropuerto. La privatización puede darse mediante dos grandes fórmulas: bien con transferencia de la propiedad -parcial o total– de los activos aeroportuarios (terrenos, instalaciones), bien sin transferencia de propiedad, como son los casos de concesión, contratos de gestión. Los propietarios privados de aeropuertos son más sensibles que los públicos a la rentabilidad de las inversiones que realizan.

No es de extrañar, pues, que la mayoría de los grandes proyectos de ampliación de aeropuertos en Asia, Oriente Medio, Europa, los realicen aeropuertos públicos.
A pesar de la profusión de literatura al respecto, la presencia pública continúa siendo hegemónica en los grandes aeropuertos europeos y también en la mayoría de los grandes concentradores asiáticos (Singapur, China, Corea, Malasia). Por el contrario,

Australia, Nueva Zelanda y Latinoamérica han experimentado un intenso proceso privatizador, que están viviendo también los países del este de Europa y recientemente la India. En Europa Occidental, la presencia privada es mayor en aeropuertos medianos y pequeños pero menor en los grandes concentradores. En la mayoría de los ejemplos de aeropuertos parcial o totalmente privatizados confluyen algunos de los siguientes aspectos: a) un sector público estructuralmente ineficaz y/o reacción al cambio, b) debilidad económica del titular del aeropuerto para acometer inversiones necesarias, y c) razones ideológicas.

Una contradicción que aparece a menudo es que existen sociedades públicas, o mayoritariamente públicas, de países concursando y ganando en procesos de privatización de aeropuertos de otros países. Es el caso de Aena, que participa, en diferente medida, en la gestión “privada” de diversos aeropuertos en Cuba, México, Colombia e incluso Inglaterra. Los aeropuertos de Frankfurt, París, Ámsterdam, Dublín, Milán y Viena, a pesar de tener una participación pública mayoritaria, vienen siendo especialmente activos en el mercado internacional de privatizaciones.

Es relevante preguntarse en este punto, ¿por qué los aeropuertos son tan atractivos para la inversión privada? ¿Y por qué el sector privado parece más interesado en los aeropuertos que en otras infraestructuras como, por ejemplo, los puertos? Entre las razones de este interés se pueden apuntar que los aeropuertos son monopolios naturales, que la demanda del transporte aéreo crece tradicionalmente por encima del PIB, que son inversiones con poco riesgo, que hay un amplio abanico de oportunidades para el desarrollo de actividades comerciales dentro de las terminales y, finalmente, que los aeropuertos ofrecen un excelente potencial de desarrollo inmobiliario en su entorno (las denominadas ciudades aeroportuarias o ciudades aeropuerto).



Todo este potencial de negocio explica por qué los aeropuertos son los actores que extraen más rentabilidad en toda la cadena del sector aeronáutico. La diferencia de rentabilidad entre los aeropuertos y las compañías aéreas que constituyen sus clientes y su razón de ser es muy importante y cada vez mayor. Asimismo, el sector del transporte aéreo es más volátil y se ve más afectado por los ciclos económicos que los aeropuertos. Esta situación explica la dialéctica entre las aerolíneas y los aeropuertos en todo el mundo.

Es razonable pensar que los aeropuertos con una importante presencia pública van a ser más activos en la captación de nuevas rutas aéreas y la consolidación de la red existente que dé accesibilidad a su territorio. Así, aeropuertos públicos como Manchester, Ámsterdam o Zúrich han puesto un importante énfasis en este objetivo.

### 3.4.2 Aerolíneas de bajo costo

Una aerolínea de bajo costo -*low-cost*, como se les conoce mundialmente- es una aerolínea que generalmente ofrece bajas tarifas a cambio de eliminar muchos de los servicios tradicionales a los pasajeros. El concepto surgió en los Estados Unidos antes de extenderse por Europa a principios de los 90 y de ahí al resto del mundo. Originalmente el término era empleado dentro de la industria de la aviación para referirse a compañías con costos de operación bajos o menores que los de la competencia. A través de los medios de comunicación de masas su significado varió y ahora define a cualquier aerolínea en la que se requiere poco dinero y la que da servicios limitados.

El modelo empresarial típico de una compañía aérea de bajo costo se basa en una única clase de pasajeros. Una flota compuesta de aviones de un único modelo, generalmente. Así se reducen los costos de entrenamiento y servicio. Un único tipo de tarifa (normalmente el precio crece a medida que se venden los billetes, lo que busca recompensar la reserva con gran anticipación). Plazas no numeradas. Los pasajeros se sientan donde eligen, acelerando el embarque. Vuelos a aeropuertos secundarios, más baratos y menos congestionados.



De este modo se evitan los retrasos debidos al tráfico y se aprovechan las menores tasas por aterrizaje de estos aeropuertos. Vuelos cortos y con muchas frecuencias. Rutas simplificadas, potenciando los viajes por etapas en lugar de los engorrosos transbordos en los concentradores de las compañías. Así se eliminan molestias por retrasos en la llegada de pasajeros o por pérdida de equipajes procedentes de los vuelos de conexión, potenciación de la venta directa de billetes, especialmente a través de Internet, evitando tasas y comisiones de las agencias de viajes y de los sistemas informatizados de reserva. Los empleados trabajan realizando múltiples tareas.

Por ejemplo, hay auxiliares de vuelo que también limpian el avión o controlan la entrada en la puerta de embarque, reduciéndose así los costos de personal. El cátering (servicio de alimentación institucional o alimentación colectiva) "*gratuito*" a bordo y otros servicios complementarios desaparecen o pasan a ser de pago. Esto representa un beneficio adicional para la aerolínea. Además de políticas agresivas de acaparamiento de combustible: las compañías compran grandes cantidades de combustible a bajo precio, de forma que si éste aumenta, dicho crecimiento no repercute directamente sobre el precio del billete.

Las tasas de aeropuerto y los cargos por emisión son descontados del precio anunciado, de forma que éste parece menor de lo que en realidad es. Bajos costos de operación o menores que los de la competencia tradicional.

Las compañías de bajo costo suponen una seria amenaza a las aerolíneas tradicionales de "servicio completo", pues el alto costo operativo de estas últimas no les permite

competir de forma efectiva con los precios de los billetes ofertados por las aerolíneas de bajo costo, y éste es el factor más importante para los clientes.

De 2001 a 2003, periodo en que la industria de la aviación fue duramente golpeada por el terrorismo, la guerra y el SARS, la gran mayoría de las aerolíneas tradicionales sufrieron fuertes pérdidas mientras las compañías de bajo costo se mantuvieron en positivo.

En el mercado de los destinos vacacionales, las aerolíneas de bajo costo también compiten con los vuelos chárter de pasajeros. Sin embargo, la inflexibilidad de los chárters (sobre todo en lo que a la duración de la estancia se refiere) los ha hecho impopulares entre los viajeros. Se ha intentado extrapolar la dinámica del bajo coste a otras industrias y negocios como la hotelera o el software.

Las ABC (Aerolíneas de Bajo Costo) son un modelo de negocios fundamentado en una muy alta eficiencia y en la tecnología de punta, con el objeto de poder utilizar al máximo los equipos a un índice de productividad muy elevado.

La súbita expansión de este tipo de compañías tiene que ver con aspectos tales como: la emisión de boletos electrónicos, la disminución del personal, la simplificación del servicio, la operación desde aeropuertos alternativos con tasas más baratas, la venta de comida a bordo, la cancelación de todo servicio superfluo y básicamente, todo aquello que permita ofrecer un boleto lo más barato posible. De tal tamaño es el fenómeno que las estimaciones señalan que actualmente uno de cada ocho vuelos se realiza en aerolíneas de bajo costo y éstas concentran el 15 por ciento de asientos disponibles a nivel mundial.



### 3.4.3 Handling

"*Handling*" es una palabra inglesa que significa "*manejo*". El sector de la aviación la ha incorporado a su vocabulario para definir la asistencia que se realiza en los aeropuertos a: pasajeros, equipaje, carga y correo, tripulaciones, aviones, vuelos.

El Handling de pasajeros comprende la atención a los clientes desde el mismo momento en que llegan al aeropuerto hasta que entran en el avión, donde la tripulación del vuelo está a su disposición. Es decir: Facturación Determinación y cobro de exceso de equipaje, salas VIP, embarque, tránsitos, objetos perdidos, gestión de incidencias.

Existe otro tipo de Handling que es el de equipaje y de carga, el cual se refiere al tratamiento del equipaje de los pasajeros, así como de la carga que lleva el avión. El tratamiento del equipaje se divide en tres partes: a) Carreteo, que es el traslado de la carga desde la terminal al avión y viceversa; b) Carga y descarga en el avión y c) Entrega en las cintas (equipaje) y en la terminal de carga (mercancías y correo).

A la asistencia que se realiza al avión se le conoce como Handling de rampa, e incluye una coordinación la cual se encarga de la supervisión y registro de las operaciones de atención al avión, tanto a pie de pista como en contacto con la tripulación. El Handling de rampa es encargado también de la elaboración de la hoja de carga y centrado, donde se especifica que es lo que lleva el avión dentro, su peso, incluidos el lastre y repuestos de la aeronave, además de cómo se coloca el peso total dentro del aparato.

Todo esto es fundamental para determinar correctamente el centro de gravedad del avión, que es vital para garantizar la seguridad del vuelo. Dentro de este tipo de Handling, se manejan los mensajes operativos, que no es más que la elaboración y envío de información con los datos del vuelo (hora de llegada, carga, número de pasajeros) a la escala de destino y a la base de operaciones de la compañía asistida. En el embarque y desembarque del avión es necesaria siempre por medidas de seguridad, la presencia de escaleras aunque los pasajeros entren y salgan del avión por pasarela.

Otras actividades de las que se encarga el Handling de rampa son: Grupo neumático y aire acondicionado, traslado de pasajeros, traslado de tripulaciones, push-back que no es más que un grupo de máquinas que empujan hacia a tras a la aeronave puesto que esta no tiene reversa.

### 3.4.4 Condiciones generales del transporte de pasajeros



Cuando los pasajeros compran un billete para un vuelo con cualquier compañía aérea (aerolínea), formalizan un contrato de transporte con dicha compañía, en el cual se aplican ciertas condiciones generales, las cuales se basan en la definición y el manejo de cada uno de los elementos existentes en la aerolínea. Algunos de estos elementos son: billetes (tickets), escalas realizadas, tarifas, reservas, facturación, carga (equipaje, combustible y pasajeros), entre otros.

El término pasajero se refiere a cualquier persona, excepto los miembros de la tripulación, para ser transportada en una aeronave con el consentimiento de la aerolínea.

Las condiciones generales de Handling son aplicables a todos los vuelos operados bajo el código de línea de cada aerolínea, considerando a ésta última como el transportista. El código de línea o aerolínea se refiere al código emitido por la IATA (Asociación Internacional de Transporte Aéreo) que identifica a cada compañía miembro de ésta asociación utilizando dos o más caracteres alfabéticos, numéricos o alfanuméricos, y que puede encontrarse, entre otros lugares, en el billete.

Bajo el código de línea, el equipaje se refiere a los artículos, efectos y otras pertenencias personales de los pasajeros que son necesarias y apropiadas para vestir y usar cómodamente durante el viaje. El Handling de equipaje comprende al equipaje facturado y no facturado. Como equipaje facturado se entiende por aquel que el

transportista tiene la custodia exclusiva y para el cual ha expedido una etiqueta de equipaje y es transportado en la misma aeronave que el pasajero. El equipaje no facturado es todo aquel que no cuenta con dicha etiqueta y que además debe alojarse debajo del asiento anterior o en un compartimento en la cabina para dicho propósito.

La facturación tiene como objetivo efectuar las anotaciones del equipaje en el billete (ticket/boleto), mostrando la información del viaje del pasajero, es decir, destino, número de vuelo, hora programada de salida así como el logotipo del transportista. Dentro de la facturación el Agente de Handling proporciona información de las dimensiones requeridas del equipaje u otros detalles. Si el pasajero no lleva equipaje, se anota una “X” en el billete. La etiqueta de facturación del equipaje emitida en el mostrador de facturación se adjunta al billete del pasajero.

El equipaje es pesado, y no debe rebasar el peso máximo requerido que es de: 40 Kg. para pasajeros en Primera clase y 30 Kg. para pasajeros en clase Turista, es indicado en el billete del pasajero. Los efectos personales son transportados en cabina como tales, sin ser pesados ni etiquetados además de que no se anexan al registro en el billete.

El equipaje en cabina se limita a una pieza por pasajero bajo las siguientes condiciones: Peso máximo: 06 Kg/3Lb. Dimensión máxima: 50 x 45 x 25 cm (20 x 16 x 10 pulgadas), y es etiquetado como “Equipaje de cabina”.



El transportista toma en cuenta de forma independiente el equipaje de mano de cada pasajero que es provisto de una etiqueta con la leyenda “Equipaje de cabina facturado”, certificando que el equipaje de cabina ha sido pesado y aceptado dentro de los límites de peso y volumen, si no cuenta con la etiqueta de “Equipaje de cabina” es considerado como exceso de equipaje.

Dentro del término equipaje, se encuentran los equipos deportivos que los pasajeros suelen llevar, para los cuales el personal de facturación maneja como peso máximo 15 Kg. y como dimensiones no deben exceder de 400 x 65 x 5 cm (158 x 25 x 1 pulgadas).

En muchas ocasiones, los pasajeros al momento de viajar llevan consigo, a parte de su equipaje, animales que lo acompañan. Los animales de compañía, son aceptados para transporte siempre y cuando sean sujetos a las regulaciones del transportista. Si las dimensiones máximas del contenedor/caja del animal, no exceden de 50 x 45 x 25 cm (20 x 18 x 10 pulgadas) y no pesan más de 6 Kg. (13 Lb.). En el caso en el que el pasajero viaje con más animales y junto con su equipaje no rebasan los 40 Kg. en Primera clase y los 30 Kg. en clase Turista, la facturación se hace sin ningún inconveniente, sin embargo, si rebaza estos límites, se considera como exceso de equipaje.

El exceso de equipaje es transportado a bordo de cualquiera de los vuelos, dependiendo de la capacidad de las bodegas, tanto en peso como cubicaje, y el encargado del transporte de este es el Fletador.

Durante el embarque de la aeronave, es común que exista exceso de peso, dada esta situación, la carga necesita ser reducida de acuerdo a la siguiente prioridad: a) Repuestos de la compañía; b) Personal que viaje con billetes sujetos a espacio; c) Equipaje sin pasajeros; d) Exceso de equipaje; e) Equipaje del pasaje; f) Personal de la compañía viajando en comisión de servicio; g) Pasajeros no incluidos en el listado; h) Repuestos AOG.

Para llevar a cabo el procedimiento de embarque de la aeronave, se manejan Tarjetas de Embarque, en las cuales se pegan etiquetas en dos colores diferentes, como medio de revisar el número total de pasajeros embarcados, y es responsabilidad del Agente de Handling comprobar que el número de pasajeros en la salida de la terminal es el mismo que se encuentra anotado en la hoja de carga.

### 3.4.5 Hoja de carga

La hoja de Carga se usa generalmente en todos los vuelos que se realizan. Tiene diferentes misiones, algunas de estas son: Cargar el avión de forma correcta, asegurando que el avión no excede ninguno de sus límites operativos (como el MTOW, MZFW o la MLW). Repartir bien esta carga para comprobar después que está dentro de un centro de gravedad operable para el avión. Informar al capitán de la distribución de la carga, así como el centro de gravedad, que será importante a la hora de configurar el avión para el despegue.



Con toda esta información, compuesta por el peso total en cada compartimiento de carga, peso total de los pasajeros, el momento que tiene el avión, el capitán puede ajustar el timón de profundidad de la aeronave adecuadamente para cerciorarse de un correcto comportamiento del avión durante las diferentes fases del vuelo. Esto es crítico en el momento del despegue, hasta el punto de alargar más de lo que la pista da. Si el timón se ajusta mal, no dará el efecto “palanca” para poder levantar el morro con más facilidad.

En la Figura 3.4.5.1 se muestra un ejemplo de hoja de carga del vuelo de SwissAir SR0111 de Nueva York a Ginebra. Es posible ver como se detallan los pasajeros, el peso en los compartimientos, los diferentes pesos dependiendo de la fase del vuelo y algunos otros detalles.

Las hojas de carga varían dependiendo del transportista y de la aeronave, puesto que las aeronaves no son iguales y no tienen la misma capacidad, por ello existen diferentes formatos que dependen del modelo de la aeronave y sus limitaciones.

Es responsabilidad del Agente de Handling el manejo de la información procedente de la hoja de carga, es decir, es el que se encarga de la verificación de la información de la carga dentro de la aeronave. El llenado de la hoja de carga se realiza de una forma manual en el momento en el que la información de los pasajeros que abordan la aeronave así como el peso del equipaje que lleva cada pasajero, ha sido recompilada.

```
L O A D S H E E T            CHECKED              APPROVED                EDNO
ALL WEIGHTS IN KILOS                                                      02

FROM/TO FLIGHT      A/C-REG    VERSION      CREW    DATE        TIME
JFK GVA  SR0111     HBIWF      M1130260      2/12   02SEP98     1938

                          WEIGHTS          DISTRIBUTION
LOAD IN COMPARTMENTS       21125      1/5531  2/6545   3/5520
                                      4/2789  5/740

PASSENGER/CABIN BAG        17892   174/  39/    0/   2 TTL  215 CAB
                                   PAX  10/   42/ 161 SOC
                                              BLKD



**********************************************************************************************
TOTAL TRAFFIC LOAD          39017
DRY OPERATING WEIGHT       137830
ZERO FUEL WEIGHT ACTUAL    176847      MAX  185970      ADJ
TAKE OFF FUEL               64300
TAKE OFF WEIGHT ACTUAL     241147      MAX  285990      ADJ
TRIP FUEL                   49600
LANDING WEIGHT             191547      MAX  199580      ADJ

BALANCE AND SEATING CONDITIONS          LAST MINUTE CHANGES
DOI   59      DLI   43     LIZFW    46   DEST   SPEC   CL/CPT   WEIGHT/IND
LITOW    58    MACZFW   19.8
BASED ON FUEL DENSITY        .810 KG/LTR
A10.B42.C14.D147.
SEATROW TRIM

UNDERLOAD BEFORE LMC     8033                  LMC TOTAL
**********************************************************************************************
LOADMESSAGE AND CAPTAINS INFORMATION BEFORE LMC

-GVA. 174/39/0/2. T21125. 1/5531. 2/6545. 3/5520. 4/2789. 5/740
.PAX /10/42/161.PAD/0/0/1
SI
SERVICE WEIGHT ADJ   WGT/IND
ADDITIONS
NIL
DEDUCTIONS
NIL
PANTRY CODE  M
NOTOC    YES
GVA     C    14585M     554 B 275/   4189      0       0       T       0
END LOADSHEET EDNO 02 SR0111  02SEP98  193842
```

**Figura 3.4.5.1. Hoja de carga de un Aeronave**

En el momento en el que los datos de la hoja de carga son correctos, se procede a obtener el centro de gravedad (CG) de la aeronave, para esto existen algunas herramientas computacionales, que llevan a cabo de forma automática, tomando en cuenta la carga de la aeronave, el trazado o el cálculo del centro de gravedad, por ello es muy importante que los datos establecidos en la hoja de carga sean lo más confiable

posibles, pues de esto depende la distribución de la carga para comprobar el CG y así configurar el avión para el despegue.

Los accidentes ocurridos en aviones de pasajeros y de carga producidos por problemas con la carga y centrado (W&B) son más comunes de lo que en principio podría parecer, según un estudio del Laboratorio Aeroespacial Nacional holandés. La investigación revela que, los accidentes de W&B son cada vez peor en Norteamérica.

El informe fue presentado por el asesor de Seguridad y Operaciones de Vuelo, Gerard van Es, a la Flight Safety Foundation/European Regions Airline Association en Ámsterdam (Holanda). Está basada en un estudio de la base de datos de accidentes globales entre 1970 y 2008, y muestra que la W&B fue la principal causa de 82 de los accidentes documentados. Los accidentes en aviones de pasajeros representan el 61%, y el 39% restante en aviones de carga. En ambos casos la mayoría de estos accidentes ocurren durante la fase de despegue (68% con pasajeros y 56% con carga). El riesgo mayor en aviones de pasajeros se da por hojas de carga incorrectas, pero otros riesgos incluyen situar el centro de gravedad por delante del límite, así como despegues con sobrepeso.

Las hojas de carga llegan a contener información incorrecta debido a que la información requerida o generada durante la ejecución de los procesos involucrados en la carga de la aeronave no es registrada correctamente.

## 3.5 Desarrollo del Sistema de Información

Debido a la falta de control del llenado de la hoja de carga de la aeronave dentro de una aerolínea, y tomando en cuenta el alto índice de peligro que esto representa para los pasajeros o clientes, se llevó a cabo un análisis de las actividades que se realizan para la obtención de los datos necesarios en función del control de la aeronave. Paralelo a estas actividades se evaluaron las alternativas tecnológicas que aseguraran la integridad de la información, que facilitaran la transmisión remota de los aeropuertos a las oficinas corporativas y permitieran desarrollar las aplicaciones en corto plazo, con base en soluciones previstas. Tomando en cuenta estos aspectos, se llegó a la conclusión de desarrollar un sistema de información por medio del cual se incrementa la eficiencia de la prestación de servicios aeroportuarios y comerciales, en virtud de que con él se obtiene y controla la información requerida o generada durante la ejecución de los procesos involucrados en la carga de la aeronave. Siendo necesaria la implementación de una Intranet para el mejor flujo de información proporcionando y registrando los servicios de carga de la aeronave. Así mismo se controla y da seguimiento a la relación contractual de la empresa con sus clientes tanto en materia del servicio aeroportuario como de arrendamiento comercial y de servicios complementarios.



Se define la tecnología de desarrollo del sistema, la cual consiste en lo siguiente:

- Arquitectura: cliente-servidor.
- Redes locales en oficinas corporativas (nacional y regional) con servidor Apache.
- Manejador de bases de datos: PostgreSQL
- Herramienta de desarrollo: PHP

El tema de la presente investigación se basa en el desarrollo de una aplicación web basada en una arquitectura de tres capas para una red aeroportuaria complementada por una intranet, con el objetivo de mejorar el tiempo de transmisión de información requerida para realizar los reportes de carga y balance de las aeronaves que integran la red aeroportuaria, así como el tratar de que dicha información sea lo más confiable posible.

Al desarrollar una aplicación de tres capas, una de las ventajas con las que nos encontramos es con la posibilidad de utilizar esta misma arquitectura para el desarrollo de la aplicación, como observamos anteriormente una aplicación de tres capas se conforma por la capa de presentación, la capa de negociación y la capa de datos, siendo esta ultima la capa por la que se empezara a construir la aplicación, esto debido

a que en ella se modela y desarrolla la base de datos, la cual contiene la mayor parte de la información necesaria para interactuar con la capa de negociación.

La segunda fase del desarrollo de la aplicación está orientada a la realización de la capa de negociación, en la cual se procesa todo el flujo de información procedente o dirigido a la base de datos, en esta capa también se realiza la interface con la cual interactuara el usuario, por lo tanto dentro de esta capa encontraremos código HTML y php, dependiendo del tipo de script que se requiera para cada sección.

La capa de presentación será omitida debido a que esta se ejecuta en una computadora remota, la cual accede a la aplicación por medio de una aplicación cliente, más específicamente un navegador web. En su lugar se realiza la configuración de la intranet, basada en una red local ya existente, desde la cual se puede ingresar a las zonas restringidas de la aplicación. Mientras que mediante una conexión remota, solo se puede acceder a las secciones de dominio público de la aplicación.

Una de las primeras actividades a realizar para el desarrollo de la aplicación es la delimitación de las zonas de acceso público y las zonas de acceso restringido, siendo las primeras zonas o secciones de la aplicación a las que podrá acceder cualquier persona y que contendrán información pública de la empresa, mientras que las zonas restringidas serán única y exclusivamente para acceso de personal de la empresa.



### 3.5.1 Desarrollo de la base de datos

Inicialmente se desarrolla un modelo de base de datos específico para cubrir las necesidades y características de este sistema en particular. Dicho modelo se basa en una base de datos relacional, recibiendo este nombre debido a la relación que existe entre las diversas tablas que conforman dicha base de datos.

Para modelar la base de datos se opta por utilizar el software MySQL Workbench, de licencia GPL, esta decisión se realiza en base a la capacidad de dicho software de realizar el modelado gráficamente.

Comenzamos creando una base de datos de nombre "*red_aeroportuaria*", el comando para crear la base de datos es CREATE DATABASE *red_aeroportuaria*. Ya creada la base de datos se prosigue a crear cada una de las tablas que la conforman. Iniciando alfabéticamente por la tabla "*aeronaves*".

```
CREATE DATABASE red_aeroportuaria;
```

El comando utilizado para crear dicha tabla es CREATE TABLE, y cuenta con diversos parámetros a especificar, entre ellos el nombre de cada columna que conforman los registros dentro de esta, así como el tipo de dato, la longitud y algún parámetro

adicional como por ejemplo si es una llave primara, un valor nulo o si se auto incrementa para el caso de un identificador.

De esta manera el código fuente para la creación de la tabla aeronaves se muestra a continuación:

```
CREATE TABLE aeronaves(
id_aeropuerto INT(9) NOT NULL AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(45) NOT NULL,
compañía VARCHAR(45) NOT NULL FOREIGN KEY,
id_aeronave INT(9) NOT NULL FOREIGN KEY
);
```

Como se ve en el código, para esta tabla en específico, se encuentra el identificador respectivo llamado "*id_aeronave*" que acepta solo valores numéricos de una longitud de nueve dígitos, no acepta valores nulos, es llave primaria y se auto incrementa automáticamente con cada registro que se ingresa.

El segundo campo, denominado "*fabricante*" tiene un tipo de datos varchar, el cual acepta caracteres alfanuméricos de una longitud de 45 dígitos, y no puede ser nulo, esto debido a la necesidad de mantener la integridad de la información. Este tipo de datos es utilizado debido a que en caso de no utilizar los 45 caracteres declarados, la memoria utilizada solo será la que cubra la cantidad de caracteres usados y el sobrante se reutilizara, es decir, si se declaran 45 caracteres y solo se utilizan cinco, la memoria de almacenamiento utilizada será la de estos cinco caracteres y la memoria destinada a los otros 40 será reciclada.



Por último encontramos el campo "*compañía*", donde el tipo de dato utilizado nuevamente es VARCHAR, el cual no acepta caracteres nulos. Ahora bien, dentro del campo de "*fabricante*" se almacena el nombre del fabricante de la aeronave y en el campo "*compañía*" se establece el nombre de la compañía a la que pertenece dicha aeronave. Gráficamente podemos observar dicha tabla en la Figura 3.5.1.1


**Figura 3.5.1.1. Esquema grafico de la tabla Aeronaves**

Observando el modelo lógico obtenemos una tabla de datos de la siguiente manera.

| id_aeronave | fabricante | compañía |
|---|---|---|
| | | |
| | | |

Una vez creada dicha tabla procedemos a crear una nueva tabla llamada “aeropuerto”, en esta tabla se almacenara todo la información pertinente respecto a cada uno de los aeropuertos en los cuales se encuentra trabajando nuestra compañía. En esta tabla es donde empezamos a observar las características relacionales de nuestro modelo de base de datos.

El código fuente para realizar lo descrito anteriormente, se desarrolla de la siguiente manera.

```
CREATE TABLE aeropuerto(
id_aeropuerto INT(9) NOT NULL AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(45) NOT NULL,
compañía VARCHAR(45) NOT NULL FOREIGN KEY,
id_aeronave INT(9) NOT NULL FOREIGN KEY
);
```

Como se mencionó con anterioridad, cada una de las tablas que conforman nuestra base de datos cuenta con un identificador el cual es numérico y se auto incrementa conforme aumenta la cantidad de registros que se encuentran dentro de la tabla.



El registro “*nombre*” se refiere al nombre del aeropuerto o aeropuertos en los cuales las aeronaves tocan tierra y pueden cargar combustible o pasajeros así como sus respectivos equipajes, el tipo de dato utilizado nuevamente es VARCHAR debido a las características antes mencionadas.

Los registros “*id_aeronave*” y “*compañía*” son parte de la tabla “*aeronaves*” y se presentan como llaves foráneas, es decir, registros que conforman una tabla ajena, y a su vez, forman parte de una tabla nueva. Este tipo de registros se alteran, cuando la tabla original contenedora es modificada.
En este caso en particular, la información contenida en “*compañía*” e “*id_aeronave*” es posible modificarla desde la tabla aeronaves.

Gráficamente podemos observar dicha tabla en la Figura 3.5.1.2.


**Figura 3.5.1. 2. Esquema grafico de la tabla Aeropuerto**

Observando el modelo lógico obtenemos una tabla de datos de la siguiente manera.

| id_aeropuerto | Nombre | compañía | id_aeronave |
|---|---|---|---|
| | | | |
| | | | |

Continuando con la creación de las tablas por orden alfabético, procedemos a crear la tabla "*carga*" que como su nombre lo indica hace referencia a la información de la carga que será trasladada por la empresa.



El código para la tabla "*carga*" es el siguiente:

```
CREATE TABLE carga(
id_carga INT(9) NOT NULL AUTO_INCREMENT PRIMARY KEY,
guia VARCHAR(45) NOT NULL,
peso INT(9) not null,
compañia VARCHAR(45) NOT NULL FOREIGN KEY,
origen VARCHAR(45) NOT NULL FOREIGN KEY,
destino VARCHAR(45) NOT NULL FOREIGN KEY,
id_aeronave INT(9) NOT NULL FOREIGN KEY
);
```

Esta tabla es relativamente pequeña, ya que cuenta con tres registros propios, como son el ID, el número de guía que se almacena en la columna "*guía*", con un tipo de dato VARCHAR de 45 caracteres y que no permite campos nulos, el peso de la carga, expresado en datos numéricos con una capacidad máxima de nueve dígitos, dicho campo tampoco puede ser nulo.

Las llaves foráneas utilizadas en esta tabla son: "*compañía*", "*id_aeronave*", que ya han sido comentadas anteriormente, y se agregan dos llaves foráneas nuevas que son "*origen*" y "*destino*", utilizadas en la tabla "*vuelos*" para determinar el origen de cada vuelo así como su respectivo destino. Estas llaves foráneas se agregan a la tabla "*carga*" para llevar un seguimiento detallado del origen de la carga, el destino, el

número de guía, la compañía que se encuentra manipulando la carga, así como el identificador único de la aeronave en la que es transportada. Todo esto para tener una referencia exacta de la situación de la carga, que es as parte fundamental de esta aplicación.

Gráficamente podemos observar dicha tabla en la Figura 3.5.1.3

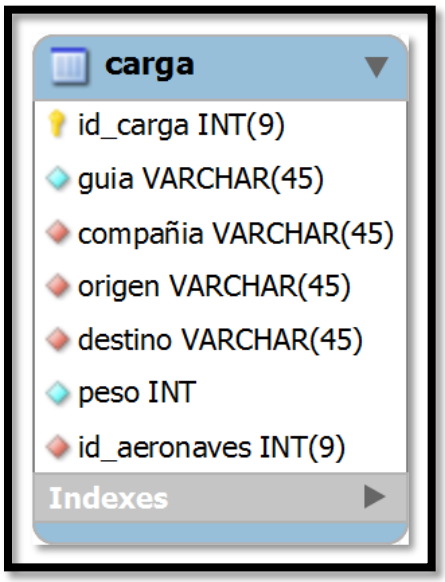



**Figura 3.5.1.3. Esquema grafico de la tabla Carga**



Observando el modelo lógico obtenemos una tabla de datos de la siguiente manera.

| id_carga | guia | compañia | origen | destino | peso | Id_aeronave |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |

Uno de los factores importantes a considerar respecto a la organización de un vuelo o a la gestión aeroportuaria es el combustible, un tema importante también de la aplicación desarrollada.

```
CREATE TABLE combustible(
id_combustible INT(9) NOT NULL AUTO_INCREMENT PRIMARY KEY,
fecha DATE NOT NULL,
litros INT(9) NOT NULL,
factura INT(9) NOT NULL,
id_aeronave INT(9) NOT NULL FOREIGN KEY
);
```

Un control adecuado respecto al combustible es uno de los principios fundamentales para la gestión aeroportuaria, esto nos permitirá saber con exactitud la cantidad de combustible utilizado, cuanto combustible se cargó en cada aeronave, y con esto se

obtiene el cálculo del centro de gravedad de cada aeronave, información que necesita el piloto de la aeronave antes de despegar. Por todo esto, fue necesario crear una tabla llamada "*combustible*" para facilitar la gestión del mismo. Al igual que todas las demás tablas cuenta con su identificador numérico de nueve dígitos ("*id_combustible*"), contiene el campo "*fecha*" con el tipo de dato DATE, que es especifico del lenguaje de programación SQL, y permite almacenar fechas en diferentes formatos, así como realizar operaciones o búsquedas más específicas en base a dicho parámetro.

El campo "*litros*" hace referencia a la capacidad en litros que un aeronave puede consumir, el valor nuevamente es numérico con una limitación de nueve dígitos, del mismo modo se encuentra el campo factura, el cual es numérico, con un límite de nueve dígitos que muestra el número de factura referente al combustible de un día o una aeronave en específico. Lo anterior íntegramente relacionado con la llave foránea "*id_aeronave*", que permite dar un seguimiento preciso del uso del combustible, además de saber la cantidad que consumió, que aeronave lo utilizó, cuándo se compró y el número de factura que demuestra dicha operación.

Gráficamente podemos observar dicha tabla en la Figura 3.5.1.4.

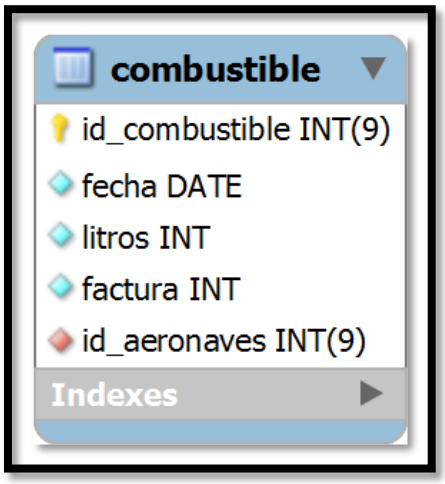



**Figura 3.5.1.4. Esquema grafico de la tabla Combustible**



Observando el modelo lógico obtenemos una tabla de datos de la siguiente manera.

| id_combustibe | fecha | litros | factura | id_aeronave |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |

Hasta el momento se han creado cuatro de las seis tablas que conforman la base de datos. La siguiente tabla creada es la de "*personal*", referida a los empleados encargados de la información alusiva a la carga de la aeronave.

El código para realizar esto se describe a continuación.

```
CREATE TABLE personal(
id_personal INT(9) NOT NULL AUTO_INCREMENT PRIMARY KEY,
nombre VARCHAR(45) NOT NULL,
cargo VARCHAR(45) NOT NULL,
compañia VARCHAR(45) NOT NULL FOREIGN KEY,
id_aeronave INT(9) NOT NULL FOREIGN KEY
);
```

En el código anterior se encuentra el identificador único que diferencia a un empleado (“*id_personal*”) para esta tabla en específico, además hallamos un campo específico para el nombre completo del empleado que maneja un tipo de dato VARCHAR de una longitud de 45 caracteres debido a que este registro incluye el nombre y ambos apellidos.

El “*cargo*” se encuentra almacenado en una columna homónima también, con un tipo de dato VARCHAR de 45 caracteres alfanuméricos. Del mismo modo incluye dos llaves foráneas, que son: “*compañía*”, la cual permite relacionar esta tabla con la tabla de “*aeronaves*”; “*id_aeropuerto*”, que relaciona directamente la tabla actual con la tabla “*aeropuerto*”, creada anteriormente.

Gráficamente podemos observar dicha tabla en la Figura 3.5.1.5.



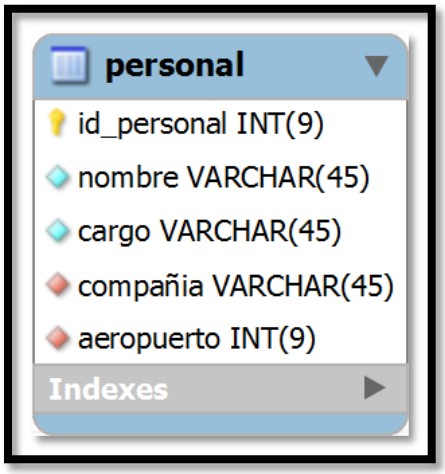



**Figura 3.5.1.5. Esquema grafico de la tabla Personal**

Observando el modelo lógico obtenemos una tabla de datos de la siguiente manera.

| id_personal | nombre | cargo | compañía | Id_aeropuerto |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |

La función de esta tabla (“*personal*”) es gestionar de forma sencilla la elaboración de la hoja de carga.

Por último, se crea la tabla “*vuelos*”, que maneja la información referente al origen y destino de cada vuelo, así como el identificador de la aeronave (”*id_aeronave*”) y el identificador de cada vuelo (“*id_aeronave*). El código utilizado para realizar esto es:

```
CREATE TABLE vuelos(
id_vuelo INT(9) NOT NULL AUTO_INCREMENT PRIMARY KEY,
origen VARCHAR(45) NOT NULL,
destino VARCHAR(45) NOT NULL FOREIGN KEY,
id_aeronave INT(9) NOT NULL FOREIGN KEY
);
```

La columna “*origen*” especifica el punto de partida de la aeronave, así como su destino. En ambos casos, se encuentran registrados dentro de la tabla con el mismo nombre, los aeropuertos autorizados para el despegue o descenso de la aeronave, que son ligados a su vez a la columna “*id_aeronave*”, la cual nos muestra el número de aeronave específica para cada vuelo. Así mismo, se tiene la columna “*id_vuelo*” que sirve como identificador de cada uno de los diferentes vuelos.

Gráficamente podemos observar dicha tabla en la Figura 3.5.1.6.




**Figura 3.5.1.6. Esquema grafico de la tabla Vuelos**

Observando el modelo lógico obtenemos una tabla de datos de la siguiente manera.

| id_vuelo | origen | destino | id_aeropuerto |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

Una vez que se crearon cada una de las tablas necesarias para la base de datos nuestro modelo queda como se muestra en la Figura 3.5.1.7.

En este modelo es posible observar cada una de las tablas en su forma gráfica, así como las relaciones inherentes entre ellas.

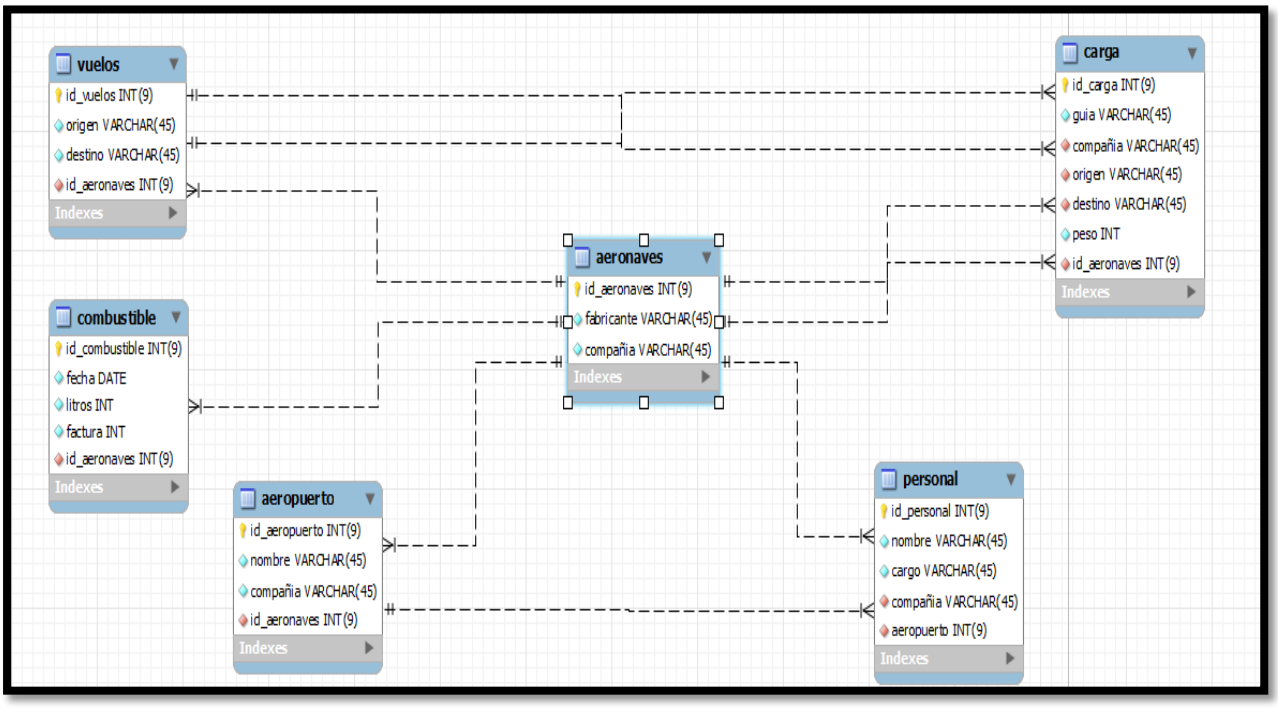



**Figura 3.5.1.7. Modelo de la base de datos de la aplicación**



Al tener la base de datos creada, se procede a la elaboración de la interface orientada a la gestión de los datos, es decir, a los formularios utilizados para ingresar datos.

## 3.5.2 Desarrollo de la capa de negocio de la aplicación

Antes de empezar, es necesario definir dos tipos de páginas en la aplicación, las páginas de negociación o proceso y las páginas de presentación o interface, las páginas de proceso, serán escritas exclusivamente en el lenguaje de programación php y se utilizan únicamente para el procesamiento y flujo de datos desde y hacia la base de datos.

Las páginas de interface están escritas en php además de código HTML el cual se desarrolla en el estándar XHTML, las cuales se encargaran única y exclusivamente de la estética de dicha interface, mientras que XHTML se utilizara para reducir la carga de procesamiento del servidor y facilitar la legibilidad del código e identificación de cada tipo de página.

Las páginas de proceso son:

- *db_conn.php*

- *process.php*

- *login.php*

A continuación se describe el desarrollo de las páginas de proceso

La página dedicada al procesamiento de esta información debe encontrarse en lenguaje PHP al menos para la aplicación en cuestión, ya que esto nos permitirá manipular la información enviada lejos de los ojos de cualquier curioso y aumentando la velocidad de procesamiento dado que esto se realiza del lado del servidor y el cliente solo interviene para enviar los datos.

Una vez que la información se encuentra del lado del servidor debe de ser procesada, lo que la mayor parte del tiempo significa revisar que dicha información no contenga código malicioso, y una vez que esto ha sido validado procedemos a realizar la conexión a la base de datos y a insertar debidamente cada dato en su respectiva tabla

Para solicitar información realizamos un procedimiento similar solo que en este caso lo único que se analiza es el patrón de búsqueda, una vez analizado dicho patrón procedemos a solicitar la información pertinente a la base de datos, formatear dicha información y enviarla a la maquina cliente como respuesta a una petición especifica.

Las conexiones a la base de datos se llevan a cabo por medio del lenguaje de programación PHP el cual cuenta con comandos específicos para llevar a cabo esta tarea. Dado que la conexión a la base de datos no va a ser permanente sino más bien intermitente lo más recomendable es crear un solo archivo con los datos de conexión y mandarlo llamar/ejecutar cada que sea necesario, este tipo de prácticas nos evita escribir demasiado código ya que esta rutina podrá ser llamada desde cualquier lugar dentro de la red empresarial.



A esta rutina le asignaremos el nombre de *db_conn.php* y en ella escribiremos el siguiente código.

```
<?php
function conectar(){
      $user = "nombre de usuario";
      $host = "localhost";
      $password = "clave de acceso";
      $database = "nombre de la base de datos";

      $conexion = pg_connect($host, $user, $password)
      or die ("Error al conectar a Postgre");

      $seleccionar = pg_connect($database, $conexion)
      or die ("Error al seleccionar la base de datos");
}
?>
```

En estas líneas de código se crean primero las variables correspondientes utilizadas para conectarnos a la base de datos, la variable user se refiere el nombre con el que se encuentra registrado el administrador de la base de datos, la variable host hace referencia a la ubicación del servidor de bases de datos y puede tomar como valores localhost si la base de datos se encuentra dentro del mismo servidor que la aplicación, en caso contrario contendrá la dirección del servidor de bases de datos remoto, la variable password contiene la contraseña de acceso asignada al usuario que declaramos anteriormente y por último la variable database contiene el nombre de la base de datos que vamos a utilizar.

Hecho esto procedemos de declarar dos variables nuevas, la primera es conectar que contiene la función *pg_connect*, la cual es una función específica de php que facilita la conexión a la base de datos y a la cual se le deben de asignar tres argumentos diferentes que son: ubicación del servidor de bases de datos, el usuario y la contraseña de acceso, como todo esto ya lo hemos convertido a variables lo único que necesitamos hacer en este caso en particular es colocar esas variables dentro de los paréntesis designados para dichos argumentos, y además agregamos las palabras reservadas *or die* e incluimos un mensaje de texto el cual se mostrara en caso de que se la función no se ejecute correctamente o exista algún error en los datos.

La siguiente variable que creamos es seleccionar la cual contiene la función *pg_select_db*, al igual que la función anterior es especifica de PHP y permite seleccionar la base de datos que vamos a utilizar en base a dos argumentos necesarios, el primero es el resultado de la función *pg_connect* que se encuentra en la variable conectar y el segundo argumento es el nombre de la base de datos que vamos a utilizar con esa conexión, nuevamente agregamos las palabras reservadas ***or die***, en este caso con un mensaje diferente.



Estos mensajes como se expuso anteriormente nos permite conocer en donde se produjo un error en caso de que este exista, por tanto podemos observar que el segundo mensaje hace referencia a un error de selección de base de datos, mientras que el primero hace referencia a un error de conexión, así dependiendo del tipo de error que lleguemos a observar seremos capaces de encontrarlo con mayor facilidad.

Se debe de ser muy cuidadoso al momento de manejar este tipo de archivos ya que en ellos se encuentra por lo menos una contraseña para el acceso a la base de datos lo cual permitiría a cualquier persona ajena que lo obtenga modificar la base de datos con consecuencias catastróficas.

Para evitar este tipo de errores lo más recomendable es colocar este archivo en un directorio ajeno a los archivos comunes de un *website*, dicho directorio solo debe de poder ser visto por el administrador del sistema.

Una vez creado este archivo procedemos a crear el archivo de procesamiento de datos al cual le asignaremos el nombre de *process.php* y contendrá las siguientes líneas de código.

```
<?php
include 'db_con.php';
extract($_POST);
conexion();
$insertar = "INSERT INTO red_aeroportuaria SET campo1
='$variable1', campo2 = '$variable2', campo3 = '$variable3', campo4
= '$variable4', campo5 = '$variable5'";
$consulta = pg_query($insertar);
unset($_POST);
?>
```

Podemos observar que en la primer línea del código declaramos la utilización del archivo que creamos anteriormente para conectarnos a la base de datos, después utilizamos la función *extract* la cual convierte el arreglo asociativo en variables independientes, una vez hecho esto ejecutamos la función conectar, la cual como vimos con anterioridad nos conecta a la base de datos y selecciona la tabla que utilizaremos para insertar los datos.

Una vez hecho esto procedemos a crear una nueva variable llamada en este caso *$peticion* la cual contendrá en forma de cadena una consulta SQL habitual de inserción de datos, podemos apreciar que dicha consulta se encuentra entre comilla doble como todos las cadenas en PHP la diferencia en este caso es que cada una de las variables se encuentra dentro de comillas simples, esto se lleva acabo de esta manera, debido a que al colocar comillas simples a una variable dentro de una cadena de caracteres automáticamente php convierte esas variables en su contenido, lo cual nos deja una consulta completamente en caracteres alfanuméricos sin importar que información contengan dichas variables, esto puede resultar un poco confuso pero después de analizar detenidamente la estructura de una consulta SQL y su interacción con PHP podemos notar la sencillez detrás de todo esto, ya que no necesitaremos escribir datos con cada consulta solo la variable y los datos que esta contenga serán ingresados directamente en la base de datos.



También podemos notar que en vez de utilizar una consulta típica de SQL se utiliza el comando *SET* esto se hace con la finalidad de clarificar el código fuente y en caso de que exista un error de SQL de datos podamos encontrarlo con mayor facilidad

Nuevamente se crea una nueva variable esta vez con el nombre de $consulta la cual contiene la función *pg_query* que al igual sus predecesoras es una función típica de PHP que nos facilita las transacciones con SQL y esta variable contendrá el resultado de dicha consulta.

Por último utilizamos la función *unset* la cual destruirá todo el contenido de la variable *$_POST* y comenzaremos a reciclar la memoria utilizada para este proceso.

Habiendo hecho esto podemos revisar la base de datos de ser necesario para confirmar la inserción de los datos en la tabla especificada y continuar construyendo la aplicación.

El siguiente paso en el desarrollo de la aplicación se enfoca en la capa de aplicación, es decir en la programación y procesamiento de datos realizado del lado del servidor, en el lenguaje de programación PHP.

Lo primero que debemos de fundamentar es un acceso restringido a la aplicación, esto se realiza mediante variables de sesión, las cuales nos permitirán limitar el acceso a secciones de la aplicación con las cuales solo deberán interactuar una cantidad reducida de usuarios.

Para realizar esta restricción de acceso utilizaremos un formulario de identificación, este contiene únicamente dos campos, el primero para el nombre de usuario y el segundo para la contraseña de dicho usuario, una vez que han sido introducidos estos datos se enviaran mediante el método *POST* a un script PHP llamado en este caso en particular *login.php* el cual hará uso del script anterior para conectarse a la base de datos y verificar que existan ese nombre de usuario y contraseña y que esta última coincida con el usuario que se pretende ingresar en el sistema, una vez confirmada la validez de estos datos se creara una variable de sesión la cual permitirá que dicho usuario acceda a las zonas restringidas libremente.

El script para realizar esto es el siguiente:

```
<?php
include 'db_con.php';
if (!$_POST){
      header("Location:index.php");
      exit();
}
else{
      extract($_POST);
      conectar();
      $consulta = "SELECT * FROM personal WHERE id_personal = '$id'
      AND password = '$password'";
      $busqueda = pg_query($consulta);
      if(!$row = pg_fetch_array($busqueda)){
            $error = "Los datos son incorrectos";
      }
      else{
            $_SESSION['id'] = $row['id_personal'];
            header("Location:registered.php");
            exit();
      }
}
?>
```



La explicación de este script es bastante simple aunque a primera vista luzca complicada.

En la línea 2 encontramos nuevamente la declaración del archivo de conexión a la base de datos, en la tercer hasta la séptima línea del script encontramos una sentencia condicional, la cual verifica que exista la variable *$_POST*, de lo contrario nos redirigirá

automáticamente a la página principal de la aplicación, además agregamos la función *exit* la cual se encargara de cancelar cualquier proceso que se esté ejecutando, esto con el fin de evitar que cualquier persona pueda acceder directamente a este script y generemos un agujero de seguridad.

El re direccionamiento del usuario se lleva a cabo mediante la función *header*, a la cual se le asigna como argumento la dirección hacia donde será re direccionado el usuario, en este caso una página de acceso común.

A partir de la octava línea del script encontramos el procedimiento a realizarse en caso de que exista la variable *$_POST* , lo primero que haremos nuevamente es extraer todas las variables que conforman dicho arreglo, una vez hecho esto, procedemos a conectarnos a la base de datos mediante la función *conectar()* y crearemos la consulta SQL la cual busca dentro de la tabla personal toda la información almacenada, donde el *id_personal* y el password sean los mismos que el que enviamos mediante el formulario.

Ya creada la consulta y guardada en una variable, esta última se pasa como argumento a la función *pg_query* la cual nuevamente almacenara el resultado de dicha consulta en una variable llamada *$busqueda* en esta ocasión.

A partir de este punto el script comienza a volverse un poco más complicado, ya que encontramos nuevamente una sentencia condicional, la cual verifica que exista la variable llamada *$row*, la cual contiene el resultado de la función *pg_fetch_array* a la que se le asigno como argumento la variable *$busqueda*.



La función *pg_fetch_array* toma los valores regresados por la base de datos y que se encuentran almacenados en la variable *$busqueda* y en base a dichos valores crea un arreglo asociativo, los cuales se almacenan en la variable *$row*, en caso de no existir estos valores se mostrara un mensaje de error donde se indica al usuario que sus datos ya sean id o contraseña no son válidos o no existen.

En caso contrario, procedemos a crear la variable de sesión *$_SESSION[‘id’] = $row[‘id_personal’]*, esta variable de sesión contendrá el valor ubicado en el arreglo *$row* con la clave o nombre id, y se asignara a la variable de sesión llamada también id, debemos de recordar que las variables de sesión también son arreglos asociativos y que cada una de esas asociaciones se conforman por pares *clave => valor*, lo cual nos permite de ser necesario crear una variable de sesión tan grande como deseemos, únicamente indicando la clave o claves donde iremos almacenando los demás datos.

Una vez creada la variable de sesión el usuario es redirigido automáticamente a la página *registered.php*, y eliminaremos cualquier proceso en ejecución una vez más mediante la función *exit( ).*

Una vez que hemos hecho esto, deberemos recordar colocar la función *session_start()* en la primer línea de cada una de las páginas que serán de acceso restringido, esta

función se utiliza para indicarle al servidor que en dicha página debe permitir el uso de variables de sesión.

La utilización de variables de sesión para restringir el acceso a determinadas secciones de la aplicación se basa en que si dicha sesión no es generada o la variable de sesión se encuentra vacía, el servidor nos redirigirá automáticamente a una página de acceso público, mediante la función *header*.

### 3.5.3 Desarrollo de las interfaces de la aplicación

Para realizar esto se utiliza el lenguaje de marcado XHTML en formato STRICT 1.0, esto con el fin de estandarizar la creación del sitio web que nos permitirá utilizar la aplicación para gestionar adecuadamente cada vuelo.

La ventaja más sobresaliente de XHTML, anteriormente descrita, es la estandarización del código así como la capacidad de validar que dicho código se apega a un estándar lo cual permite visualizar correctamente dicho sitio web independientemente del navegador que se utilice.

Al crear los formularios es necesario enfatizar y observar que, el método de envió, bien puede ser *POST* o *GET*, el atributo *action* contiene la dirección o nombre del archivo de procesamiento de la información y por ultimo preferentemente es necesario asignar a cada campo un nombre (*name*) referente al dato que se envía por él, puesto que al enviar esta información a la página de procesamiento en PHP se convierte en un arreglo de datos conocido como "*asociativo*" es decir, el nombre del campo pasa a ser el nombre de una variable y la información que se haya enviado mediante este campo resulta ser el contenido de dicha variable.



Cada uno de los formularios para ingresar datos es dirigido a una página dedicada exclusivamente al procesamiento de dicha información, por cuestiones de transparencia la información es enviada mediante el método *POST*, método que nos permite enviar información de un documento XHTML a cualquier sitio de internet sin que el usuario pueda modificar o alterar de forma sencilla dicha información.

El código del formulario "*aeronaves*" donde se hace el registro de las aeronaves, es el siguiente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<title>Untitled Document</title>
</head>
<body>
<form method="post" action="registro_aeronaves.php"
```

```
name="aeronaves">
      <table width="200" border="1" align="center">
      <tr>
            <td>Codigo Aeronave</td>
            <td><input type="text" name="id_aeronaves" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td>Fabricante:</td>
            <td><input type="text" name="fabricante" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td>Compañia</td>
            <td><input type="text" name="compañia" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td><input type="submit" value="enviar" /></td>
        </tr>
      </table>
</form>
</body>
</html>
```

El siguiente formulario es el de "*aeropuerto*" que genera el código a continuación escrito:



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<title>Untitled Document</title>
</head>
<body>
<form method="post" action="registro_aeropuerto.php"
name="aeronaves">
      <table width="200" border="1" align="center">
      <tr>
            <td>Codigo Aeropuerto:</td>
            <td><input type="text" name="id_aeropuerto" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td>Nombre:</td>
            <td><input type="text" name="nombre" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td>Compañia:</td>
            <td><input type="text" name="compañia" size="45"
            maxlength="45" /></td>
        </tr>
```

```
        <tr>
            <td>Codigo de Aeronave:</td>
            <td><input type="text" name="id_aeronaves" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td><input type="submit" value="enviar" /></td>
        </tr>
      </table>
</form>
</body>
</html>
```

El formulario "*carga*" está formado con el siguiente código HTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<title>Untitled Document</title>
</head>
<body>
<form method="post" action="registro_carga.php" name="aeronaves">
      <table width="200" border="1" align="center">
      <tr>
            <td>Codigo:</td>
            <td><input type="text" name="id_carga" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td>Guia:</td>
            <td><input type="text" name="guia" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td>Peso:</td>
            <td><input type="text" name="Peso" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td>Compañia:</td>
            <td><input type="text" name="compañia" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td>Origen:</td>
            <td><input type="text" name="origen" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td>Destino:</td>
            <td><input type="text" name="destino" size="45"
            maxlength="45" /></td>
```

```
        </tr>
        <tr>
            <td>Codigo de Aeronave:</td>
            <td><input type="text" name="id_aeronaves" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td><input type="submit" value="enviar" /></td>
        </tr>
      </table>
</form>
</body>
</html>
```

El control del combustible está dado por un formulario con el mismo nombre "*combustible*"

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<title>Untitled Document</title>
</head>
<body>
<form method="post" action="registro_combustible.php"
name="aeronaves">
      <table width="200" border="1" align="center">
      <tr>
            <td>Codigo:</td>
            <td><input type="text" name="id_carga" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td>Fecha:</td>
            <td><input type="text" name="fecha" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td>Litros:</td>
            <td><input type="text" name="Litros" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td>Numero de factura:</td>
            <td><input type="text" name="factura" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td>Codigo de Aeronave:</td>
            <td><input type="text" name="id_aeronaves" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
```

```
            <td><input type="submit" value="enviar" /></td>
        </tr>
      </table>
</form>
</body>
</html>
```

A continuación se muestra el código generado para el formulario del "*personal*":

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<title>Untitled Document</title>
</head>
<body>
<form method="post" action="registro_personal.php"
name="aeronaves">
      <table width="200" border="1" align="center">
      <tr>
            <td>Codigo:</td>
            <td><input type="text" name="id_carga" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td>Nombre:</td>
            <td><input type="text" name="nombre" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td>Cargo:</td>
            <td><input type="text" name="cargo" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td>Compañia:</td>
            <td><input type="text" name="compañia" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td>Codigo de Aeropuerto:</td>
            <td><input type="text" name="id_aeropuerto" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td><input type="submit" value="enviar" /></td>
        </tr>
      </table>
</form>
</body>
</html>
```

Se termina la elaboración de los formularios, con la creación del código HTML para el formulario de "*vuelos*":

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<title>Untitled Document</title>
</head>
<body>
<form method="post" action="registro_vuelos.php" name="aeronaves">
      <table width="200" border="1" align="center">
      <tr>
            <td>Codigo de Vuelo:</td>
            <td><input type="text" name="id_carga" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td>Origen:</td>
            <td><input type="text" name="origen" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td>Destino:</td>
            <td><input type="text" name="destino" size="45"
            maxlength="45" /></td>
        </tr>
        <tr>
            <td>Codigo de Aeronave:</td>
            <td><input type="text" name="id_aeronaves" size="9"
            maxlength="9" /></td>
        </tr>
        <tr>
            <td><input type="submit" value="enviar" /></td>
        </tr>
      </table>
</form>
</body>
</html>
```

# Conclusiones Generales y Aportaciones

## Conclusiones Generales

Mediante el análisis del desarrollo de un sistema de gestión, es posible determinar cuáles son los procesos operativos que requieren de un análisis exhaustivo con la finalidad de optimizarlos y simplificarlos.

Toda vez que estos procesos hayan sido actualizados, su implementación mediante un sistema informático, provee a la empresa, la capacidad de cumplir con estos procesos adecuadamente, además de brindar a los empleados una herramienta que les permita entre otras cosas simplificar sus labores y desarrollar sus operaciones con mayor eficiencia bajo un estándar que beneficia a todos y cada uno de los departamentos y personas que integran el sistema de gestión.

El contar con información, en tiempo real, fiable y precisa, en el momento en el que se está brindando el servicio, permite incrementar la velocidad con la que dichos procesos son realizados, incrementando a su vez la cantidad de servicios brindados, lo cual se verá reflejado en la calidad y cantidad de servicios, la disminución de errores durante los mismos y finalmente, en una perspectiva a mediano plazo, la eliminación de costos de operación innecesarios.



## Aportaciones

Este sistema de gestión, aportará a la empresa, una infra estructura tecnológica de bajo costo, lo cual es posible, debido principalmente al software utilizado y que se encuentra desarrollado bajo una perspectiva de licenciamiento Open Source, el cual implica que no existe un gasto económico en el mismo, reduciendo el costo del sistema, única y exclusivamente a la mano de obra del sistema y los requisitos de hardware para su implementación.

Adicionalmente el sistema cuenta con un gestor de bases de datos, con tecnología de punta, equiparable al gestor de bases de datos Oracle y sin embargo se encuentra libre del enorme gasto económico que Oracle presenta, al ser un software cuya licencia más simple y limitada tiene un costo de 70 euros.

Con las herramientas computacionales ocupadas en el desarrollo del sistema será posible obtener una mayor eficiencia en la comunicación y el intercambio de

información entre departamentos o personal que se encuentra laborando dentro de la red aeroportuaria y las oficinas corporativas.

Con el desarrollo de la Aplicación para una Red Aeroportuaria, se obtendrán resultados evidentes en las tareas que se realizan, como son:

- Archivar formulario de tramitación.
- Al ser archivado el formulario los procedimientos o revisiones de peso y balance, se obtienen de forma exitosa y más confiable.
- Documentar la tarea. Llenar y completar toda la documentación en el archivo del Operador/solicitante.

# Bibliografía

**Apache.** (2013, Febrero 20). Apache HTTP Server Project. [Online]. Disponible: http://httpd.apache.org/ABOUT_APACHE.html

**Azundris.** (2013, Febrero 24). MySQL Dump, My Life with MySQL. [Online]. Disponible: http://mysqldump.azundris.com/uploads/mysql-workbench.png

**Campos Paré, R., Casillas Santillán, L. A., Costal Costa, D., Gibert Ginestá, M., Martín Escofet, C., y Perez Mora, O.** (2005). Bases de Datos (1ª Edicion). España: Formación de Posgrado.

**Codd, E.** (1970). A Relational Model of Data for Large Shared Data Banks. USA: P. Baxendale.

**Coulouris, G., Dollimore, J., y Kindberg, T.** (2001). Sistemas Distribuidos: Conceptos y Diseño (1ª Edicion). España: Addison Wesley.

**DataPrix.** (2013, Marzo 25). DataPrix, Knowledge Is The Goal. [Online]. Disponible: http://www.dataprix.com/8-cliente-grafico-pgadmin3



**Ecured.** (2013, Marzo 25). EcuRed Conocimiento con todos y para todos. [Online]. Disponible: http://www.ecured.cu/index.php/PgAdmin3

**Elmasri, R., y B. Navathe, S.** (1997). Sistemas de Bases de Datos - Conceptos Fundamentales (2ª Edicion). Mexico D.F.: Addison - Wesley Iberoamericana.

**Foundation, A. S.** (2013, Febrero 20). Apache HTTP Server Project. [Online]. Disponible: de http://httpd.apache.org/

**Gourley, D., y Totty, B.** (2002). HTTP The Definitive Guide (1ª Edicion). USA: O'Reailly Media.

**Group, N. W.** (2013, Febrero 15). The Internet Engineering Task Force. [Online]. Disponible: http://www.ietf.org/rfc/rfc2616.txt

**Group, P. G.** (2013, Marzo 24). PostgreSQL, The world's most advanced open source database. [Online]. Disponible: http://www.postgresql.org/docs/current/static/plpgsql-overview.html

**Group, T. P.** (2013, Febrero 15). PHP. [Online]. Disponible: http://us2.php.net/history

**Group, T. P.** (2013, Febrero 25). PostgreSQL. [Online]. Disponible: http://www.postgresql.org/about/history/

**Lamarca Lapuente, M. J.** (2013, Marzo 24). Hipertexto: El nuevo concepto de documento en la cultura de la imagen. [Online]. Disponible: http://www.hipertexto.info/documentos/b_datos.htm

**Lujan Mora, S.** (2002). Programación de Aplicaciones Web: Historia, Principios Básicos y Clientes Web (1ª Edicion). España: Club Universitario.

**M. Brandon, D.** (2008). Software Engineering for Modern Web Applications: Methodologies and Technologies (1ª Edicion). USA: Information Science Reference.

**Machado, A., & Gil, J. C.** (2013, Marzo 23). BASESDEDATOS'S WEBLOG. [Online]. Disponible: de http://basesdedatos.wordpress.com/modelos-logicos-basados-en-registros/

**Marshall, J.** (2013, Febrero 15). HTTP Made Really Easy. [Online]. Disponible: http://www.jmarshall.com/easy/http/

**Martínez, R.** (2013, Marzo 25). PostgreSQL - es, Portal en español sobre PostgreSQL. [Online]. Disponible: http://www.postgresql.org.es/sobre_postgresql

**Netcraft.** (2013, Febrero 20). Netcraft. [Online]. Disponible: http://news.netcraft.com/archives/2013/05/03/may-2013-web-server-survey.html

**Netcraft.** (2013, Febrero 19). php.net. [Online]. Disponible: http://php.net/usage.php

**Niq.** (2013, Febrero 20). Httpd Wiki. [Online]. Disponible: http://wiki.apache.org/httpd/FAQ

**Nolasco Calderón, L., y Huerta Ramírez, N.** (2013, Abril 30). SG Software Gurú. [Online]. Disponible: http://sg.com.mx/content/view/222

**Obe, R., y Hsu, L.** (2012). PostgreSQL Up and Running (1ª Edicion). USA: O'Reilly Media.

**pgAdmin.** (2013, Marzo 25). pgAdmin, PostgreSQL Tools. [Online]. Disponible: http://www.pgadmin.org/

**PHP.** (2013, Febrero 18). PHP. [Online]. Disponible: http://us.php.net/manual/en/faq.general.php

**Pressman, R. S.** (2002). Ingeniería del Software, Un Enfoque Practico (5ª Edicion). España: MC Graw Hill.

**Sánchez, C.** (2013, Febrero 24). ONess Project. [Online]. Disponible: http://oness.sourceforge.net/proyecto/html/ch03s02.html

**Sánchez, J.** (2004). Principios Sobre Bases de Datos Relacionales. España: Independiente.

**Silberschatz, A., F. Korth, H., y Surdarshan, S.** (2002). Fundamentos de Bases de Datos (4ª Edicion). España: MC Graw Hill.

**Tanenbaum, A.** (2003). Redes de Computadoras (4ª Edicion). Mexico: Pearson Prentice Hall.

**w3.org.** (2013, Febrero 15). W3C. [Online]. Disponible: http://www.w3.org/protocols