1.1 MARCO CONTEXTUAL

Actualmente el Sistema Nacional de Educación Superior Tecnológica está integrado por la D.G.E.S.T. y 77 planteles. Entre ellos se encuentra el Instituto Tecnológico de Jiquilpan.

El ITJ inicia sus actividades el 14 de Febrero de 1977, ubicando sobre la carretera nacional S/N Km. 202 en la ciudad de Jiquilpan Michoacán, con código postal 59510 y el teléfono 01(353)53 3-0237.

El ITJ hace de las áreas tecnológicas y administrativas una excelencia educativa, buscando así la integración a las actividades socioeconómicas de la comunidad. Consolidando su prestigio como máxima casa de estudios de la región, para así mismo ampliar la cobertura educativa en su zona de influencia; además incrementa el desarrollo de la investigación científica y tecnológica, atendiendo las necesidades y prioridades tanto locales como regionales, entre otras.

Los departamentos que conforman a la institución son:

1- Gestión Tecnológica y Vinculación
2- Planeación, Programación y Presupuestación
3- Comunicación y Difusión
4- Actividades Extraescolares
5- Servicios Escolares
6- Centro de Información
7- Recursos Humanos
8- Recursos Financieros
9- Recursos Materiales y Servicios
10- Centro de Cómputo

11-Ingeniería Industrial
12-Ingeniería Química y Bioquímica
13-Ciencias Básicas
14-Ciencias Económico-Administrativas
15-Sistemas y Computación
16-Desarrollo Académico
17-División de Estudios Profesionales
18-Ciencias de la Tierra

El esquema organizacional del ITJ se divide en una Dirección, 3 subdirecciones y 18 departamentos.

Dirección: su función es administrar la presentación de los servicios educativos de la educación tecnológica conforme a la D.G.E.S.T. (Ver figura 1.1)

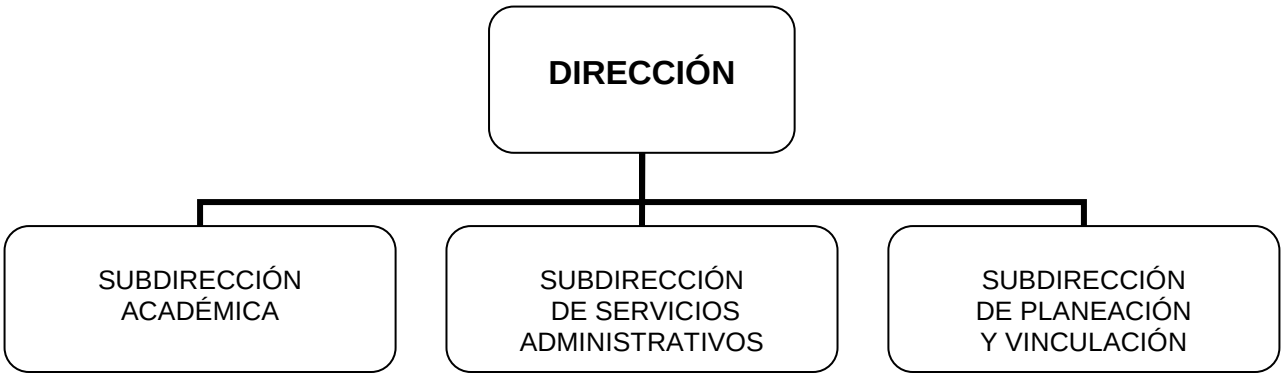

Figura 1.1_ Dirección con sus respectivas Subdirecciones

Las tres subdirecciones son:

Subdirección de Planeación y Vinculación que coordina, supervisa y evalúa todas las actividades realizadas en el plantel, lo cual favorece al desempeño de los departamentos que están a su cargo. (Ver figura 1.2)

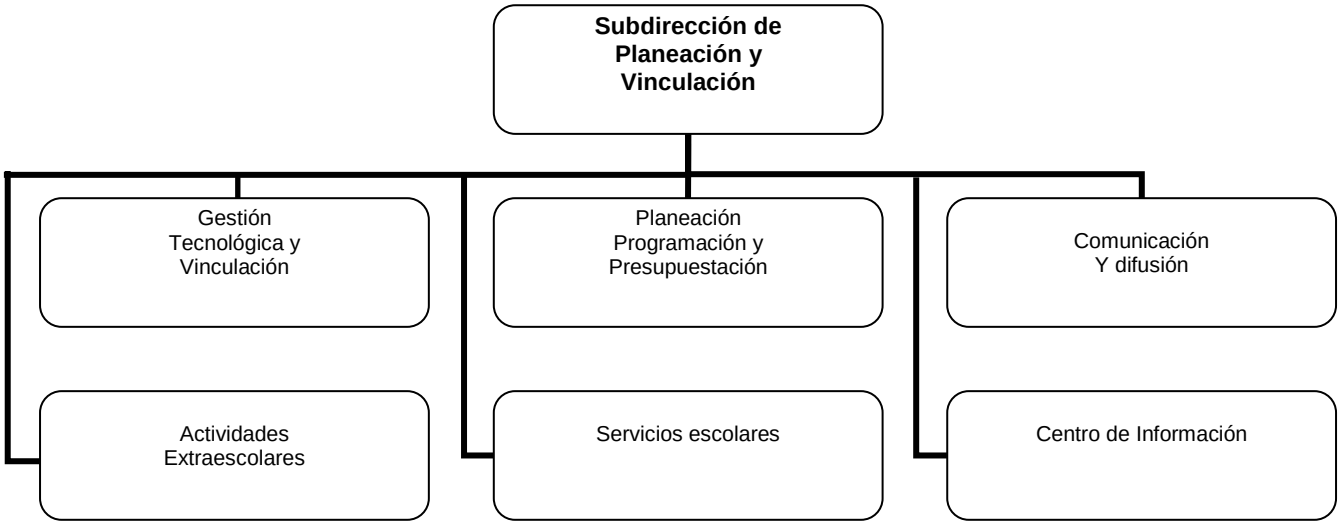

Figura 1.2_ Subdirección de Planeación y Vinculación

Subdirección Administrativa, verifica el cumplimiento de los procedimientos establecidos para la administración de los recursos necesarios para la operación del Instituto en conjunto con sus departamentos. (Ver Figura 1.3)

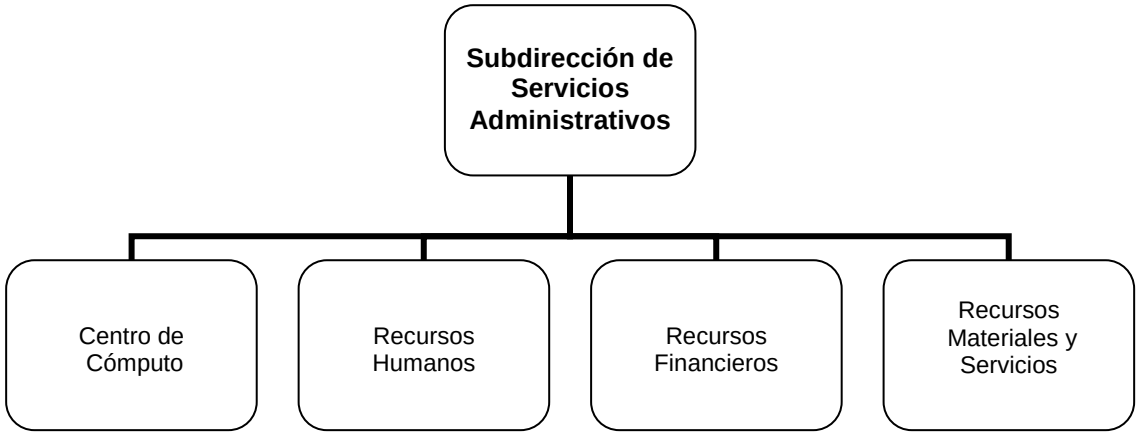

Figura 1.3_ Subdirección de Servicios Administrativos

Subdirección Académica coordina las actividades de docencia, investigación y vinculación del ITJ, así como los estudios profesionales que ofrecen los departamentos que componen esta subdirección. (Ver figura 1.4)

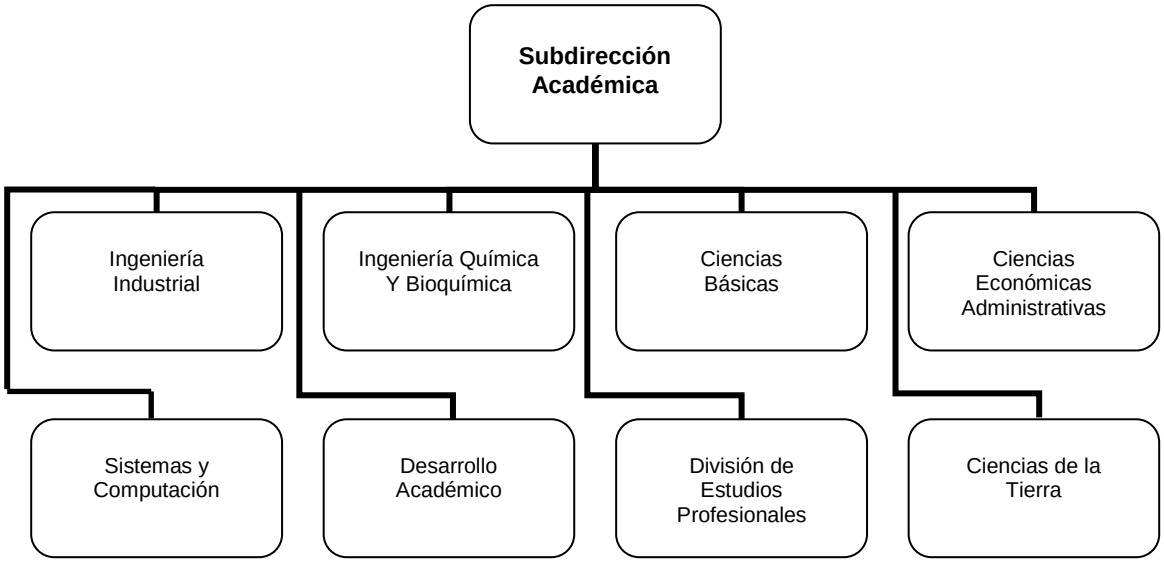

Figura 1.4_ Subdirección Académica

1.2 ESTRATEGIAS PARA EL DESARROLLO DEL SISTEMA

Los sistemas de información basados en computadoras sirven para diversas finalidades que van desde el procesamiento de las transacciones de una empresa, hasta proveer de la información necesaria para decidir sobre asuntos que se presentan con frecuencia. En algunos casos los factores que deben considerarse en un proyecto de sistemas de información son: la tecnología de comunicaciones que se va a utilizar, el impacto del nuevo sistema sobre los docentes de la institución y las características específicas que el sistema debe tener, todas estas situaciones están representadas por tres distintos enfoques para el desarrollo de los sistemas de información basados en computadoras:

1.- Método del ciclo de vida para el desarrollo de sistemas.

2.- Método del desarrollo del análisis estructurado.

3.- Método del prototipo de sistemas.

Esto tiene como finalidad explorar cada enfoque, abordando las características del método y las condiciones bajo las que es probable que se obtenga el mayor beneficio para la institución.

1.2.1 CICLO DE VIDA CLÁSICO PARA EL DESARROLLO DE SISTEMAS

El desarrollo de sistemas, es un proceso formado por las etapas de análisis y diseño, comienza cuando la administración o algunos miembros del personal encargado de desarrollar sistemas, detectan un sistema dentro de la institución que necesita mejoras. El método del ciclo de vida para el desarrollo de sistemas (SDLC), es el conjunto de actividades que los analistas, diseñadores y usuarios llevan a cabo para desarrollar e implantar un sistema de información, este método se compone por 6 fases que son:

1.- Investigación preliminar

2.- Determinación de los requerimientos del sistema

3.- Diseño del sistema

4.- Desarrollo del software

5.- Pruebas de los sistemas

6.- Implantación y evaluación

Un aspecto fundamental en el método clásico es comprender todas las facetas importantes del sistema que se encuentra bajo estudio, para ello es necesario seguir una secuencia lógica durante la ejecución de este proceso.

Dentro del método clásico la estrategia a seguir es de lo general a lo particular, lo cual permite enfocar el sistema como un usuario final y como el usuario de la parte integral, para lo cual se ilustra el algoritmo a bloques de un sistema como tal. (Ver figura 1.5)

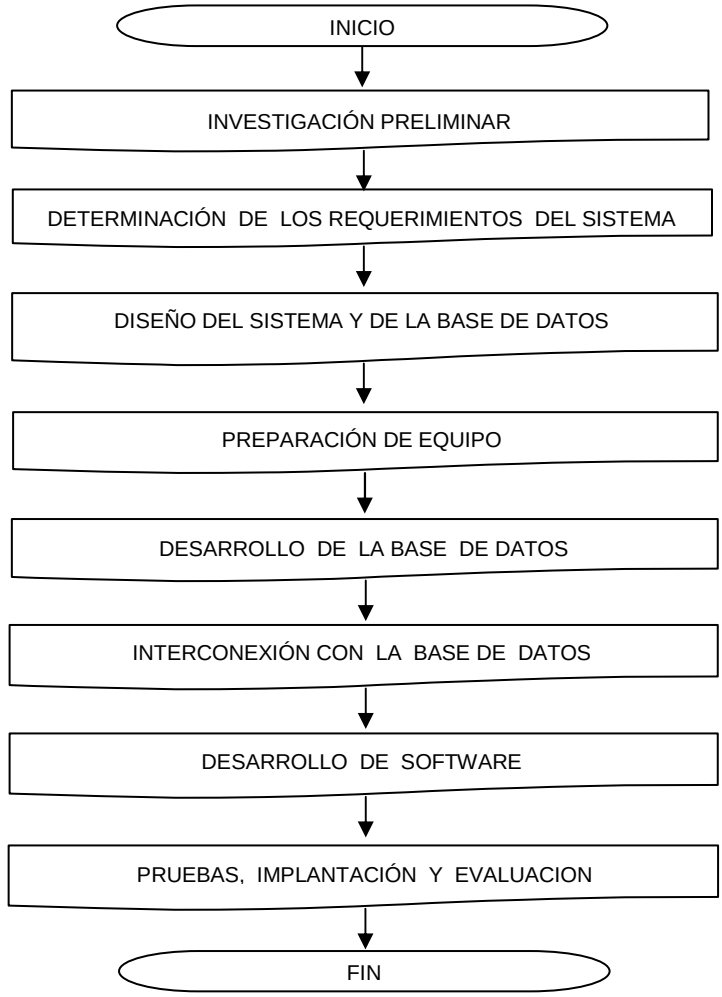

Figura 1.5_ Diagrama General de bloques del Desarrollo de Sistemas.

El diagrama de la figura 1.5, ejemplifica el desarrollo de un sistema, como se observa este consta de diversas actividades.

La investigación preliminar, consiste en el estudio y análisis de las características que determina el sistema, con el fin de conocer los requerimientos que presentan actualmente, lo cual nos permite tener los principales factores para el desarrollo del mismo.

Posteriormente se analizan los requerimientos del sistema así como los procesos que se realizan, con el fin de conocer el funcionamiento y flujo de información. Con base en dichos elementos se elabora el diseño de la Base de Datos, una vez realizado se comienza con la preparación del equipo de cómputo para el desarrollo del software.

La siguiente actividad a realizar es la creación de la base de datos con base en el diseño desarrollado anteriormente; posteriormente se genera el proceso de interconexión a la base de datos por medio de código generado sobre el leguaje elegido para ello. Partiendo del punto anterior se lleva a cabo el desarrollo de los módulos de programación necesarios para la manipulación de la información contenida en la base de datos; después se prosigue con las interfases y estructuras generales de emisión de la información para con ello concluir con el desarrollo del software.

Las pruebas de implantación son la última actividad para el desarrollo del Sistema, estas se realizan con base al funcionamiento de la organización que requiere el sistema. Estas pruebas se realizan con la finalidad de confirmar y respaldar la adaptabilidad de implantación del sistema.

1.2.2 MÉTODO DE DESARROLLO POR ANÁLISIS ESTRUCTURADO

Muchos especialistas en sistemas de información reconocen la dificultad de comprender de manera completa sistemas grandes y complejos. Este método tiene como finalidad superar esta dificultad por medio de:

- La división del sistema en componentes y
- La construcción de un modelo del sistema

Este incorpora elementos tanto de análisis como de sistemas, es por ello que el análisis estructurado se concentra en especificar lo que se requiere que realice el sistema o la aplicación. No establece cómo se cumplirán los requerimientos o la forma en que se implantará la aplicación, mas bien permite que observen los elementos lógicos es decir que hará el sistema, separados de los componentes físicos (PC´s, terminales, Sistemas de almacenamiento, etc.) Después de esto se puede desarrollar un diseño físico eficiente para la situación donde será utilizado.

1.2.3 MÉTODO DEL PROTOTIPO DE SISTEMAS

Este método hace que el usuario participe de manera más directa en la experiencia de análisis y diseño que cualquiera de los ya presentados (ciclo de vida del desarrollo de sistemas y análisis estructurado). Tal como se indicó, la construcción de prototipos es muy eficaz bajo las circunstancias correctas. Sin embargo, al igual que los otros métodos, el método es útil sólo si se emplea en el momento adecuado y en la forma apropiada.

Además los prototipos permiten evaluar situaciones extraordinarias donde los encargados de diseñar e implementar sistemas no tienen información ni experiencia, donde existen situaciones de riesgo o costos elevados y aquellas donde el diseño propuesto es novedoso y aun no ha sido probado.

En realidad, el prototipo es un modelo piloto o de prueba; el diseño evoluciona con el uso. Y auque el prototipo es un sistema que funciona, esta diseñado para ser modificado con facilidad. En algunos casos, aquellos donde el sistema será utilizado con poca frecuencia, el prototipo puede, de hecho, convertirse en el sistema terminado. Una vez que existe acuerdo en los requerimientos o diseños formulados, el sistema puede ser reprogramado para alcanzar mayor rapidez en su ejecución o para tener todas las características deseadas.

1.2.4 MÉTODO DEL ESPIRAL

El modelo espiral para la ingeniería de software ha sido desarrollado para cubrir las mejores características tanto del ciclo de vida clásico, como de la creación de prototipos, añadiendo al mismo tiempo un nuevo elemento: el análisis de riesgo. El modelo representado mediante:

1. Planificación: determinación de objetivos, alternativas y restricciones.
2. Análisis de riesgo: análisis de alternativas e identificación/resolución de riesgos.
3. Ingeniería: desarrollo del producto del "siguiente nivel",
4. Evaluación del cliente: Valorización de los resultados de la ingeniería.

El paradigma del modelo en espiral para la ingeniería de software es actualmente el enfoque más realista para el desarrollo de software y de sistemas a gran escala. Utiliza un enfoque evolutivo para la ingeniería de software, permitiendo al desarrollador y al cliente entender y reaccionar a los riesgos en cada nivel evolutivo. Utiliza la creación de prototipos como un mecanismo de reducción de riesgo, pero lo que es más importante, permite a quien lo desarrolla aplicar el enfoque de creación de prototipos en cualquier etapa de la evolución de prototipos.

1.3 SISTEMAS OPERATIVOS

1.3.1 WINDOWS

En 1985 Microsoft lanzó Windows, un sistema operativo que ampliaba las prestaciones de MS-DOS e incorporaba por primera vez una interfaz gráfica de usuario. Windows 2.0, que salió a la venta en 1987, mejoraba el rendimiento y ofrecía un nuevo aspecto visual. Tres años más tarde apareció una nueva versión, Windows 3.0, a la que siguieron Windows 3.1 y 3.11. Estas versiones, que ya venían preinstaladas en la mayoría de los equipos, se convirtieron rápidamente en los sistemas operativos más utilizados de todo el mundo. En 1990 Microsoft pasó a ser la empresa líder de programas informáticos.

Windows es un sistema que aprovecha la potencia de los procesadores, ha sido diseñado para adaptarse a las nuevas tecnologías, ofrece compatibilidad con varias plataformas (OS/2, Unix y versiones anteriores a él mismo), soporta el multiprocesamiento simétrico, buen rendimiento y conectividad, seguridad y al no estar encasillado en ningún modelo estándar de Sistema Operativo tiene la capacidad de combinar las ventajas del modelo cliente/servidor, puede correr además sobre múltiples arquitecturas con un mínimo de cambios, permite que varios procesos sean ejecutados simultáneamente en varios procesadores y estos no se apropien de recursos del sistema por tiempo indefinido, sino por tratamiento del sistema.

1.3.1.1 BENEFICIOS DE WINDOWS

El ambiente operativo Windows ofrece los siguientes beneficios:

- Corre sobre múltiples arquitecturas de hardware y plataformas.

- Es compatible con aplicaciones hechas en plataformas anteriores, es decir que corrieran la mayoría de las aplicaciones existentes hechas sobre versiones anteriores a la actual.
- Reúne los requisitos gubernamentales para POSIX (Portable Operating System Interface for Unix).
- Reúne los requisitos de la industria y del gobierno para la seguridad del Sistema Operativo.
- Fácilmente adaptable al mercado global soportando código Unicode.
- Es un sistema que corre y balancea los procesos de forma paralela en varios procesadores a la vez.
- Sistema Operativo de memoria virtual.

Comparte con el resto de los Sistemas Operativos avanzados, múltiples tareas, las cuales están asociadas a los modos actuales soportados por los microprocesadores. Estos modos proporcionan a los programas que corren dentro de ellos, diferentes niveles de privilegios para acceder al hardware o a otros programas que están corriendo en el sistema. Además de que Windows usa un modo privilegiado (Kernel) y un modo no privilegiado (Usuario).

1.3.2 UNIX

El sistema operativo Unix tiene su origen en los laboratorios Bell de AT&T en los años 60'^{S}, los cuales trabajaban en un sistema operativo nuevo llamado MULTICS (Multiplexed Information and Computing System). Este proyecto fue un fracaso, pero los integrantes del equipo adquirieron una gran experiencia durante su desarrollo.

Uno de los integrantes del equipo, Kem Thompson, escribió un juego llamado "Space Travel" y desarrolló un sistema operativo para poder jugar con él, consiguió que dos personas pudieran jugar simultáneamente con este sistema

operativo, que por un juego de palabras en comparación con MULTICS, lo llamó UNICS.

Este sistema UNICS estaba escrito en ensamblador, lo que dificultaba que se pudiera usar en máquinas con distintos procesadores. Viendo el problema, Kem Thompson y Denis Ritchie crearon un lenguaje de programación de alto nivel, el lenguaje “C”, en el cual reescribieron todo el sistema operativo lo que permitió que se pudiera usar en prácticamente cualquier tipo de ordenador de la época. Sólo las partes críticas seguían en ensamblador.

Más tarde un decisión judicial obligó a AT&T a dejar de vender su sistema operativo. Esta compañía dejó las fuentes del sistema operativo a diversas universidades, las cuales junto con otras empresas, continuaron el desarrollo del sistema operativo Unix e hicieron que tuviera una enorme difusión. Aunque Unix todavía lo poseía AT&T, la compañía hizo poco comercialmente con él hasta mediados de la década de 1980.

Existen muchas versiones de Unix, ya que diversas empresas han desarrollado sus propios sistemas operativos basados en el, por citar algunos: AIX, HPUX, Solaris, SunOS, IRIX, Xenix, SCO, FreeBSD, Linux.

1.3.3 SOLARIS

Sistema operativo de la empresa Sun Microsystems basado inicialmente en el sistema UNIX BSD de la Universidad de Berkeley, del cual uno de sus fundadores fue programador en sus tiempos universitarios.

Más adelante incorporó funcionalidades del System V, convirtiéndose prácticamente en un sistema operativo totalmente basado en System V.

Quizá sea uno de los UNIX comerciales más usados, principalmente en el entorno Internet. Solaris es una evolución del sistema anterior SunOS de la compañía.

Solaris funciona principalmente sobre la arquitectura SPARC en 32 y 64 bits (esta última conocida como UltraSparc) de la misma compañía y sobre la arquitectura Intel, aunque ésta se suele usar con fines didácticos, y escasamente en entornos de producción. Proporcionó desde sus primeros momentos un excelente soporte para aplicaciones de red basados en protocolos IP, y fue el primer entorno donde se desarrolló el sistema Java, donde hasta la actualidad tiene un excelente rendimiento.

Aporta prácticamente todas las funcionalidades típicas de los sistemas UNIX en entorno servidor, como Sockets , Multitarea, Threads, entorno de ventanas basado en X-Windows en el que se pueden usar diferentes escritorios como Open Look CDE o más recientemente GNOME.

En los últimos tiempos la compañía ha puesto en marcha una clara estrategia de acercamiento entre Solaris y Linux desarrollando productos que permiten ejecutar programas de Linux en Solaris.

El Ambiente Operativo Solaris 8 es un sistema operativo líder consagrado, el cual ofrece sólida estabilidad y previsibilidad. Establece estándares, proporcionando a los ambientes multiplataforma de misión crítica la confiabilidad, disponibilidad, agilidad, servicio tipo mainframe y del cómputo de 64 bits, a una fracción del costo de un mainframe.

La estructura modular del Ambiente Operativo Solaris 8 permite la instalación de nuevas características a medida que se encuentran disponibles. Con el Sistema Operativo Solaris 8, Sun eleva la universalidad a un nivel más alto,

ofreciendo tecnologías que pueden conectar a miles de millones de usuarios y dispositivos a través de Internet.

1.3.3.1 BENEFICIOS DE SOLARIS 8

El ambiente operativo Solaris 8 ofrece los siguientes beneficios:

Seguridad: IPSec, autenticación Kerberos v5, tarjetas inteligentes, control de acceso basado en funciones, un sistema más sólido y los eventos de auditoria de usuarios centralizados ofrecen la seguridad que exigen los ambientes punto-com.

Alta disponibilidad: Actualización y diagnóstico mientras el sistema está activo, registro del sistema de archivos, análisis previsible de fallas, reconfiguración dinámica y clustering que proporcionan alta disponibilidad.

Escalabilidad: El Ambiente Operativo Solaris 8 es escalable desde una PC hasta una supercomputadora, ofreciendo soporte de 1 a más de 1.000 procesadores, clusters de 2, 4 y 8 nodos, ambientes de 32 y 64 bits, el nuevo protocolo de direcciones Ipv6 y multithreading avanzado.

Interoperabilidad: Solaris 8 soporta Arquitectura Intel y las plataformas SPARC™, integrando servicios de Windows NT, compatibilidad con Linux, conectividad transparente entre múltiples plataformas, compatibilidad binaria comprobada y soporte para aplicaciones existentes.

Facilidad de uso: Las herramientas basadas en navegadores Web, el protocolo de localización de servicios, las capacidades de consola remota, la arquitectura hot desk, el soporte Plug-and-Play para los dispositivos del cliente y el

ambiente común de PC, proporcionan una instalación y administración más fáciles y económicas.

Universal: Solaris 8 es un sistema operativo verdaderamente universal, proyectándose como óptimo para la Plataforma Java™ 2, ediciones Standard y Enterprise, con soporte para 37 idiomas, la moneda Euro y ofreciendo conectividad entre múltiples plataformas.

Agilidad: Este sistema operativo proporciona la agilidad punto-com, al ofrecer numerosas funciones que hacen más eficientes las tareas de desarrollo, incluyendo el control de nomenclatura de los archivos principales, la capacidad para crear aplicaciones de administración remota a través de la Web que se ejecutan en navegadores Web estándares, agentes para simplificar la instalación y administración de las aplicaciones nativas de Solaris y Java.

Gratuito: Para promover la innovación, Sun no cobrará por la licencia de derecho para uso del software en sistemas con ocho o menos procesadores.

1.4 LENGUAJES PARA WEB

1.4.1 HTML

Después de la creación de Internet y la creciente demanda que este presentaba se requerían nuevas aplicaciones, nuevas formas de comunicar diferentes equipos informáticos. Así que en CERN, la NSCA y diversos organismos pusieron manos a la obra y para poder dar formato a los datos presentes en el documento Web, se desarrolló un lenguaje específico, HTML (Hyper Text Markup Language ó Lenguaje de Marcas de Hiper Texto), que permitía asignar un formato especial de presentación a los elementos del documento contenidos entre unas etiquetas especiales, denominadas marcas o tags.

1.4.2 ¿QUÉ ES EL HTML?

HTML es un lenguaje sencillo, pensado para presentar información en la World Wide Web. HTML (Hyper Text Markup Language), como su nombre indica es un lenguaje de marcas para la creación de hipertextos. Por hipertexto entenderemos texto con una presentación agradable, con inclusión de elementos multimedia (gráficos, video, audio) y con la presencia de hiperenlaces que permiten relacionar otras fuentes de información en documentos hipertextos.
Las marcas del lenguaje HTML especifican:

- La estructura lógica del documento:
  - Cabeceras, tipos y tamaños de las fuentes.
  - Párrafos de texto.
  - Centrado.
  - Enumeraciones o listas.
  - Formularios.
  - Tablas.

- Distintos estilos que definen el texto:
  - Negrita.
  - Cursiva.
  - Diferentes efectos: (direcciones de correo, citas textuales, etc.).
- Inclusión de hipertextos para acceder a otros documentos relacionados.
- Inclusión de imágenes y ficheros multimedia.

Por el momento no existe un estándar de HTML, ya que tanto Netscape como Microsoft se empeñan en incluir directivas ó etiquetas que solo funcionan con sus navegadores, de cualquier manera existen diferentes revisiones o niveles de estandarización, el 1.0, el 2.0 y el 3.0, lo que produce que algunos navegadores no comprendan en su totalidad el contenido de un documento.

1.4.3 ETIQUETAS DEL LENGUAJE HTML

El lenguaje HTML es un lenguaje de marcas, las cuales son fragmentos de texto destacado de una forma especial que permiten la definición de las distintas instrucciones de HTML, tanto los efectos a aplicar sobre el texto como las estructuras del lenguaje. A dichas marcas se les denomina etiquetas y son la base principal del lenguaje HTML. El documento HTML es un fichero texto con etiquetas que variarán la forma de su presentación.

Una etiqueta será un texto incluido entre los símbolos menor que < y mayor que >. El texto incluido dentro de los símbolos será explicativo de la utilidad de la etiqueta. Por ejemplo:

<B> Letra Negrita, del inglés Bold (negrita).

<TABLE> Definirá una tabla.

<IMG> Inclusión de una imagen.

Existen normalmente dos tipos de etiquetas:

* Abiertas: Las cuales constan de una etiqueta de inicio y otra de fin, la de fin contendrá el mismo texto que la de inicio añadiéndole al principio una barra inclinada /. El efecto que define la etiqueta tendrá validez para todo lo que este incluido entre las etiquetas de inicio y fin, ya sea texto plano o otras etiquetas HTML. Por ejemplo:

<ETIQUETA> Elementos Afectados por la Etiqueta </ETIQUETA>

* Cerradas: estas etiquetas no necesitarán la de fin, son aquellas en las que el final este implícito, por ejemplo <P> párrafo,
 salto de línea ó <IMG> inclusión de una imagen. Definen un efecto que se producirá en un punto determinado sin afectar a otros elementos.

1.4.3.1 ATRIBUTOS DE LAS ETIQUETAS

Las etiquetas pueden presentar modificadores a los que se les llama atributos, las cuales permiten definir diferentes posibilidades de la instrucción HTML. Estos atributos se definirán en la etiqueta de inicio y consistirán normalmente en el nombre del atributo y el valor que toma, separados por un signo de igual. El orden en que se incluyan es indiferente, no afectando al resultado. Cuando el valor que toma el atributo tiene más de una palabra deberá expresarse entre comillas, en otro caso no será necesario. Igualmente una etiqueta podría presentar uno ó varios atributos. Por ejemplo en este caso la etiqueta HR presenta tres atributos:

<HR ALIGN=LEFT SIZE=5 WIDTH=50%>

1.4.3.2 ESTRUCTURA BÁSICA DE UN DOCUMENTO HTML

Un documento HTML está definido por una etiqueta de apertura <HTML> y una de cierre </HTML>. Dentro de este se dividen dos partes fundamentales la

cabecera, delimitada por la etiqueta <HEAD> y el cuerpo, delimitado por la etiqueta <BODY>. Por tanto la estructura de un documento HTML será:

<HTML>	Indica el inicio del documento.
<HEAD>	Inicio de la cabecera del documento.
<TITLE></TITLE>	Inicio / Final del título del documento.
Definiciones de la cabecera	
</HEAD>	Final de la cabecera del documento.
<BODY>	Inicio del cuerpo del documento.
Instrucciones HTML	
</BODY>	Final del cuerpo del documento.
</HTML>	Indica el final del documento.

1.4.4 FORMATO DE LAS URL

1.4.4.1 DEFINICIÓN

URL es el acrónimo de (Uniform Resource Locator), localizador uniforme de recursos y permite localizar o acceder de forma sencilla a cualquier recurso de la red, desde el navegador de la WWW. Con la WWW se pretende unificar el acceso a información de servicios que antes eran incompatibles entre sí, tratando de conseguir que los servicios de Internet sean accesibles, de esta forma desde un mismo programa se puede tener acceso a todos los recursos de una forma uniforme y permite que los documentos HTML incluyan enlaces a otras fuentes de información en servicios como FTP, gopher, WAIS, etc.

1.4.4.2 USO Y FORMATO DE URL

Las URL se utilizarán para definir el documento de destino de los hiperenlaces, para referenciar los gráficos y cualquier otro fichero que se desee incluir dentro de un documento HTML.

Cada elemento de Internet tendrá una URL que lo defina, ya se encuentre en un servidor de la WWW, FTP, gopher o las News. El formato de una URL será:

servicio://máquina.dominio:puerto/camino/fichero

El servicio será alguno de Internet, estos pueden ser: http: (HyperText Transport Protocol), https: (HyperText Transport Protocol Secure), ftp: (File Transfer Protocol), gopher, wais, news, telnet, malito.

La máquina Dominio indicará el servidor que proporciona el recurso, en este caso se utilizará el esquema IP para identificar la máquina, que será el nombre de la máquina y el dominio.

En el caso del ITJ el dominio siempre será itjiquilpan.edu.mx Por tanto un nombre válido de máquina será www.itjiquilpan.edu.mx.

El puerto TCP es opcional y lo normal es no ponerlo, si el puerto es el mismo que se utiliza por el servicio. Solo se utilizará cuando el servidor utilice un puerto distinto al de por defecto.

El camino será la ruta de directorios que hay que seguir para encontrar el documento que se desea referenciar. Para separar los subdirectorios se utiliza la barra de UNIX /, se usa por convenio al ser este tipo de máquinas las más usadas como servidores.

La extensión de los ficheros será de gran importancia, ya que por ella sabe el servidor el tipo de documento que se accede e indica al cliente (navegador) la forma en que debe tratarse ese documento. Para definir los tipos de documentos se utiliza los tipos MIME. Las extensiones más normales con sus tipos correspondientes se muestran en la Tabla 1.1.

Tipo MIME	Extensión	Tipo de fichero
text/html	.html ó .htm	documento HTML
text/plain	.txt	Por defecto, texto plano
image/gif	.gif	imagen de formato GIF
image/jpeg	.jpg ó .jpeg	Imagen de formato JPEG

Tabla 1.1_ Extensiones de Archivos.

1.5 JAVASCRIPT

JavaScript comenzó siendo LiveScript, pero Netscape le cambio el nombre, posiblemente porque estaba configurado en Java. El nombre confunde a la gente, aunque, se espera un acercamiento entre Java y JavaScript mayor del que existe ahora. De hecho hay una pequeña confrontación entre estos lenguajes, a pesar de tener una estructura muy similar.

El lenguaje JavaScript fue creado por Netscape en 1996 y estaba incluido en su navegador Netscape Navigator (NN) 2.0, que contaba con un intérprete que leía y ejecutaba el lenguaje JavaScript incluido en sus páginas HTML. La popularidad de este lenguaje ha crecido desde entonces, y ahora es soportado por la mayoría de los navegadores de Microsoft y Netscape y por otros navegadores como Opera.

La buena noticia es que esto significa que JavaScript puede ser usado por la mayoría de los navegadores modernos. Pero esto no impide las diferentes maneras como se implementa JavaScript en sus navegadores, aunque el código es muy similar.

JavaScript permite crear aplicaciones específicamente orientadas a su funcionamiento en la red Internet. Usando JavaScript, se pueden crear páginas HTML dinámicas que procesen la entrada del usuario y que sean capaces de gestionar datos persistentes usando objetos especiales, archivos y bases de datos relacionales.

Con este lenguaje se pueden construir aplicaciones que varían desde la gestión de la información corporativa interna y su publicación en Intranets hasta la gestión masiva de transacciones de comercio electrónico. Es importante reseñar que JavaScript puede utilizar una tecnología propietaria de Netscape, denominada

LiveConnect, con el propósito de que las aplicaciones JavaScript puedan tener acceso a aplicaciones basadas en objetos distribuidos CORBA y Java, pese a la similitud de nombres, JavaScript no es Java.

Las aplicaciones cliente y servidor en JavaScript comparten el mismo núcleo de lenguaje. Este núcleo corresponde con ECMA-262, el lenguaje de scripts estándar de la Oficina de Estándares de la Unión Europea, con algunos añadidos extra. Aunque Javascript de cliente y de servidor comparten el mismo conjunto base de funciones y características, en algunos casos se utilizan de distinta forma. Los componentes de este lenguaje son los siguientes:

* Núcleo de JavaScript (Core JavaScript).
* JavaScript para Cliente.
* JavaScript para Servidor.

JavaScript para cliente engloba el núcleo del lenguaje y algunos elementos adicionales como, por ejemplo, una serie de objetos predefinidos que sólo son relevantes para la ejecución del mismo en el contexto de un cliente Web. Así mismo, dicho lenguaje para servidor incluye también el núcleo de lenguaje y los objetos predefinidos y funciones necesarias para el correcto funcionamiento en el marco de un servidor.

1.5.1 JAVASCRIPT PARA APLICACIONES CON SERVIDOR

En el servidor, JavaScript también está integrado en páginas HTML. Las sentencias de JavaScript del servidor pueden realizar multitud de tareas:

- Conectarse a bases de datos relacionales de varios fabricantes.
- Compartir información entre usuarios de una aplicación.
- Acceder a los ficheros del servidor.
- Comunicarse con otras aplicaciones a través de LiveConnect y Java.

Las aplicaciones JavaScript del servidor se compilan generando archivos binarios. Existen servicios especiales de este lenguaje en el servidor:

- Servicio de Gestión de Sesiones.
- Servicio de Bases de Datos LiveWire.

1.5.2 JAVASCRIPT PARA APLICACIONES CLIENTE

Los clientes Web que soportan JavaScript, tales como el Netscape Navigator (desde la versión 2.0) o el Microsoft Internet Explorer (desde la versión 3.0) pueden interpretar sentencias JavaScript colocadas en un documento HTML. Cuando el cliente Web solicita una página de este tipo, el servidor envía por la red al cliente el contenido completo del documento, incluyendo todos los códigos HTML y las sentencias JavaScript que pudieran existir en él, éste es interpretado en su totalidad por el cliente Web en tiempo de ejecución.

Las sentencias JavaScript colocadas en una página Web pueden dar respuesta a eventos de usuario, tales como la pulsación de un botón del ratón (clic), la entrada de datos en un formulario y la navegación por una página.

Por ejemplo, se puede crear una función JavaScript que permita verificar que la información introducida por el usuario en un campo de entrada de datos de un formulario (número de teléfono, código postal, número de tarjeta de crédito, etc.) tiene el formato correcto. En este caso, lo importante es que, sin necesidad de realizar ninguna transmisión de datos por la red, se puede validar dicha información, mostrando al usuario un cuadro de diálogo en caso de que ésta sea incorrecta.

Clientes que no soportan JavaScripts

Estos clientes no admiten la etiqueta HTML <SCRIPT>. Consideran todo su contenido como texto normal, por ello, se hace preciso ocultar el código a

clientes que no lo soporten. Para evitar esto, se utilizan los comentarios de HTML entre las etiquetas <SCRIPT> y </SCRIPT>. Otra forma de conocer si un cliente soporta JavaScript es insertar el código <NOSCRIPT>...</NOSCRIPT>. De modo que los navegadores que no soporten JavaScript ejecuten las sentencias HTML alternativas incluidas dentro de esta etiqueta.

Integración de JavaScript en documentos HTML

Se ofrece aquí un primer ejemplo en el que se ilustra la integración directa de código JavaScript en un documento HTML:

```
<HTML>
	<HEAD>
		<TITLE>Primer ejemplo de JavaScript</TITLE>
	</HEAD>
	<BODY>
		<B>Esto es texto normal de un documento HTML</B> <BR>
		<SCRIPT LANGUAGE="JavaScript">
			document.write ("Texto generado desde JavaScript")
		</SCRIPT>
		<BR><B>Esto es, de nuevo, HTML</B>
	</BODY>
</HTML>
```

Para poder ver el resultado de su ejecución, bastará con cargar dicho documento con cualquiera de los clientes Web antes mencionados. El resultado se compone de tres líneas de texto:

Esto es texto normal de un documento HTML

Texto generado desde JavaScript

Esto es, de nuevo, HTML

En realidad no se trata de un script útil, puesto que todo lo que ofrece se podría haber hecho en HTML directamente, y sin duda, con mayor comodidad. Sólo se trata de mostrar el funcionamiento del código <SCRIPT>. En efecto, cualquier elemento que quede delimitado por las etiquetas <SCRIPT> y </SCRIPT> se considera código JavaScript.

En este caso particular, se ha utilizado document.write, una de las funciones más importantes de JavaScript, que permite escribir algo en el documento actual.

Archivos de código JavaScript

El atributo SRC del código SCRIPT del lenguaje HTML permite especificar un archivo que contiene el código JavaScript en lugar de incrustar el código JavaScript en el documento HTML. Por ejemplo:

```
<HEAD>
<SCRIPT SRC="comun.js">
</SCRIPT>
</HEAD> .....
```

Este atributo es especialmente útil para compartir funciones entre numerosos documentos HTML. Las sentencias JavaScript del interior de un código <SCRIPT SRC= ... > se ignoran a menos que la inserción cause un error. Se incluye una sentencia que muestre un mensaje de error en caso de no poder cargar el archivo de código. Los archivos JavaScript externos solo pueden tener código JavaScript y ninguna sentencia HTML.

1.5.3 VARIABLES

Las variables son palabras que se incluyen en el código JavaScript, que incluyen la información que se desee, su principal rutina es para simplificar el uso de información larga, en constante cambio de valor y para almacenar información que obtengamos del usuario, JavaScript reconoce los siguientes valores:

- Valores numéricos.
- Valores lógicos (true, false).
- Cadenas de caracteres.
- Valor null.
- Valor no definido (undefined).

Además de estos valores, existen otros elementos propios de JavaScript como los objetos y las funciones. JavaScript trata de forma dinámica los datos. Se puede realizar la siguiente operación: var Valor = 50

Y después asignar a un Valor, un valor de tipo cadena de caracteres:

Valor = "Ahora está lloviendo"

Las Variables pueden comenzar por un carácter o un guión bajo (_). Para establecer una variable simplemente se escribe el nombre que se quiera para la variable, luego un signo = y finalmente el valor inicial que tendrá. Cuando a una variable no se le asigna un valor, tiene valor indefinido.

Si se le pone un valor, puede ocurrir que:

- Si fue declarada sin "var", se produce un error en tiempo de ejecución.
- Si fue declarada con "var", devuelve el valor NaN(Not a Number).

Veamos un ejemplo:

```
function f1() {
        return y-2;
        }          f1() // Esta llamada a f1 provoca un error en tiempo de ejecución
function f2() {
          return var y-2;
        }                  f2() // devuelve el valor NaN
```

Las variables pueden ser:

* Globales: Cuando se le asigna un valor fuera de una función. El uso de "var" es opcional.
* Locales: Se realiza la operación de asignación dentro de una función. El uso de "var" es obligatorio.

Por último, es bueno saber que se puede acceder a una variable de un documento HTML de un FRAME desde otro con la instrucción parent: parent.MiVariable

Captura

JavaScript permite definir eventos y asignarlos a objetos por encima de los elementos donde nacen dichos eventos. Para definir estos eventos, los objetos window, document y layer utilizan los siguientes métodos:

- **captureEvents:** Captura eventos del tipo que se especifique.
- **releaseEvents:** Ignora la captura del tipo especificado.
- **routeEvent:** Envía el evento capturado a un objeto.

Ahora vamos a ver la secuencia de captura, definición y activación de un gestor de eventos:
Debemos especificar el tipo de eventos que queremos capturar:

```
window.captureEvent(Event.CLICK [| Event.* | Event.*])
```

De este modo, todos los eventos de tipo click que se produzcan serán capturados. Nótese que se pueden especificar varios eventos, siempre separados por | que significa or. Tras capturar dicho(s) evento(s), se debe especificar una función que realice las acciones asociadas a dicho evento:

```
function evento_nombre([parámetros]){
    acciones
      return true o false
```

La función devolverá true cuando la acción sea posible y false en caso contrario. Ya tenemos el evento capturado y la función asociada, sólo queda asignar al evento la función especificada:
window.onClick=evento _ nombre;

1.6 PHP

PHP fue creado por Rasmus Lerdorf a finales de 1994, aunque no hubo una versión utilizable por otros usuarios hasta principios de 1995. Esta primera versión se llamó, Personal Home Page Tools. Al principio, PHP sólo estaba compuesto por algunas macros que facilitaban el trabajo a la hora de crear una página Web. A mediados de 1995, se creó el analizador sintáctico y se llamó PHP/F1 Versión 2, y sólo reconocía el texto HTML y algunas directivas de MySQL. El crecimiento de PHP desde entonces ha sido exponencial, y han surgido versiones nuevas como la actual, PHP3, PHP4 y PHP5.

El objetivo final es conseguir la integración de las paginas HTML con aplicaciones que corran en el servidor como procesos integrados en el mismo, y no como un proceso separado, como ocurría con los CGIs. Igualmente interesa que dichas aplicaciones sean totalmente independientes del navegador (lo que no ocurría con otros lenguajes basados en scripts, como JavaScript o VisualBasic Script), independientes de la plataforma y de la Base de Datos.

Perl ha sido el lenguaje que ha servido como estándar para construir CGIs durante mucho tiempo, y aún sigue siendo una de las mejores soluciones para desarrollar aplicaciones Web portables, ya que trabaja en cualquier servidor Web que soporte CGIs, y sobre cualquier plataforma que soporte Perl, incluso ha servido para desarrollar módulos que extienden la funcionalidad de los servidores. PHP, está más orientado a conexiones entre páginas Web y servidores donde se almacenan toda clase de Bases de Datos.

PHP es un lenguaje de programación soportado por HTML. La sintaxis está heredada de C, Java y Perl. Este lenguaje está orientado para los constructores de páginas Web, permitiéndoles crear páginas dinámicamente generadas de

forma rápida. Oficialmente, es un preprocesador de hipertextos, ¿pero qué significa?

Para ilustrar esto podemos ver un simple ejemplo:

```
<html>
        <head>
                <title>Example</title>
        </head>
        <body>
                <?php echo "Hi, Esto es un Script PHP";?>
        </body>
</html>
```

Esto es muy parecido a cualquier otro Script escrito en Perl o C. El código de PHP está incluido en tags especiales "<?", "?>". Lo que hace diferente a PHP es que el código se ejecuta siempre en el servidor. Así, al ejecutar el script anterior, el cliente recibirá sólo los resultados de la ejecución por lo que es imposible para el cliente acceder al código que generó la página. Por ello se pueden hacer con muchos tipos de bases de datos como: Adabas D, dBase, Empress, FiclePro, informix, Solid, Sybase, Veloces, Unix dbm, mSQL, MySQL, Oracle, PosgreSQL. Es decir este lenguaje esta preparado para soportar los accesos a las BD antes mencionadas. Además de esto, PHP soporta la utilización de otros protocolos como IMAP, SNMP, NNTP, POP3 o HTTP a nivel de socket.

1.6.1 SEGURIDAD Y PHP

PHP es un intérprete que puede ser incluido en un servidor Web como un módulo o como un CGI binario. Con él se pueden realizar accesos a ficheros, conexiones de red, etc. PHP está diseñado para ser más seguro que cualquier otro lenguaje de programación de CGIs, como Perl o C.

CGI binario, este método lo que hace es instalar PHP en el directorio cgi-bin. Esto permite a PHP reaccionar ante diversos tipos de ataques. Por ejemplo, al acceder al sistema de ficheros mediante la línea: http://my.host/cgi-

bin/php?/etc/passwd. En http, todo lo que este detrás del símbolo “?”, es la línea de argumentos que el interfaz CGI interpreta. Curiosamente, si a un sistema Linux, le pasas la instrucción /etc/passwd, el sistema intenta ejecutar este comando y esto puede ser un fallo en la seguridad. Otro posible ataque, intenta acceder a los ficheros del servidor Web a los que no se debe tener acceso. Para evitar esto, existen opciones de configuración que redirigen todas las peticiones al intérprete de PHP, forzando un chequeo de acceso al fichero que se pide. Algunas de estas opciones de seguridad son:

1) Si se activa la opción disable-force-cgi-redirect se obliga a que tanto las peticiones del tipo http://my.host.cgi-bin/php/dir/script.php3 como las peticiones del tipo http://my.host/dir/ script.php3 sean analizadas por el intérprete PHP.
2) Otras opciones posibles en la configuración, es combinar la directiva Action y AddHandler mediante estas opciones se configura la redirección de las llamadas para que sean interpretadas. Esta opción ha sido probada en Apache y a este servidor se refiere. Action php3-script /cgi-bin/php Addhandler php3-script.php3
3) La tercera opción es utilizar las directivas doc_root y user_dir. Estas directivas se utilizan en servidores Web que no disponen de la facilidad del redireccionamiento. Por ejemplo un script no se ejecuta correctamente, en este caso, el código se muestra en pantalla y esto puede violar la propiedad intelectual de ese script. Para solucionar esto, se colocan todos los scripts PHP ejecutables en un directorio, que indica la directiva doc_root asegurando así que todo lo que esté en ese directorio será ejecutado y nunca mostrado al usuario. Si esta directiva se combina con user_dir se permitirá ejecutar, ante llamadas del tipo: http://my.host/~user/doc.php3 ficheros que estén en el directorio que indica user_dir bajo el directorio /home/user/.

Otra práctica muy segura es mantener la instalación del intérprete fuera del arbol web. Si esto es así, se deberán hacer los ficheros php ejecutables, modificando los atributos del fichero y además se deberá incluir en la primera línea del script la dirección del intérprete, #!/usr/local/bin/php por ejemplo.

Módulo, en el caso de tener PHP instalado como un módulo del servidor Apache, este hereda todas las características del servidor. Esta opción es la menos utilizada.

1.6.1.1 CRIPTOGRAFÍA

Viene del griego y significa “escritura secreta”. Históricamente cuatro grupos de personas han utilizado y contribuido al arte de la criptografía: los militares, el cuerpo diplomático, los redactores de los periódicos y los amantes.

De éstos, la milicia ha tenido el papel más importante y ha moldeado el campo a través de los siglos. Hasta la llegada de las computadoras, una de las principales restricciones de la criptografía había sido la capacidad del empleado encargado de la codificación para realizar las transformaciones, con frecuencia en un campo de batalla y con poco equipo. Estos requerimientos en conflicto han dado lugar al siguiente modelo de encriptación para un cifrado de clave simétrica (Ver figura 1.6).

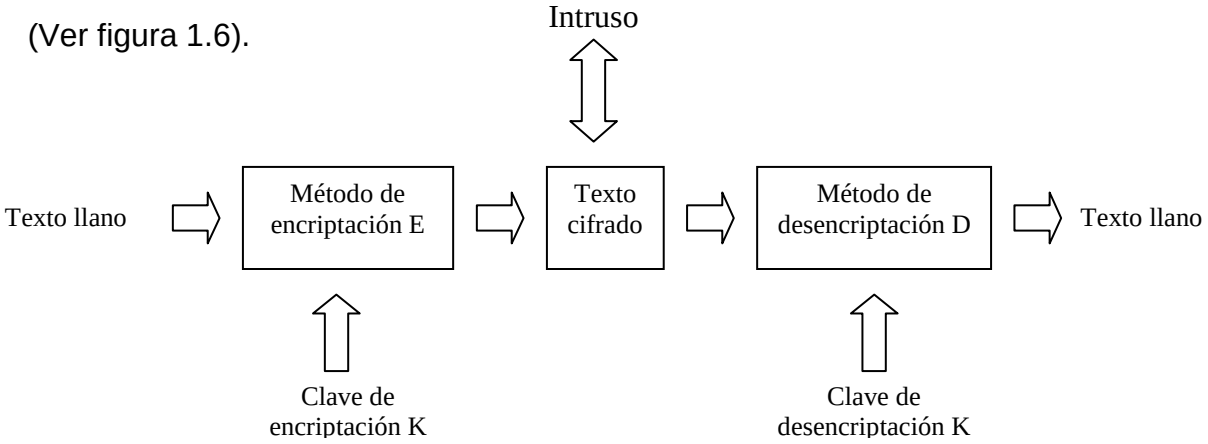

Figura 1.6_ Modelo de Encriptación.

Los mensajes por encriptar, conocidos como texto llano, son transformados por una función parametrizada por una clave. El resultado del proceso de encriptación es conocido come texto cifrado y es transmitido posteriormente. Suponemos que el intruso copia con exactitud todo el texto cifrado, sin embargo a diferencia del destinatario original, el intruso no conoce la clave de desencriptación y no puede desencriptar con facilidad el texto cifrado.

El arte de descifrar los mensajes (criptoanálisis) y el arte de crear los cifrados (criptografía), se conocen en conjunto como criptología. Para Encriptar claves con PHP dispone de una función que efectúa una encriptación de claves, al igual que ocurre en el fichero /etc/passwd de Linux, se trata de la función crypt, que encripta una cadena mediante el algoritmo DES.

string crypt (string cad [string semilla])

crypt(), encripta una cadena utilizando el método estándar de encriptación del Unix DES. Los argumentos son una cadena a encriptar y una cadena semilla de 2 caracteres en que basar la encriptación. Si el argumento de semilla no se proporciona, será generado aleatoria mente por PHP.

Algunos sistemas operativos soportan más de un tipo de encriptación. De hecho, algunas veces la encriptación estándar DES es sustituida por un algoritmo de encriptación basado en MD5. El tipo de encriptación es disparado por el argumento semilla. En tiempo de instalación, PHP determina la capacidad de la función de encriptación y aceptará semillas para otros tipos de encriptación. Si no se proporciona la semilla, PHP intentará generar una semilla estándar DES de 2 caracteres por defecto, excepto si el tipo de encriptación estándar del sistema es MD5, en cuyo caso se generará una semilla aleatoria compatible con MD5.

PHP fija una constante llamada CRYPT_SALT_LENGTH que le especifica si su sistema soporta una semilla de 2 caracteres o si se debe usar la semilla de

12 caracteres del NDS. La función estándar de encriptación crypt() contiene la semilla como los dos primeros caracteres de la salida. En los sistemas en los que la función crypt() soporta múltiples tipos de encriptación, las siguientes constantes son fijadas de 0 ó 1 dependiendo si está disponible el tipo dado:

CRYPT_STD_DES - Encriptación DES estándar con semilla de 2 caracteres.

CRYPT_EXT_DES - Encriptación DES extendida con semilla de 9 caracteres.

CRYPT_MD5 - Encriptación MD5 con semilla de 12 caracteres y comenzando por 1.

CRYPT_BLOWFISH - Encriptación DES extendida con semilla de 16 caracteres y comenzando por 2.

No hay función de desencriptado porque crypt() utiliza un algoritmo de una sola vía.

MD5, es la quinta de una serie de compendios de mensaje diseñados por Ronald Rivest. Opera truncando los bits de una manera tan complicada que cada bit de salida es afectado por cada bit de entrada. Ahora comienza el cálculo. Cada ronda toma un bloque de 512 bits de entrada y lo mezcla por completo con el búfer de 128 bits. Se introduce también una tabla construida a partir de la función seno.

El objetivo de usar una función conocida como el seno, no es por que sea mas aleatoria que un generador de números aleatorios, sino para evitar cualquier sospecha de que el diseñador construyo una puerta trasera ingeniosa por la que solo el puede entrar. Se hacen cuatro rondas por cada bloque de entrada. Este proceso continua hasta que todos los bloques de entrada se han consumido. El contenido de búfer de 128 bits forma el compendio de mensaje.

1.6.2 BASES DE LA SINTAXIS

Inserción de PHP en HTML

<? .... ?> Sólo si se activa la función short_tags() o la bandera de configuración short_open_tag.

<?php .... ?> ó <script lenguaje="php"> .... </script> Sólo si se activan los tags para ficheros 'asp' con la bandera de configuración asp_tags.

Separación de instrucciones

Las instrucciones se separan con ‘ ; ' , en el caso de ser la última instrucción no es necesario el punto y coma.

Comentarios

Los comentarios en PHP pueden ser como en C o C++, /*...*/ ó // . Otro tipo de comentario de una línea es #, que comentará la línea en que aparezca pero sólo hasta el tag ?> que cierra el código php.

Juggling

Una variable en PHP, define su tipo según el contenido y el contexto en el que se utilice, es decir, si se asigna una cadena a una variable, el tipo de esa variable será string. Si a esa misma variable se el asigna un número, el tipo cambiará a entero. Para asegurarte de que una variable es del tipo adecuado se utiliza la función settype (). Para obtener el tipo de una variable se utiliza la función gettype (). También es posible utilizar el mecanismo del casting tal y como se utiliza en C.

1.6.3 VARIABLES

Las variables en PHP mantienen una serie de características referentes a:

* Conceptos básicos.
* Variables predefinidas.
* Ámbito de una variable.
* Variables variable.
* Variables Externas a PHP.
* Conceptos básicos.

Los conceptos a tomar en cuenta en PHP con las variables son los siguientes:

- Cualquier nombre de variable está precedido por el símbolo $.
- En PHP las variables siempre se asignan por valor, aunque en PHP4 ya existen métodos para asignaciones por referencia (&).

Cuando existe un form en HTML, inmediatamente después de ser enviado, dentro del ámbito PHP se crea automáticamente una variable por cada uno de los objetos que contiene el form. Si se activa la directiva <?php_track_vars?> o con la variable track_vars, todo lo enviado por los métodos POST y GET estará en las variables $HTTP_POST_VARS y $HTTP_GET_VARS. Las variables de entorno, tales como $HOME, para entornos Linux, se pueden utilizar desde PHP.

1.6.4 CONSTANTES

Las constantes en PHP tienen que ser definidas por la función define() y además no pueden ser redefinidas con otro valor. Además, existen una serie de variables predefinidas denominadas:

- _FILE_: Fichero que se está procesando.
- _LINE_: Línea del fichero que se está procesando.
- _PHP_VERSION: Versión de PHP.
- PHP_OS: Sistema operativo del cliente.
- TRUE: Verdadero.
- FALSE: Falso.
- E_ERROR: Error sin recuperación.
- E_WARNING: Error recuperable.
- E_PARSE: Error no recuperable (sintaxis).
- E_NOTICE: Puede tratarse de un error o no, normalmente permite continuar la ejecución.

Todas las constantes que empiezan por "E_"se utilizan normalmente con la función error_reporting (). Ejemplo:

```
<?php
define("CONSTANTE", "hello world.");
echo CONSTANTE;
?>
```

1.6.5 EXPRESIONES Y OPERADORES

En PHP una expresión es cualquier cosa que contenga un valor. Las expresiones más simples son las variables y las constantes, otras más complicadas serán las funciones, puesto que cada función devuelve un valor al ser invocada, es decir, contiene un valor, por lo tanto, es una expresión.

Todas las expresiones en PHP son exactamente igual que en C. Los operadores abreviados, los incrementos, etc., son exactamente iguales. Incluso existen otros operadores adicionales como el operador "." que concatena valores de variables, o el operador "===" denominado operador de identidad, que devolverá verdadero si las expresiones a ambos lados del operador contienen el mismo valor y son del mismo tipo. Por último, el operador "@" sirve para el control de errores. Para poder ver como funciona el operador @, veamos un ejemplo:

```
<?php
        $res = @mysql_query("select nombre from clientes")
or die   ("Error en la selección, '$php_errormsg'");
?>
```

Este ejemplo, utiliza el operador @ en la llamada a mysql_query y en caso de dar un error, se salvará el mensaje devuelto en una variable denominada php_errormsg. Esta variable contiene el mensaje de error de cada sentencia y si ocurre otro error posterior, se borra el valor con la nueva cadena.

PHP mantiene también los operadores """, que sirven para ejecutar un comando del sistema tal y como hace la función system() por ejemplo: Las diferencias con C son los operadores de referencia, & y *, puesto que las operaciones por referencias no existen en PHP·, aunque si son posibles en PHP4, y en PHP existen dos operadores and y dos operadores or que son: 'and', '&&' y 'or', '||' respectivamente, que se diferencian en la precedencia de cada uno.

1.6.6 ESTRUCTURAS DE CONTROL

La mejor forma de resumir cada una de las opciones que ofrece PHP para las estructuras de control es mediante la siguiente tabla (Ver Tabla 1.2).

Estructura	Alternativa
If, if else, if elseif	if: endif;
While	while:
endwhile;	
For	for: endfor;
Do... while	
foreach (array as $value)	
foreach (array as $key=>$value)	
(Solo PHP4 y no PHP3)	
Switch	switch:
endswitch;	
Continue	
Break	
require() (Necesitan estar dentro de tags PHP)	
include() (Necesitan estar dentro de tags PHP)	

Tabla 1.2_ Estructuras de Control de PHP.

Una nota sobre include() y require(), si se desea incluir un fichero de forma condicional, es mejor utilizar include(), sin embargo, si la línea donde está una instrucción require() no se ejecuta nada de ese fichero.

Además, si en un bucle se ejecutan varias veces una instrucción require(), el intérprete lo incluirá una sola vez, en cambio si es include(), se incluirá el fichero cada vez que se ejecute la instrucción. Como apunte final, se debe saber que en un fichero que va a ser requerido, se puede incluir una instrucción return al final como si esta instrucción devolviera un valor (sólo en PHP3), si se trata de include, se puede poner al final del fichero la instrucción return, tanto en PHP3 como en PHP4, aunque con algunas diferencias. Así, require(), reemplaza su llamada por el contenido del fichero que requiere, e include(), incluye y evalúa el fichero especificado.

1.6.7 FUNCIONES

Funciones definidas por el usuario; un ejemplo puede ser:

```
Function foo ($a1, $a2, ..., $aN)
 {
   echo "Función ejemplo"
   return $value;
 }
```

Dentro de una función puede aparecer cualquier cosa, incluso otra función o definiciones de clase. En PHP3 es necesario que una función esté declarada antes de ser referenciada, y en PHP4 esto no es necesario. No es posible realizar sobrecarga de funciones o número variable de argumentos en PHP3, pero sí en PHP4, aunque esto se puede simular en PHP3 pasando un array de argumentos.
Respecto al paso de argumentos, son pasados por valor y por referencia, para pasarlos por este último hay que indicarlo, se puede hacer de dos formas

diferentes, en la definición de la función, anteponiendo el símbolo & al argumento que corresponda, en este caso la llamada será igual que la llamada a una función normal, o manteniendo la definición de la función normal y anteponer un & delante del argumento que corresponda en la llamada a la función.

PHP permite el mecanismo de argumentos por defecto. Un ejemplo de esta característica es:

```
Function hacerCafe($tipo="capuchino")
        {
          return "he hecho un café $tipo\n";
        }
```

En la llamada a esta función se obtendrá una frase u otra según se llame: echo hacerCafe() ó echo hacerCafe("expreso"). En el caso de tratarse de una función con argumentos por defecto y argumentos normales, los argumentos por defecto deberán estar agrupados al final de la lista de argumentos. En PHP4 el número de argumentos de una función definida por el usuario, puede ser variable, se utilizan las funciones func_num_args(), func_get_arg() y func_et_args().

Valores devueltos

PHP puede devolver cualquier número de valores, sólo hará falta recibir estos argumentos de la forma adecuada. Ejemplo:

```
function numeros()
  {
    return array(0,1,2);
  }
list ($cero, $uno, $dos) = numeros();
```

Funciones Variables

Las funciones variables pueden ser una potente herramienta en el procesamiento dinámico de un script. Ejemplo:

```
<?php
        funcion foo()
          {
            echo "En foo()<br>\n"
```

```
    }
  function bar ($arg ='')
    {
      echo " bar();El argumento ha sido '$arg'.<br>\n"
    }
  $func = 'foo';
  $func();
  $func='bar';
  $func('test');
?>
```

1.6.8 MANEJO DE ERRORES

En PHP hay cuatro tipos de errores:

- Funciones de Error (1)
- Warnings (2)
- Errores Parse (4)
- Notices (8)

El nivel de error por defecto es 7 (1+2+4), pero esto puede ser modificado en el fichero de configuración php3.ini con la directiva error_reporting. Cualquier expresión en PHP se puede llamar con la "@", al principio invocará a la función de manejo de errores, y si track_errors está activada, el error podremos encontrarlo en la variable $php_errormsg.

1.6.9 MANEJO DE CONEXIONES

En PHP las conexiones que se mantienen pueden tener tres estados, Normal (0), Aborted (1) y Timeout (2). En un script normal, el estado es NORMAL, cuando el cliente desconecta, el estado pasa a ser ABORTED y si el límite impuesto por PHP-imposed ha transcurrido, (set_time_limit(), el tiempo por defecto es 30 segundos) el estado es TIMEOUT. Una función muy útil para estos casos, es connection_status() que devuelve el estado de la conexión. Las conexiones

persistentes son enlaces SQL que no se cierran cuando la ejecución del script termina. El comportamiento de estas conexiones es el siguiente; cuando se solicita una conexión de este tipo, PHP comprueba si existe una conexión de este mismo tipo o por el contrario, se trata de una nueva conexión. En caso de que exista, se procede a su uso, y en caso inverso, la conexión se crea. Dos conexiones se consideran iguales cuando están realizadas sobre el mismo servidor, con el mismo usuario y la misma contraseña.

En realidad, estas conexiones permanentes, no proporcionan ningún tipo de funcionabilidad adicional frente a conexiones temporales, debido a la forma en que los servidores Web funcionan. Aún así se utilizan debido a la eficiencia, al tiempo de establecimiento de la conexión, y a que si se tiene una sola conexión sobre el servidor, irá mucho más rápido que si se tienen 10 conexiones temporales, puesto que la carga que soporta es diferente.

Funciones De Php Mysql

La lista de algunas funciones disponibles es la siguiente:

mysql_connect: Abre una conexión con el servidor MySQL

mysql_close: Cierra la conexión MySQL.

mysql_create_db: Crea una base de datos en el gestor de Bases de Datos.

mysql_drop_db: Realiza una operación Drop sobre una base de datos.

mysql_db_query: Realiza una consulta a una base de datos.

mysql_error: Devuelve el mensaje de error asociado a un código concreto relacionado con la última operación MySQL realizada.

mysql_fetch_array: Introduce el resultado en un array asociativo.

mysql_free_result: Libera la memoria de los resultados.

mysql_query: Envía una consulta SQL a MySQL.

mysql_result: Obtiene los datos resultados.

1.7 APACHE

Apache es el servidor Web hecho con excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. El nombre Apache tiene un origen un poco discutido, algunos dicen que viene de "A Patchy Server" debido a numerosos patchs del principio, los instigadores de este proyecto dicen que tomaron el nombre en memoria de los Apaches por su gran adaptabilidad al terreno. Un grupo del CERN (Centro Europeo de Investigación Nuclear) desarrolló el concepto servidor/cliente HTTP. Una vez terminado su trabajo de investigación, confiaron esto a una universidad americana (NSCA).

La historia de Apache se remonta a febrero de 1995, donde empieza el proyecto del grupo Apache, el cual esta basado en el servidor Apache http, la aplicación original de NCSA.

El desarrollo de esta aplicación, se estancó por algún tiempo tras la marcha de Rob McCool por lo que varios webmaster siguieron creando sus parches para sus servidores Web, hasta que se contactaron vía correo electrónico para seguir en conjunto el mantenimiento del servidor Web, fue ahí cuando formaron el grupo Apache.

Fueron Brian Behlendorf y Cliff Skolnick quienes a través de una lista de correo, coordinaron el trabajo y lograron establecer un espacio compartido de libre acceso para los desarrolladores. Fue así como fue creciendo el grupo Apache, hasta lo que es hoy.

La primera versión, sus sucesivas evoluciones y mejoras alcanzaron gran implantación como software de servidor, inicialmente solo para sistemas operativos UNIX y fruto de esa evolución es la versión para Windows. Apache es

una muestra, de que el trabajo voluntario y cooperativo dentro de Internet es capaz de producir aplicaciones de calidad profesional difíciles de igualar.

La licencia Apache es una descendiente de la licencias BSD, no es GPL. Esta licencia permite hacer lo que quieras con el código fuente, siempre que les reconozcas su trabajo. La popularidad de este software libre, se debe a las siguientes razones:

* Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.

* Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto, esto le da una gran transparencia a este software.

* Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que sean instalados en caso de ser necesitados. Otra cosa importante es que cualquiera que posea una experiencia en la programación de C o Perl puede escribir un modulo para realizar una función determinada.

* Apache trabaja con gran cantidad de Perl, PHP y otros lenguajes de script. Perl destaca en el mundo del script y Apache, utiliza su parte de Perl tanto con soporte CGI como con soporte mod perl. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas.

* Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurarlo para que ejecute un determinado script cuando ocurra un error en concreto.

*Tiene una alta configurabilidad en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en el servidor.

1.7.1 DESCRIPCIÓN DE LA ARQUITECTURA EN MÓDULOS DEL APACHE

El servidor Apache es un software que esta 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:

- 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 atenderlas.
- 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.

1.7.2 DIRECTORIOS VIRTUALES EN APACHE

Un directorio virtual es un directorio que se encuentra en un directorio distinto del que se mapea en la URL. El directorio virtual no se tiene que encontrar, necesariamente, dentro del árbol de directorios que se crea a partir de

DocumentRoot, sino que se puede encontrar en cualquier otra ubicación, incluso se podría encontrar en un servidor distinto.

1.7.2.1 ALIAS

Permite la definición de directorios virtuales, por ejemplo, cuando se escribe tec. itjiquilpan.edu.mx/docente, la carpeta docente no se tiene que encontrar necesariamente dentro de la carpeta raíz itjiquilpan, sino que se encuentra en una ubicación distinta, y fuera del árbol de subdirectorios de la directiva DocumentRoot.

Sintaxis:

Alias NombreFicticio UbicacionReal

Ejemplo:

Alias /docente "/var/www/html/proyecto/docente"

El directorio docente no se encuentra dentro del directorio itjiquilpan, en la carpeta DocumentRoot, sino en la carpeta "/var/www/html/proyecto/docentes".

Por defecto vienen creados dos redirecciones con Alias:

* Icons: Establece la carpeta donde se encuentran los iconos que utilizará el Apache, para mostrar el contenido de los directorios.
* Manual: que apunta a la carpeta donde está instalado el manual del Apache en caso de que se hubiera elegido la opción durante la instalación.

Cualquier ajuste que se requiera realizar, ya sea para configurar sitios de red virtuales u otra funcionalidad adicional, se puede realizar sin tocar el fichero principal de configuración, utilizando cualquier fichero con extensión *.conf dentro del directorio /etc/httpd/conf.d/httpd.conf, de tal modo que si, por ejemplo, se

quiere añadir el alias para un directorio localizado en /var/ftp/pub/ y el cual se visualiza como el directorio /pub/ en Apache, solo bastaría crear un fichero que se denomina arbitrariamente como el fichero /etc/httpd/conf.d/aliases.conf con el siguiente contenido:

```
Alias /pub /var/ftp/pub
```

Si trata de acceder hacia este nuevo directorio virtual con el navegador, notará que no está permitido el acceso. Para poder acceder deberá haber un documento índice en el interior (index.html, index.php, etc) o bien que dicho directorio sea configurado para mostrar el contenido del siguiente modo:

```
Alias /pub /var/ftp/pub
    <Directory "/var/ftp/pub">
        Options Indexes Includes FollowSymLinks
        AllowOverrride all
    </Directory>
```

El parámetro Indexes, indica que se deberá mostrar el contenido del directorio. El parámetro FollowSymLinks posibilita poder colocar enlaces simbólicos dentro del directorio los cuales se seguirán. El parámetro Includes específica que se permite la utilización de los SSI (Server Side Includes) que posibilitan utilizar funciones como autenticación. El parámetro AllowOverrride all posibilita utilizar ficheros .htaccess.

1.8 MANEJADOR DE BASE DE DATOS

1.8.1 ¿QUÉ ES Y PARA QUE SIRVE SQL?

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 las operaciones básicas de una forma universal.

De eso trata el SQL (Structured Query Language) que no es más que un lenguaje estándar de comunicación con bases de datos.

Es 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...).

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.

MySQL empezó como un proyecto de programación hará unos 10 años, cuando un programador sueco decidió crear su propio gestor de datos para la

aplicación que estaba desarrollando. Hasta ese momento usaba MySQL, pero vio que quedaba incompleto en algunos aspectos que el podría mejorar directamente.

Pronto MySQL fue el número 1 en rendimiento, en el ámbito de las bases de datos de código abierto. Posteriormente se creo una empresa sueca, llamada Tux que se dedicó a crear nuevas versiones de MySQL y comercializarlo en ciertos casos. Actualmente MySQL AB ha absorbido el proyecto MySQL

.

1.8.2 CARACTERÍSTICAS DE MySQL

- Es un servidor de bases de datos SQL, multiusuario y multi-threaded.
- Puede usar más de una CPU si hay disponible.
- Las principales metas del diseño de MySQL son velocidad, robustez y fácil manejo.
- Maneja grandes volúmenes de información. Se han reportado bases de datos con 50'000.000 de registros.
- Es muy rápido para unir tablas. Aún si son de diferentes Bases de Datos.
- Fuente abierta pero con restricciones de licencia de uso.
- Lenguajes que se comunican con MySQL: C, C++, Eiffel, Java, Perl, PHP, Python and TCL.