



ESCUELA DE INFORMÁTICA

PROGRAMA: TÉCNICO EN DESARROLLO DE

SOFTWARE

SUBMÓDULO

PROGRAMACION PARA LA WEB I

DP03 Ver. 02

NOMBRE Y APELLIDOS ESTUDIANTE

---

## CONTENIDO

pág.

<b>PLANEADOR DE CLASES.....</b>	<b>11</b>
<b>PACTO PEDAGÓGICO.....</b>	<b>22</b>
<b>CLASE NO. 1 PACTO PEDAGÓGICO.....</b>	<b>22</b>
<b>INTRODUCCIÓN.....</b>	<b>24</b>
<b>CONCEPTOS GENERALES .....</b>	<b>25</b>
<b>Internet .....</b>	<b>25</b>
<b>SERVICIOS DE INTERNET.....</b>	<b>25</b>
<b>CLIENTE WEB .....</b>	<b>28</b>
<b>SERVIDOR WEB.....</b>	<b>28</b>
<b>representación Gráfica de la arquitectura cliente web – servidor web .....</b>	<b>29</b>
<b>APLICACIONES WEB .....</b>	<b>30</b>
Aplicaciones en el lado del cliente .....	30
Aplicaciones en el lado del servidor .....	30
<b>ALOJAMIENTO WEB .....</b>	<b>30</b>
<b>DOMINIO DE INTERNET .....</b>	<b>31</b>
<b>SUBDOMINIOS .....</b>	<b>32</b>
<b>DOMINIOS DE NIVEL SUPERIOR.....</b>	<b>32</b>
TLD Geográfico (ccTLD) .....	33
Dominios de Internet genérico (gTLD) .....	33
TLD de infraestructura .....	33
<b>página web .....</b>	<b>33</b>
<b>sitio web .....</b>	<b>33</b>
<b>Editores de página Web.....</b>	<b>33</b>
<b>EL LENGUAJE HTML.....</b>	<b>35</b>
<b>INTRODUCCIÓN.....</b>	<b>35</b>

<b>ETIQUETAS HTML .....</b>	<b>36</b>
Etiqueta de inicio y fin de página.....	36
Etiquetas de cabecera .....	36
Etiqueta de cuerpo de página .....	39
Etiquetas para manipular el texto .....	40
 <b>CLASE NO. 2 .....</b>	 <b>41</b>
Listas ordenadas y sin ordenar: Numeración y viñetas .....	41
Caracteres en HTML.....	42
Etiquetas para manipular contenido multimedia .....	49
Hipervínculos .....	52
Marcos flotantes.....	52
Tablas en HTML .....	53
 <b>CLASE NO. 3 .....</b>	 <b>55</b>
Formularios HTML .....	55
Etiquetas para crear divisiones .....	57
 <b>CLASE NO. 4 .....</b>	 <b>58</b>
<b>EJEMPLO GENERAL DE HTML.....</b>	<b>58</b>
<b>EJERCICIOS PROPUESTOS .....</b>	<b>64</b>
<b>HTML DINÁMICO (DHTML – DYNAMIC HTML).....</b>	<b>65</b>
 <b>CLASE NO. 5 .....</b>	 <b>65</b>
<b>¿QUE ES DHTML? .....</b>	<b>65</b>
<b>HOJAS DE ESTILO CSS .....</b>	<b>65</b>
Estilos .....	66
Capas .....	66
Aplicación directa de estilos en etiquetas.....	67
Redefinición de etiquetas .....	67
Etiqueta <style>...</style> .....	67
Separar CSS de HTML .....	69
Estilos de clase.....	71
Estilos de capas.....	71
 <b>CLASE NO. 6 .....</b>	 <b>71</b>
<b>JAVASCRIPT .....</b>	<b>71</b>
Inserción de código Javascript en documentos HTML .....	72
Diálogos de Javascript .....	73
Finalización de instrucciones en Javascript.....	73

Declaración de Variables en Javascript .....	73
Inicialización de variables .....	74
Tipos de datos en Javascript.....	74
Manejo de comentarios.....	75
Operadores.....	75
Estructuras de control .....	77
Bifurcación de control .....	82
<b>CLASE NO. 7 .....</b>	<b>83</b>
Funciones definidas por el usuario.....	83
Arreglos .....	84
La Clase String .....	91
La clase Date para Fechas y Horas .....	95
La clase Math .....	96
Otras funciones de Javascript.....	99
Redireccionamiento de páginas .....	100
Objetos en Javascript .....	101
Archivos externos Javascript .js .....	102
Ejemplo general sobre Javascript .....	102
<b>EJERCICIOS PROPUESTOS .....</b>	<b>108</b>
<b>INTRODUCCIÓN AL LENGUAJE PHP.....</b>	<b>111</b>
<b>CLASE NO. 8.....</b>	<b>111</b>
<b>¿QUÉ ES PHP? .....</b>	<b>111</b>
<b>RESEÑA HISTÓRICA .....</b>	<b>111</b>
Versiones de PHP.....	112
Usos de PHP .....	113
<b>SERVIDOR WEB HTTP APACHE.....</b>	<b>113</b>
<b>BASES DE DATOS EN PHP .....</b>	<b>114</b>
MySQL.....	114
phpMyAdmin.....	114
<b>INSTALACIÓN Y CONFIGURACIÓN DE XAMPP .....</b>	<b>115</b>
XAMPP .....	115
Instalación de XAMPP .....	116
<b>PUBLICAR PÁGINAS EN EL SERVIDOR .....</b>	<b>117</b>
¿Dónde se ejecuta el código de PHP?.....	117
Trabajo con PHP.....	117
<b>DESARROLLO DE PÁGINAS WEB DINÁMICAS CON PHP .....</b>	<b>121</b>

<b>CLASE NO. 9 .....</b>	<b>121</b>
<b>ESTRUCTURA GENERAL DE UNA PÁGINA PHP .....</b>	<b>121</b>
Etiqueta PHP .....	121
<b>SALIDA DE INFORMACIÓN EN PHP .....</b>	<b>121</b>
<b>TIPOS DE DATOS EN PHP .....</b>	<b>122</b>
Enteros .....	122
Flotantes .....	122
String .....	122
Arrays .....	122
Objetos .....	122
<b>VARIABLES EN PHP .....</b>	<b>123</b>
<b>MANEJO DE COMENTARIOS .....</b>	<b>123</b>
<b>OPERADORES .....</b>	<b>123</b>
Operadores aritméticos .....	123
Operadores relacionales .....	124
Operadores lógicos o booleanos .....	124
Operadores aritméticos/asignación .....	125
<b>ESTRUCTURAS DE CONTROL .....</b>	<b>125</b>
Condicionales .....	126
Selector múltiple o estructura caso .....	127
Ciclos .....	129
<b>Bifurcación de control .....</b>	<b>132</b>
Instrucción exit .....	132
Instrucción break .....	132
Instrucción continue .....	132
Instrucción return .....	133
<b>PASO DE PARÁMETROS POR URL .....</b>	<b>133</b>
<b>Funciones para validar datos en PHP .....</b>	<b>135</b>
Función isset() .....	135
Funcion empty() .....	135
<b>MANIPULACIÓN DE CADENAS DE CARACTERES .....</b>	<b>135</b>
<b>MANIPULACIÓN DE FECHAS Y HORAS .....</b>	<b>136</b>
Funciones para el manejo de fechas y horas .....	136
Configurar la fecha y hora local .....	137
Formatos de fecha y hora de la función date .....	138
<b>ARREGLOS .....</b>	<b>140</b>

<b>CLASE NO. 10 .....</b>	<b>143</b>
<b>Funciones matemáticas .....</b>	<b>143</b>
Funciones para cálculos matemáticos en PHP .....	143
Números aleatorios .....	144
<b>FUNCIONES DEFINIDAS POR EL USUARIO .....</b>	<b>147</b>
Declaración de prototipos de función en PHP .....	147
Pasar parámetros por valor o por referencia .....	148
<b>BASES DE DATOS EN PHP. MYSQL .....</b>	<b>151</b>
<b>CLASE NO. 11 .....</b>	<b>151</b>
<b>CONCEPTOS GENERALES SOBRE BASES DE DATOS .....</b>	<b>151</b>
<b>Sistemas de Gestión de Base de Datos (SGBD).....</b>	<b>151</b>
<b>LENGUAJE ESTRUCTURADO DE CONSULTA SQL.....</b>	<b>152</b>
Consultas de selección .....	152
Funciones de agregación .....	152
Consultas de acción.....	153
<b>MySQL .....</b>	<b>153</b>
Tipos de datos en MySQL.....	155
Claves foráneas en MySQL .....	158
<b>CLASE NO. 12 .....</b>	<b>160</b>
<b>phpMyAdmin .....</b>	<b>160</b>
<b>CLASE NO. 13 .....</b>	<b>172</b>
<b>Funciones de php para acceder a mysql .....</b>	<b>172</b>
<b>EJEMPLO SOBRE BASES DE DATOS CON PHP.....</b>	<b>173</b>
<b>CLASE NO. 14 .....</b>	<b>177</b>
<b>FUNCIONALIDADES DE PHP .....</b>	<b>182</b>
<b>CLASE NO. 15 .....</b>	<b>182</b>
<b>SESIONES .....</b>	<b>182</b>
Función session_start() .....	183
Función session_destroy() .....	183

session_id() .....	183
Ejemplo: manejo de sesiones .....	183
<b>CLASE NO. 16 .....</b>	<b>187</b>
<b>PHP FILE UPLOADS .....</b>	<b>187</b>
Ejemplo: subir imágenes al servidor.....	188
<b>CLASE NO. 17 .....</b>	<b>190</b>
<b>ENVÍO DE CORREO ELECTRÓNICO CON PHP .....</b>	<b>190</b>
Ejemplo.....	190
Configurar Mercury Mail/32 de XAMPP para Windows para enviar correos externos .....	191
<b>CLASE NO. 18 .....</b>	<b>192</b>
<b>EJERCICIOS PROPUESTOS .....</b>	<b>192</b>
<b>BIBLIOGRAFÍA Y CIBERGRAFÍA .....</b>	<b>196</b>
<b>Textos .....</b>	<b>196</b>
<b>Enlaces en Internet .....</b>	<b>196</b>

## LISTA DE TABLAS

Tabla 1. Planeador de clases.....	21
Tabla 2. Tipos de información en la etiqueta meta .....	37
Tabla 3. Tabla de Códigos de Caracteres en HTML .....	49
Tabla 4. Operadores aritméticos en Javascript .....	75
Tabla 5. Operadores relacionales o de comparación en Javascript .....	76
Tabla 6. Operadores lógicos o booleanos en Javascript .....	76
Tabla 7. Operadores aritméticos/asignación en Javascript .....	77
Tabla 8. Operadores aritméticos en PHP.....	124
Tabla 9. Operadores relacionales o de comparación en PHP .....	124
Tabla 10. Operadores lógicos o booleanos en PHP.....	125
Tabla 11. Operadores aritméticos/asignación en PHP .....	125
Tabla 12. Caracteres que son reconocidos en la cadena del parámetro formato .....	139
Tabla 13. Tipos de datos numéricos en MySQL y su tamaño de almacenamiento.....	156
Tabla 14. Tamaños y formatos de fecha en MySQL para el tipo timestamp.....	157
Tabla 15. Tipos de datos de fecha y hora en MySQL.....	157
Tabla 16. Tipos de datos de fecha en MySQL y su tamaño de almacenamiento .....	158
Tabla 17. Diferencia de almacenamiento entre los tipos Char y VarChar.....	158



## LISTA DE FIGURAS

Figura 1. Representación de una conexión a la red Internet .....	25
Figura 2. Logo de algunos de los navegadores mas populares: Mozilla Firefox, Microsoft Internet Explorer, Opera, Google Chrome, Netscape Navigator y Safari.....	28
Figura 3. Algunos servidores web: Apache, Cherokee, IIS7, Lighttpd y Thttpd .....	28
Figura 4. Arquitectura Cliente Servidor. Representación de la comunicación entre un cliente web (navegador) y un servidor web (Apache en la figura b) .....	29
Figura 5. “Granja de servidores”, enormes bancos de servidores en habitaciones acondicionadas en los cuales se alojan diferentes páginas web .....	30
Figura 6. Distintos tipos de TLD (Top Level Domain – Dominio de Nivel Superior) .....	31
Figura 7. Logos de algunos editores para página web: Geany, Programmers Notepad, Texpad, Notepad++, Crimson, Dreamweaver y Aptana Studio respectivamente .....	34
Figura 8. Código fuente HTML de una página web visualizado desde el navegador Firefox .....	36
Figura 9. Casi cualquier tipo de imagen puede mostrarse en una página web utilizando la etiqueta img.....	49
Figura 10. Diferentes recursos multimedia .....	50
Figura 11. Esquema para la creación de tablas HTML.....	53
Figura 12. Formulario HTML para el ingreso de datos personales en una página web .....	55
Figura 13. Árbol de directorios (sistema de archivos) para el sitio del ejemplo.....	59
Figura 14. Algunos logos empleados para identificar al lenguaje de las Hojas de Estilos CSS ....	66
Figura 15. Logos del Lenguaje de programación PHP .....	111
Figura 16. Logos del Servidor Web HTTP Apache.....	113
Figura 17. Logos del Servidor de bases de datos MySQL .....	114
Figura 18. Logo de phpMyAdmin .....	115
Figura 19. Logos del Servidor XAMPP.....	115
Figura 20. Panel de Control de XAMPP 1.6.8 .....	116
Figura 21. Subárbol de directorios de xampp/htdocs visualizado en el Explorador de Windows	118
Figura 22. Página principal de Localhost de XAMPP 1.7.1, en inglés y visualizada en Firefox ..	119

Figura 23. Caricatura de un archivador con un ratón (mouse) conectado representando una base de datos.....	151
Figura 24. Editor de línea de comandos (shell, DOS) de Windows. El indicador del sistema (prompt) muestra que se está ejecutando MySQL .....	154
Figura 25. Vista en Firefox de la página principal de phpMyAdmin (versión 2.9.0.2) en Localhost .....	160
Figura 26. Vista de la base de datos inmobiliaria .....	161
Figura 27. Vista de la Estructura de la tabla propiedades de la base de datos inmobiliaria .....	162
Figura 28. Advertencia por no especificar un índice para la tabla .....	163
Figura 29. Creación de un índice .....	164
Figura 30. Vista estructura de la tabla propiedades para su modificación .....	164
Figura 31. Establecer un campo entero (int) como autoincremental.....	165
Figura 32. Creación de la tabla ciudades en la base de datos inmobiliaria .....	165
Figura 33. Especificación de campos en la tabla ciudades .....	165
Figura 34. Advertencia por no especificar un índice para la tabla .....	166
Figura 35. Creación de un índice .....	166
Figura 36. Vista de la base de datos “inmobiliaria” .....	167
Figura 37. Tablas de la base de datos “inmobiliaria” .....	167
Figura 38. Vista de la estructura de la tabla “ciudades” .....	168
Figura 39. Vista Insertar de la tabla “ciudades”. Aquí puede ingresar los registros de la tabla ..	168
Figura 40. Opciones del cuadro de lista “y luego” .....	169
Figura 41. Enlaces para ir a las vistas Examinar y Estructura .....	169
Figura 42. Vista “Examinar” de la tabla “ciudades” .....	169
Figura 43. Tablas de la base de datos “inmobiliaria”: Seleccionar tabla “propiedades” .....	170
Figura 44. Vista de la estructura de la tabla “propiedades” .....	171
Figura 45. Vista “Insertar” de la tabla “propiedades” con valores ingresados .....	171

## PLANEADOR DE CLASES

SEMANA/FECHA	CONOCIMIENTOS, COMPRENSIONES Y RECOLECCIÓN DE EVIDENCIAS	ACCIONES DESARROLLAR A	FORMAS DE ENSEÑANZA Y APRENDIZAJE
PROGRAMADA			
<b>Semana 1</b> <b>Febrero 4-10</b>	Pacto Pedagógico y diagnóstico de aprendizajes previos	Al finalizar la clase estará en capacidad de: *Identificar y comprender los acuerdos y reglas de juego durante el desarrollo del submódulo *Identificar los aspectos en que se deben fortalecer algunos conceptos previos que se requieren en el semestre	Clase Magistral Ejercicios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.

<b>Semana 2</b> <b>Febrero 11-17</b>	<b>Scripts del lado del cliente con DHTML</b> 1. Identificación de las características de los scripts del lado del cliente a. Introducción • Reseña histórica de Internet (conceptos y evolución) • Servicios de Internet • Elementos típicos de una aplicación • Evolución de la programación de aplicaciones • Servidores Web y clientes Web • Conceptos sobre la programación Web • Tipos de Aplicaciones Web • Tecnologías para el desarrollo de aplicaciones Web	Al finalizar la clase estará en capacidad de: *Identificar la evolución y las tecnologías que componen Internet y la Web.	Clase Magistral
<b>Semana 3</b> <b>Febrero 18-24</b>	b. Introducción al HTML • Páginas y sitios web • Conceptos básicos de HTML • Archivos .html • Editores para la creación de páginas web • Publicar páginas web en el cliente o navegador web • Etiquetas fundamentales: Cabeceras y cuerpo de	Al finalizar la clase estará en capacidad de: *Aplicar el lenguaje HTML en la creación de páginas web. *Identificar las principales etiquetas, atributos y valores admitidos en HTML.	Clase Magistral Ejercicios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.

	<p>la página</p> <ul style="list-style-type: none"> <li>• Etiquetas para el manejo de texto</li> <li>• Atributos de las etiquetas y valores admitidos</li> <li>• Inserción de imágenes con HTML</li> <li>• Inserción de animaciones y video con HTML</li> <li>• Creación de tablas</li> </ul>		
<b>Semana 4</b> <b>Febrero 25-Marzo 3</b>	<ul style="list-style-type: none"> <li>• Hipervínculos</li> <li>• Marcos flotantes</li> <li>• Formularios en HTML</li> </ul>	<p>Al finalizar la clase estará en capacidad de:</p> <ul style="list-style-type: none"> <li>*Aplicar el lenguaje HTML en la creación de páginas web.</li> <li>*Utilizar hipervínculos para enlazar recursos de URL</li> <li>*Utilizar marcos flotantes para mostrar información</li> <li>*diseñar interfaces para solicitar información al usuario</li> </ul>	<p>Clase Magistral</p> <p>Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.</p>
<b>Semana 5</b> <b>Marzo 4-10</b>	<p>c. Extensiones HTML: DHTML (Dynamic HTML): Javascript y Hojas de Estilos CSS</p> <ul style="list-style-type: none"> <li>• Conceptos fundamentales</li> <li>• Modelo de Objetos de Documento (D.O.M.)</li> <li>• Inserción de código JavaScript en documentos HTML</li> <li>• Archivos .js</li> </ul>	<p>Al finalizar la clase estará en capacidad de:</p> <ul style="list-style-type: none"> <li>*Reconocer la importancia del lenguaje Javascript en páginas web (HTML)</li> <li>*Implementar Javascript en páginas web (HTML)</li> <li>*Identificar las principales</li> </ul>	<p>Clase Magistral</p> <p>Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.</p>

		<ul style="list-style-type: none"> <li>Ejemplos de funcionamiento de JavaScript en páginas Web</li> <li>Variables y constantes</li> <li>Tipos de datos</li> <li>Operadores</li> <li>Enviar contenido al navegador desde Javascript</li> <li>Cuadros de diálogo alert, prompt, confirm</li> <li>Leer datos del usuario con Javascript (procesar información de cuadros de diálogo y formularios)</li> <li>Estructuras de control de flujo: Definiciones, Condicionales (if, switch), Ciclos (for, while, do while)</li> <li>Bifurcación de control: break, exit, return, continue</li> </ul>	características de Javascript *Crear programas con Javascript	
<b>Semana 6</b> <b>Marzo 11-17</b>	<ul style="list-style-type: none"> <li>Funciones: conceptos generales</li> <li>Funciones del lenguaje para conversión de tipos</li> <li>Funciones definidas por el usuario</li> <li>Recolección de evidencias</li> </ul>	Al finalizar la clase estará en capacidad de: *Implementar funciones con el lenguaje Javascript en páginas web (HTML)		
<b>Semana 7</b> <b>Marzo 18-24</b>	<ul style="list-style-type: none"> <li>Arreglos</li> <li>Objetos en Javascript: String, Date y Math</li> <li>Redireccionamiento de páginas</li> <li>Recolección de evidencias</li> </ul>	Al finalizar la clase estará en capacidad de: *Utilizar distintos objetos de Javascript en la creación de aplicaciones web *Utilizar Javascript para controlar los accesos a las páginas web	Clase Magistral Ejercicios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.	

<b>Semana Abril 1-7</b>	<b>8</b>	<ul style="list-style-type: none"> <li>Recolección de evidencias</li> </ul> <p>Hasta esta semana Primera Recolección de Evidencias</p>	<p>Al finalizar la clase estará en capacidad de:</p> <ul style="list-style-type: none"> <li>*Entregar las actividades propuestas como evidencia de aprendizaje</li> </ul>	<p>Recolección de pruebas de conocimiento, desempeño y producto</p>
<b>Semana Abril 8-14</b>	<b>9</b>	<ul style="list-style-type: none"> <li>Introducción a las Hojas de Estilos CSS</li> <li>Inserción de código CSS en documentos HTML</li> <li>Archivos .css</li> <li>Definición de estilos de etiqueta</li> <li>Definición de estilos de clase</li> <li>La etiqueta &lt;span&gt; para aplicar CSS</li> <li>La etiqueta &lt;div&gt; para maquetar con CSS</li> <li>Combinar Javascript con CSS</li> </ul>	<p>Al finalizar la clase estará en capacidad de:</p> <ul style="list-style-type: none"> <li>*Identificar la importancia del lenguaje de Hojas de Estilos CSS</li> <li>*Identificar las formas de aplicar CSS en páginas HTML, así como sus principales propiedades y valores</li> <li>*Utilizar CSS para mejorar el aspecto y funcionalidad de las páginas HTML</li> </ul>	<p>Clase Magistral Ejercicios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.</p>
<b>Semana Abril 15-21</b>	<b>10</b>	Semana Académica (Reunión de Colectivos-Elaboración Mapa de Riesgo)		
<b>Semana Abril 22-28</b>	<b>11</b>	<p><b>Scripts Del Lado Del Servidor Con El Lenguaje Php</b></p> <p>2. Identificación de las características de los scripts del lado del servidor</p> <p>a. Conceptos básicos sobre servidores web</p> <ul style="list-style-type: none"> <li>Servidores Web</li> <li>Servidor web Apache</li> <li>Otros servidores web: IIS (Internet Information</li> </ul>	<p>Al finalizar la clase estará en capacidad de:</p> <ul style="list-style-type: none"> <li>*Reconocer la importancia del lenguaje PHP en las aplicaciones web</li> <li>*Identificar las tecnologías básicas para el desarrollo de aplicaciones con PHP</li> <li>*Identificar las principales</li> </ul>	<p>Clase Magistral Ejercicios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.</p>

	<p>Services), Cherokee, lighttpd, tthttpd</p> <ul style="list-style-type: none"> <li>• Distribuciones para PHP: XAMPP, AppServ, Wamp, EasyPHP</li> <li>• Servidor XAMPP (Paquete que distribuye libre y gratuitamente Apachefriends.org, incluye, entre otras aplicaciones: Servidor Web Apache, Servidor de base de datos MySQL, Lenguaje PHP, Servidor SMTP Mercury Mail, Servidor FTP FileZilla y gestor de base de datos PHPMyAdmin)</li> <li>• Descarga e Instalación de XAMPP</li> <li>• Ejecución de páginas en el cliente /Ejecución de páginas en el servidor</li> <li>• Páginas web dinámicas</li> <li>• Ejemplos de funcionamiento de páginas PHP</li> </ul> <p>b. El lenguaje de programación web PHP</p> <ul style="list-style-type: none"> <li>• Definición y conceptos básicos sobre el lenguaje PHP</li> <li>• Archivos .php</li> <li>• Variables y constantes</li> <li>• Tipos de datos</li> <li>• Operadores</li> <li>• Envío de contenido al navegador con PHP</li> </ul>	<p>características de PHP</p> <ul style="list-style-type: none"> <li>*Instalar el entorno de trabajo de PHP</li> <li>*Iniciar el servidor web Apache para el trabajo con PHP</li> <li>*Crear programas con PHP</li> </ul>	
--	--	---	--



<b>Semana 12</b> <b>Abril 29 - Mayo 5</b>	<ul style="list-style-type: none"> <li>• Estructuras de control: condicionales (if, switch), ciclos (for, while, do while)</li> <li>• Bifurcación de control: break, exit, return, continue</li> <li>• Paso de variables (parámetros) entre páginas por URL y por formulario</li> <li>• Funciones para el manejo de fechas y horas</li> <li>• Funciones definidas por el usuario</li> <li>• Arreglos</li> <li>• Funciones matemáticas.</li> <li>• Generación de números aleatorios</li> <li>• Recolección de evidencias</li> </ul>	<p>Al finalizar la clase estará en capacidad de:</p> <ul style="list-style-type: none"> <li>*Reconocer las estructuras de control del lenguaje PHP</li> <li>*Trasferir datos entre páginas</li> <li>*Configurar fechas y horas en las páginas web</li> <li>*Utilizar funciones en el diseño de aplicaciones web</li> <li>*Reconocer las principales funciones matemáticas y utilizarlas en la creación programas con PHP</li> </ul>	<p>Clase Magistral Ejercicios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.</p>
<b>Semana 13</b> <b>Mayo 6-12</b>	<p><b>Gestión de bases de datos con el motor de base de datos MySQL.</b></p> <p>3. Utilización del motor de Bases de datos MySQL para la gestión de bases de datos</p> <p>a. Diseño e implementación de bases de datos con MySQL</p> <ul style="list-style-type: none"> <li>• Conceptos generales sobre bases de datos</li> <li>• Motor de bases de datos MySQL</li> <li>• Gestores de bases de datos para MySQL: PHPMyAdmin, MySQL-Front, Consola de MySQL</li> <li>• Repaso general de SQL</li> <li>• Consultas de selección de registros (select from where)</li> </ul>	<p>Al finalizar la clase estará en capacidad de:</p> <ul style="list-style-type: none"> <li>*Reconocer la importancia del motor de bases de datos MySQL para la creación de aplicaciones web</li> <li>*Identificar las tecnologías utilizadas para trabajar con MySQL</li> <li>*Aplicar los principales conceptos de bases de datos y SQL para la gestión de información en MySQL</li> </ul>	<p>Recolección de pruebas de conocimiento, desempeño y producto</p>

	<ul style="list-style-type: none"> <li>• Funciones de agregación (sum, count, avg, max, min)</li> <li>• Consultas de acción (insert into, update set, delete)</li> <li>• Creación de tablas</li> <li>• Campos o atributos</li> <li>• Registros o tuplas</li> <li>• Tipos de datos</li> <li>• Índices</li> <li>• Claves principales y foráneas - Integridad referencial</li> <li>• Propiedades de los campos</li> <li>• Importar y exportar bases de datos desde MySQL</li> </ul>		
<b>Semana 14</b> <b>Mayo 13-19</b>	<ul style="list-style-type: none"> <li>• Recolección de evidencias</li> </ul> Hasta esta semana Segunda Recolección de Evidencias	Al finalizar la clase estará en capacidad de: *Entregar las actividades propuestas como evidencia de aprendizaje	Clase Magistral Ejercicios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.
<b>Semana 15</b> <b>Mayo 20-26</b>	<ul style="list-style-type: none"> <li>• Acceso a los registros desde PHP</li> <li>c. Creación de aplicativos con PHP y MySQL</li> <li>• Mostrar información de una base de datos en una página web</li> <li>• Cambiar los contenidos de una página web dinámicamente mediante el uso</li> </ul>	Al finalizar la clase estará en capacidad de: *Reconocer la forma en que PHP accede a los datos almacenados en MySQL *Aplicar los principales conceptos de bases de datos, SQL y de PHP para la creación de aplicaciones para gestionar información	Clase Magistral Ejercicios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.

	consultas SQL • Principales funciones de PHP para acceder a la información almacenada en el motor de bases de datos MySQL • Taller: Sistema para el manejo de usuarios (Ingreso, Modificación, Retiro, Consulta, Listado)	almacenada en MySQL	
<b>Semana 16</b> <b>Mayo 27 - Junio 2</b>	• Taller: Sistema para el manejo de usuarios (Ingreso, Modificación, Retiro, Consulta, Listado)	Al finalizar la clase estará en capacidad de: *Aplicar los principales conceptos de bases de datos, SQL y de PHP para la creación de aplicaciones para gestionar información almacenada en MySQL	Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.
<b>Semana 17</b> <b>Junio 3-9</b>	<b>Utilización de funcionalidades del lenguaje PHP</b>  4. Uso de funcionalidades de PHP a. Añadir seguridad a los sitios web mediante el uso de sesiones • Conceptos generales sobre las sesiones • Variables de sesión • Funciones para el manejo de sesiones • Añadir seguridad a las páginas mediante el uso de sesiones • Taller: sistema para iniciar sesión, a partir de la base de datos de usuarios	Al finalizar la clase estará en capacidad de: *Identificar diferentes funcionalidades que ofrece el lenguaje PHP *Aplicar las funciones apropiadas en el manejo de sesiones con PHP para agregar seguridad a las aplicaciones web	Clase Magistral Ejercios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.

<b>Semana 18</b> <b>Junio 10-16</b>	<p>b. Subir archivos al servidor: PHP File Uploads</p> <ul style="list-style-type: none"> <li>• Configuración del formulario para seleccionar los archivos de subida al servidor</li> <li>• Funciones para cargar archivos en el servidor</li> <li>• Taller: sistema para subir imágenes al servidor y crear una galería a partir ellas (pueden estar asociadas a la base de datos usuarios para subir, por ejemplo, la foto de éstos)</li> </ul> <p>c. Envío de correo electrónico con PHP</p> <ul style="list-style-type: none"> <li>• Servidores SMTP</li> <li>• Configuración del servidor SMTP Mercury Mail</li> <li>• Función mail para el envío de correo electrónico</li> <li>• Taller: Sistema para enviar correo masivo a partir de la base de datos de usuarios</li> <li>• Recolección de evidencias</li> </ul>	<p>Al finalizar la clase estará en capacidad de:</p> <ul style="list-style-type: none"> <li>*Identificar diferentes funcionalidades que ofrece el lenguaje PHP</li> <li>*Utilizar las funciones apropiadas para subir archivos al servidor y enviar correos desde una aplicación web</li> </ul>	<p>Clase Magistral Ejercicios prácticos realizados por el docente y ejercicios prácticos realizados por el estudiante con la orientación del docente.</p>
<b>Semana 19</b> <b>Junio 17-23</b>	<ul style="list-style-type: none"> <li>• Recolección de evidencias</li> </ul> <p>Hasta esta semana Tercera Recolección de Evidencias</p>	<p>Al finalizar la clase estará en capacidad de:</p> <ul style="list-style-type: none"> <li>*Entregar las actividades propuestas como evidencia de aprendizaje</li> </ul>	<p>Recolección de pruebas de conocimiento, desempeño y producto</p>

<b>Semana</b> <b>Junio 24-30</b>	<b>20</b>	Planes de Mejoramientos	Al finalizar la clase estará en capacidad de: *Entregar las actividades propuestas como evidencia de aprendizaje	Realización de pruebas de conocimiento, desempeño y producto
-------------------------------------	-----------	----------------------------	--	--

Tabla 1. Planeador de clases

## PACTO PEDAGÓGICO

### CLASE No. 1 PACTO PEDAGÓGICO

Cada docente que recibe un grupo cuenta con la posibilidad, y lo que puede entenderse a su vez como una herramienta pedagógica, de definir los límites de su clase y de pactar, de entrada, las condiciones que permitirán el logro de los objetivos propuestos, la adquisición de la responsabilidad individual y colectiva frente a éstos, y la regulación de los vínculos e interacciones grupales para una favorable convivencia al interior del grupo.

Al establecimiento explícito de dichas condiciones en el aula de clase se le puede denominar PACTO PEDAGÓGICO, el cual se define como el acuerdo inaugural del curso, pues consiste principalmente en aquel compromiso que de antemano se “PACTA” entre el docente y sus estudiantes, y recíprocamente entre los estudiantes y el docente desde el primer día de clase, lo que determina simbólicamente el devenir futuro del curso.

Es importante que el docente dedique la primera hora de clase de su submódulo, al inicio de cada semestre, para el establecimiento del PACTO PEDAGÓGICO, ya que es el momento oportuno para sentar las reglas de juego. Se espera entonces, que el docente una vez se presente al grupo, de inicio a la determinación de ciertos aspectos, dentro de los cuales deben estar contemplados los siguientes:

#### 1. **Presentación del docente**

#### 2. **Conducta de entrada:** El primer día los estudiantes quieren básicamente tres cosas:

- **Relaciones:**

Los estudiantes quieren sentirse cómodos con sus compañeros de clase. Sentir que hacen parte del grupo y que pueden desarrollar una red de amigos. Por lo tanto lo invitamos a:

- a. Desarrollar una actividad corta donde ellos realicen una significativa presentación de lo que son, lo que hacen y quieren ser, para que conozcan a sus compañeros.
- b. Reconocer a los estudiantes a través del aplauso, la felicitación o unas palabras motivadoras, teniendo en cuenta a aquellos que trabajan y estudian, que son padres de familia; o que tienen una característica específica de acuerdo con la dinámica desarrollada en el grupo; hacerlos sentir que son valorados por la decisión de transformar su vida a través de la formación técnica. Algunas frases motivadoras pueden ser: HOY EMPIEZAS A TRANSFORMAR TU VIDA; BIENVENIDO, TE ESTÁBAMOS ESPERANDO.

- **Visión de Programa:**

Este es el motivador más poderoso de todos. Parte de esa visión es el mercado laboral, y ese día los estudiantes están muy abiertos en mirar lo que ellos conocen, y lo que es y no es realista con respecto al programa que eligieron y al plan de estudios del mismo. Ellos quieren saber qué habilidades y competencias necesitarán tener para lograr una carrera laboral exitosa. Por tal razón esta es la oportunidad de dar a conocer de manera general el perfil del programa técnico.

- **Saber cómo Triunfar:**

Los estudiantes quieren saber que se requiere para ser exitosos con su programa académico. Quieren saber qué tan diferente va a ser de su experiencia en los colegios donde estuvieron u otras instituciones de educación superior a las que hayan asistido antes.

Para lograr una alta motivación en los estudiantes, el primer día es el mejor para que el docente llegue a la base de su motivación. El primer día tenemos su total atención, se empieza de cero y hay que

aprovecharlo. Dado que estamos hablando de educación para el trabajo y el desarrollo humano, la oportunidad para crear una relación de por vida es altísima ya que los estudiantes verán el aprendizaje aplicado desde el primer día.

3. **Dimensión disciplinaria:** la puntualidad y la asistencia tendrán seguimiento por medio de la “llamada a lista” los primeros cinco minutos de la clase. No se permite ingerir alimentos, comer chicles o realizar ventas durante el desarrollo de la clase. También interferir negativamente con el adecuado desenvolvimiento del submódulo, manifestando conductas o comportamientos que atenten contra el logro de los objetivos del trabajo propuesto.
4. **La asistencia:** aclarar la importancia de la asistencia a clase o en el caso de que se presente la inasistencia del alumno, justificarla con una excusa escrita (médica o laboral) o en caso de ser verbal, demostrar su validez.
5. **La puntualidad:** definir la hora en que se comenzará y finalizará la clase regularmente. Los estudiantes que tengan dificultades para cumplir estos horarios pueden aclararlo desde el principio y comprometerse a mantenerse al día en el desarrollo del submódulo.
6. **La evaluación:** oficializar la metodología de la recolección de evidencias, los momentos empleados para este proceso y las dimensiones que serán evaluadas: conocimiento, desempeño y producto, así como el porcentaje respectivo. De igual forma, anunciar el tiempo de devolución de las evidencias, ser muy claro en los plazos acordados para la entrega de trabajos por parte de los estudiantes y fijar las pautas y derroteros con los cuales deben ser presentados. Aclarar el tipo de sanciones cuando se presenten eventuales fraudes.
7. **El conducto regular:** establecer las instancias regulares a las cuales el estudiante podrá recurrir en el caso de presentarse algún inconveniente o desacuerdo significativo.
8. **Los recursos logísticos:** velar por el uso adecuado de los materiales didácticos, los equipos, los muebles y elementos del salón de clase.
9. Los estudiantes del primer semestre tendrán una inducción especial en la cual se les presentará el perfil, los objetivos, la justificación y las áreas de estudio del programa que van a realizar, igualmente se clarificarán las dudas que surjan al respecto.
10. **MOMENTOS DE RECOLECCIÓN DE EVIDENCIAS:** Los estudiantes tendrán 3 momentos de recolección de evidencias de conocimiento, desempeño y producto, con sus respectivos mejoramientos:
  - La primera en las primeras 6 semanas
  - La segunda entre la 7 y la semana 12
  - La tercera entre la 13 y la semana 18
11. Los estudiantes pueden habilitar en los siguientes casos:
  - Si pierden una instancia de un submodulo(conocimiento, desempeño o producto)
  - Si pierde dos instancias de un submódulo
  - Si pierde dos instancias en submódulos distintos
  - Si pierde tres instancias en dos submodulos distintos(dos en uno y uno en otro)
  - Si pierde cuatro instancias en dos submódulos distintos(dos en uno y dos en otro)

En los demás casos debe repetir los submódulos en donde haya perdido las instancias de evaluación.

## INTRODUCCIÓN

Esta guía tiene como finalidad, ser un material de apoyo para los estudiantes de programación web, y además servir como una referencia de consulta, en donde encontrará, además de los conceptos teóricos acerca de la creación de sitios web, ejemplos resueltos y ejercicios propuestos al final de cada capítulo.

El texto se divide en varias partes, partiendo del diseño de páginas y sitios web con *HTML*, en donde se estudia lo relacionado a las interfaces de usuario y los elementos (*etiquetas*) más importantes a emplear en la creación de sitios. Siguiendo luego con *DHTML*, se muestra como utilizar las poderosas características de *Javascript* y *CSS* para crear sitios web más funcionales que mejoran sustancialmente las interfaces de usuario, sin tener que acceder a los recursos de un servidor. Posteriormente se estudia la creación de páginas web dinámicas con *PHP* y como trabajar con bases de datos *MySQL* utilizando este lenguaje.

Si el estudiante es principiante en el desarrollo web, es aconsejable leer la guía en el orden indicado, aunque puede estudiar primero los temas relacionados con PHP y MySQL y luego los de DHTML (Javascript y CSS), dependiendo del interés. En cualquier caso debe haber estudiado el tema relacionado con HTML.

Otro aspecto importante a tener en cuenta, tiene que ver con la tarea de hacer tanto los ejemplos resueltos como los ejercicios propuestos, por parte del estudiante, pues hacerlos, fortalecen los conceptos teóricos y permiten ganar experiencia al enfrentarse a diferentes problemas, además, allí encontrará explicaciones y notas adicionales de los conceptos definidos.

Cabe anotar además que el enfoque de este texto está orientado a la utilización de herramientas de *software libre* (*free software*), por una parte, por lo apasionante de este mundo, y por otro, por las enormes ventajas que trae el desarrollo con este tipo de tecnologías. También emplearemos herramientas de libre uso y gratuitas (*Freeware* o *Software gratuito*), y que no son software libre<sup>1</sup>.

---

<sup>1</sup> No sobra decir que este texto también es libre, esto es, puede ser copiado, modificado y redistribuido según la GNU GFDL (Licencia de Documentación Libre de GNU, una "hermana" de la Licencia Pública General GNU/GPL que protege el software de intentos de apropiación que restrinjan sus libertades a los usuarios); esto siempre y cuando se conserve la autoría del mismo (Jaime Montoya) y las referencias bibliográficas que allí aparecen.



## CONCEPTOS GENERALES

### INTERNET

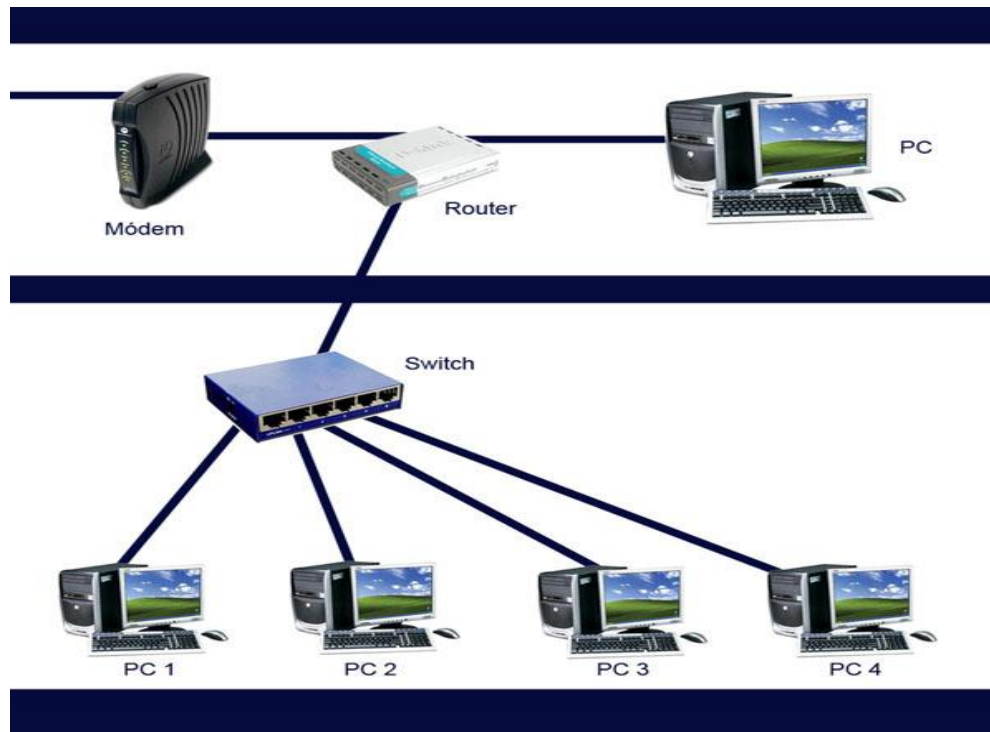


Figura 1. Representación de una conexión a la red Internet

Es una red de redes que permite la comunicación entre dos o más computadores alrededor del mundo. Internet es un conjunto descentralizado de redes de comunicación interconectadas, que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial. Sus orígenes se remontan a 1969 como un proyecto del departamento de defensa de Estados Unidos (E.U.) cuando se estableció la primera conexión de computadoras, conocida como ARPANET, entre tres universidades en California y una en Utah. Internet se compone de un conjunto de servicios, algunos de ellos se comentan a continuación.

### SERVICIOS DE INTERNET

Internet está compuesta de una serie de tecnologías que nos permiten hacer una gran cantidad de tareas; estas tecnologías se conocen como servicios de Internet y han ido apareciendo desde su creación, uno de los más conocidos es la *WWW*, que comúnmente se confunde con la Internet misma, pero que solo es una de las partes que constituyen la red. Veamos cuales son estos servicios:

**Correo electrónico (e-mail).** Utiliza el protocolo *SMTP (Simple Mail Transfer Protocol)*, para la recepción y envío de correspondencia electrónica. Existen empresas que prestan el servicio de creación y manejo de cuentas de correo gratuitas como Yahoo! Mail de Yahoo!, Outlook (antes Hotmail) de Microsoft y Gmail de Google entre otros, mientras que otras cobran por dicho servicio.

En un informe de 2012 Yahoo! Mail continúa siendo la empresa que registra mayor número de usuarios en Estados Unidos seguido de cerca por Outlook y Gmail, mientras que a nivel mundial el orden es Gmail, Outlook y Yahoo!<sup>2</sup>

**IRC (Internet Relay Chat).** Fue el primer servicio que apareció y nos permite enviar y recibir mensajería instantánea, o en otras palabras, conversar en línea. Este servicio trabaja en tiempo real y actualmente está integrado en cuentas de correo; además de texto, intercambia sonido e imagen y permite transferir archivos. Se conocen comúnmente como chat, algunos son públicos y otros privados. Algunos sistemas de chat populares son: Yahoo! Messenger, Hotmail Messenger (Windows Live), Google Talk, LatinChat, etc.

**Emulación de terminal TELNET.** Se utiliza para conectarse a equipos remotos mediante la red emulando una terminal del equipo al que se realiza la conexión. Como ejemplo podríamos citar las acciones que debían realizarse cuando accedíamos a Internet por telefonía conmutada, en donde era necesario especificar un número de teléfono, un usuario y una contraseña. Con las conexiones modernas, este proceso se realiza automáticamente y es invisible a los usuarios.

**Transferencia de archivos.** Utiliza el protocolo FTP (File Transfer Protocol), se usa para enviar o recibir archivos (de cualquier tipo) entre dos equipos conectados a la red. Los sistemas FTP pueden estar dentro de los navegadores como extensiones o ser aplicaciones independientes. La tarea de descargar o subir archivos no siempre es realizada por FTP, ésta también puede ser hecha por HTTP; se diferencia en que las descargas por FTP no se pierden al apagar la máquina, solo se pausan, mientras que las realizadas por HTTP si lo hacen. Algunos programas comunes para FTP son eMule, KazaA, eDonkey, Ares, FireFTP (componente para Firefox), FileZilla, etc.

**Servicio de nombres de dominio DNS (Domain Name Service).** En realidad es un servicio que raramente se utiliza solo; es usado por otros, como TELNET, FTP, WWW, etc. para conseguir las direcciones IP (numéricas) de las máquinas remotas a partir de los nombres de dominio.

**Gopher.** Un servicio de información basado en servidores y que sirve de interfaz para otros servicios de información.

**WAIS (Wide Area Information Service).** Se trata de otro servicio de información basado en bases de datos de archivos que permiten su rápida localización.

**Finger.** Es un servicio de identificación de usuarios.

**WWW (World Wide Web o W3).** Un servicio basado en HTTP (HyperText Transfer Protocol o Protocolo de Transferencia de Hipertexto), el último (de los 5 principales, aparece en 1990) y más popular, integra actualmente a la mayoría de servicios. Gracias a www, podemos visualizar los contenidos basados en HTML (HyperText Markup Language – Lenguaje de Marcado de Hipertexto), o en otras palabras, páginas web.

Su creador, el físico inglés Tim Barners-Lee, ante la necesidad de distribuir e intercambiar información acerca de sus investigaciones de una manera más efectiva, desarrolló las ideas fundamentales que estructuran la web. Él y su grupo crearon en su proyecto en 1989 y 1990, cinco tecnologías para el funcionamiento de la web:

---

<sup>2</sup> Wikipedia, la enciclopedia libre. Correo Yahoo!. En Internet: [http://es.wikipedia.org/wiki/Correo\\_Yahoo!](http://es.wikipedia.org/wiki/Correo_Yahoo!)

- El Localizador Uniforme de Recursos o sistema para la localización de objetos *URL* (*Uniform Resource Locator*)
- El lenguaje *HTML* (*HyperText Markup Language* o Lenguaje de Marcado de Hipertexto), el lenguaje predominante en la creación de páginas web
- Un navegador web donde visualizar los documentos basados en *HTML* llamado *WorldWideWeb* desarrollado con *NEXTSTEP*
- El protocolo *HTTP* (*HyperText Transfer Protocol*),
- El servidor web *httpd* (*HyperText Transfer Protocol daemon*)

En el pasado, Tim Berners-Lee se ha opuesto a la creación de nombres de dominio nuevos como el '.xxx' y el '.mobi'. De hecho, cuando el '.mobi' nació, él era el mayor detractor. Él argumenta que todo el mundo debería de acceder a las mismas web, independientemente de si usase un ordenador o un móvil. Básicamente lo que no le gustaba a Tim del .mobi es que éste sería para que se accediese únicamente con los móviles, ya que él desarrolló la web como una forma de comunicación universal y no veía necesario el desarrollo del .mobi únicamente para el uso en móviles.

También hubo una pelea entre diferentes gobiernos y el *ICANN* (*Internet Corporation for Assigned Names and Numbers - Corporación de Internet para la Asignación de Nombres y Números*) sobre la propiedad de los nombres de los dominios, sobre todo con el ".com". Tim apoya que nadie tenga los nombres de los dominios, sino que estos sean un recurso público.

Tim también dejó claro que el nombre o la propiedad de los dominios no era el aspecto más importante en el proceso de estandarización, sino que eran más importantes los estándares de vídeo, codificación, estándares abiertos de comunicación de datos, subida de datos científicos y clínicos o la propagación de información entre países.

En 1994 entró en el Laboratorio de Ciencias de la Computación e Inteligencia Artificial del MIT (Massachusetts Institute of Technology). Se trasladó a EE.UU. y puso en marcha el *W3C* (*World Wide Web Consortium*), que dirige actualmente. El W3C es un organismo internacional de estandarización de tecnologías Web dirigido conjuntamente por el Instituto Tecnológico de Massachusetts (MIT), el ERCIM francés y la Universidad de Keiō en Japón. Este organismo decidió que todos sus estándares fuesen libres, es decir, que los pudiese utilizar todo el mundo libremente sin coste alguno, lo que sin lugar a dudas fue una de las grandes razones para que la Web haya llegado a tener la importancia que tiene hoy en día.

En su libro *Tejiendo la red*, publicado en 1999, Tim Berners-Lee explica por qué la tecnología web es libre y gratis. Se considera al mismo tiempo el inventor y el protector de la web.

Este texto está orientado a trabajar con algunas de las más importantes tecnologías web que existen actualmente y básicamente se concentrará en como desarrollar aplicativos para el servicio *www*.

**Sistema de ficheros de red NFS (Network File System).** Es un sistema que permite a equipos físicamente distantes, compartir discos y directorios mediante la técnica denominada *RPC* (*Remote Procedure Call*), que hace que tales recursos aparezcan como si estuvieran en el propio sistema.

**Servicios de Información de Red, NIS (Network Information Services).** También basados en RPC, permite que varios sistemas puedan compartir una misma base de datos situada remotamente; por ejemplo, varios sistemas pueden compartir bases de datos con el mismo archivo de seguridad (password file), lo que facilita su gestión centralizada.

**Servicios "R" tales como rlogin, rsh y otros.** Utilizan la idea de acuerdos entre sistemas (hosts trusting), que permite ejecutar comandos y otras órdenes en equipos remotos sin requerir un password.

## CLIENTE WEB



Figura 2. Logo de algunos de los navegadores mas populares: Mozilla Firefox, Microsoft Internet Explorer, Opera, Google Chrome, Netscape Navigator y Safari

Es un programa o aplicación capaz de interpretar código *HTML* y *DHTML* (scripts del lado del cliente), permitiendo visualizar diferentes contenidos, entre los que se tienen texto con formato, imágenes, etc. Es conocido como *Navegador* o *Browser*, algunos son: Mozilla Firefox, Netscape Navigator (fue el primer navegador comercial, el proyecto fue cancelado en 2008), Google Chrome, Microsoft Internet Explorer, Opera y Safari, entre otros.

## SERVIDOR WEB



Figura 3. Algunos servidores web: Apache, Cherokee, IIS7, Lighttpd y Thttpd

Es un programa que implementa el protocolo *HTTP (Hypertext Transfer Protocol)*. Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas Web o páginas *HTML (Hypertext Markup Language)*: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

Sin embargo, el hecho de que HTTP y HTML estén íntimamente ligados no debe dar lugar a confundir ambos términos. HTML es un formato de archivo y HTTP es un protocolo.

Cabe destacar el hecho de que la palabra servidor identifica tanto al programa como a la máquina en la que dicho programa se ejecuta. Existe, por tanto, cierta ambigüedad en el término, aunque no será difícil diferenciar a cuál de los dos nos referimos en cada caso. Aquí nos referiremos siempre a la aplicación, a no ser que se indique lo contrario.

Un servidor Web se encarga de mantenerse a la espera de peticiones HTTP llevadas a cabo por un cliente HTTP que solemos conocer como navegador. El navegador realiza una petición al servidor y éste le responde con el contenido (flujo HTML) que el cliente solicita. A modo de ejemplo, al teclear `http://www.wikipedia.org` en nuestro navegador, éste realiza una petición HTTP al servidor de dicha dirección. El servidor responde al cliente enviando el código HTML de la página; el cliente, una vez recibido el código, lo interpreta y lo muestra en pantalla. Como vemos con este ejemplo, el cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página; el servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma. Algunos servidores web importantes son: Apache, IIS (Internet Information Services, Tecnología de Microsoft), Cherokee. Otros servidores web, más simples pero más rápidos son: Lighttpd y Thttpd.

### REPRESENTACIÓN GRÁFICA DE LA ARQUITECTURA CLIENTE WEB – SERVIDOR WEB

Los conceptos combinados sobre clientes y servidores web, los cuales constituyen lo que se conoce como *Arquitectura Cliente Servidor*, se pueden resumir gráficamente de la siguiente manera en las figuras 4a y 4b:

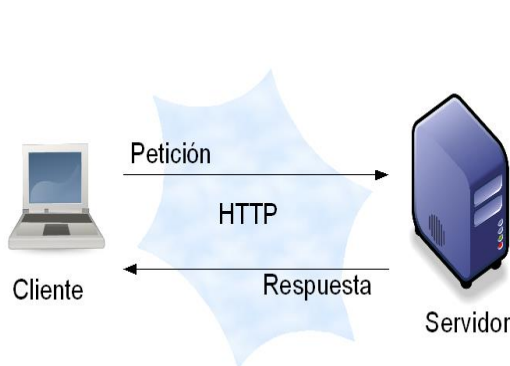


Figura a

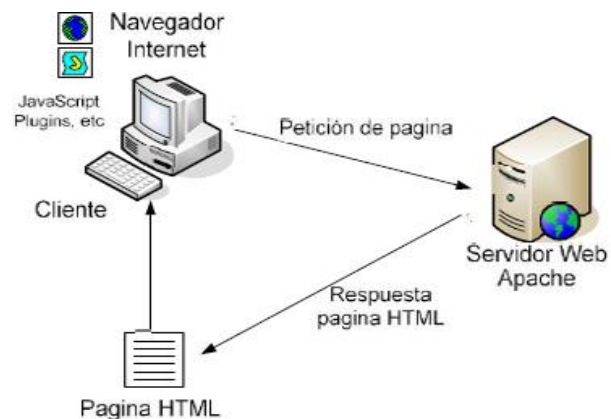


Figura b.

Figura 4. Arquitectura Cliente Servidor. Representación de la comunicación entre un cliente web (navegador) y un servidor web (Apache en la figura b)



## **APLICACIONES WEB**

Sobre el servicio Web clásico podemos disponer de aplicaciones Web. Éstas son fragmentos de código que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP. Hay que distinguir entre:

**Aplicaciones en el lado del cliente:** el cliente Web es el encargado de ejecutarlas en la máquina del usuario. Son las aplicaciones tipo JavaScript, JScript, VBScript, ActionScript (Flash), Java (applets), etc. El servidor proporciona el código de las aplicaciones al cliente y éste, mediante el navegador, las ejecuta. Es necesario, por tanto, que el cliente disponga de un navegador con capacidad para ejecutar aplicaciones (también llamadas scripts). Normalmente, los navegadores permiten ejecutar aplicaciones escritas en lenguaje Javascript y Java, aunque pueden añadirse más lenguajes mediante el uso de plugins.

**Aplicaciones en el lado del servidor:** el servidor Web ejecuta la aplicación; ésta, una vez ejecutada, genera cierto código HTML; el servidor toma este código recién creado y lo envía al cliente por medio del protocolo HTTP.

Las aplicaciones de servidor suelen ser la opción por la que se opta en la mayoría de las ocasiones para realizar aplicaciones Web. La razón es que, al ejecutarse ésta en el servidor y no en la máquina del cliente, éste no necesita ninguna capacidad adicional, como sí ocurre en el caso de querer ejecutar aplicaciones Javascript, Java u otro en el cliente de la máquina. Sin embargo, al crear sitios, lo mejor es combinar ambos tipos de aplicaciones, ya que cada tecnología tiene propósitos específicos, y uno no reemplaza al otro, por el contrario, se complementan para crear sitios robustos y potentes. Algunos lenguajes utilizados para desarrollar aplicaciones Web del lado del servidor son: PHP, ASP, ASP.NET, Perl, CGI, JSP (Tecnología Java).

## **ALOJAMIENTO WEB**

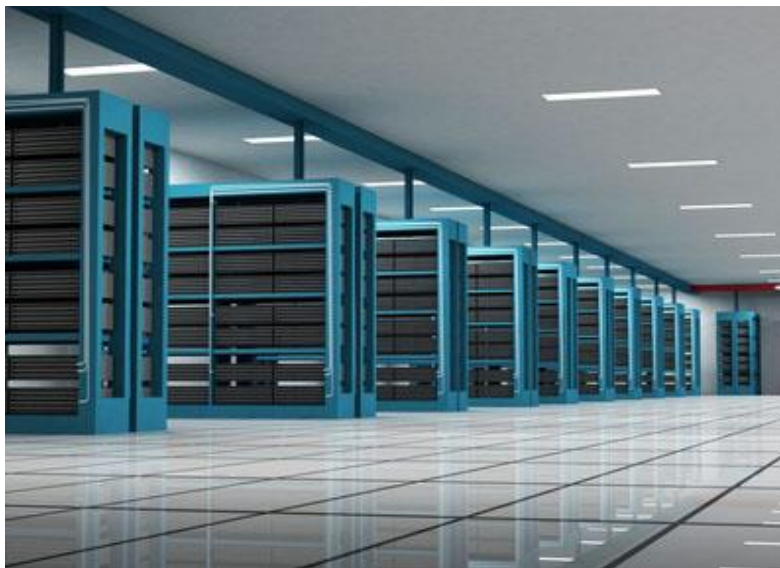


Figura 5. “Granja de servidores”, enormes bancos de servidores en habitaciones acondicionadas en los cuales se alojan diferentes páginas web

El alojamiento Web (en inglés, *Web Hosting*) es el servicio que provee a los usuarios de Internet de un sistema para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía Web. Los *Web Host* son compañías que proporcionan espacio de un servidor a sus usuarios.

## DOMINIO DE INTERNET



Figura 6. Distintos tipos de TLD (Top Level Domain – Dominio de Nivel Superior)

### ¿Qué es un dominio?

Un dominio de Internet es una red de identificación asociada a un grupo de dispositivos o equipos conectados a la red Internet.

Un dominio de Internet suministra nombres de equipo más fácilmente recordados que la dirección IP (*Internet Protocol*) numérica. Permiten a cualquier servicio moverse a otro lugar diferente en la topología de Internet, que tendrá una dirección IP diferente.

El propósito principal de los nombres de dominio en Internet y del sistema de nombres de dominio (DNS), es traducir las direcciones IP de cada nodo activo en la red, a términos memorizables y fáciles de encontrar. Esta abstracción hace posible que cualquier servicio (de red) pueda moverse de un lugar geográfico a otro en la red Internet, aun cuando el cambio implique que tendrá una dirección IP diferente.

Sin la ayuda del sistema de nombres de dominio, los usuarios de Internet tendrían que acceder a cada servicio web utilizando la dirección IP del nodo (por ejemplo, sería necesario utilizar `http://192.0.35.11` en vez de `http://misitio.com`).

### Ejemplo

A continuación se ilustra la diferencia entre una URL (Uniform Resource Locator o Localizador Uniforme de Recursos) y un nombre de dominio:

URL: `http://www.innovasistemas.co/`  
Nombre de dominio: `innovasistemas`

## **SUBDOMINIOS**

Un subdominio es un subgrupo o subclasificación del nombre de dominio el cual es definido con fines administrativos u organizativos, que podría considerarse como un dominio de segundo nivel. Normalmente es una serie de caracteres o palabra que se escriben antes del dominio. Un subdominio es un dominio dentro de otro dominio. Se puede tener subdominios ilimitados y no necesitan ser registrados.

En Internet se podría decir que el subdominio se utiliza para referirse a una dirección web que trabaja como un anexo (o sitio relacionado) de un dominio principal. Por ejemplo un subdominio puede representarse de la siguiente forma:

[http://subdominio.dominio\\_principal.com/](http://subdominio.dominio_principal.com/)

ó también:

[http://www.subdominio.dominio\\_principal.com/](http://www.subdominio.dominio_principal.com/)

Dentro de la estructura del servidor se refleja como un directorio, el cual contiene la información a mostrar, por lo cual también se puede acceder desde una URL como esta:

[http://www.dominio\\_principal.org/subdominio/](http://www.dominio_principal.org/subdominio/)

Los subdominios son proporcionados por las empresas que prestan servicios de hospedaje de páginas web (Hosting), aunque algunas lo proporcionan con distintas restricciones, ya sea por número de subdominios permitidos o por el servicio que prestan. Por ejemplo, existen empresas que regalan un subdominio al momento de registrar un blog con ellos (como Google con Blogspot) o algunas otras que prestan servicios de hospedaje gratuito. Otras empresas registran sitios relacionados con la organización para ejecutar otras aplicaciones. También encontramos compañías que venden hospedajes en subdominios. Generalmente al adquirir un hosting pago, se tiene la posibilidad de crear subdominios, el número depende de las características que ofrezca éste y de las necesidades que se tengan para contratar la cantidad que se requiera.

## **Ejemplos**

1.

URL: <http://gides.innovasistemas.co>

Dominio: innovasistemas

Subdominio: gides

2.

URL: <http://notas.cesde.edu.co>

Dominio: cesde

Subdominio: notas

## **DOMINIOS DE NIVEL SUPERIOR**

Cada nombre de dominio termina en un Dominio de Nivel Superior (TLD: Top Level Domain), que es siempre o bien uno de una pequeña lista de nombres genéricos (tres o más caracteres), o un código territorial de dos caracteres basado en la ISO-3166 (hay pequeñas excepciones y los nuevos códigos se integran caso por caso).

## **Ejemplos**



1.

URL: <http://www.innovasistemas.co>  
TLD: co

2.

URL: <http://www.google.com>  
TLD: com

3.

URL: <http://www.cesde.edu.co>  
TLD: edu

La *Internet Assigned Numbers Authority* (IANA) actualmente clasifica los dominios de nivel superior en tres tipos:

**TLD Geográfico (ccTLD):** Usados por un país o territorio dependiente. Tiene dos o más letras. Por ejemplo: co (correspondiente a Colombia), ar (correspondiente a Argentina), br (correspondiente a Brasil), mx (correspondiente a México), ve (correspondiente a Venezuela), etc.

**Dominios de Internet genérico (gTLD):** Usado (en teoría) por una clase particular de organizaciones. Por ejemplo com (para organizaciones comerciales), org (para organizaciones no gubernamentales), gov (para entidades del gobierno), edu (para instituciones educativas), mil (para entidades militares), tv (para empresas de televisión), co (para comunidades, comercio, compañías, etc.) y net (utilizado por diferentes tipos empresas) entre otros. Algunos de éstos están restringidos solo para ciertos tipos de organizaciones. En los últimos años han aparecido han aparecido nuevos TLD debido a la gran cantidad de páginas y sitios web en el mundo, algunos de los más recientes son co, mobi, info, biz, aero, name, pro, coop, museum y cc entre otros.

**TLD de infraestructura:** El TLD arpa es el único confirmado.

#### **Nota**

Los nombres de dominio no son sensibles a las mayúsculas.

#### **PÁGINA WEB**

Es un documento basado en HTML/XHTML accesible generalmente mediante el protocolo HTTP de Internet y que puede visualizarse en un cliente web (navegador).

#### **SITIO WEB**

Es una colección de páginas web relacionadas y comunes a un dominio o subdominio en la World Wide Web en Internet, junto con los demás archivos que éstas utilicen, como imágenes, animaciones, documentos, etc. En general, un sitio web es un contenedor de elementos que se relacionan e interactúan para mostrar cierto contenido en los navegadores web.

#### **EDITORES DE PÁGINA WEB**

Para escribir el código HTML o de otro lenguaje para la web, puede emplearse cualquier editor de texto (pero no un procesador de textos); algunos que pueden utilizarse son:

**Editores avanzados WYSIWYG<sup>3</sup>:** Ofrecen características avanzadas para el desarrollo de páginas y aplicativos web, algunos son: Aptana Studio (basado en el popular IDE Eclipse), Dreamweaver, Expression Web, PHP Designer, OpenOffice.org.

**Editores semi avanzados:** Ofrecen algunas funcionalidades que ayudan en la edición del código, entre ellos tenemos el Programmer's Notepad, Notepad++, Textpad, Crimson, Geany.

**Editores sencillos:** Ofrecen muy pocas opciones de edición de código o ninguna, algunos de éstos son: *gedit* de Linux, *Bloc de notas* de Windows, Edit del D.O.S.

Algunos de estos editores son productos de software libre y otros no, y no todos son gratuitos, por lo que tendrá que pagar por el uso de algunos de ellos. Decidir con cual editor trabajar es una cuestión de gusto que solo se define al experimentar con varios de éstos. Estos editores también pueden ser utilizados para editar código PHP, Javascript y CSS entre otros. Cabe aclarar que las páginas HTML son archivos con extensión *.html*, la cual debe especificar a la hora de guardar por primera vez.



Figura 7. Logos de algunos editores para página web: Geany, Programmers Notepad, Textpad, Notepad++, Crimson, Dreamweaver y Aptana Studio respectivamente

<sup>3</sup>**WYSIWYG** es el acrónimo de *What You See Is What You Get* (en inglés, "Lo Que Ves Es Lo Que Obtienes"). Se aplica a los procesadores de texto y otros editores de texto con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final.

## EL LENGUAJE HTML

### INTRODUCCIÓN

*HTML*, siglas de *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un *script* (por ejemplo Javascript), el cual puede afectar el comportamiento de los navegadores web y otros procesadores de HTML.

HTML también es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores).

El lenguaje HTML está compuesto de unos elementos llamados **etiquetas**, que se encierran entre los símbolos menor y mayor (<,>), éstos se encargan de establecer ciertos parámetros en las páginas (cabecera del documento) y modificar la apariencia del documento (cuerpo de la página); las etiquetas pueden contener **atributos**, que cambian su apariencia por medio de **valores** que se encierran dentro de comillas. La sintaxis utilizada es:

`<nombre_etiqueta atributo1="valor" atributo2="valor"...>`

*Elementos afectados por la etiqueta*

...

`</nombre_etiqueta>`

### Notas

- La etiqueta que se encuentra al final: `</nombre_etiqueta>`, indica cierre o finalización de ésta, así, lo que esté entre el par de etiquetas del mismo tipo, será afectado por ella. Sin embargo, algunas etiquetas no poseen una etiqueta de cierre, para lo cual se recomienda poner una barra inclinada (/ o slash) al final de ésta, así:  
`<nombre_etiqueta atributo1="valor" atributo2="valor".../>`
- Algunas etiquetas no poseen atributos y en algunos casos no los requieren, si los hay, el orden no importa siempre y cuando se ubiquen después del nombre de la etiqueta. Se debe separar con un espacio en blanco cada atributo dentro de la etiqueta.
- Las etiquetas pueden anidarse pero no solaparse. Por ejemplo: (Las etiquetas utilizadas se comentan más adelante)  
- `<b><i>Hola, bienvenidos a las páginas HTML </b></i>`  
Aquí se presenta un solapamiento de etiquetas, esto es una mala técnica que debe evitarse.  
- `<b><i>Hola, bienvenidos a las páginas HTML </i></b>`  
Aquí la etiqueta `<i>` se encuentra anidada dentro de la etiqueta `<b>`. Esto significa que la primera etiqueta en abrir será la última en cerrar. También cabe anotar, que hay que tener cuidado al anidar las etiquetas, ya que no todas lo admiten, dependiendo del tipo de

etiqueta utilizada y de la sección donde se haga; en la práctica se ganará experiencia para aplicar correctamente todos los elementos del lenguaje HTML.

En resumen, podemos decir que las etiquetas HTML son elementos que permiten darle la apariencia, que hasta cierto punto queremos, a las páginas Web que desarrollamos. Por ejemplo, si queremos mostrar un texto en negrita, realizamos lo siguiente:  
<b>Este texto aparecerá en negrita</b>

El texto que está encerrado entre <b> </b> aparecerá en negrita al publicar una página que contenga dicho código, ya que este es el objetivo de dicha etiqueta.

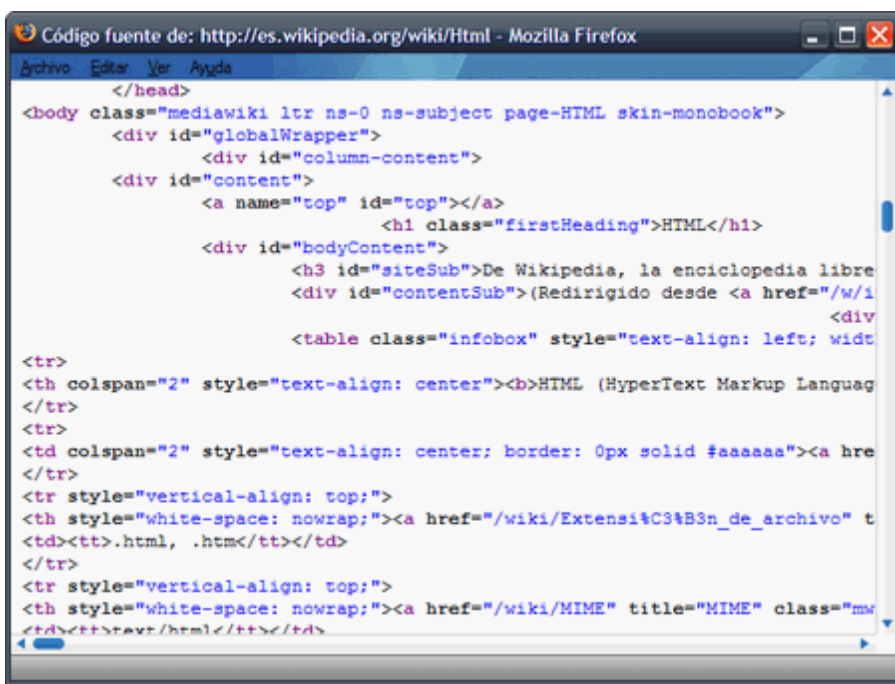


Figura 8. Código fuente HTML de una página web visualizado desde el navegador Firefox

## ETIQUETAS HTML

A continuación se presentan las principales etiquetas del HTML y se ilustra su uso con un ejemplo.

### Etiqueta de inicio y fin de página

#### Etiqueta <html>...</html>

Indica el inicio y fin de un documento HTML. Generalmente, es la primera (y última) etiqueta que encontramos en una página HTML.

### Etiquetas de cabecera

### **Etiqueta <head>...</head>**

Especifica el encabezado de una página HTML. Entre ellas pueden indicarse hojas de estilos CSS, scripts, vínculos a otros archivos, el título de la página, información del sitio, etc.

### **Etiqueta <title>...</title>**

Especifica el título de la página Web; éste aparecerá en la barra de título de la ventana del navegador (o en la pestaña donde se abra la página). Se ubica en la cabecera del documento.

### **Etiqueta <meta/>**

La etiqueta *<meta/>* se utiliza para añadir información sobre la página. Esta información puede ser utilizada por los buscadores. Éstos consultan la información de la etiqueta *<meta>* de las páginas, buscando coincidencias con lo que el usuario pretende encontrar.

A través de esta etiqueta pueden especificarse los atributos *name* y *content*. El atributo *name* indica el tipo de información, y el atributo *content* indica el valor de dicha información.

Para indicar el tipo de información podemos utilizar cualquier palabra que deseemos, como puede ser "Autor", "Palabras clave", "Descripción", etc. Pero debido a que la mayoría de buscadores están en inglés, es preferible que el tipo de información se especifique en inglés.

Los tipos de información más utilizados se muestran en la siguiente tabla:

<b>Tipo</b>	<b>Significado</b>
author	Autor de la página
classification	Palabras para clasificar la página en los buscadores
description	Descripción del contenido de la página
generator	Programa utilizado para crear la página
keywords	Palabras clave

Tabla 2. Tipos de información en la etiqueta meta

La etiqueta *<meta />* no necesita etiqueta de cierre. Para cada etiqueta *<meta>* solo es posible indicar un tipo de información y su valor, pero es posible insertar varias etiquetas *<meta>* en un mismo documento.

La etiqueta *<meta/>* ha de estar entre las etiquetas *<head>* y *</head>*.

Por ejemplo, el siguiente código indica que el autor de la página es innovasistemas, que la página trata sobre un curso de HTML gratuito, y especifica algunas palabras clave a ser consultadas por los buscadores:

### **Ejemplo**

```
<html>
<head>
...
<meta name="author" content="innovasistemas"/>
<meta name="description" content="Curso de HTML gratuito"/>
<meta name="keywords" content="código HTML etiqueta página web gratis curso"/>
</head>
...
```

La etiqueta `<meta/>` también se utiliza para indicarle al navegador alguna información o alguna acción que debe realizar. En este caso se utiliza el atributo `http-equiv`, en lugar del atributo `name`.

Por ejemplo, supongamos que por algún motivo queremos que nuestra página se actualice automáticamente cada 30 segundos. En ese caso, deberíamos utilizar la acción Refresh (actualizar). El siguiente código ilustra esto:

### Ejemplo

```
<html>
<head>
...
<meta http-equiv="Refresh" content="30"/>
</head>
...
```

Ahora imagine que se cambia la dirección en la que se encuentra nuestra página web, y queremos que cuando algún usuario visite la página en la dirección antigua, a los 5 segundos el navegador lo redirija automáticamente a la dirección nueva. En ese caso podríamos insertar el siguiente código en la página que se encuentra en la dirección antigua:

### Ejemplo

```
<html>
<head>
...
<meta http-equiv="Refresh" content="5; URL=http://www.innovasistemas.org/index.php"/>
</head>
...
```

De este modo, el navegador realizaría la función de actualizar la página, pero cargando la que se encuentre en la nueva dirección (URL: <http://www.innovasistemas.org/index.php>).

### Etiqueta `<style>...</style>`

Permite especificar definiciones de hojas de estilos CSS. Esta etiqueta debe ubicarse en la cabecera del documento. El atributo *type* especifica el tipo de estilo. La sintaxis para su manejo es:

```
<style type="text/css">
<!--
  Etiqueta1, Etiqueta2 : {propiedad1:valor}
  Etiqueta3 : {propiedad1:valor;...;propiedadS:valor}
  Etiqueta4 : {propiedad1:valor;...;propiedadT:valor}
  .Clase1 : {propiedad1:valor;...;propiedadT:valor}
  //-->
</style>
```

### Etiqueta <script>...</script>

Permite insertar código de script en un documento web. Algunos atributos son:

**language:** Especifica el lenguaje de script a utilizar.

**type:** Especifica el tipo de script a utilizar

**src:** Permite indicar un archivo externo donde se encuentra el script. En el caso de Javascript, pueden crearse archivos con extensión **.js** y luego ser incluidos en la página mediante este atributo.

### Etiqueta <link />

Es conocida por la mayoría de los desarrolladores web, ya que se emplea para indicar los archivos CSS que se desean emplear en una página web para definir la presentación de la página; para implementarla, hay que tener presente que debe ubicarse dentro de las etiquetas <head>...</head>. En caso de tener varias hojas de estilos en diferentes archivos, podemos invocar varias veces esta etiqueta:

### Ejemplo

```
<link rel="stylesheet" type="text/css" href="miestilocss.css" />
```

### Nota

Más adelante iremos retomando algunas etiquetas vistas, así mismo, estudiaremos otros atributos de estos elementos cuando estemos trabajando tanto con DHTML como con PHP.

## Etiqueta de cuerpo de página

### Etiqueta <body>...</body>

Especifica el cuerpo del documento HTML. Entre este par de etiquetas va la mayoría de elementos que mostrará la página Web. Su atributo *bgcolor* permite establecer el color de fondo de la página, el cual puede indicarse con su nombre en inglés (por ejemplo "blue" para azul) o con un número en hexadecimal. Otro atributo es *background*, el cual permite especificar una imagen de fondo para la página. Sin embargo para aplicar formato a las páginas, se recomienda utilizar las nuevas construcciones con CSS, que se comentarán mas adelante.

### Comentarios en HTML

Los comentarios permiten documentar partes del código y/o "esconder" parte de éste cuando no se desea eliminarlo pero que tampoco se quiere ejecutar. Existen dos formas de hacer comentarios en HTML.

**Primera forma (etiqueta `<comment>...</comment>`)**

`<comment>` *Esto es un comentario. Solo se admite en Internet Explorer* `</comment>`

**Segunda forma** (Se recomienda utilizar este modo, ya que funciona en todos los navegadores)

`<!--` *Esto es un comentario que aplica a los estándares de la web -->*

**Etiquetas para manipular el texto**

**Etiqueta `<br />`**

Crea un salto de línea o “retorno de carro” en la página.

**Etiqueta `<hn>...</hn>`**

Esta etiqueta permite crear títulos de distinto nivel indicados por un número *n* comprendido entre 1 y 6. La etiqueta `<h1>...</h1>` muestra el texto con el mayor tamaño, la etiqueta `<h2>...</h2>` muestra el texto un poco mas pequeño y así sucesivamente.

**Etiqueta `<hr/>`**

Crea una línea (regla o barra horizontal) separadora en la página Web (observe que esta etiqueta no tiene una etiqueta de finalización). Puede modificarse su apariencia mediante los atributos *size* (tamaño), *width* (ancho), *align* (alineación) y *color* (color).

**Etiqueta `<strong>...</strong>` y Etiqueta `<b>...</b>`**

El objetivo de estas etiquetas es poner el texto en negrita.

**Etiqueta `<em>...</em>` y Etiqueta `<i>...</i>`**

El objetivo de estas etiquetas es poner el texto en cursiva (itálica).

**Etiqueta `<u>...</u>`**

El objetivo de esta etiqueta es subrayar el texto.

**Etiqueta `<s>...</s>` y Etiqueta `<strike>...</strike>`**

El objetivo de estas etiquetas es tachar el texto.

**Etiqueta `<sup>...</sup>`**

El texto encerrado entre este par de etiquetas aparece como superíndice (exponente o potencia en notaciones algebraicas). Es útil también para hacer notas al pie.

**Etiqueta `<sub>...</sub>`**

El texto encerrado entre este par de etiquetas aparece como subíndice.

**Etiqueta `<font>...</font>`**

Permite establecer la fuente del texto encerrado entre este par de etiquetas. El atributo *face* se utiliza para indicar la el tipo de fuente, *size* su tamaño y *color* su color (los tamaños no deben especificarse como se hacen en procesadores de texto ya que una tamaño de 12 puede ser un tipo de fuente muy grande a mostrar en la página). Como se ha comentado ya, se recomienda utilizar CSS para establecer el formato del sitio.



**Etiqueta <center>...</center>**

Centra los elementos que se encuentren entre el par de etiquetas.

**Etiqueta <p>...</p>**

Permite especificar un nuevo párrafo en el documento. Con el atributo *align* puede indicarse la alineación del texto mediante los valores *left* (por defecto), *right*, *justify* y *center*.

**CLASE No. 2**

**Listas ordenadas y sin ordenar: Numeración y viñetas**

Una lista, o listado, es una enumeración de dos o más elementos que suelen disponerse de tal forma que se facilite la distinción entre ellos. Es común ver las listas en trabajos escritos, ya sea utilizando viñetas (listas no ordenadas), o números (listas ordenadas, las cuales pueden aparecer con números arábigos, romanos o letras del alfabeto). En HTML también hay otro tipo de listas: las listas de glosario o de definición. A continuación se describe como crear estos tipos de listas.

**Listas ordenadas: Etiqueta <ol>...</ol>**

Esta etiqueta engloba las opciones que se mostrarán enumeradas. El atributo *type* permite especificar el tipo de numeración que aparecerá; los valores para este atributo son: *i*, *I* (números romanos en minúsculas o mayúsculas respectivamente); *a*, *A* (numeración con letras en minúsculas o mayúsculas respectivamente), *1* (numeración arábica) que es el valor por defecto.

**Listas sin ordenar: Etiqueta <ul>...</ul>**

Esta etiqueta engloba las opciones que se mostrarán con viñetas. El atributo *type* permite especificar el símbolo que aparecerá; los valores para este atributo son *bullet* (por defecto), *circle* y *square*.

**Opciones de la lista: Etiqueta <li>...</li>**

Permite especificar las opciones de una lista; se anida dentro de <ol>...</ol> o dentro de <ul>...</ul>.

**Nota**

Las listas pueden anidarse, basta con colocar la etiqueta <ol> o <ul> dentro de cualquiera de éstas y listo.

**Listas de definiciones: Etiquetas <dt>...</dt>, <dl>...</dl> y <dd>...</dd>**

Cada elemento de la lista está compuesto por un término y una definición y cada una de estas partes tiene su propia etiqueta. Estas listas se engloban con las etiquetas <dl>...</dl>. Para el término se usa la etiqueta <dt>...</dt> y para la definición la etiqueta <dd>...</dd>. Generalmente el navegador colocará el término y definición en dos líneas diferentes pero esto se puede evitar añadiendo a la etiqueta de apertura el atributo *compact*: <dl compact>...</dl>.

**Etiqueta <marquee>...</marquee>**

Establece una marquesina que se desplazará a lo largo de la pantalla. Algunos atributos de esta etiqueta son:

**align:** Los valores para este atributo son *top*, *middle*, *bottom*. Indica si el texto del interior de la marquesina se alinea en la zona alta (top), en la baja (bottom) o en el centro (middle) de la misma.

**bgcolor:** "Código de color". Indica el color del fondo de la marquesina.

**direction:** Los valores para este atributo son *left* (por defecto), *right*, *up*, *down*. Indica hacia que lugar se desplaza el texto: hacia la izquierda (left), hacia la derecha (right), hacia arriba (up) y hacia abajo (down).

**height:** Número o porcentaje. Indica la altura de la marquesina en puntos o porcentaje en función de la ventana del navegador.

**width:** Número o porcentaje. Indica la anchura de la marquesina en puntos o porcentaje en función de la ventana del navegador.

**loop:** Número o la palabra *infinite*. Indica el número de veces que se desplazará el texto por la marquesina. Si se indica *infinite* (o el número -1 -valor por defecto-), se desplazará indefinidamente.

**scrollDelay:** Número. Indica el número de milisegundos que tarda en reescribirse el texto por la marquesina.

**scrollAmount:** Número. Permite especificar la velocidad del desplazamiento, a mayor número mas rápidamente se desplazará la marquesina.

**behavior:** Indica la forma de desplazamiento de la marquesina: el valor *slide* hace que la marquesina se desplace hacia un lado y se detenga. El valor *alternate* hace que la marquesina vaya de un lado a otro. El valor *scroll* (valor por defecto) hace que la marquesina se desplace siempre en el mismo sentido.

Sin embargo, a la hora de crear animaciones es preferible y se recomienda hacerlo con DHTML (Javascript y CSS) o ActionScript (Flash), ya que esta etiqueta, aunque fue una novedad en HTML 3, se considera obsoleta.

### Caracteres en HTML

El siguiente cuadro muestra la Tabla de Códigos de Caracteres en HTML que pueden imprimirse en un documento web. Al emplearlos en HTML, hacemos referencia al código o al nombre del símbolo respectivo. También existe una Tabla Extendida de Códigos de Caracteres en HTML (4 caracteres) que pueden encontrar en la red, como en la URL:

<http://webusable.com/CharsExtendedTable.htm>.

Código	Nombre	Símbolo	Descripción
&#00;			No usado
&#01;			No usado
&#02;			No usado
&#03;			No usado
&#04;			No usado
&#05;			No usado
&#06;			No usado
&#07;			No usado
&#08;			No usado
&#09;			Tab Horizontal
&#10;			Line feed

**ESCUELA DE INFORMÁTICA**  
**TÉCNICO EN DESARROLLO DE SOFTWARE**  
**SUBMÓDULO PROGRAMACIÓN PARA LA WEB I**



&#11;			No usado
&#12;			No usado
&#13;			Retorno de Carro
&#14;			No usado
&#15;			No usado
&#16;			No usado
&#17;			No usado
&#18;			No usado
&#19;			No usado
&#20;			No usado
&#21;			No usado
&#22;			No usado
&#23;			No usado
&#24;			No usado
&#25;			No usado
&#26;			No usado
&#27;			No usado
&#28;			No usado
&#29;			No usado
&#30;			No usado
&#31;			No usado
&#32;			Espacio
&#33;		!	Exclamación
&#34;	&quot;	"	Comillas dobles
&#35;		#	Signo de Número
&#36;		\$	Signo de Dólar
&#37;		%	Signo de Porcentaje
&#38;	&amp;	&	Ampersand
&#39;		'	Apóstrofe
&#40;		(	Paréntesis izquierdo
&#41;		)	Paréntesis derecho
&#42;		*	Asterisco
&#43;		+	Signo Más
&#44;		,	Coma
&#45;		-	Guión
&#46;		.	Periodo (punto)
&#47;		/	Slash
&#48;		0	0
&#49;		1	1
&#50;		2	2
&#51;		3	3
&#52;		4	4
&#53;		5	5
&#54;		6	6
&#55;		7	7
&#56;		8	8
&#57;		9	9

**ESCUELA DE INFORMÁTICA**  
**TÉCNICO EN DESARROLLO DE SOFTWARE**  
**SUBMÓDULO PROGRAMACIÓN PARA LA WEB I**



&#58;		:	Dos puntos
&#59;		;	Punto y coma
&#60;	&lt;	<	Menor que
&#61;		=	Signo igual
&#62;	&gt;	>	Mayor que
&#63;		?	Interrogación (cerrada)
&#64;		@	Arroba
&#65;		A	A
&#66;		B	B
&#67;		C	C
&#68;		D	D
&#69;		E	E
&#70;		F	F
&#71;		G	G
&#72;		H	H
&#73;		I	I
&#74;		J	J
&#75;		K	K
&#76;		L	L
&#77;		M	M
&#78;		N	N
&#79;		O	O
&#80;		P	P
&#81;		Q	Q
&#82;		R	R
&#83;		S	S
&#84;		T	T
&#85;		U	U
&#86;		V	V
&#87;		W	W
&#88;		X	X
&#89;		Y	Y
&#90;		Z	Z
&#91;		[	Corchete izquierdo
&#92;		\	Backslash
&#93;		]	Corchete derecho
&#94;		^	Circunflejo (caret)
&#95;		–	Barra horizontal (subrayado)
&#96;		`	Acento agudo
&#97;		a	A
&#98;		b	B
&#99;		c	C
&#100;		d	D
&#101;		e	E
&#102;		f	F
&#103;		g	G

**ESCUELA DE INFORMÁTICA**  
**TÉCNICO EN DESARROLLO DE SOFTWARE**  
**SUBMÓDULO PROGRAMACIÓN PARA LA WEB I**



&#104;		h	H
&#105;		i	I
&#106;		j	J
&#107;		k	K
&#108;		l	L
&#109;		m	M
&#110;		n	N
&#111;		o	O
&#112;		p	P
&#113;		q	Q
&#114;		r	R
&#115;		s	S
&#116;		t	T
&#117;		u	U
&#118;		v	V
&#119;		w	W
&#120;		x	X
&#121;		y	Y
&#122;		z	Z
&#123;		{	Llave izquierda
&#124;			Barra vertical
&#125;		}	Llave derecha
&#126;		~	Tilde
&#127;		•	No usado
&#128;		€	No usado
&#129;		◆	No usado
&#130;		,	No usado
&#140;		Œ	No usado
&#141;		◆	No usado
&#142;		Ž	No usado
&#143;		◆	No usado
&#144;		◆	No usado
&#145;		‘	No usado
&#146;		’	No usado
&#147;		“	No usado
&#148;		”	No usado
&#149;		•	No usado
&#150;		—	No usado
&#151;		—	No usado
&#152;		~	No usado
&#153;		™	No usado
&#154;		š	No usado
&#155;		›	No usado
&#156;		œ	No usado

**ESCUELA DE INFORMÁTICA**  
**TÉCNICO EN DESARROLLO DE SOFTWARE**  
**SUBMÓDULO PROGRAMACIÓN PARA LA WEB I**



&#157;		◆	No usado
&#158;		◆	No usado
&#159;		ÿ	No usado
&#160	&nbsp;		Espacio sin ruptura (Non-breaking Space)
&#161;	&iexcl;	¡	Exclamación invertida
&#162;	&cent;	¢	Signo de Centavo
&#163;	&pound;	£	Signo de Libra Esterlina
&#164;	&curren;	¤	Signo de Moneda General
&#165;	&yen;	¥	Signo de Yen
&#166;	&brvbar;	¦	Barra vertical partida
&#167;	&sect;	§	Signo de Sección
&#168;	&uml;	¨	Diéresis
&#169;	&copy;	©	Copyright
&#170;	&ordf;	ª	Ordinal Femenino
&#171;	&laquo;	«	Ángulo izquierdo
&#172;	&not;	¬	Signo No
&#173;	&shy;		Guión suave
&#174;	&reg;	®	Marca registrada
&#175;	&macr;	¯	Acento macron
&#176;	&deg;	°	Signo de Grado
&#177;	&plusmn;	±	Más / menos
&#178;	&sup2;	²	Signo de "al cuadrado" (superscript 2)
&#179;	&sup3;	³	Signo de "al cubo" (superscript 3)
&#180;	&acute;	´	Acento agudo
&#181;	&micro;	µ	Signo de Micro
&#182;	&para;	¶	Párrafo
&#183;	&middot;	·	Punto medio
&#184;	&cedil;	¸	Cedilla
&#185;	&sup1;	¹	Signo de "a la 1" (superscript 1)
&#186;	&ordm;	º	Ordinal Masculino
&#187;	&raquo;	»	Ángulo derecho
&#188;	&frac14;	¼	Fracción un-cuarto
&#189;	&frac12;	½	Fracción un-medio
&#190;	&frac34;	¾	Fracción tres-cuartos
&#191;	&iquest;	¿	Interrogación abierta
&#192;	&Agrave;	À	A mayúscula, acento grave
&#193;	&Aacute;	Á	A mayúscula, acento agudo
&#194;	&Acirc;	Â	A mayúscula, acento circunflejo
&#195;	&Atilde;	Ã	A mayúscula, tilde
&#196;	&Auml;	Ä	A mayúscula, diéresis
&#197;	&Aring;	Å	A mayúscula, anilloanillo

**ESCUELA DE INFORMÁTICA**  
**TÉCNICO EN DESARROLLO DE SOFTWARE**  
**SUBMÓDULO PROGRAMACIÓN PARA LA WEB I**



&#198;	&AElig;	Æ	AE mayúscula diptongo (ligature)
&#199;	&Ccedil;	Ç	C mayúscula, cedilla
&#200;	&Egrave;	È	E mayúscula, acento grave
&#201;	&Eacute;	É	E mayúscula, acento agudo
&#202;	&Ecirc;	Ê	E mayúscula, acento circunflejo
&#203;	&Euml;	Ë	E mayúscula, diéresis
&#204;	&Igrave;	Ì	I mayúscula, acento grave
&#205;	&Iacute;	Í	I mayúscula, acento agudo
&#206;	&Icirc;	Î	I, acento circunflejo
&#207;	&Iuml;	Ï	I mayúscula, diéresis
&#208;	&ETH;	Ð	Letra Eth mayúscula, (Islandia)
&#209;	&Ntilde;	Ñ	N mayúscula, tilde
&#210;	&Ograve;	Ô	O mayúscula, acento grave
&#211;	&Oacute;	Ó	O mayúscula, acento agudo
&#212;	&Ocirc;	Ö	O mayúscula, acento circunflejo
&#213;	&Otilde;	Õ	O mayúscula, tilde
&#214;	&Ouml;	Ö	O mayúscula, diéresis
&#215;	&times;	×	Signo de Multiplicar
&#216;	&Oslash;	Ø	O mayúscula, slash
&#217;	&Ugrave;	Ù	U mayúscula, acento grave
&#218;	&Uacute;	Ú	U mayúscula, acento agudo
&#219;	&Ucirc;	Û	U mayúscula, acento circunflejo
&#220;	&Uuml;	Ü	U mayúscula, diéresis
&#221;	&Yacute;	Ý	Y mayúscula, acento agudo
&#222;	&THORN;	Þ	Letra Thorn, (Islandia)
&#223;	&szlig;	ß	s minúscula sostenida, (Alemania)
&#224;	&agrave;	à	a minúscula, acento grave
&#225;	&aacute;	á	a minúscula, acento agudo
&#226;	&acirc;	â	a minúscula, acento circunflejo
&#227;	&atilde;	ã	a minúscula, tilde
&#228;	&auml;	ä	a minúscula, diéresis
&#229;	&aring;	å	a minúscula, anillo
&#230;	&aelig;	æ	ae minúsculae diptongo
&#231;	&ccedil;	ç	c Small c, cedilla
&#232;	&egrave;	è	e minúscula, acento grave
&#233;	&eacute;	é	e minúscula, acento agudo
&#234;	&ecirc;	ê	e minúscula, acento circunflejo
&#235;	&euml;	ë	e minúscula, diéresis
&#236;	&igrave;	ì	i minúscula, acento grave

**ESCUELA DE INFORMÁTICA**  
**TÉCNICO EN DESARROLLO DE SOFTWARE**  
**SUBMÓDULO PROGRAMACIÓN PARA LA WEB I**



&#237;	&iacute;	í	i minúscula, acento agudo
&#238;	&icirc;	î	i minúscula, acento circunflejo
&#239;	&iuml;	ï	i minúscula, diéresis
&#240;	&eth;	ð	eth minúscula, (Islandia)
&#241;	&ntilde;	ñ	n minúscula, tilde
&#242;	&ograve;	ò	o minúscula, acento grave
&#243;	&oacute;	ó	o minúscula, acento agudo
&#244;	&ocirc;	ô	o minúscula, acento circunflejo
&#245;	&otilde;	õ	o minúscula, tilde
&#246;	&ouml;	ö	o minúscula, diéresis
&#247;	&divide;	÷	Signo de División
&#248;	&oslash;	ø	o minúscula, slash
&#249;	&ugrave;	ù	u minúscula, acento grave
&#250;	&uacute;	ú	u minúscula, acento agudo
&#251;	&ucirc;	û	u minúscula, acento circunflejo
&#252;	&uuml;	ü	u minúscula, diéresis
&#253;	&yacute;	ý	y minúscula, acento agudo
&#254;	&thorn;	þ	thorn minúscula, (Islandia)
&#255;	&yuml;	ÿ	y minúscula, diéresis
&#338;	&OElig;	Œ	OE mayúscula
&#339;	&oelig;	œ	oe minúscula
&#352;	&Scaron;	Š	S mayúscula con caron
&#353;	&scaron;	š	s minúscula con caron
&#376;	&Yuml;	Ÿ	Y mayúscula con diéresis
&#710;	&circ;	ˆ	acento circunflejo
&#732;	&tilde;	˜	tilde minúscula
&#8194;	&ensp;		espacio en
&#8195;	&emsp;		espacio em
&#8201;	&thinsp;		espacio fino
&#8204;	&zwnj;		non-joiner ancho cero
&#8205;	&zwj;		joiner ancho cero
&#8206;	&lrm;		marca izquierda-derecha
&#8207;	&rlm;		marca derecha-izquierda
&#8211;	&ndash;	–	dash en
&#8212;	&mdash;	—	dash em
&#8216;	&lsquo;	‘	comilla simple izquierda
&#8217;	&rsquo;	’	comilla simple derecha
&#8218;	&sbquo;	‚	coma bajo-9 simple
&#8220;	&ldquo;	“	comilla doble izquierda
&#8221;	&rdquo;	”	comilla doble derecha
&#8222;	&bdquo;	„	coma bajo-9 doble
&#8224;	&dagger;	†	Daga
&#8225;	&Dagger;	‡	doble daga



&#8240;	&permil;	% <sub>00</sub>	por mil
&#8249;	&lsaquo;	◁	ángulo simple izquierdo
&#8250;	&rsaquo;	▷	ángulo simple derecho
&#8264;	&euro;	€	Euro
&#8282;		☐	marca registrada
&#8217;		'	comilla simple
&#8230;		...	puntos suspensivos

Tabla 3. Tabla de Códigos de Caracteres en HTML

## Etiquetas para manipular contenido multimedia

### Etiqueta <img/>



Figura 9. Casi cualquier tipo de imagen puede mostrarse en una página web utilizando la etiqueta img

Permite poner una imagen en la página; algunas extensiones permitidas son: jpg, gif, png y bmp, aunque se recomienda no utilizar imágenes con este último formato ya que es pesado y los formatos actuales han mejorado la calidad de éstas. El atributo *src* se utiliza para especificar la ruta (la cual debe ser relativa y no absoluta) donde se encuentra la imagen, los atributos *height* y *width* permiten establecer el alto y ancho respectivamente de la imagen y con el atributo *alt* podemos mostrar un texto alternativo cuando se pase el Mouse sobre la imagen (dado que en algunos navegadores no se reconoce este atributo, utilizamos entonces el atributo *title*). También puede especificarse el atributo *dynsrc* en vez de *src* para indicar que se desea ver una animación en formatos .avi, .mpg y .wmv entre otros. Esta etiqueta tampoco tiene una etiqueta de finalización.

### Etiqueta <embed/>

Se utiliza para poner recursos multimedia en una página web, tales como videos y música; admite los formatos mp3, wmv, mpg y avi entre otros. Algunos atributos son:

**src:** Especifica la ubicación del recurso multimedia.

**autostart:** De tipo booleano; indica si la animación se reproduce o no automáticamente. Los valores válidos son “*true*” y “*false*”.

**loop:** De tipo booleano; indica si la reproducción se repite o no al finalizar ésta.

Actualmente la etiqueta `<embed />` es reconocida por diferentes navegadores e incluso puede usarse para sitios que se visualizan en el sistema operativo Linux. Para poner videos .flv no podemos utilizar esta etiqueta, por lo que es necesario utilizar la etiqueta `<object>...</object>` que

se comenta a continuación (con esta etiqueta pueden publicarse videos y animaciones de cualquier tipo, así como sonido; esto puede lograrse fácilmente utilizando un editor *WYSIWYG*.).

**Etiqueta <object>...</object>**



Figura 10. Diferentes recursos multimedia

Con objeto de normalizar la inclusión de archivos no nativos en los navegadores web se decidió sustituir las diferentes etiquetas que realizaban este papel (APPLET, BGSOUND, EMBED, etc.), y que no pertenecían a los estándares web, por una etiqueta general que fuera capaz de incrustar en el navegador todo tipo de archivos. La etiqueta elegida en el estándar HTML 4.0 fué *OBJECT*, a la que se dotó de suficientes atributos y flexibilidad para poder realizar correctamente su trabajo. Debido a esto, la propuesta ha sido usar esta etiqueta también para incluir ficheros de audio de todo tipo en las páginas web.

Ahora bien, la aceptación e implementación de la misma ha tenido variantes según el navegador que se utilice y el objeto que se va a incrustar. De este forma, Internet Explorer ha realizado su propia implementación de la etiqueta *object*, incluyendo en ella referencias a filtros y componentes ActiveX específicos para los ficheros de audio. Por su lado, los navegadores Netscape no soportan correctamente esta etiqueta para archivos de este tipo.

Restringiéndonos a Internet Explorer, la polémica sigue, ya que en diferentes manuales nos encontramos diferentes formas de incrustar sonidos mediante *object*, unas que funcionan bien, y otras que no. ¿Por qué sucede esto? Porque Microsoft ha ido usando la etiqueta *object* para implementar todo un gran conjunto de componentes propios, que además han ido adaptándose a las diferentes versiones de Internet Explorer.

Como regla general, válida no sólo para incrustar ficheros de sonido, sino también para otros tipos, la etiqueta *object* va a definir un objeto o componente externo encargado de la reproducción del archivo, que en el caso de Internet Explorer suele ser algún tipo de control ActiveX. Mediante *object* se instancia el objeto, se declara su URL y sus principales propiedades generales, y mediante un conjunto de etiquetas especiales, *PARAM*, se le van pasando los valores que necesita para su correcto funcionamiento o para su configuración deseada.

*La sintaxis general de la etiqueta object, para el caso de ficheros de sonido, es del tipo:*

```
<object atributo1="valor" atributo2="valor" ... atributoN="valor">
<param name="nombre" value="valor">
<param name="nombre" value="valor">
...
</object>
```

Los principales atributos de **object**, en referencia a archivos de audio, son:

**classid="identificador\_objeto"**: Fija la URL del objeto o componente externo necesario para reproducir el archivo de audio, y la implementación CLSID de los controles ActiveX necesarios.

**type="tipo\_fichero"**: Atributo importante, que declara el tipo de archivo de audio que estamos usando.

**width="w"**: Determina el ancho visible de la consola, en píxeles.

**height="h"**: Determina el alto visible de la consola, en píxeles.

**align="top/bottom/center/baseline/left/right/texttop/middle/absmiddle/absbottom"**: Análogo al de la etiqueta *img* y otras, define la alineación horizontal o vertical de la consola respecto de los elementos de la página.

**hspace="hs"**: Que establece la separación horizontal, **vspace="vs"**, que establece la separación vertical, en píxeles, entre la consola y los elementos de la página que la rodean. Análoga a sus equivalentes de la etiqueta *IMG*.

**autostart="true/false"**: Determina si el fichero de audio debe empezar a reproducirse por sí sólo al cargarse la página o si por el contrario será preciso la actuación del usuario (o de código de script) para que comience la audición.

**standby="mensaje"**: Presenta en pantalla un mensaje al usuario mientras el fichero se carga.

En cuanto a los elementos **param**, los más importantes son:

**param name="FileName" value="ruta\_archivo"**: Determina la ruta o URL del archivo de audio a reproducir. No es necesario utilizar sólo archivos WAV o MID, pudiendo reproducirse también archivos MP3 o Real Audio. El reproductor del primero lo incluye Internet Explorer en ActiveMovie (componente de Windows Media).

**param name="autostart" value="true/false"**: Indica al navegador si se debe empezar a reproducir el sonido automáticamente al cargar la página o si por el contrario será preciso que el usuario pulse el botón Play para ello.

No son estos todos los atributos y parámetros posibles. Es más, en cuanto nos metemos en componentes Microsoft, podemos encontrarnos multitud de configuraciones posibles, que nos van a permitir fijar muchos aspectos de los mismos. Se deja a cada uno la posibilidad de profundizar en el estudio de aquellos componentes y propiedades que necesite, pero sabiendo que con los elementos vistos arriba tenemos más que suficiente para presentar un archivo de audio en nuestra página web.

### Ejemplo

```
<object classid="CLSID:05589FA1-C356-11CE-BF01-00AA0055595A" width="150" height="175"
type="audio/midi">
<param name="FileName" value="../../sonidos/xfiles.mid">
```

```
<param name="autostart" value="true">

</object>
```

Una aplicación importante que se le da a esta etiqueta es poder utilizarla para insertar en una página web películas y videos Flash (archivos *.swf* y *.flv* respectivamente); el siguiente ejemplo muestra como insertar animaciones *.swf* en una página web.

### Ejemplo

```
...
<object type="application/x-shockwave-flash" data="animaciones/pelicula.swf" width="900"
height="220">
  <param name="movie" value="animaciones/pelicula.swf" />
  <param name="quality" value="high">
  <param name="loop" value="true" />
  <param name="play" value="true" />
  <param name="wmode" value="transparent" />
</object>
...
```

### Hipervínculos

Es uno de los principales elementos de HTML, pues son los que nos permiten navegar a través de la web, enlazando un sinnúmero de documentos y recursos de *URL (Uniform Resource Locator – Localizador Uniforme de Recursos)*. Para crear hipervínculos utilizamos la siguiente etiqueta.

#### Etiqueta **<a>...</a>**

Se utiliza para especificar un vínculo o enlace a un recurso URL. Posee varios atributos:

**href:** Utilizado para hacer referencia a otra URL cuando se pulsa el texto o imagen encerrados entre el par de etiquetas

**name:** Utilizado para identificar un anclaje. El nombre puede ser invocado por otras etiquetas de URL con el signo # añadido al nombre.

**title:** Muestra un texto alternativo cuando el Mouse se pasa sobre el enlace.

**target:** Especifica donde se abrirá el recurso URL, este puede ser una nueva ventana, una pestaña o un marco.

### Marcos flotantes

#### Etiqueta **<iframe>...</iframe>**

Un *iframe* o *marco flotante* sirve para crear un espacio dentro de la página donde se puede incrustar otra web. Es un rectángulo cuyas dimensiones debe especificar el desarrollador en la propia etiqueta *iframe*, que tiene asociada una página web que se carga en dicho espacio. Esa página web tendrá sus propios contenidos y estilos, independientes del contexto donde se está mostrando. Además será perfectamente funcional: si tiene enlaces se mostrarán en ese mismo espacio y si tiene scripts o aplicaciones dentro, se ejecutarán también de manera autónoma en el espacio reservado al *iframe*. Los marcos flotantes vienen a reemplazar, de una manera muy

práctica, a los *frames*, utilizados anteriormente y controvertidos por los inconvenientes que causaban. Algunos atributos de la etiqueta *iframe*, son:

**name:** Asigna un nombre al marco flotante. Podemos utilizar este nombre para referirnos a él con el atributo *target* de los hipervínculos, o mediante Javascript.

**src:** Permite especificar una página web que se abrirá por defecto en el *iframe*.

**width:** Define el ancho del *iframe*.

**height:** Define la altura del *iframe*.

**id:** Especifica el identificador del *iframe*, y poder referirnos a él desde Javascript.

**frameborder:** Define si queremos o no que haya un borde en el *iframe*. Los valores posibles son 0 | 1. *frameborder="0"* indicaría que no queremos borde y *frameborder="1"* que sí.

**scrolling:** Indica si se quiere que aparezcan barras de desplazamiento para ver los contenidos del *iframe* completo, en el caso que no aparezcan en el espacio reservado para el *iframe*. Los valores posibles son: yes | no | auto. El valor "yes" es para que aparezcan siempre las barras de desplazamiento o scroll bars, "no" sirve para que no aparezcan nunca y "auto" es para que aparezcan sólo cuando son necesarias (es el valor por defecto).

**marginwidth:** Define el margen a izquierda y derecha que debe tener la página que va dentro del *iframe*, con respecto al borde. Este margen va en pixeles, pero prevalecerá el margen que pueda tener asignada la página web que mostremos en el frame flotante.

**marginheight:** Lo mismo que *marginwidth*, pero en este caso para el tamaño del margen por la parte de arriba y abajo.

**margin:** Especifica la alineación del *iframe*, igual que se especifica para las imágenes.

**style y class:** Los atributos para definir el aspecto del *iframe* por medio de hojas de estilo CSS.

## Tablas en HTML

### Etiqueta <table>...</table>

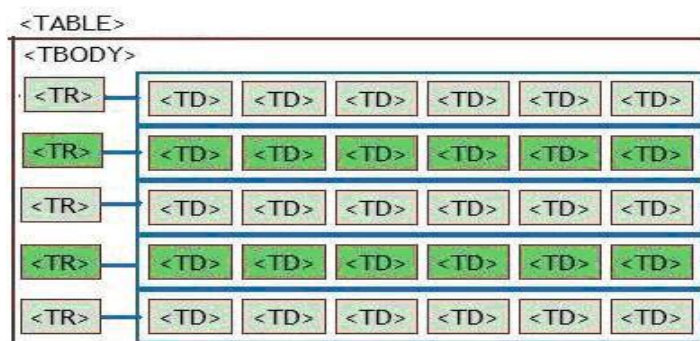


Figura 11. Esquema para la creación de tablas HTML

Permite crear tablas en una página Web. Las tablas se utilizan para organizar la presentación del contenido mostrado en nuestro sitio. La creación de tablas implica especificar otras etiquetas que se anidan en la etiqueta *<table>* con el fin de especificar las filas y las columnas que tendrá la tabla, estas etiquetas son *<tr>*, *<td>* y *<th>* que se enuncian a continuación.

### Etiqueta <tr>...</tr>

Permite especificar una nueva fila en la tabla.

#### Etiqueta <td>...</td>

Permite especificar una nueva celda en una fila de la tabla.

#### Etiqueta <th>...</th>

Se comporta igual a la anterior; se utiliza generalmente para encabezados de columna, ya que muestra el texto en negrita.

Veamos a continuación algunos atributos útiles para la construcción de las tablas. Empecemos viendo atributos que nos permiten modificar una celda en concreto o toda una fila:

**align:** Justifica el texto de la celda del mismo modo que si fuese el de un párrafo.

**valign:** Podemos elegir si queremos que el texto aparezca arriba (*top*), en el centro (*middle*) o abajo (*bottom*) de la celda.

**bgcolor:** Da color a la celda o fila elegida.

**bordercolor:** Define el color del borde.

Otros atributos que pueden ser únicamente asignados a una celda y no al conjunto de celdas de una fila son:

**background:** Nos permite colocar un fondo para la celda a partir de un enlace a una imagen.

**height:** Define la altura de la celda en píxeles o porcentaje.

**width:** Define la anchura de la celda en píxeles o porcentaje.

**colspan:** Expande una celda horizontalmente (combina columnas).

**rowspan:** Expande una celda verticalmente (combina filas).

#### Nota

El atributo *height* no funciona en todos los navegadores, además, su uso no está muy extendido. Las celdas por lo general tienen el alto que necesitan para que se ajuste todo el contenido que se le haya insertado, es decir, crecen lo suficiente para que quepa lo que hemos colocado dentro.

El atributo *width* si funciona en todos los navegadores y lo tendremos que utilizar, quizás, constantemente. Si le asignamos un ancho a la celda, éste será respetado, y si dicha celda tiene mucho texto o cualquier otro contenido, ésta crecerá hacia abajo todo lo necesario para que el contenido que hemos colocado se ajuste.

#### Etiqueta <caption>...</caption>

Esta etiqueta, que también se anida dentro de la tabla, permite especificar un texto que se mostrará en la parte superior o inferior de ésta mediante los valores *top* (por defecto) y *bottom*.

Así mismo, la tabla tiene una serie de atributos que modifican su aspecto, estos son:

**align:** Alinea horizontalmente la tabla con respecto a su entorno.

**background:** Nos permite establecer un fondo para la tabla a partir de un enlace a una imagen.

**bgcolor:** Da color de fondo a la tabla.

**border:** Define el número de píxeles del borde principal o el grosor de éste.

**bordercolor:** Define el color del borde.

**cellpadding:** Define, en píxeles, el espacio entre los bordes de la celda y el contenido de la misma.

**cellspacing:** Define el espacio entre los bordes (en píxeles).


**height:** Define la altura de la tabla en píxeles o porcentaje.

**width:** Define el ancho de la tabla en píxeles o porcentaje.



CLASE No. 3

Formularios HTML



**Datos Personales**

Nombre \*

Apellido \*

Passaporte o I.D.Card No.

Sexo

Edad

E-Mail \*

Ciudad

Estado/ Provincia

Zip / Código Postal

País

nº Fax

Por favor, ingrese al menos uno de los siguientes teléfonos

Casa

Oficina

Cómo prefiere que nos contactemos? Marque abajo una opción:

☐ E-Mail ☐ Teléfono ☐ Fax

Figura 12. Formulario HTML para el ingreso de datos personales en una página web

Los formularios son una característica del estándar HTML que permite a los autores recolectar información provista por los visitantes. Éstos son útiles para recoger información personal, de contacto, preferencias u opiniones, o cualquier tipo de entrada por parte del visitante que el autor pueda necesitar. Cabe anotar sin embargo, que es necesario utilizar un lenguaje de script (del lado del cliente o el servidor) para procesar la información que se ingrese en un formulario, ya que HTML solo ofrece la interfaz, nada mas.

**Etiqueta <form>... </form>**

Permite reunir varios elementos o controles de formulario, actuando como contenedor; algunos atributos son:

**method:** Indica cómo se enviarán los datos del formulario. El valor "post" hace que se envíen los datos al agente de procesamiento almacenándolos en el cuerpo del formulario, en tanto que el valor "get" (por defecto) envía los datos agregándolos como parámetros a la dirección URL y separándolos de la dirección con un signo de interrogación (?), si hay varios datos, se separan con un signo de ampersand (&).

**action:** Indica la dirección a la que se enviará la información (un script CGI -Common Gateway Interface o Interfaz de Entrada Común- o dirección de correo electrónico -mailto:dirección\_de\_correo\_e@dominio-)

**name:** Establece un nombre para el formulario.

**id:** Establece un identificador para el formulario.

**enctype:** Especifica cómo se codifican los datos del formulario.

**accept:** Se usa para establecer tipos MIME (*Multipurpose Internet Mail Extensions* o *Extensiones de Correo de Internet Multipropósito*) para los datos que el formulario puede enviar.

Sintaxis básica para la etiqueta `<form>...</form>`

```
<form method="post" | "get" action="url" name="nombre_formulario"
id="identificador_formulario"> ... </form>
```

### Ejemplo

Mostramos como se podría configurar la etiqueta `<form>...</form>`

1.

```
<form method="post" action="mailto:micorreo@nose.tal"> ... </form>
```

2.

```
<form method="get" action="http://www.misitio.tal/cgi-bin/script.cgi">...</form>
```

La etiqueta `<form>...</form>` actúa como contenedor para almacenar elementos que permiten al usuario seleccionar o introducir datos. Todos los datos se enviarán a la dirección URL indicada en el atributo *action* de la etiqueta, por el método indicado en el atributo *method*. Los formularios admiten cualquier elemento HTML anidado (como texto, imágenes, tablas y enlaces), pero los elementos interactivos son los que interesan. Estas etiquetas de formulario que permiten crear los diferentes controles poseen también los atributos *name* e *id* los cuales especifican el nombre y el identificador del elemento respectivamente y que pueden ser invocados desde PHP y DHTML para procesarlos de alguna forma; éstas son:

### Etiqueta `<input>`

Permite crear varios tipos de controles según el valor especificado en su atributo **type**. Los valores para este atributo son:

**checkbox:** Crea una casilla de verificación o de elección. Las casillas de elección pueden adoptar uno de dos estados: *checked* (seleccionado) o *unchecked* (no seleccionado). Cuando la casilla es seleccionada, el par nombre/valor se envía al CGI.

**hidden:** Este campo, que el navegador no muestra, se utiliza para definir una configuración única que se enviará al CGI como par nombre/valor. En él, se guarda información oculta a los usuarios.

**file:** Permite al usuario especificar una ruta de archivo, que lleva al archivo que se enviará con el formulario. Los tipos de archivo que pueden ser enviados deben especificarse utilizando el atributo *accept* de la etiqueta de formulario.

**image:** Crea un *botón de envío personalizado* que muestra la imagen especificada con el atributo *src*.

**password:** Crea un *cuadro de texto* donde los caracteres escritos aparecen como asteriscos o puntos para camuflar el texto de entrada.

**radio:** Crea un *botón de opción o de "radio"* que permite al usuario elegir entre varias opciones. Cada uno de estos botones debe tener el mismo atributo *name* (mas no necesariamente *id*) para formar un grupo (agruparlos). El par nombre/valor del botón de radio seleccionado se enviará al CGI. Al aplicar el atributo *checked* para uno de estos botones, se definirá como seleccionado de



forma predeterminada. Para crear un grupo diferente de estos elementos en un mismo formulario, basta con asignar otro valor en el atributo *name* (e *id*).

**reset:** Crea un *botón de restauración* para quitar todos los elementos en el formulario y restablecer a sus valores predeterminados.

**submit:** Crea un *botón de envío* para enviar el formulario (con todos sus datos) a la URL especificada en el atributo *action* del formulario. El texto en el botón puede definirse usando el atributo *value*.

**text:** Crea un *cuadro de texto* para escribir una línea de texto. El tamaño de la caja puede definirse usando el atributo *size* y la extensión máxima del texto con el atributo *maxlength*. Para especificar un valor por defecto, utilizamos el atributo *value*, el cual además, guarda el valor ingresado como texto en la caja. El atributo *readonly* determina si el campo es de solo lectura, usándose así: *readonly="readonly"*.

**button:** Crea un botón que puede programarse con un lenguaje de script (como Javascript) llamando a uno de sus manejadores de evento.

#### **Etiqueta <textarea>...</textarea>**

Crea un cuadro de texto multilínea. Algunos atributos, además de *name* e *id*, son:

**rows:** Especifica el número de filas o renglones visibles en el cuadro de texto.

**cols:** Especifica el número de columnas visibles en el cuadro de texto.

**wrap:** Permite especificar si el cuadro de texto realizará automáticamente saltos de línea al llegar al extremo de él. Los valores para este atributo son: "*off*", "*hard*", "*soft*".

#### **Etiqueta <select>...</select>**

Crea listas (cuadros de lista y cuadros combinados) de opciones múltiples. También posee los atributos *name* e *id*, otros son:

**size:** Numérico. Especifica el alto de la lista. Si se indica el valor "1", se crea un cuadro combinado o desplegable; si se indica un valor mayor a 1, se crea un cuadro de lista de tamaño fijo.

**multiple:** De tipo booleano. Especifica si la lista acepta múltiple selección de elementos (aplica para cuadros de listas de tamaño fijo). Al implementarlo, se sigue esta forma: *multiple="multiple"*.

Para especificar las opciones de la lista utilizamos la etiqueta <option>...</option>:

#### **Etiqueta <option>...</option>**

Permite especificar las opciones de la lista. El atributo *selected*, de tipo booleano, aplicado en una de las opciones, indica que la opción aparecerá seleccionada por defecto (debe utilizarse así: *selected="selected"*). El atributo *value* guarda un valor que es enviado al script especificado en el atributo *action* del formulario.

Hasta aquí en lo que concierne a formularios, mas adelante retomaremos estos conceptos en ejemplos de HTML y aplicaciones web tanto del lado del cliente como del servidor y veremos las formas de acceder a estos elementos para procesar la información que contienen.

#### **Etiquetas para crear divisiones**

Actualmente existen etiquetas que permiten organizar la información en un documento HTML, remplazando las rígidas tablas y que tantos inconvenientes traían a la hora de maquetar páginas. Con estas etiquetas y la ayuda de CSS, se pueden maquetar páginas web y lograr presentaciones agradables al usuario.

### Etiqueta <div>...</div>

Permite estructurar un documento HTML. Se trata de un contenedor que incluye texto, imágenes, tablas, etc. Lo que se encuentre entre las etiquetas <div> y </div> se considera una capa y puede posicionarse en cualquier parte del documento HTML de forma absoluta o relativa mediante CSS.

Admite los atributos id, utilizado para identificar la capa, y style que permite definir sus estilos. Si se desea establecer los estilos de una capa en la cabecera o en un archivo .css, se ha de incluir su nombre, precedido del símbolo de almohadilla o numeral (#):

### Ejemplo

```
<div id="capa1">Texto dentro de una capa </div>
```

### Etiqueta <span>...</span>

Sirve para crear pequeñas divisiones en la página; es muy útil para aplicar estilos CSS a una pequeña parte de una página HTML. Por ejemplo, con ella podríamos hacer que una parte de un párrafo se coloree en rojo indicando el estilo apropiado. Este importante elemento puede utilizarse en esos casos donde queremos aplicar estilos CSS y no se tenga una etiqueta en particular para hacerlo.

### Ejemplo

```
<span class="miestilo">Hola </span><!--Debe definirse el estilo "miestilo" para usarlo -->
<br/>
<span style="text-decoration:underline;font-weight:bold;">Hola de nuevo</span>
```

## CLASE No. 4

### EJEMPLO GENERAL DE HTML

A continuación se ilustrará el uso de las etiquetas del HTML en una página web. Se asume un árbol de directorios como el que se muestra a continuación (los textos en negrita representan carpetas y en cursiva, archivos):

```
-misitio
  --pagina1.html
  --pagina2.html
  --imagenes
    ---img1.jpg
  --documentos
    ---dcto1.doc
  --videos
    ---video1.wmv
```

La siguiente figura muestra también el árbol de directorios (los rectángulos representan carpetas):

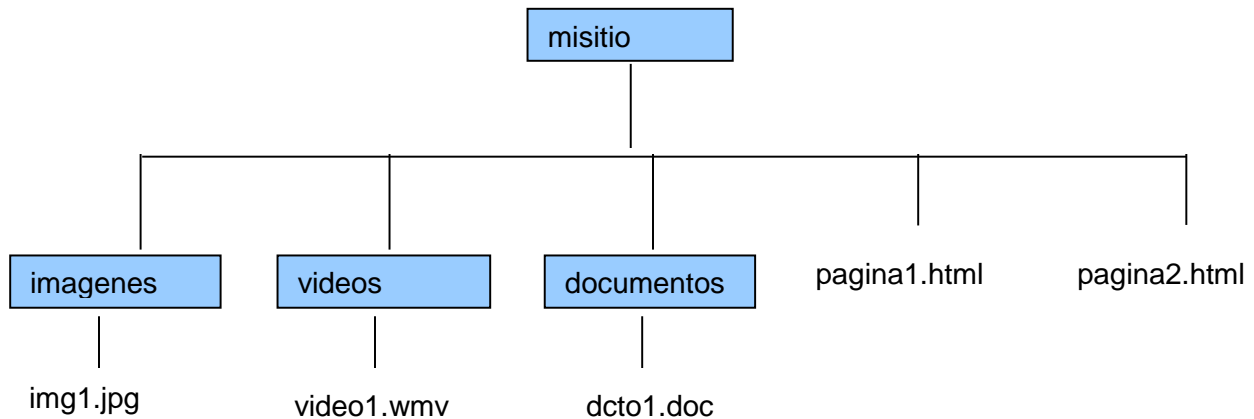


Figura 13. Árbol de directorios (sistema de archivos) para el sitio del ejemplo

### Solución

Cree el árbol de directorios especificado arriba. Luego escriba el código mostrado en el editor de su preferencia y guarde el archivo como *pagina1.html*. Saque una copia de éste, renómbrela como *pagina2.html* y cambie algunos aspectos de esta nueva página (como el color de fondo, por ejemplo); según lo mostrado en el árbol, guarde en las carpetas correspondientes los archivos que se indican (u otros, recordando hacer los respectivos cambios en el código).

Para publicar la página (ejecutarla), basta con abrir el archivo .html en un navegador de Internet, como Mozilla Firefox, Opera, Google Chrome, Internet Explorer (7.0 ó superior), Safari, Netscape Navigator, etc.

### Notas

- Los nombres de archivos y carpetas del sitio no deben contener caracteres especiales ni espacios en blanco, y es muy importante además, que los reference en el código tal y como se llaman en el sistema operativo, respetando mayúsculas y minúsculas, con esto evitará inconvenientes cuando publique el sitio utilizando Linux o Unix.
- HTML no es sensible a las mayúsculas, sin embargo se recomienda indicar sus palabras reservadas en minúscula.

### Ejemplo

```
<!--Este ejemplo ilustra el manejo de las principales etiquetas html -->
```

```
<html>
<head>
<title>Primera página HTML</title>
</head>
```

```
<body bgcolor="#3399FF">
```

```
<a href="#abajo" name="arriba" title="Ir abajo">Ir abajo</a>
```

```
<br/>
Hola, bienvenid@s a la web

<hr/>
<h1 align="center">Bienvenido a la programación web (título primer nivel)</h1>
<br/>

<h2>título segundo nivel</h2>
<br/>
<hr/>

<b>Este texto está en negrita</b>
<br/>
<strong>Este texto también está en negrita</strong>
<hr/>

<i>Este texto está en cursiva</i>
<br/>
<em>Este texto también está en cursiva</em>
<hr/>

<u>Este texto está subrayado</u>
<hr/>

<s>Este texto está tachado</s>
<br/>

<strike>Otro texto tachado</strike>
<hr/>

Superíndices:
<br/>

z<sup>2</sup>=x<sup>2</sup>+y<sup>2</sup>
<br/>
<br/>

Subíndices:
<br/>
c<sub>k</sub>=a<sub>i</sub>+b<sub>j</sub>
<hr/>

<center>
<font face="Comic Sans MS" color="#CC0033" size="8">
Texto con formato y centrado
</font>
</center>
<hr/>
```

```
<marquee behavior="alternate" direction="right" loop="-1">Esta es una marquesina</marquee>
<hr/>
```

```

```

```

<hr/>
```

```
<p align="right">Este texto aparece alineado a la derecha</p>
<hr/>
```

Manejo y aplicaciones de los hipervínculos (los anclajes con nombre se encuentran en el inicio y fin de la página)

```
<br/>
```

```
<a href="http://www.google.com" title="Ir a Google" target="_blank">Ir al sitio Google</a>
<br/>
```

```
<a href="pagina2.html" title="Abrir mi página">Ir a página 2</a>
<br/>
```

```
<a href="imagenes/img1.jpg" title="Ver imagen">Ver imagen</a>
<br/>
```

```
<a href="documentos/dcto1.doc" title="Descargar">Descargar documento</a>
<br/>
```

```
<a href="mailto:micorreo@nose.tal" title="Escríbeme">Contactar al correo</a>
<hr/>
```

```
<center>
```

```
<form name="form1" id="form1">
```

```
  <table width="400" border="1" bgcolor="yellow">
```

```
    <caption align="bottom">Datos del formulario</caption>
```

```
    <tr>
```

```
      <th colspan="2">Información</th>
```

```
    </tr>
```

```
    <tr>
```

```
      <td>Nombre</td>
```

```
      <td><input type="text" name="txtnom" id="txtnom" value="Pedro" maxlength="30"
size="50"/></td>
```

```
    </tr>
```

```
<tr>
  <td>Apellido</td>

  <td><input type="text" name="txtape" id="txtape" value="Zapata" readonly="readonly"
size="50"/></td>
</tr>

<tr>
  <td>Contraseña</td>

  <td><input type="password" name="txtpass" id="txtpass" maxlength="20" size="30"/></td>
</tr>

<tr>
  <td>Sexo</td>
  <td>
    <input type="radio" name="sexo" id="sexo" value="Femenino" checked="checked"/>Femenino
    &nbsp;
    <input type="radio" name="sexo" id="sexo" value="Masculino"/>Masculino
  </td>
</tr>

<tr>
  <td>Ocupación</td>
  <td>
    <input type="checkbox" name="trabaja" id="trabaja" value="Trabaja"
checked="checked"/>Trabaja &nbsp;
    <input type="checkbox" name="estudia" id="estudia" value="Masculino"/>Estudia
  </td>
</tr>

<tr>
  <td>Ciudad</td>
  <td>
    <select name="ciudad" id="ciudad" size="1">
      <option>Bogotá</option>
      <option selected="selected">Medellín</option>
      <option>Cali</option>
    </select>
  </td>
</tr>

<tr>
  <td>Fruta favorita</td>
  <td>
```

```
<select name="lstfruta" id="lstfruta" size="3" multiple="multiple">
  <option value="Mango">Mango</option>
  <option value="Pera">Pera</option>
  <option value="Uva">Uva</option>
  <option value="Manzana">Manzana</option>
  <option value="Banano">Banano</option>
</select>
</td>
</tr>

<tr>
  <td>Comentarios</td>
  <td><textarea name="txtcom" id="txtcom" cols="30" rows="10">Escriba aquí su
comentario</textarea></td>
</tr>

<tr>
  <td colspan="2" align="center">
    <input type="submit"/>&nbsp;
    <input type="reset"/>&nbsp;
    <input type="button" name="btn1" id="btn1" value="Mensaje" onclick="alert('Hola, esto es
Javascript');"/>
  </td>
</tr>

</table>

</form>

</center>

<hr/>
<h3>Marcos flotantes</h3>
<br/>
<iframe name="marco" src="pagina2.html" width="700" height="700"></iframe>
<br/>
<a href="http://www.google.com" title="Ir a Google" target="marco">Abrir sitio Google en el marco
flotante</a>
<hr/>

<embed src="videos/video1.avi" autostart="false" loop="true"/>
La etiqueta embed no está disponible para Linux, no pertenece al estándar HTML
<hr/>

<a href="#arriba" name="abajo" title="Ir arriba">Ir arriba</a>
<hr/>

</body>
</html>
```

## **EJERCICIOS PROPUESTOS**

Diseñe sitios web que permitan solucionar los siguientes requerimientos

1. Diseñe un sitio web para su hoja de vida virtual. Éste debe contener 4 páginas para informar acerca de sus datos personales, estudios realizados, experiencia laboral y referencias personales. El sitio debe contener un index o página de inicio y permitir la navegación entre las páginas. Además, debe tener un enlace que permita ir a una página para el registro de usuarios (para esto, diseñe un formulario apropiado) y otro hacia una página con una galería de 9 imágenes o más; al pulsar clic sobre una de las imágenes, éstas deben verse en un tamaño mayor.
2. Diseñe un sitio web para una empresa (escoja el tipo). Éste debe estar compuesto de las siguientes páginas: “Quiénes son” (misión, visión, objetivos, historia, etc.), “Productos” (la tienda; muestre cada producto con imágenes, y que éstas al ser pulsadas se vean mas grandes y con información acerca del producto, puede ser dentro de un marco flotante), “Registrarse” (contiene un formulario para registrar los usuarios que ingresan al sitio), “Contáctenos” (contiene información general para ubicar la empresa y un formulario para enviar sugerencias y comentarios del sitio), “Enlaces externos” (contiene enlaces relacionados con el sitio), además debe tener un “Index” (proponga un diseño apropiado para el sitio).
3. *Revista virtual – Centro de noticias.* Este ejercicio consiste en crear un sitio para la publicación de una revista virtual o un centro de noticias de un tema de preferencia: Ciencia, computación e informática, deportes, naturaleza, etc. El sitio debe contener información general como la comentada en los ejercicios anteriores y ser llamativo visualmente.
4. Sistema de encuestas. Diseñe un sitio web que permita realizar diferentes tipos de encuestas para una firma encuestadora. Generalmente manejan al menos 10 tipos de encuestas, para lo cual se le pide ser muy creativo ya que cada una debe presentarse de forma diferente. Algunos temas que desarrollan tienen que ver con la tecnología, con el deporte y con la economía entre otros. Las opciones para las encuestas pueden ser varias y no siempre son de selección múltiple con única respuesta.

### **Sugerencia**

Puede crear una plantilla para cada sitio, así se ahorrará trabajo, y recuerde cual es el fin del sitio y que éste es para un público determinado. Además, puede aplicar otros elementos a cada sitio (videos, marcos flotantes, etc.)



---

## HTML DINÁMICO (DHTML – DYNAMIC HTML)

<b>CLASE No. 5</b>
--------------------

### **¿QUE ES DHTML?**

Es una extensión del HTML que permite crear páginas web mas funcionales, agregando las poderosas características de JavaScript y CSS. El HTML Dinámico o DHTML (del inglés *Dynamic HTML*) designa el conjunto de técnicas que permiten crear sitios web interactivos utilizando una combinación de lenguaje HTML estático, un lenguaje interpretado en el lado del cliente (como JavaScript), el lenguaje de (CSS) y la jerarquía de objetos de DOM (Document Object Model o Modelo de Objeto de Documento).

Una página de HTML Dinámico es cualquier página web en la que los scripts en el lado del cliente cambian el HTML del documento, después de que éste haya cargado completamente, lo cual afecta a la apariencia y las funciones de los objetos de la página. La característica dinámica del DHTML, por tanto, es la forma en que la página interactúa con el usuario cuando la está viendo, siendo la página la misma para todos los usuarios.

En contraste, el término más general "página web dinámica" lo usamos para referirnos a alguna página específica que es generada de manera diferente para cada usuario, en cada recarga de la página o por valores específicos de variables de entorno. Este término no debe ser confundido con DHTML. Estas páginas dinámicas son el resultado bien de la ejecución de un programa en algún tipo de lenguaje de programación en el servidor de la página web (como por ejemplo PHP, Perl o ASP.NET), el cual genera la página antes de enviarla al cliente, o bien de la ejecución en la parte cliente de un código que crea la página completa antes de que el programa cliente (usualmente, un navegador) la visualice.

En una página DHTML, una vez ésta ha sido cargada completamente por el cliente, se ejecuta un código (como por ejemplo en lenguaje JavaScript) que tiene efectos en los valores del lenguaje de definición de la presentación, logrando así una modificación en la información presentada o el aspecto visual de la página mientras el usuario la está viendo.

Entre los usos más habituales del DHTML están el hacer menús desplegables, imágenes que cambian al pasar el cursor sobre ellas, objetos en movimiento, botones que permiten desplazar el texto que se está mostrando, textos explicativos que aparecen al situar el cursor sobre ciertas palabras clave, cronómetros, etc.

Otro uso interesante de esta tecnología es la creación de juegos de acción que utilizan el navegador web para funcionar, aunque tradicionalmente este tipo de desarrollos han sido complicados debido a las diferencias en el lenguaje y las características soportadas por los distintos navegadores existentes. Recientemente los navegadores más populares han empezado a soportar estándares comunes, como el DOM, lo cual ha facilitado mucho la creación de este tipo de aplicaciones.

### **HOJAS DE ESTILO CSS**



Figura 14. Algunos logos empleados para identificar al lenguaje de las Hojas de Estilos CSS

Permiten al usuario disponer de una colección de documentos HTML o de un espacio completo en la Web que utilizan uno o varios formatos de estilo, es decir, lo que anteriormente suponía la tediosa tarea de insertar muchas etiquetas `<font>`, gráficos y otras mezclas de elementos en un documento HTML.

Una hoja de estilo no sólo se aplica al estilo de página, sino también al estilo de las etiquetas utilizadas en la página.

Las hojas de estilo vienen a intentar volver a separar en un documento el estilo lógico del estilo físico, dejando este último en bloques de definición de estilos separados de la estructura del documento.

CSS son las siglas de "Cascade StyleSheet". Se trata de una especificación sobre los estilos físicos aplicables a un documento HTML, y trata de dar la separación definitiva del estilo lógico (estructura) y el estilo físico (presentación) del documento.

### **Estilos**

El estilo lógico se refiere a la lógica del documento: cabeceras, párrafos y demás, no se preocupa de la apariencia final, sino de la estructura del documento. Por el contrario, el estilo físico no se preocupa de la estructura del documento, sino por la apariencia final: párrafos con un cierto tipo de letra, tablas con un determinado color de fondo, etc.

La finalidad de las hojas de estilo es crear unos estilos físicos, separados de las etiquetas HTML, en lugar de tenerlos como parámetros de las etiquetas, y aplicarlos en los bloques de texto en los que se quieran aplicar. Estos estilos podrán ser modificados en algunas ocasiones desde JavaScript, y esto empieza a darnos un poco más de interactividad.

### **Capas**

Por otra parte, tenemos las capas, que vienen a darnos la solución al problema de poner elementos justo en la posición que queramos, evitándonos tener que hacer artificios para obtener el resultado buscado. Una capa será una parte más del documento que puede ser situada en cualquier posición del mismo, consiguiendo que se solape sobre algunos elementos si es lo que necesitamos, adecuando sus márgenes y otras propiedades a lo que queramos hacer. La etiqueta para la creación de capas es `<div>...</div>`, en ella podemos indicar la posición de la capa dentro del documento. Las capas son utilizadas actualmente para la maquetación de sitios

web, entrando a reemplazar el diseño por tablas que aun es usado, permitiendo organizar los contenidos desplegados. Además, es posible crear hojas de estilos CSS para las capas.

### Aplicación directa de estilos en etiquetas

Tenemos varias posibilidades para definir un estilo: especificarlo directamente en la etiqueta en la que queremos usarlo, definirlo aparte y aplicarlo en las etiquetas que queramos, o definir estilos globales para las etiquetas (que podrán ser cambiados en las que no se desee aplicarlos).

Para aplicar un estilo a una etiqueta concreta, usaremos la sintaxis:

```
<etiqueta style="propiedad1:valor;...;propiedadN:valor"> ... </etiqueta>
```

Etiqueta es la etiqueta de HTML en la que queremos dar una apariencia concreta (<p>, <b>, <i>, ...). *style* es el parámetro que indica que vamos a aplicar el estilo definido a continuación a la etiqueta en la que se encuentra. La definición del estilo son pares *propiedad:valor* separados por punto y coma. *Propiedad* será la característica de la etiqueta que queramos modificar (el color, el tamaño de la fuente, el tipo de letra, ...) y *valor* es el valor que queremos darle (color negro, 8 puntos de tamaño de letra, ...).

Por ejemplo, si tenemos un texto en negrita y queremos que salga con un tamaño de letra 14 y en color rojo, haremos:

La negrita que vemos <b style="font-size:14pt;color:red"> es más grande y está en rojo</b>, cuyo efecto es así:

La negrita que vemos es más grande y está en rojo.

### Redefinición de etiquetas

Suele ocurrir es que queremos definir estilos que se apliquen a todas las etiquetas del documento, es decir, queremos que todo el documento tenga un cierto tipo de letra, que las tablas tengan otro, que las cabeceras tengan un color determinado, para ello, definiremos estilos globales por medio de la etiqueta <style> ... </style> como sigue:

### Etiqueta <style>...</style>

Permite especificar definiciones de hojas de estilos CSS. Esta etiqueta debe ubicarse en la cabecera del documento. El atributo *type* especifica el tipo de estilo. La sintaxis para su manejo es:

```
<style type="text/css">
<!--
  Etiqueta1, Etiqueta2 : {propiedad1:valor}
  Etiqueta3 : {propiedad1:valor;...;propiedadS:valor}
  Etiqueta4 : {propiedad1:valor;...;propiedadT:valor}
  .Clase1 : {propiedad1:valor;...;propiedadT:valor}
//-->
</style>
```

Podemos aplicar el mismo estilo a varias etiquetas, escribiéndolas separadas por comas y, a continuación, la especificación del estilo según pares *propiedad:valor* separados por punto y coma y encerrados entre llaves { }. En un bloque de estilo global podremos definir cuantos estilos

queramos. Aparece un .Clase1; se refiere a las llamadas clases, que nos permitirán que una etiqueta concreta tenga una apariencia distinta a la definida como global.

Es recomendable que definamos estos estilos globales dentro de la cabecera del documento (entre <head>...</head>) para asegurarnos de que se aplicarán a todas las etiquetas para las que se haya definido un estilo. Veamos un ejemplo.

### Ejemplo

```
<html>
<head>
<title> Ejemplo con bloque de estilo </title>
<style type="text/css">
<!--
body {font-family:verdana,sans-serif;font-size:x-small;
margin-left:0.25in; margin-right:0.25in}
h2 {font-family:verdana,sans-serif;font-size:14pt;color:red}
b, td {font-family:verdana,sans-serif;font-size:x-small;color:olive}
th {font-family:verdana,sans-serif;font-size:x-small;
color:white;background-color:#0080c0}
pre, tt, code {font-family:courier new,courier;font-size:9pt;color:maroon}
//-->
</style>
</head>
<body bgcolor=white>
<h2>Prueba de definición de estilos en un bloque aparte</h2>
como puede verse, la apariencia de esta página queda
completamente definida por los estilos que hemos
especificado en el bloque style en la cabecera del
documento. los márgenes son más amplios de lo habitual,
la <b>negrita</b> tiene un tamaño y un color fijos, los
trozos de texto en teletipo como <tt>este fragmento</tt>
también tienen definida su fuente, tamaño y color,
y vamos a ver cómo quedan las tablas, para finalizar
el ejemplo: <p>

<center>
<table border=1 cellspacing=2 cellpadding=2>
<tr> <th>cabecera 1</th> <th>cabecera 2</th> </tr>
<tr> <td>celda (1,1)</td> <td>celda (1,2)</td> </tr>
</table>
</center>
</body>
</html>
```

### Ejemplo

```
<html>
<head>
<title>Aplicación de estilos</title>
```

```
<style type="text/css">
<!--
a {color:#CC0000; font:Georgia, "Times New Roman", Times, serif; font-style:italic}
i,b {color:#000099; font-size:36px; font-family:Geneva, Arial, Helvetica, sans-serif}
-->
</style>
</head>
<body>
<p>&nbsp;</p>
<p><strong style="font-size:14px; color:#FF0000; font:Stencil ">Estilo en la etiqueta</strong></p>
<p><a href="controles.htm">Abrir una página</a></p>
<p><i>Este texto en cursiva tiene un estilo aplicado</i></p>
<p><b>Texto con el mismo estilo aplicado en el último ejemplo</b></p>
<p>&nbsp;</p>
</body>
</html>
```

### Separar CSS de HTML

Como sabemos definir estilos globales, sería interesante tenerlos definidos en un archivo aparte, pues si queremos dotar a todas las páginas de los mismos estilos, no es tarea grata copiar y pegar la definición de los estilos en cada una de las páginas.

Afortunadamente, sí podemos definir los estilos en un fichero distinto al documento HTML, y después referenciarlo desde el propio documento HTML. Esto lo haremos con la siguiente etiqueta, dentro de la cabecera del documento (<head>...</head>)

```
<link rel="stylesheet" type="text/css" href="url_hoja.css">
```

Veamos un ejemplo que consta de dos archivos: uno .html y el otro .css.

### Ejemplo: ejemplo1.html

```
<html>
<head>
<title> Ejemplo con hoja de estilo externa </title>
<link rel="stylesheet" type="text/css" href="ejemplo1.css">
</head>
<body bgcolor=white>
<h2>Prueba de definición de estilos en una hoja de
estilo externa</h2>
<span class="miestilo">
Como puede verse, la apariencia de esta página queda
completamente definida por los estilos que hemos
especificado en el bloque style en la cabecera del
documento. los márgenes son más amplios de lo habitual,
la <b>negrita</b> tiene un tamaño y un color fijos, los
trozos de texto en teletipo como <tt>este fragmento</tt>
también tienen definida su fuente, tamaño y color,
y vamos a ver cómo quedan las tablas, para finalizar
el ejemplo: <p>
```

```
</span>
<center>
<table border=1 cellspacing=2 cellpadding=2>
<tr> <th>cabecera 1</th> <th>cabecera 2</th> </tr>
<tr> <td>celda (1,1)</td> <td>celda (1,2)</td> </tr>
</table>
</center>
<h3 class="miestilo">Otro texto con el estilo de clase aplicado sobre otra etiqueta</h3>
</body>
</html>
```

Si quisiéramos que otros documentos tuvieran el mismo estilo, sólo tendríamos que usar la etiqueta `<link>` para aplicarlos. Esa es la ventaja de las hojas de estilo externas. Observe que los archivos independientes de hojas de estilos tienen la extensión `.css`.

**Ejemplo:** ejemplo1.css

```
/*Observe como se comenta en CSS*/
/* Manual de CSS de WebEstilo.com */
/* Definición de estilos en un archivo aparte */

/* Estilo para el documento */
body {font-family:verdana,sans-serif;font-size:x-small;
margin-left:0.25in; margin-right:0.25in}

/* estilo para la cabecera de nivel 2 */
h2 {font-family:verdana,sans-serif; font-size:14pt;color:red}

/* estilos para otras etiquetas */
b, td {font-family:verdana,sans-serif;font-size:x-small;
color:olive}
th {font-family:verdana,sans-serif;font-size:x-small;
color:white;background-color:#0080c0}
pre, tt, code {font-family:courier new,courier;
font-size:9pt;color:maroon}

/*Este es un estilo de clase*/
.miestilo{

    text-align:justify;

    font-family:Arial, Helvetica, sans-serif;

    font-size:12px;

    color:#0066FF;

}
```

### Estilos de clase

Observe el uso de estilos de clase en el ejemplo anterior. Para definirlos es necesario anteponer un punto (.) antes del nombre; estos estilos son personalizados y pueden crearse los que quieran, igualmente pueden aplicarse en la etiqueta que se desee. Al llamar el estilo de clase se utiliza el atributo class, el cual está presente en todas las etiquetas, indicando como valor el nombre del estilo de clase sin el punto.

### Estilos de capas

Los estilos CSS pueden aplicarse a las etiquetas *div*, con esto podemos ajustar, o en otras palabras, maquetar nuestros sitios y publicar los contenidos como deseemos. En el siguiente ejemplo se tienen tres definiciones de estilos para tres etiquetas div, las cuales fueron creadas en alguna página y cuyo identificador en la hoja de estilos debe tener como prefijo el símbolo #:

### Ejemplo

```
#contenedor {  
margin: 0 auto;  
width: 922px;  
}  
  
#cabecera {  
width: 900px;  
float: left;  
padding: 10px;  
margin: 10px 0px 5px 0px;  
background-color: green;  
}  
  
#contenido{  
float: left;  
margin: 0px 5px 5px 0px;  
padding: 10px;  
width: 456px;  
display: inline;  
}
```

## CLASE No. 6

### JAVASCRIPT

JavaScript es un lenguaje de scripting basado en objetos, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. JavaScript es un dialecto de ECMAScript y se caracteriza por ser un lenguaje basado en prototipos, con entrada dinámica y con funciones de primera clase. JavaScript ha tenido influencia de múltiples lenguajes y se diseñó con una sintaxis similar al lenguaje de programación Java, aunque más fácil de utilizar para personas que no programan.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.

El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, la que desarrolló los primeros navegadores web comerciales. Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.

Tradicionalmente, se venía utilizando en páginas web HTML, para realizar operaciones y en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se ejecuta en el agente de usuario (cliente web), al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Inicialmente los autores lo llamaron Mocha y más tarde LiveScript pero fue rebautizado como JavaScript en un anuncio conjunto entre Sun Microsystems y Netscape, el 4 de diciembre de 1995.

En 1997 los autores propusieron JavaScript para que fuera adoptado como estándar de la European Computer Manufacturers Association ECMA, que a pesar de su nombre no es europeo sino internacional, con sede en Ginebra. En junio de 1997 fue adoptado como un estándar ECMA, con el nombre de ECMAScript. Poco después también como un estándar ISO.

JScript es la implementación de ECMAScript de Microsoft, muy similar al JavaScript de Netscape, pero con ciertas diferencias en el modelo de objetos del navegador que hacen ambas versiones sean incompatibles con frecuencia.

Para evitar estas incompatibilidades, el World Wide Web Consortium (W3C) diseñó el estándar Document Object Model (DOM, ó Modelo de Objetos del Documento), que incorporan Konqueror, las versiones 6 de Internet Explorer y Netscape Navigator, Opera la versión 7, y Mozilla Application Suite, Mozilla desde su primera versión.

### **Inserción de código Javascript en documentos HTML**

Para especificar código Javascript en una página web, debemos utilizar la etiqueta `<script>...</script>`, que preferiblemente se ubica en el encabezado de la página, haciendo que el script sea global (en caso de ser necesario, puede ubicar la etiqueta en otros puntos de la página, aunque también se acostumbra hacerlo directamente en las etiquetas).

#### **Etiqueta `<script>...</script>`**

Permite insertar código de script en un documento web. Algunos atributos son:

**language:** Especifica el lenguaje de script a utilizar.

**type:** Especifica el tipo de script a utilizar

**src:** Permite indicar un archivo externo donde se encuentra el script. En el caso de Javascript, pueden crearse archivos con extensión .js y luego ser incluidos en la página mediante este atributo.

**Ejemplo:** *hola\_mundo.html*



El clásico ejemplo “*hola mundo*”, permite ilustrar la salida estándar (impresión por pantalla) del lenguaje. Aquí se muestran dos formas de mostrar contenido: con la instrucción *document.write* (método *write* del objeto *document*, el cual genera HTML o despliega contenido en la ventana del navegador) y con la función *alert()*, que muestra una ventana con un mensaje.

```
<html>
<head>
<title>Página con Javascript</title>
<script type="text/javascript" language="javascript">
<!--
  document.write("<b>Hola, bienvenidos a Javascript</b>");
  alert('¡Hasta pronto!');
-->
</script>
</head>
<body>
<p>Esta página contiene un script de Javascript</p>
</body>
</html>
```

### Diálogos de Javascript

Javascript utiliza tres tipos de diálogos con diferentes finalidades. Todos ellos son métodos (funciones) del objeto window del DOM (*Document Object Model – Modelo de Objeto de Documento*).

#### Función alert()

Muestra una ventana de diálogo con un mensaje y un botón.

#### Función confirm([mensaje])

Muestra una caja de diálogo con un mensaje y dos botones: *Aceptar* y *Cancelar*. Devuelve *true* si presiona el botón *Aceptar* y *false* en otro caso.

#### Función prompt([mensaje\_diálogo], valor\_por\_defecto )

Muestra una ventana de diálogo con un mensaje y dos botones: *Aceptar* y *Cancelar*. Devuelve la cadena ingresada por el usuario si presiona el botón *Aceptar* y *null* en otro caso. Esta función se utiliza para ingresar información del usuario.

### Finalización de instrucciones en Javascript

Cada línea en el código de Javascript representa una instrucción y debe terminar en *punto y coma* (;); aunque Javascript no obliga a su uso, es una buena técnica hacerlo. En caso de que incluya dos instrucciones en el mismo renglón, es posible que tenga que utilizar obligatoriamente el *punto y coma* (;) para separarlas.

### Declaración de Variables en Javascript

Las variables son espacios en memoria donde almacenamos información de algún tipo. Javascript no obliga a la declaración de variables, sin embargo es una buena técnica hacerlo y puede hacerse en cualquier parte del script. Sintaxis:

```
var <lista de variables>;
```

Los nombres de variables deben ser según las reglas sugeridas de la lógica de programación. También debe tenerse presente que Javascript es un lenguaje sensible a las mayúsculas y minúsculas.

El alcance de las variables depende de en que parte del script se definen o utilizan: las variables que están definidas por fuera de las funciones, se consideran globales a la página, esto es, pueden ser accedidas desde cualquier parte del código ("hacia abajo", después de definirla) y desde cualquier función; mientras que las variables que se encuentran en las funciones definidas por el usuario, tienen un alcance local, ésta desaparece una vez deja de ejecutarse la función.

### **Inicialización de variables**

Se hace de la misma manera que en otros lenguajes, utilizando el operador (asimétrico) de asignación *igual* (=). Sintaxis:

`<variable> = <valor o expresión>;`

### **Nota**

En los ejemplos sobre Javascript, en general, vamos a escribir solo el código del script; en caso de ser necesario, y en algunas ocasiones, se indicará el código HTML.

### **Ejemplo**

```
var x, y, nombre;  
x=0;  
nombre="Pedro Zapata";  
y=true;  
alert("Nombre: " + nombre);  
document.write("x=" + x + "y=" + y);
```

### **Nota**

Observe el operador **+** como es utilizado para concatenar cadenas, mas adelante se trata este tema. Se dice que este operador está sobrecargado, pues además de esta operación, también puede utilizarse para realizar sumas aritméticas.

### **Tipos de datos en Javascript**

Los tipos de cada variable en Javascript no están tan claros como en Java, C o C++. El intérprete asigna el tipo de una variable según el uso que se esté haciendo de ella; sin embargo, el manejo de tipos en Javascript es muy flexible, pudiendo en un script asignar valores de distintos tipos a una misma variable sin que se afecte la ejecución del código o el rendimiento del programa.

Algunos tipos que soporta el lenguaje son:

#### **Enteros**

Permite almacenar números sin decimales.

#### **Flotantes**

Este tipo de dato es utilizado para guardar números que pueden o no tener una parte decimal, tal como 3.141592, 2.8182, etc.

### String

Javascript considera las cadenas como objetos de la clase String, por lo que se puede acceder a un conjunto de métodos y propiedades que esta contine. Las cadenas son secuencias de caracteres alfanuméricos (incluyendo los signos especiales) y pueden estar delimitadas por comillas (") o apóstrofes (').

### Booleanos

Javascript permite trabajar con variables lógicas (booleanas) que admiten dos valores, que a la vez son constantes del lenguaje: *true* y *false*.

### Arrays

Los arreglos en Javascript utilizan índices (index) para acceder a sus elementos. Javascript solo soporta arreglos unidimensionales (vectores), aunque con algunos "trucos" de programación, pueden simularse arreglos bidimensionales (matrices). Los arreglos al ser considerados en Javascript como objetos, obligatoriamente deben ser declarados, lo cual se hace de la siguiente forma:

```
var <nombre_arreglo>=new Array([tamaño]);
```

### Ejemplo

```
var vector=new Array();
```

### Manejo de comentarios

Un comentario es una instrucción que es ignorada por el intérprete a la hora de publicar la página; permiten hacer documentaciones o explicaciones de partes del código, o "tapar" algo que no queremos que se ejecute pero que no deseamos eliminar. En Javascript los comentarios pueden realizarse de dos formas:

```
// Esto es un comentario.
```

```
/*
```

```
Esto es otro comentario. Puede utilizarse cuando el comentario ocupa varias líneas de código.
```

```
*/
```

### Operadores

Javascript cuenta con los siguientes operadores:

#### Operadores aritméticos

Utilizados para realizar cálculos matemáticos

Nombre Operador	Símbolo	Prioridad
Negación unaria	-	Alta
Multiplicación	*	Media
División	/	
Módulo	%	
Suma	+	Baja
Resta	-	

Tabla 4. Operadores aritméticos en Javascript

### Notas

- La prioridad se refiere al orden en que los operadores se efectúan en una expresión aritmética: los de mayor prioridad se efectúan primero.
- Las operaciones encerradas entre paréntesis se efectúan primero, por lo que tienen mayor prioridad.

### Operadores relacionales

Permiten realizar comparaciones entre dos expresiones, devolviendo un valor booleano (true o false); esto son:

Nombre Operador	Símbolo	Prioridad
Igual	==	Alta
Diferente	!=	
Mayor que	>	Media
Menor que	<	
Mayor o igual que	>=	Baja
Menor o igual que	<=	

Tabla 5. Operadores relacionales o de comparación en Javascript

### Operadores lógicos o booleanos

Permiten conectar expresiones de comparación y realizar operaciones lógicas. El valor devuelto (true o false) depende del conectivo lógico utilizado, según las leyes del álgebra proposicional y booleana; estos son:

Nombre Operador	Símbolo	Prioridad
Negación	!	Alta
Conjunción	&&	Media
Disyunción		Baja

Tabla 6. Operadores lógicos o booleanos en Javascript

### Operadores aritméticos/asignación

En los ejemplos anteriores aparecieron algunas expresiones con combinaciones de operadores. Javascript implementa varios operadores que abrevian la escritura normal de expresiones aritméticas, veamos cuales son.

Nombre Operador	Símbolo	Ejemplo
Incremento en una unidad	++	Ejemplo: Escribir: a++ Equivale a escribir:

Decremento en una unidad	--	a=a+1 Igual ocurre con el operador --
Acumulador de sumas	+=	Ejemplo: Escribir: s+=3 Equivale a escribir: s=s+3 De manera similar se comportan los operadores -= *= /= %=
Acumulador de restas	-=	
Acumulador de productos	*=	
“Acumulador” de divisiones	/=	
“Acumulador” de módulos	%=	
Inicialización (asignación simple)	=	A = 4;

Tabla 7. Operadores aritméticos/asignación en Javascript

### Estructuras de control

En los lenguajes de programación, las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa. Con ellas se puede:

- De acuerdo a una condición, ejecutar un grupo u otro de sentencias.
- Ejecutar un grupo de sentencias un número determinado de veces
- Interrumpir la ejecución normal del programa

Todas las estructuras de control tienen un único punto de entrada y un único punto de salida. Éstas se pueden clasificar en: decisión, iteración y de control avanzadas.

### Condicionales

Se utilizan para tomar decisiones a partir de valores booleanos obtenidos de la comparación de expresiones lógicas. Veamos como se implementan en Javascript:

#### Condicional simple y condicional compuesto

Instrucción if - else

Evalúa una determinada condición o expresión de comparación, en caso de ser verdadera, se ejecuta un bloque de instrucciones. Si dicha condición no se cumple, esto es, es falsa, entonces ninguna de las instrucciones es ejecutada a no ser que se especifique la cláusula *else*.

Sintaxis

```
if (<expresión_de_comparación>)
{
```

```
    bloque de instrucciones Javascript;  
}  
[else  
{  
    bloque de instrucciones Javascript;  
}]
```

### Ejemplo

```
var valor1, valor2, resultado;  
valor1=5;  
valor2=7;  
  
if (valor1 > valor2)  
{  
    resultado = valor1 + valor2;  
}  
else  
{  
    resultado = valor2-valor1;  
}  
document.write(resultado);
```

### Nota

Observe el uso de llaves { } para encerrar ciertas partes del código, éstas nos indican inicio y fin de una estructura o bloque. Cuando la estructura de control que requiere llaves solo contiene una instrucción, puede omitirse su uso.

### Selector múltiple o estructura caso

Permite la ejecución de un bloque de instrucciones en función del valor que tome una expresión. Esta es la Estructura Caso o el Selector Múltiple utilizado en lógica de programación como alternativa al condicional simple cuando se tienen más de dos salidas lógicas.

### Instrucción switch<sup>4</sup>

Sintaxis

```
switch (<expresión>)  
{  
    case valor1:  
        Bloque de instrucciones valor 1;  
        break;  
  
    case valor 2:  
        Bloque de instrucciones valor 2;  
        break;
```

---

<sup>4</sup> La instrucción break utilizada en cada caso (case) se comenta en la subsección Bifurcación de control, que se encuentra mas adelante.

```
...
case valor N:
    Bloque de instrucciones valorN;
    break;

[default:
    Bloque de instrucciones por defecto;]
}
```

### Ejemplo

```
var usuario=prompt("Ingresa tu nombre de usuario","Escribe aquí");
var apellido=prompt("Ingresa tu apellido","Escribe aquí");
switch (usuario)
{
    case "JORGE":
        alert( "hola " + usuario + " como estás hoy");
        break;

    case "ANA":
        alert("hola " + usuario + " bienvenida");
        alert("Tu apellido es " + apellido);
        break;

    case "PEDRO":
        alert("Hola " + usuario + " , ya era hora");
        break;

    default:
        alert("No estás autorizado" + apellido + " ," + usuario);
}
```

### Notas

- Si un usuario escribe en los campos respectivos uno de los nombres: JORGE, ANA o PEDRO, aparecerá un mensaje de bienvenida acompañado del nombre que suministró, en caso contrario aparecerá un texto que dice "No estás autorizado".
- Observe la manera como se puede solicitar información del usuario con la función *prompt*.

### Ciclos

Los ciclos o bucles son estructuras de control que repiten un grupo de instrucciones mientras se cumpla una condición. Javascript cuenta con tres tipos de ciclos.

#### Ciclo while

Permite la repetición de un bloque de instrucciones un determinado número de veces de acuerdo a una condición: si ésta es verdadera, las instrucciones del ciclo se repiten, en caso contrario finaliza la ejecución de éste y continúa con la siguiente instrucción después del ciclo. Es posible que las sentencias del bucle no se lleguen a ejecutar nunca, ya que

antes de proceder a interpretar la primera instrucción se evalúa la condición, y si ésta resulta ser falsa, no entrará en las instrucciones del bloque.

Sintaxis

```
while (<condición>)  
{  
    Bloque de instrucciones;  
}
```

**Ejemplo**

```
var i=1;  
while(i <= 10)  
{  
    if(i % 2==0)  
        alert(i + " es par");  
    else  
        alert(i + " es impar");  
    if (i == 5)  
    {  
        alert( "El ciclo se interrumpe");  
        break;  
    }  
    i++;  
}  
alert( "Primera instrucción una vez finalizado el ciclo");
```

**Nota**

La instrucción break, hace que el ciclo se interrumpa una vez la variable i tenga el valor de 5.

**Ciclo do while**

Es similar al ciclo while, con la diferencia de que la condición se evalúa al final del ciclo, garantizando que las instrucciones dentro de él se ejecutarán por lo menos una vez.

Sintaxis

```
do  
{  
    Bloque de instrucciones;  
} while (<condición>);
```

**Ejemplo**

```
var i=1;  
do  
{
```



```
if(i % 2==0)
    alert(i + " es par");
else
    alert(i + " es impar");

    i++;
}while(i<=10)

}
alert( "Primera instrucción una vez finalizado el ciclo");
```

### Ciclo for

Tiene como fin repetir un bloque de instrucciones mientras cumpla una condición preestablecida. Se deben indicar tres parámetros: La condición que determina si se debe seguir ejecutando o no el bucle (condición), una condición que vaya haciendo cambiar algún parámetro que varíe el cumplimiento de la condición anterior (actualización) y por supuesto, una expresión que determine cuál es la situación de partida en el cumplimiento de dicha condición (inicialización).

#### Sintaxis

```
for (<inicialización>; <condición>; <actualización>)
{
    Bloque de instrucciones;
}
```

### Ejemplos

1. Imprimir varias líneas aplicando formato.

```
var i;
for (i = 1; i <= 10; i++)
{
    document.write("Línea número: <b>" + i + "</b><br/>");
}
```

2. Este ejemplo crea una tabla dinámicamente. Cuando hablamos de tabla dinámica, significa que se crea en tiempo de ejecución y no en tiempo de diseño.

```
var i;
document.write("<table width='200' border='1'><tr><td>Número </td></tr>");

for (i = 1; i <= 10; i += 1)
{
    document.write("<tr><td>" + i + "</td></tr>");
}

document.write("</table>");
```

```
var i, c, n;
n=parseInt(prompt("Ingrese el dato n", "0"));
document.write("<h2>Cuadrado de los números de 1 a " + n + "</h2>");

for (i = 1; i <= n; i = i + 1)
{
    c = i * i;
    document.write("<b>Número: " + i + "&nbsp;&nbsp;&nbsp;</b>Cuadrado <em>" + c +
"</em><br/>");
}
```

```
var i = 0;
while (i == 0)
{
    i++;
    document.write($i);
    if (i == 1)
```

```
continue;  
}
```

### Instrucción return

Devuelve el control del programa a una función o porción de programa que llama; a su vez puede devolver un valor a dicha función. En la subsección de funciones definidas por el usuario se ilustrará el uso de esta instrucción.

## CLASE No. 7

### Funciones definidas por el usuario

Al desarrollar aplicaciones nos vemos en la necesidad de utilizar la misma porción de código en varias ocasiones; para optimizar la programación y evitar tener que repetir segmentos de código, los lenguajes implementan los subprogramas, secuencias de código a las que podemos invocar pasándoles de 0 a N parámetros y que tras ejecutar una serie de operaciones, nos pueden devolver 0, 1 o mas valores. Normalmente, cuando devuelven un valor se les llama funciones, y en los demás casos, procedimientos. Javascript trata todos los subprogramas como funciones; si la función no devuelve ningún valor, podría verse como un procedimiento.

En Javascript, como en cualquier otro lenguaje de alto nivel, existen muchas funciones ya implementadas, pero en esta subsección vamos a ver como crear funciones e implementarlas en los sitios web.

### Definir Funciones

Para definir o declarar funciones, utilizamos la palabra reservada **function**, a continuación el nombre de la función y entre paréntesis los parámetros que recibirá. En el caso de que devuelva la función algún valor, se utiliza la sentencia return seguida de dicho valor.

El cuerpo de una función en Javascript es como se muestra en la siguiente sintaxis:

```
function <nombre_función>([Lista_de_argumentos])  
{  
    Cuerpo de la función;  
    ...  
    [return <valor_de_retorno>;]  
}
```

### Nota

Como buena técnica, se recomienda definir el prototipo de las funciones en las primeras líneas de código, antes de que estas sean invocadas o escribirlas en archivos independientes (.js) que luego puede incluir en las páginas requeridas mediante la etiqueta <script>:

```
<script type="text/javascript" src="nombre_archivo.js"></script>
```

Así, puede crear librerías propias que contengan diversas funciones (puede tener varias funciones en un mismo archivo, pero es conveniente que estén agrupadas por temáticas). Lógicamente, para invocar las funciones que se encuentren en archivos .js, debe incluirlos en los archivos HTML mediante la etiqueta comentada.

### **Pasar parámetros a una función**

Javascript solo permite manejar el paso de argumentos por valor, por lo que no pueden ser modificados en la función, esto es, no existen los parámetros por referencia o dirección.

### **Ejemplos**

1. Sumar dos números utilizando una función y en otra función mostrar el resultado

```
function sumar_numeros(a, b) //a y b son parámetros formales
{
    var s=a+b;
    return s;
}

function imprimir_suma(s)
{
    document.write("<p>La suma es " + suma + "</p>");
    alert("La suma es " + s);
}

var x=parseFloat(prompt("Ingrese un número", "0"));
var y=parseFloat(prompt("Ingrese otro número", "0"));
var suma=sumar_numeros (x, y); /*x e y son parámetros actuales. También pueden pasarse
constantes como parámetros, ejemplo: sumar_numeros(5,8)*/
imprimir_suma(suma);
```

2. Llamar a una función que muestre un saludo al usuario.

```
function imprimir_saludo()
{
    alert("Hola a tod@s");
}

imprimir_saludo();//Función sin parámetros que no retorna valor alguno.
```

### **Arreglos**

En los lenguajes de programación existen estructuras de datos especiales que nos sirven para guardar información más compleja que simples variables. Una estructura típica en todos los lenguajes son los arreglos. Existen varios tipos: unidimensionales (vectores), bidimensionales (matrices) y n-dimensionales (n>2). Sin embargo, Javascript solo soporta vectores, aunque pueden simularse matrices con algunos "trucos" de programación. Los vectores permiten almacenar varios valores del mismo tipo utilizando un mismo nombre de variable e identificando

cada elemento con un índice que representa la posición de éste en el vector, el cual va desde cero (0) -primera posición del vector- hasta el total de elementos del arreglo menos uno (1)

Los arreglos deben ser declarados en Javascript de la siguiente forma:

```
var <nombre_arreglo> = new Array([tamaño]);
```

Al crear arreglos estamos empleando el objeto Array, el cual contiene propiedades y métodos, algunos se enuncian a continuación.

### **Propiedades de los Arreglos**

#### **length**

Cuando se crea un arreglo con el constructor `Array()` o se define un *arreglo literal* se tiene una propiedad especial llamada *length* la cual especifica cuantos elementos contiene el arreglo. La propiedad `length` de un arreglo se actualiza automáticamente para mantener su consistencia cuando se agregan nuevos elementos o se sobre escribe el arreglo.

### **Métodos de los Arreglos**

#### **push()**

Este método inserta uno o más elementos al final del arreglo y regresa el último valor que insertó.

#### **pop()**

Este método elimina el último elemento que haya en el arreglo, decrementando la longitud del arreglo y regresando el valor que eliminó.

#### **sort()**

El método `sort()` de un arreglo, ordena los elementos de un arreglo.

#### **reverse()**

El método `reverse()`, ordena al revés los elementos de un arreglo.

#### **indexOf(<elemento>)**

Devuelve la posición de un elemento dentro de un arreglo.

#### **lastIndexOf(<valor>)**

Devuelve la posición del último elemento encontrado en un arreglo dependiendo de determinado valor.

### **Ejemplo**

Ilustración del uso de vectores.

```
var i
var vector1 = new Array(); //vector sin un número de posiciones definido
var vector2 = new Array(3); //vector de 3 posiciones.
var nombres = ["Pedro","Ana"]; //Crea el vector y lo inicializa.
var edades = new Array(28, 30, 54); //Crea el vector y lo inicializa.
```

```
alert("Vector nombres");
alert(nombres[0]);
alert(nombres[1]);

vector2[0]=5;
vector2[1]=10;
vector2[2]=15;

alert("Vector 1");

for(i=0; i<5; i++)
{
    vector1[i]=prompt("Ingrese un dato numérico para el vector 1", "0");
    alert("El dato ingresado fue " + vector1[i]);
}

alert("Vector 2 y edades");
for(i=0; i<vector2.length; i++) //la propiedad length guarda el número de elementos del vector
{
    alert("Edad " + i + ": " + edades[i]);
    alert("Dato " + i + ": " + vector1[i]);
}

alert("Total elementos vector edades: " + edades.length);
```

### Arreglos bidimensionales (matrices)

Como se mencionó anteriormente, las matrices pueden simularse en Javascript. Las matrices son arreglos de dos (2) dimensiones y para referirse a un elemento deben utilizarse dos índices, el primero para indicar la fila y el segundo para especificar la columna. A continuación se ilustra el manejo de matrices en Javascript con un ejemplo.

### Ejemplo

Esta página muestra la implementación de matrices con JavaScript. Este lenguaje solo admite arreglos unidimensionales (vectores), pero pueden simularse matrices haciendo que cada posición de un vector sea a su vez otro vector. Debe tener un número de filas menor o igual al número de columnas, ya que de lo contrario se generan resultados inesperados.

En este ejemplo se ilustra como crear matrices y mostrar sus elementos, además se imprimen algunas áreas típicas con un color diferente: diagonal principal y secundaria, triangular inferior y superior y un triángulo.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Language" content="es-co" />
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
<title>Matrices</title>

<script type="text/javascript">
var m=n=0;
var mat=new Array();
var mat1=new Array();

function crear()
{
    var i,j;
    //m y n se convierten a enteros para realizar operaciones de comparación numéricas
    m=parseInt(document.getElementById("txtnf").value);
    n=parseInt(document.getElementById("txtn").value);

    if(m<=n)
    {
        //Creación de la matriz

        for(i=0;i<n;i++)
        {
            mat[i]=new Array(m);
            mat1[i]=new Array(m);
        }

        //Llenado de la matriz con números aleatorios
        for(i=0;i<m;i++)
        {
            for(j=0;j<n;j++)
            {
                mat[i][j]=Math.round(Math.random()*10);
            }
        }

        limpiar_matriz();
        mostrar();
    }
    else
    {
        alert("Debido a las limitaciones de JavaScript para manejar matrices, debe ingresar un
        número de filas menor o igual al del número de columnas");

        document.getElementById("txtnf").focus();
    }
}

//-----
function mostrar()
{
    var i,j,contenido;
```

```
contenido="";

if(m>0 && n>0)
{
    contenido="<center><table border='1'>";

    for(i=0;i<m;i++)
    {
        contenido+="<tr>";

        for(j=0;j<n;j++)
        {
            contenido+="<td width='20' align='center'>" + mat[i][j] + "</td>";
        }

        contenido+="</tr>";
    }

    contenido+="</table></center><br>";
    document.body.innerHTML+=contenido;
}

else
    alert("No ha creado la matriz");
}

//-----
function diagonales()
{
    var i;
    if(m>0 && n>0 && m==n)
    {
        for(i=0;i<m;i++)
        {
            mat1[i][i]=mat[i][i];
            mat1[i][m-i-1]=mat[i][m-i-1];
        }
        mostrar_area('green');
    }
    else
        alert("No ha creado la matriz o esta no es cuadrada");
}

//-----
function triangular_inferior()
{
    var i,j;

    if(m>0 && n>0 && m==n)
```



```
{
  for(i=1; i<m; i++)
  {
    for(j=0; j<i; j++)
    {
      mat1[i][j]=mat[i][j];
    }
  }
  mostrar_area("blue");
}
else
  alert("No ha creado la matriz o esta no es cuadrada");
}

//-----
function triangular_superior()
{
  var i,j;

  if(m>0 && n>0 && m==n)
  {
    for(i=0; i<m-1; i++)
    {
      for(j=i+1; j<n; j++)
      {
        mat1[i][j]=mat[i][j];
      }
    }
    mostrar_area("pink");
  }

  else
    alert("No ha creado la matriz o esta no es cuadrada");
}

//-----
function triangulo1()
{
  var i,j;

  if(m>0 && n>0 && m==n)
  {
    for(i=0; i<m/2; i++)
    {
      for(j=i; j<n-i; j++)
      {
        mat1[i][j]=mat[i][j];
      }
    }
  }
}
```

```
    mostrar_area("red");
}

else
    alert("No ha creado la matriz o esta no es cuadrada");
}

//-----Muestra la matriz con elementos "sombreados"-----
function mostrar_area(color1)
{
    var contenido;
    contenido="";
    contenido="<center><table border='1'>";

    for(i=0;i<m;i++)
    {
        contenido+="<tr>";
        for(j=0;j<n;j++)
        {
            if(mat1[i][j]!=-1)
            {
                contenido+="<td width='20' align='center'><font color='"+ color1 + "'><b>" + mat[i][j] +
"</b></font></td>";
            }

            else
            {
                contenido+="<td width='20' align='center'>" + mat[i][j] + "</td>";
            }
        }
        contenido+="</tr></table></center><br>";
        document.body.innerHTML+=contenido;
        limpiar_matriz();
    }

}

//-----
function limpiar_matriz()
{
    for(i=0;i<m;i++)
    {
        for(j=0;j<n;j++)
        {
            mat1[i][j]=-1;
        }
    }
}

//-----
```

```
function restablecer()
{
    parent.location.href="matrices.html";
}

</script>
</head>
<body>
<form method="post" action="">
<h1 align="center">Matrices</h1><br />
<table style="width: 100%">
<tr>
<td width="225" style="width: 225px">Número de filas</td>
<td width="512"><input name="txtnf" type="text" id="txtnf" /></td>
</tr>
<tr>
<td style="width: 225px">Número de columnas</td>
<td><input name="txtnc" type="text" id="txtnc" /></td>
</tr>
<tr>
<td colspan="2">
<input name="btncrear" type="button" value="Crear matriz" onclick="crear()" />
<input name="btnmostrar" type="button" value="Mostrar" onclick="mostrar()" />
<input name="btndiagppal" type="button" value="Diagonales" onclick="diagonales()" />
<input name="bnttriangulo1" type="button" id="bnttriangulo1" value="Triángulo arriba"
onclick="triangulo1()" />
<input name="bnttriangularinferior" type="button" id="bnttriangularinferior" value="Triangular
inferior" onclick="triangular_inferior()" />
<input name="bnttriangularsuperior" type="button" id="bnttriangularsuperior" value="Triangular
superior" onclick="triangular_superior()" />
<input name="btnrestablecer" type="button" id="btnrestablecer" value="Restablecer página"
onclick="restablecer()" />
</td>
</tr>
</table>
</form>
</body>
</html>
```

### La Clase String

En Javascript, las cadenas son tratadas de manera similar a como lo hace Java, aunque sin el rigor de éste; es decir, las cadenas son *objetos* de la clase *String*, y como tal, pueden acceder a *propiedades* y *métodos* de dicha clase. Veamos algunos de ellos.

### Propiedades de la clase String

La clase String sólo tiene una propiedad: length. Veamos que hace dicha propiedad.

### **length**

Se encarga de almacenar el número de caracteres de la cadena.

### **Métodos de la clase String**

Los objetos de la clase String tienen a disposición diversos métodos para realizar muchas tareas interesantes. Primero vamos a ver una lista de los métodos que nos permitirán el tratamiento de datos ya sea para validar, buscar, etc., y luego vamos a ver otra lista de métodos que permiten dar una mejor apariencia al texto.

#### **charAt(<indice>)**

Devuelve el carácter que hay en la posición indicada como índice. La primera posición de una cadena es la cero (0).

#### **indexOf(<cadena>[,inicio])**

Devuelve la posición de la primera vez que aparece el carácter o cadena indicado por el primer parámetro, en una cadena. Si no encuentra el carácter o la cadena en el string, devuelve -1. El segundo parámetro es opcional y sirve para indicar a partir desde que posición se desea que empiece la búsqueda.

#### **lastIndexOf(<carácter>[,desde])**

Busca la posición de un carácter exactamente igual a como lo hace la función indexOf pero desde el final en lugar del principio. El segundo parámetro indica el número de caracteres desde donde se busca, igual que en indexOf.

#### **replace(<substring\_a\_buscar>,<nuevoStr>)**

Implementado en Javascript 1.2, sirve para reemplazar porciones del texto de un string por otro texto, por ejemplo, podríamos utilizarlo para reemplazar todas las apariciones del substring "xxx" por "yyy". El método no reemplaza en el string, sino que devuelve un resultante de hacer ese reemplazo. Acepta expresiones regulares.

#### **split(<separador>)**

Este método sólo es compatible con Javascript 1.1 en adelante. Sirve para crear un vector a partir de una cadena en el que cada elemento es la parte de la cadena que está separada por el separador indicado por el parámetro del método.

#### **substr(<inicio>,num\_car>)**

Devuelve una subcadena que empieza en el carácter indicado por el argumento inicio; el número de caracteres a substraer está determinado por el segundo argumento, num\_car.

#### **toLowerCase()**

Convierte todos los caracteres de una cadena a minúsculas.

#### **toUpperCase()**

Convierte todos los caracteres de una cadena a mayúsculas.

#### **toString()**

Este método lo tienen todos los objetos y se usa para convertirlos en cadenas.

Hasta aquí hemos visto los métodos que nos ayudarán a tratar cadenas. Ahora vamos a ver otros métodos que sirven para aplicar estilos a un texto. Utilizar estos métodos es similar a usar las etiquetas HTML respectivas. Veamos.

**anchor(<name>)**

Convierte en un ancla (sitio a donde dirigir un enlace) una cadena de caracteres usando como el atributo name de la etiqueta <a> lo que recibe por parámetro.

**big()**

Aumenta el tamaño de letra de la cadena. Es similar a encerrar una cadena dentro de la etiqueta <big>...</big>.

**blink()**

Hace que parpadee el texto, es similar a utilizar la etiqueta <blink>. Solo aplica en navegadores Netscape Navigator.

**bold()**

Realiza lo mismo que las etiquetas <b>...</b> o <strong>...</strong>, que establecen un texto a negrita.

**fixed()**

Permite utilizar una fuente monoespaciada, etiqueta <tt>...</tt>.

**fontcolor(<color>)**

Establece el color de la fuente especificado como argumento. Es similar a utilizar la etiqueta <font color="color">...</font>.

**fontsize(<tamaño>)**

Establece un tamaño para la fuente según el número indicado como argumento. Es similar a utilizar la etiqueta <font size="número">...</font>.

**italics()**

Establece la fuente a cursiva. Es similar a utilizar las etiqueta <i>...</i> o <em>...</em>.

**link(<url>)**

Pone el texto como un enlace a la URL indicada como argumento. Es similar a utilizar la etiqueta <a href="url">...</a> con el atributo href indicado como parámetro.

**small()**

Es similar a utilizar la etiqueta <small>...</small>, para establecer un texto de menor tamaño

**strike()**

Similar a utilizar la etiqueta <strike>...</strike> o <s>...</s>, que sirve para que el texto aparezca tachado.

**sub()**

Establece el texto a subíndice. Similar a utilizar la etiqueta <sub>...</sub>.

### **sup()**

Establece el texto a superíndice. Similar a utilizar la etiqueta <sup>...</sup>.

### **Ejemplo**

Aplicaremos los métodos y propiedades de la clase String para determinar si una frase es o no un palíndromo. Un palíndromo es una frase que puede leerse de igual forma en cualquier sentido (de izquierda a derecha y viceversa). Algunos palíndromos son:

amad a la dama

anita lava la tina

dábale arroz a la zorra el abad

adán nada

anilina

```
<html>
<head>
<title>Frases palíndromos</title>

<script type="text/javascript">

function palindromo()
{
    var frase=document.getElementById("texto").value.toUpperCase();
    frase=quitar_espacios(frase);

    if (comparar_caracteres(frase))
        alert(document.getElementById("texto").value + " es palíndromo");
    else
        alert(document.getElementById("texto").value + " no es palíndromo");
}

//*****

function quitar_espacios(frase)
{
    var i=0;

    while (i<frase.length)
    {
        if (frase.substr(i,1)==' ')
            frase=frase.substr(0,i) + frase.substr(i+1,frase.length-i-1);
        else
            i=i+1;
    }
    return frase;
}

//*****

function comparar_caracteres(frase)
```

```
{
  var i, sw;
  i=0;
  sw=true;

  while (i<=frase.length/2 && sw)
  {
    if (frase.substr(i,1)==frase.substr(frase.length-i-1,1))
      i=i+1;
    else
      sw=false;
  }
  return (sw);
}

</script>

</head>
<body>
<form name="form1" method="post" action="">
  <strong>Ingresa su frase:</strong>
  <input name="texto" type="text" id="texto" size="50" maxlength="100">
  <input name="boton" type="button" id="boton" value="Comprobar" onClick="palindromo()">

</form>
</body>
</html>
```

### La clase Date para Fechas y Horas

Javascript tiene a disposición la clase *Date* que permite la manipulación de fechas y horas. Para manipular sus datos, la clase cuenta con una serie de métodos, veamos cuales son.

#### **getDate()**

Devuelve el día del mes de 1 a 31

#### **getDay()**

Devuelve el día de la semana de 0 a 6

#### **getMonth()**

Devuelve el mes actual de 0 a 11, si queremos mostrar la fecha en formato dd/mm/yyy, tendremos que sumar uno a este valor.

#### **getFullYear()**

Devuelve el año en formato YYYY

#### **getYear()**

Devuelve el año en formato YY

#### **getHours()**

Devuelve la hora de 0 a 23

**getMinutes()**

Devuelve los minutos de 0 a 59

**getSeconds()**

Devuelve los segundos de 0 a 59

**getMilliseconds()**

Devuelve los milisegundos (0-999)

**getTime()**

Devuelve la fecha Unix (Número de milisegundos desde la medianoche del 1 de enero de 1970)

**getTimezoneOffset()**

Zona horaria del visitante

**Ejemplo**

Uso de la clase Date y de algunos de sus métodos.

```
<script type="text/javascript">

    ahora=new Date();
    alert("Hoy es: " + ahora);
    micumple=new Date("October 03, 1999 10:05:00");
    alert("Mi cumpleaños: " + micumple);
    alert("Día del mes: " + ahora.getDate());
    alert("Mes actual: " + ahora.getMonth());
    alert("Hora actual: " + ahora.getHours());

</script>
```

**La clase Math**

La clase Math es una de las clases nativas de Javascript, proporcionando los mecanismos para realizar diferentes operaciones matemáticas. Algunas operaciones se resuelven rápidamente con los operadores aritméticos que ya conocemos, como la resta o la suma, pero hay una serie de operaciones matemáticas adicionales que se tienen que realizar usando la clase Math, como pueden ser, calcular un coseno trigonométrico o efectuar una raíz cuadrada.

No es necesario instanciar un objeto (variable) de esa clase, se trabaja directamente con ella. Esto se permite porque las propiedades y métodos de la clase Math son lo que se conoce como propiedades y métodos de clase y para utilizarlos se opera a través de la clase en lugar de los objetos. Dicho de otra forma, para trabajar con la clase Math, no debemos utilizar la instrucción *new*, en cambio, utilizaremos el nombre de la clase para acceder a sus propiedades y métodos.



---

**Propiedades de la clase Math**

Las propiedades guardan valores que probablemente necesitemos en algún momento si estamos haciendo cálculos matemáticos. Es probable que estas propiedades resulten un poco raras a las personas que desconocen las matemáticas avanzadas, pero los que las conozcan, sabrán de su utilidad.

**E**

Número E o constante de Euler, la base de los logaritmos neperianos.

**LN2**

Logaritmo neperiano de 2.

**LN10**

Logaritmo neperiano de 10.

**LOG2E**

Logaritmo en base 2 de E.

**LOG10E**

Logaritmo en base 10 de E.

**PI**

Conocido número cuyo valor aproximado a 6 cifras decimales es  $\pi = 3.141592\dots$ . Este número es irracional y se obtiene de dividir el perímetro de la circunferencia entre el diámetro de la misma. Se representa con la letra griega  $\pi$  (pi) dado que fueron los griegos quienes descubrieron este número hace mas de 2000 años.

**SQRT1\_2**

Raiz cuadrada de un medio.

**SQRT2**

Raiz cuadrada de 2.

**Métodos de la clase Math**

Así mismo, tenemos una serie de métodos para realizar operaciones matemáticas típicas, aunque un poco complejas. Todos los que conozcan las matemáticas a un buen nivel conocerán el significado de estas operaciones.

**abs(x)**

Devuelve el valor absoluto de un número x pasado como argumento. El valor devuelto siempre será positivo o cero.

**acos(x)**

Devuelve el arcocoseno de un número en radianes.

**asin(x)**

Devuelve el arcoseno de un numero en radianes.

**atan(x)**

Devuelve el arcotangente de un numero.

**ceil(x)**

Devuelve el menor entero igual o mayor que x. Por ejemplo, ceil(3) equivale a 3, ceil(3.4) a 4, ceil(-2.2) a -2, etc.

**cos(x)**

Retorna el coseno de un número en radiades.

**exp(x)**

Retorna el resultado de elevar el número E por un número.

**floor(x)**

Lo contrario de ceil(), devuelve el mayor entero igual o menor que x. Por ejemplo, floor(3) equivale a 3, floor(3.4) a 3, floor(-2.2) a -3, etc.

**log(x)**

Devuelve el logaritmo neperiano de un número positivo.

**max(x,y)**

Retorna el mayor de 2 números.

**min(x,y)**

Retorna el menor de 2 números.

**pow(x,y)**

Calcula la potencia: “x” *elevado a la* “y”. Requiere de dos argumentos: la base (x) y el exponente (y). Este método, además de efectuar potencias, puede ser utilizado para calcular raíces, ya que sus argumentos son de tipo real.

**random()**

Devuelve un número seudo aleatorio entre 0 y 1. Método creado a partir de Javascript 1.1.

**round(x)**

Redondea un número al entero más próximo.

**sin(x)**

Devuelve el seno trigonométrico de x en radianes.

**sqrt(x)**

Retorna la raíz cuadrada de un número positivo o cero.

**tan(x)**

Devuelve la tangente trigonométrica de x en radianes.

**Ejemplo**

Vamos a ver un sencillo ejemplo sobre cómo utilizar métodos y propiedades de la clase Math para calcular el seno y el coseno de 2 PI radianes (una vuelta completa). En el bachillerato se aprendió

que el coseno de  $2\pi$  radianes ( $2\pi$  rad) debe dar como resultado 1 y el seno 0. También se muestran otras operaciones utilizando dicha clase y la función `alert` para mostrar los resultados.

```
document.write (Math.cos(2 * Math.PI))  
document.write ("<br/>")  
document.write (Math.sin(2 * Math.PI))  
alert(Math.abs(-2));  
alert(Math.pow(2,4));  
alert(Math.pow(2,2 * Math.PI));  
alert(Math.round(2*Math.SQRT2));
```

$2\pi$  rad ( $2 * \pi$  radianes) es el resultado de multiplicar 2 por el número  $\pi$ . Ese resultado es lo que recibe el método `cos`, y da como resultado 1. En el caso del seno el resultado no es exactamente 0 pero está aproximado con una exactitud de más de una millonésima de fracción. Se escriben los del seno y coseno con un salto de línea en medio para que quede más claro. También se muestran otros métodos, y puede probar con los demás.

### Otras funciones de Javascript

Javascript también cuenta con diversas funciones y métodos para realizar diferentes tareas, entre las que se tienen *parseInt* y *parseFloat*, métodos pertenecientes al objeto *window*, estos elementos enuncian a continuación.

### Función `isNaN(x)`

Evalúa si el argumento es un número o no. En caso de no serlo devolverá `true`. Es muy útil a la hora de validar entradas numéricas.

### Método `parseInt(x)`

Convierte su argumento a un número entero. Es muy útil cuando trabajamos con cálculos aritméticos. Sin embargo, la utilización del *parseInt* para validar números en muchos casos no resulta ser la solución más efectiva, debido a que permite la presencia de letras y/o espacios, y el resultado podría no ser el esperado. Para responder a esta cuestión, basta con ensayar funcionamiento de *parseInt* en algunos ejemplos:

"123456": este String retorna como resultado el número 123456 el cual es el resultado esperado.

"123456asd": este String retorna como resultado el número 123456 a pesar de que el String contenía letras.

"asd": este String retorna como resultado *NaN* (No a Number) el cual es el resultado esperado.

"": este String vacío retorna como resultado *NaN* el cual es el resultado esperado.

" 123456asd": este String (que contiene varios espacios al principio del número y letras al final) retorna como resultado el número 123456.

" 123 123 asd" este String (que contiene espacios y letras) retorna como resultado el número 123.

### Nota

Puede ensayar rápidamente, escribiendo en la barra de direcciones del navegador, órdenes similares a esta: `javascript:alert(parseInt("123"));`

### **Método parseFloat(x)**

Devuelve su argumento convertido a un número real. Se comporta de manera similar a parseInt, con la diferencia de que parseFloat admite el primer carácter punto (.) (y no la coma (,)) que encuentre como separador de cifras decimales, por ejemplo, ensaye escribiendo en la barra del navegador: `javascript:alert(parseFloat("123.15"))`; el resultado será un mensaje mostrando: 123.15. En cambio, `javascript:alert(parseFloat("123,15"))` mostrará el mensaje: 123.

### **Redireccionamiento de páginas**

Esta operación consiste en hacer que el programa abra un hipervínculo sin que el usuario tenga que pulsar con el clic un determinado enlace. Esto ocasiona que al entrar a una página, el sistema nos enviará a otra. Las instrucciones utilizadas para esto son:

**parent.location.href=<url>;**

**window.location=<url>;**

Cualquiera de estas dos funcionan de igual forma. Otras instrucciones utilizadas en el redireccionamiento de páginas son:

**navigator.appName**, que almacena el nombre del navegador instalado en nuestra máquina.

**setTimeout(<"url">,<milisegundos>)**, que permite hacer un redireccionamiento después de cierto número de milisegundos;

Veamos algunos ejemplos

### **Ejemplos**

1. Detectar el navegador instalado y redireccionar dos sitios diferentes.

```
<html>
<head>
<title>Detectar Navegador</title>
<script type="text/javascript">

    if (navigator.appName=="Netscape")
    {
        parent.location.href="http://www.google.com";
    }
    if (navigator.appName=="Microsoft Internet Explorer")
    {
        window.location="http://www.yahoo.com";
    }
</script>
</head>
</html>
```

2. Redireccionar a un sitio después de un número determinado de segundos.

```
function redireccionar_pagina()
{
    parent.location.href=pagina;
}

var pagina = "http://www.php.net";
var milisegundos = 5000; //Equivale a 5 segundos.
setTimeout("redireccionar_pagina()",milisegundos);
```

### Objetos en Javascript

Javascript es un lenguaje basado en objetos, mas no orientado objetos, ya que no permite establecer relaciones de herencia entre éstos, a diferencia de Java, un lenguaje totalmente orientado a objetos; aunque esto no es un problema, ya que si nos permite crear clases e instanciar objetos de dichas clases; para ello, primero es necesario definir su tipo (o clase) en una función. En JavaScript, la sentencia `this` se refiere al objeto en el que se utiliza. Una vez definida la clase, podemos crear variables (instanciar objetos) de esa clase de la siguiente manera:

`<objeto> = new nombre_clase([lista_de_valores]);`

Veamos un ejemplo para ilustrar lo comentado.

### Ejemplo

Creación de la clase “persona” y manipulación de objetos de tipo “persona”. Se crea una clase con dos propiedades: nombre y edad y un método para mostrar los datos. Luego se instancia el objeto `per1` y se manipula la información para éste.

```
<html>
<head>
<title>Objetos</title>
<script type="text/javascript">

function mostrar_datos()
{
    alert("Nombre: " + this.nombre + "\nEdad: " + this.edad);
}

function persona(nom, ed)
{
    this.nombre = nom;
    this.edad = ed;
    this.mostrar_datos=mostrar_datos;
}

per1 = new persona("Pedro", 26);
per1.mostrar_datos();

</script>
</head>
</html>
```

### Archivos externos Javascript .js

Las funciones de Javascript definidas por el usuario pueden guardarse en archivos independientes de extensión .js y luego incluirlas en las páginas mediante el atributo *src* de la etiqueta de script. Así mismo, puede escribir código en estos archivos que no sea exclusivo de una función. Es recomendable crear subcarpetas en el sitio para guardar los códigos de Javascript externos, de forma similar a como se hace con los archivos .css de las hojas de estilos, logrando de esta forma una mayor organización de los contenidos de éste.

Se pueden insertar las etiquetas de script que sean necesarias en caso de tener varios archivos .js (esta es una práctica común, ya que a medida que crecen las aplicaciones web, es necesario desarrollar nuevos módulos, componentes, etc., y se vuelve muy útil clasificarlos y separarlos en diferentes archivos, facilitando la labor de tratamiento y mantenimiento del código).

La forma de la etiqueta sería la siguiente, suponiendo que el archivo .js externo se encuentra en una subcarpeta del sitio llamada "scripts":

```
<script type="text/javascript" src="scripts/archivoexterno.js"></script>
```

De esta forma, al incluir esta etiqueta en la página, se podrá disponer de todo el código Javascript que contenga el archivo .js. Cabe anotar que dentro de este par de etiquetas que llaman un script externo, no debe ponerse código, ya que de hacerlo, se generarían errores en la ejecución. De otro lado, los archivos .js solo pueden contener código Javascript, el código HTML debe ser generado dentro del mismo script como cadenas de caracteres.

### Ejemplo general sobre Javascript

Este ejemplo ilustra el uso de algunos de los aspectos comentados anteriormente, entre ellos, las estructuras de control, arreglos, cadenas y funciones. También se muestra como acceder a los datos de un formulario HTML con Javascript y utilizando divs para mostrar información.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Vectores y envío de contenido al navegador</title>

<script language="javascript" type="text/javascript">
var vec=new Array(); //Crearé un vector sin un tamaño definido
var n=0; //Significa: Vector vacío

//-----
function agregar()
{
    var dato = document.getElementById('txtdato').value;

    if (dato.indexOf(" ",0) != -1 || isNaN(dato) || dato == "")
    {
```

```
        alert("Ingrese datos numéricos. \n Para continuar, presione clic en el botón Aceptar.");
    }

    else
    {
        vec[n] = parseFloat(dato);
        n++;
    }

    document.getElementById('txtdato').value = "";
    document.getElementById('txtdato').focus();

}

//-----
function mostrar()
{
    var i, contenido;

    contenido = "<table border='1'><tr>";

        for(i=0; i<n; i++)
        {
            contenido += "<td width='20' align='center'>" + vec[i] + "</td>";
        }

    contenido += "</tr></table><br/>";
    document.getElementById("divcontenido").innerHTML += contenido;

    /*También podría utilizarse una forma similar a esta: document.body.innerHTML += contenido;
    El problema de esta forma, es que se escribe sobre el cuerpo de la página y no hay una
    forma sencilla de limpiar el anterior contenido para mostrar el nuevo */
}

//-----
function crear_lista()
{
    var i, contenido;
    contenido = "<select name='lista' id='lista'>";

        for(i = 0; i < n; i++)
        {
            contenido += "<option>" + vec[i] + "</option>";
        }

    contenido += "</select>";
    document.getElementById("divcontenido").innerHTML += contenido;
}
```

```
//-----  
function total_datos()  
{  
    return n;  
}  
  
//-----  
function suma()  
{  
    var i, s = 0;  
  
    for(i = 0; i<n; i++)  
        s = s + vec[i];  
  
    return s;  
}  
  
//-----  
function promedio()  
{  
    return suma() / n;  
}  
  
//-----  
function mayor()  
{  
    var i, m;  
    m = vec[0];  
  
    for(i = 1; i < n; i++)  
    {  
        if(vec[i] > m)  
            m = vec[i];  
    }  
    return m;  
}  
  
//-----  
function menor()  
{  
    var i, m;  
    m = vec[0];  
  
    for(i = 1; i < n; i++)  
    {  
        if(vec[i] < m)  
            m = vec[i];  
    }  
}
```



```
    return m;
}

//-----
function buscar(d)
{
    var i, pos;
    i = 0;
    pos = -1;
    d = parseFloat(d);
    while(i < n && pos == -1)
    {
        if(vec[i] == d)
            pos = i;
        else
            i++;
    }
    return pos;
}

//-----
function eliminar(pos)
{
    var i;

    for(i = pos; i < n - 1; i++)
    {
        vec[i] = vec[i+1];
    }
    n--;
}

//-----
function insertar(pos,d)
{
    var i;
    pos = parseInt(pos);
    for(i = n; i > pos; i--)
    {
        vec[i] = vec[i-1];
    }

    vec[pos] = parseFloat(d);
    n++;
}

//-----
function modificar(pos, nd)
{

```

```
var i, pos;
pos = parseInt(pos);
vec[pos] = parseFloat(nd);
}

//-----
</script>

</head>
<body>
<form name="f1" id="f1">
  <p>Ingresa un dato al vector
    <input name="txtdato" type="text" id="txtdato">
  </p>
  <p>
    <input name="btn1" type="button" id="btn1" value="Agregar" onClick="agregar();" />

    <input name="btn2" type="button" id="btn2" value="Mostrar" onClick=" if(total_datos())>0)
mostrar(); else alert('No hay datos'); " />

    <input name="btn3" type="button" id="btn3" value="Crear lista con datos del vector" onClick="
if(total_datos())>0) crear_lista(); else alert('No hay datos'); " />

    <input name="btn4" type="button" id="btn4" value="Total datos" onClick="alert('Total datos:
' + total_datos());" />

    <input name="btn5" type="button" id="btn5" value="Suma" onClick="
    if(total_datos())>0) alert('Suma: ' + suma()); else alert('No hay datos'); " />

    <input name="btn6" type="button" id="btn6" value="Promedio" onClick=" if(total_datos())>0)
alert('Promedio: ' + promedio()); else alert('No hay datos'); " />

    <input name="btn7" type="button" id="btn7" value="Mayor dato" onClick="
if(total_datos())>0) alert("Mayor dato: ' + mayor (); else alert('No hay datos'); " />

    <input name="btn8" type="button" id="btn8" value="Menor dato" onClick="
if(total_datos())>0) alert("Menor dato: ' + menor (); else alert('No hay datos'); " />

  <br/>

  <input name="btn9" type="button" id="btn9" value="Buscar" onClick="
    if(total_datos())>0)
    {
      d=prompt('Ingresa el dato a buscar:','');
      pos=buscar(d);
      if(pos!=-1)
      {
        alert('El dato ' + d + ' se encuentra en la posición: ' + pos);
      }
    }
  </input>
</form>
</body>
</html>
```

```
        else
            alert('El dato ' + d + ' no existe en el vector.');
```

```
    }
    else
        alert('No hay datos');
```

```
" />

<input name="btn10" type="button" id="btn10" value="Eliminar" onClick="
    if(total_datos())>0)
    {
        d = prompt('Ingrese el dato a eliminar:', '');
        pos = buscar(d);
        if(pos != -1)
        {
            eliminar(pos);
            alert('El dato ' + d + ' fue eliminado del vector.');
```

```
            mostrar();
        }
        else
            alert('El dato ' + d + ' no existe en el vector.');
```

```
    }
    else
        alert('No hay datos');
```

```
" />

<input name="btn11" type="button" id="btn11" value="Insertar" onClick="
    if(total_datos())>0)
    {
        pos = prompt('Ingrese la posición donde insertará el dato:', '');
        if(pos >= 0 && pos <= total_datos())
        {
            d = prompt('Ingrese el dato a insertar:', '');
            insertar(pos, d);
            alert('El dato ' + d + ' fue insertado en el vector en la posición ' + pos);
            mostrar();
        }
        else
            alert('La posición ' + pos + ' no es válida.');
```

```
    }
    else
        alert('No hay datos');
```

```
" />

<input name="btn12" type="button" id="btn12" value="Modificar" onClick="
    if(total_datos())>0)
    {
        pos = prompt('Ingrese la posición del dato a modificar:', '');
        if(pos >= 0 && pos < total_datos())
```

```
        {
            nd = prompt('Ingrese el nuevo dato:', '');
            modificar(pos, nd);
            alert('El dato en posición ' + pos + ' fue modificado');
            mostrar();
        }
    else
        alert('La posición ' + pos + ' no es válida.');
```

```
    }
    else
        alert('No hay datos');
    " />
</p>
</form>
<br/>
<div id="divcontenido"></div>
</body>
</html>
```

### EJERCICIOS PROPUESTOS

A continuación se presentan ejercicios diversos para que desarrolle utilizando lo visto de DHTML. Aplique CSS para darle una apariencia agradable a las páginas, así como Javascript para realizar validaciones, cálculos y otras tareas para la gestión de la interfaz de usuario. Implemente clases para manipular la información de los formularios. Muchos de estos ejercicios también pueden ser resueltos con PHP o combinar ambas tecnologías.

1. Leer el nombre y estado civil de M personas. Encuentre cuántas son solteras, cuántas son casadas, cuántas son divorciadas, cuántas son separadas y cuántas viudas. Determine el porcentaje que representan las personas solteras y casadas.
2. A un grupo de hombres y mujeres les realizan una prueba de conocimientos. Se desea conocer cuántas mujeres y cuántos hombres presentaron la prueba y el total de personas que se procesaron.
3. En una fábrica contratan personal que cumpla con los siguientes requisitos para laborar medio tiempo: Mayor de 18 años y menor de 50, estado civil soltero y que actualmente se encuentre estudiando. A la fábrica se presentaron M aspirantes. Encuentre cuántos cumplen con los requisitos que se piden y que porcentaje sobre el total de personas representan.
4. Calcular la suma de los números pares e impares del 1 al 100 y comparar cual de las dos sumas es mayor.
5. Leer N números desde el teclado y determinar cuántos son positivos, cuántos negativos y cuántos son cero.
6. Calcule la suma de los primeros M términos de la serie  $1 + 1/2 + 1/3 + 1/4 + \dots + 1/M$  (esta es la serie Aritmética).
7. Leer el nombre y la nota de un grupo de T estudiantes. Encuentre quien sacó la mejor nota y de cuanto fue ésta.
8. Leer el nombre y salario de un grupo de trabajadores. Muestre cuál es el empleado de más bajo salario y a cuánto equivale éste.

9. Leer un número del 1 al 7 e imprimir el día al que corresponde.
10. Hacer un algoritmo que imprima las tablas de multiplicar
11. Implementar en una página un reloj mostrando las horas, minutos y segundos, así como el día y fecha actual.
12. Lea tres números e imprímalos en orden descendente. Resuélvalo utilizando conectivos lógicos y sin ellos.
13. Intercambie los valores de tres variables así: el valor de la primera debe quedar en la segunda; el valor de la segunda variable en la tercera, y el valor de la tercera variable en la primera.
14. Una empresa con N empleados desea realizar una estadística de su nómina. Para ello ingresa el nombre, número de horas trabajadas (nht), salario básico hora (sbh) y sexo del empleado (a). Se desean conocer los siguientes datos: Salario básico (sb), valor de la retención en salud (4%) y en pensión (3.5%) sobre el básico, total retención, salario neto (sn), empleado con mejor salario y a cuánto equivale su monto, empleado con más bajo salario y a cuánto equivale su monto, promedio de salarios, monto total de salarios pagados en la empresa, porcentaje que representan tanto los hombres como las mujeres en la empresa, porcentaje que representan las personas que ganan más de un millón de pesos, total de salarios por debajo de \$500000.
15. Leer un número y determinar si es o no, un número primo. Un número es primo cuando es entero y cuando solo tiene dos (2) divisores exactos: la unidad y el mismo número.
16. Hallar el factorial de un número N. El factorial de un número se define para números enteros positivos como:  $N! = N * (N-1) * (N-2) * \dots * 3 * 2 * 1$ , por ejemplo  $5! = 5 * 4 * 3 * 2 * 1 = 120$ . Por definición  $0! = 1$  y  $1! = 1$ .
17. Leer N números. Halle el promedio de los números pares y el promedio de los impares.
18. Determine las raíces de la ecuación cuadrática  $ax^2 + bx + c = 0$
19. Una empresa tiene 5 sucursales y en cada sucursal hay M empleados. De cada empleado se conoce su cédula y salario. Encuentre: El salario promedio de cada sucursal y de toda la empresa. El empleado que gana mayor salario en cada sucursal y en toda la empresa. El empleado que gana menor salario en cada sucursal y en toda la empresa. El monto pagado por concepto de salarios en cada sucursal y en toda la empresa
20. Leer el nombre y nota final de N estudiantes de un curso y encontrar los siguientes datos: Mejor y peor nota. Promedio general de notas. Cuántos ganan y cuantos pierde y el porcentaje que representan. Cuántos obtuvieron nota de 5. Determinar si algún estudiante es de nombre "Pedro José".
21. Imprimir los términos de la serie de Fibonacci menores o iguales a 10000. La serie de Fibonacci es: 1, 1, 2, 3, 5, 8, 13, 21, 34,...
22. Leer una cantidad indeterminada de números e informar si se ingresó ordenadamente.
23. Determinar los números perfectos entre 1 y 10000. Un número es perfecto si al sumar sus divisores, excepto el mismo número, da como resultado dicho número. Por ejemplo: 6 es perfecto, pues sus divisores (excepto el 6) son: 1, 2 y 3, los cuales suman 6
24. Se tiene el nombre, número de horas diurnas (nhd), número de horas nocturnas (nhn), número de horas festivas (nhf) y salario básico hora de un grupo de empleados. Calcular el salario básico teniendo en cuenta que la hora nocturna tiene un recargo del 35% y la hora festiva un recargo del 75%. Terminar la lectura de datos cuando se ingrese el nombre "\*\*\*\*". Informar cuántos empleados se ingresaron y el promedio de salarios básicos que devengan éstos.
25. Hacer un algoritmo que permita realizar el cálculo de una llamada. Se debe ingresar el tipo de llamada, el valor del minuto y el número de minutos que una persona habla por

- teléfono. Hay tres tipos de llamada: Local, que no tiene recargo; Nacional, que tiene un recargo del 3% e Internacional, que tiene un recargo del 5%.
26. Crear una factura, donde el usuario pueda agregar y quitar productos y especificar cuantos quiere de cada uno. Mostrar una interfaz agradable e informativa para el usuario.
  27. En el ejemplo del vector encuentre también: la moda, la mediana, la desviación estándar y la productoria. El sitio debe ofrecer la posibilidad de mostrar el vector ordenado (ordenar una copia, para conservar el orden original del vector).
  28. Dada una cadena, muestre en un renglón independiente cada carácter de ésta indicando al frente si es un número o una letra / símbolo. Cada carácter debe aparecer con un color diferente.
  29. Sume y multiplique dos matrices
  30. Sobre un texto que ingresa un usuario, ofrecer la posibilidad de buscar y reemplazar palabras o frases dentro de éste.
  31. Convertir un número positivo entre 0 y 10000 a letras (implementarlo en el ejercicio de la factura para indicar el costo de ésta).
  32. Convertir un número arábigo entre 1 y 10000 a un número romano.
  33. Cree un juego de azar similar al traga monedas que muestra tres o cuatro imágenes.
  34. Realizar el cálculo de una devuelta. Debe contar con denominaciones de monedas y billetes para especificar cuantos de cada uno debe entregar. Por ejemplo si la devuelta son \$450, podría devolver 2 monedas de \$200 y una de \$50 (implementarlo en el ejercicio de la factura).
  35. Crear el juego Trique (Triqui o Tres en Línea). Debe ofrecer la posibilidad de quien inicia el juego: el usuario o la máquina. La máquina nunca puede perder, debe ganar o conducir a empate.
  36. Mostrar el equivalente de un número decimal en binario, octal y hexadecimal.
  37. Muestre diferentes áreas en matrices cuadradas: diagonales (principal y secundaria), triangular inferior y superior, triángulos, etc.
  38. Se tienen las M asignaturas de N estudiantes almacenados en vectores. Generar las M\*N notas aleatoriamente y guardarlas en una matriz. Muestre un resumen indicando: Mejor y peor nota, promedio de notas; todo esto hacerlo por estudiante y asignatura.
  39. Se tiene un vector V en el plano cartesiano, cuyas componentes rectangulares son  $X_1(2,5)$  y  $X_2(-7,4)$ . Calcule la magnitud del vector  $V=(X_1^2 + X_2^2)^{1/2}$  y el ángulo  $\beta$  que forma, donde  $\tan(\beta)=X_2/X_1$
  40. Muestre y calcule las series de las funciones trigonométricas seno y coseno. Realice la aproximación mínimo con 100 datos y luego compare con las funciones correspondientes del lenguaje.

$$\text{sen } x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \dots \quad \cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} \dots$$

*Figura. Series de las funciones seno y coseno respectivamente*

## INTRODUCCIÓN AL LENGUAJE PHP

### CLASE No. 8



Figura 15. Logos del Lenguaje de programación PHP <sup>5</sup>

### ¿QUÉ ES PHP?

PHP es un lenguaje de programación interpretado y de scripting que permite la navegación dinámica de contenidos en un servidor web. PHP es un acrónimo recursivo que significa PHP HyperText Preprocessor. Entre sus principales características se pueden destacar:

- Potente y robusto lenguaje de programación embebido en documentos HTML.
- Alto rendimiento y fácil uso.
- Dispone de librerías de conexión con la mayoría de los sistemas de gestión de bases de datos para el almacenamiento de información permanente en el servidor.
- Gratuito: no es necesario realizar ningún desembolso económico para desarrollar sistemas de información empleando este versátil lenguaje.
- Portable y multiplataforma: existen versiones del intérprete para múltiples plataformas (Windows 95, 98, 2000, XP, Vista, NT, Unix, Linux, etc.). Esto permite que las aplicaciones puedan ser portadas de una plataforma a otra sin necesidad de modificar una sola línea de código.
- Eficiente: PHP consume muy pocos recursos en el servidor, por lo que con un equipo relativamente sencillo es posible desarrollar interesantes aplicaciones.

Actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica.

### RESEÑA HISTÓRICA

Inicialmente este lenguaje fue llamado *PHP Tools* o *Personal Home Page Tools*. Fue creado originalmente por Rasmus Lerdorf en 1994 y al igual que Javascript, Java y otros, su sintaxis es muy similar a la del lenguaje C. La implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

<sup>5</sup> Las empresas, fundaciones y organizaciones de Software Libre acostumbran utilizar animales en sus logos, por ejemplo PHP utiliza un elefante, Linux un pingüino y MySQL un delfín.

PHP es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo Web y puede ser embebido dentro de código HTML. Generalmente se ejecuta en un servidor Web, tomando el código en PHP como su entrada y creando páginas Web como salida. Puede ser desplegado en la mayoría de los servidores Web y en casi todos los sistemas operativos y plataformas sin costo alguno. PHP se encuentra instalado en millones de sitios Web, y su popularidad sigue en aumento.

PHP está permanentemente en desarrollo, y desde hace varios años, diferentes empresas se han dedicado a crear frameworks para el trabajo con este potente lenguaje.

PHP fue originalmente diseñado en Perl, seguidos por la escritura de un grupo de CGI binarios escritos en el lenguaje C por el programador danés-canadiense Rasmus Lerdorf en el año 1994 para mostrar su currículum vitae y guardar ciertos datos, como la cantidad de tráfico que su página web recibía. El 8 de junio de 1995 fue publicado "Personal Home Page Tools" después de que Lerdorf lo combinara con su propio *Form Interpreter* para crear PHP/FI.

PHP ha pasado por varias versiones, entre ellas, tenemos las siguientes.

### **Versiones de PHP**

#### **PHP 3**

Dos programadores israelíes del Technion, Zeev Suraski y Andi Gutmans, reescribieron el analizador sintáctico (*parser* en inglés) en el año 1997 y crearon la base del PHP3, cambiando el nombre del lenguaje a la forma actual. Inmediatamente comenzaron experimentaciones públicas de PHP3 y fue publicado oficialmente en junio del 1998.

Para 1999, Suraski y Gutmans reescribieron el código de PHP, produciendo lo que hoy se conoce como Zend Engine o motor Zend, un *portmanteau* de los nombres de ambos, Zeev y Andi. También fundaron Zend Technologies en Ramat Gan, Israel.

#### **PHP 4**

En mayo de 2000 PHP 4 fue lanzado bajo el poder del motor Zend Engine 1.0. El día 13 de julio de 2007 se anunció la suspensión del soporte y desarrollo de la versión 4 de PHP, a pesar de lo anunciado se ha liberado una nueva versión con mejoras de seguridad, la 4.4.8 publicada el 13 de Enero del 2008. Según esta noticia se dará soporte a fallos críticos hasta el 08 de agosto de 2008.

#### **PHP 5**

El 13 de julio de 2004, fue lanzado PHP 5, utilizando el motor Zend Engine 2.0 (o Zend Engine 2). La versión más reciente de PHP es la 5.4.3 (08 de mayo de 2012), que incluye nuevas ventajas que provee el nuevo Zend Engine 2, tales como como:

- Nuevo modelo de objetos que da mejor soporte para la Programación Orientada a Objetos, que en versiones anteriores era extremadamente rudimentario, con PHP Data Objects.
- Mejoras de rendimiento.
- Mejor soporte para MySQL con extensión completamente reescrita.
- Mejor soporte a XML (XPath, DOM, etc.).
- Soporte nativo para SQLite.
- Soporte integrado para SOAP.
- Iteradores de datos.



- Manejo de excepciones
- Mejoras con la implementación con Oracle.

### PHP 6

El desarrollo de la rama 6 de PHP ha sido retrasado porque los desarrolladores decidieron que el enfoque actual para el tratamiento de cadenas Unicode no es correcto, y están considerando formas alternas para la siguiente versión. Las mejoras que se tenían para la sexta versión fueron añadidas a las versiones 5.3.0 y 5.4.0, tales como soporte para espacios de nombre, enlace estático en tiempo de ejecución, funciones lambda, clausuras, goto, traits y revinculación de clausura. Hasta el momento se desconoce cuando será su lanzamiento.<sup>6</sup>

### Usos de PHP

Los principales usos de PHP son los siguientes:

- Programación de páginas Web dinámicas, habitualmente en combinación con el motor de base de datos MySQL, aunque cuenta con soporte nativo para otros motores, incluyendo el estándar ODBC, lo que amplía en gran medida sus posibilidades de conexión.
- Programación en consola, al estilo de Perl o Shell scripting.
- Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y Qt/GTK+, lo que permite desarrollar aplicaciones de escritorio en los sistemas operativos en los que está soportado.

### SERVIDOR WEB HTTP APACHE



Figura 16. Logos del Servidor Web HTTP Apache

El servidor HTTP Apache es un software (libre) de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf eligió ese nombre porque quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de Internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, *a patchy server* (un servidor "parcheado").

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation y se encuentra actualmente en la versión 2.2.14.

---

<sup>6</sup> Wikipedia, la enciclopedia libre. PHP. En Internet: <http://es.wikipedia.org/wiki/PHP>

Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios Web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años<sup>7</sup>.

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

## **BASES DE DATOS EN PHP**

### **MySQL**



Figura 17. Logos del Servidor de bases de datos MySQL

Es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero las empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es propiedad y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

### **phpMyAdmin**

---

<sup>7</sup> Estadísticas históricas y de uso diario proporcionadas por Netcraft (<http://news.netcraft.com/>)



Figura 18. Logo de phpMyAdmin

Es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas Web, utilizando un navegador web. Actualmente puede crear y eliminar bases de datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios y exportar datos en varios formatos entre otros aspectos, además, está disponible en 50 idiomas. Se encuentra disponible bajo la licencia GPL.

Este proyecto se encuentra vigente desde el año 1998, siendo el mejor evaluado en la comunidad de descargas de SourceForge.net como la descarga del mes de diciembre del 2002. Como esta herramienta corre en máquinas con servidores web y soporte de PHP y MySQL, la tecnología utilizada ha ido variando durante su desarrollo.

## INSTALACIÓN Y CONFIGURACIÓN DE XAMPP<sup>8</sup>



Figura 19. Logos del Servidor XAMPP

### XAMPP

XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor Web Apache y los interpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl.

La instalación de PHP requiere tener presente que este es un lenguaje interpretado del lado del servidor Web, por lo que se requiere tener uno instalado para publicar las páginas .php; además PHP es capaz de gestionar la información contenida en bases de datos, por lo que necesitamos también un servidor de tal tipo. Existen diferentes distribuciones de PHP, entre ellas, XAMPP<sup>9</sup>.

Puede instalar el paquete XAMPP, que incluye Apache, PHP y MySQL (además incorpora phpMyAdmin y otros módulos y aplicaciones, entre ellas, el servidor FTP FileZilla y el servidor

<sup>8</sup> El programa **XAMPP** está liberado bajo la licencia GNU y actúa como un servidor Web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris, y MacOS X.

<sup>9</sup> Otras distribuciones para PHP son: AppServ, Wamp y Easy PHP

SMTP Mercury Mail) y que configura estas herramientas automáticamente. Permanentemente se están publicando en el sitio oficial nuevas versiones de este servidor. Puede descargarse libre y gratuitamente de:

<http://www.apachefriends.org/en/xampp.html> <sup>10</sup>

### Instalación de XAMPP

Para realizar una configuración rápida de PHP y las demás herramientas, se recomienda descargar el archivo que aparece con el enlace *Installer o exe*, esto es, el archivo ejecutable .exe. Después de descargar el archivo, ejecute el instalador y en el momento en que el asistente de instalación le indique, especifique la carpeta donde se instalará XAMPP (en el caso de Windows, se generalmente se indica c:\xampp) y active todos los servicios (Apache, MySQL y FileZilla); siga los pasos necesarios hasta el paso Finalizar. Una vez terminado el proceso de instalación, tendrá acceso para trabajar con PHP y los demás componentes.

Una vez finalizada la instalación podrá a su vez tener acceso al Panel de Control de XAMPP, el cual se muestra en la siguiente figura:

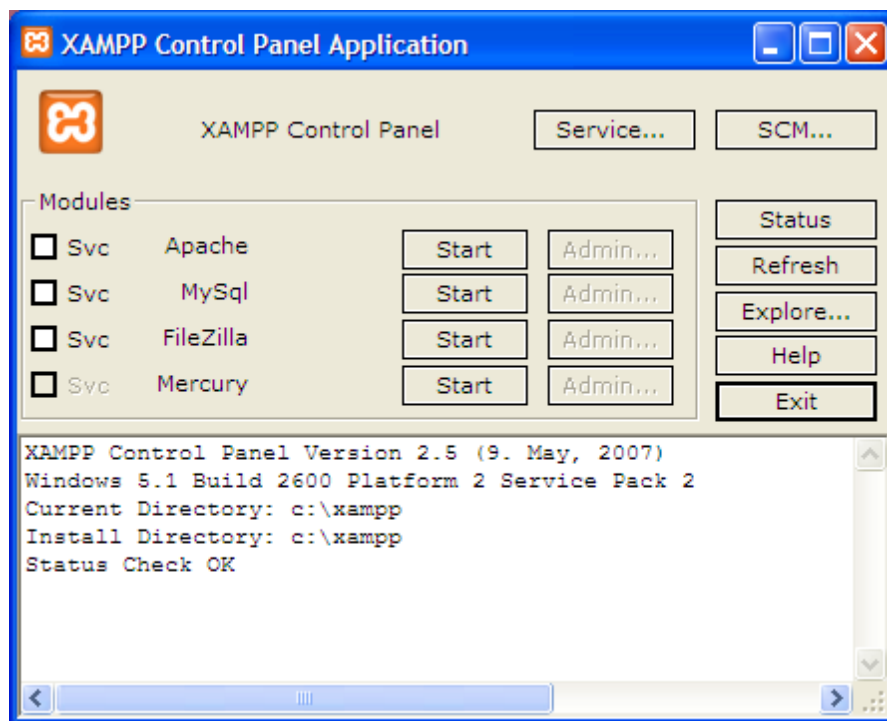


Figura 20. Panel de Control de XAMPP 1.6.8

<sup>10</sup> Al ingresar a la página encontrará varios archivos para la descarga: .zip, .tar.gz, y .exe. Para una instalación sencilla (en Windows) se recomienda el archivo ejecutable .exe (Installer) que configura las herramientas necesarias para trabajar con PHP de manera automática. Claro que si trabaja en una plataforma Linux, debe descargar el paquete comprimido.

Desde allí podrá realizar diferentes acciones entre ellas:

- Ver los servicios instalados con el botón *Service...*
- Acceder a los servicios de Windows con el botón *SCM...*
- Iniciar y/o detener los distintos módulos (servicios) con los botones *Start* y/o *Stop* respectivamente.
- Abrir aplicaciones de administración de los respectivos módulos con los botones *Admin*. (El botón Admin de Apache abre la página principal de *Localhost* en nuestro navegador predeterminado, para nuestro caso mostrará la página de nuestro XAMPP, verificando así que Apache funciona correctamente).
- Obtener el estado actual con el botón *Status*.
- Refrescar los servicios con el botón *Refresh*.
- Abrir la carpeta del servidor (*document root*) con el botón *Explore...*
- Obtener ayuda con el botón *Help*.
- Salir y cerrar la aplicación XAMPP Control Panel con el botón *Exit*.

#### **Notas**

- Si tiene otro servidor Web instalado en la máquina, Apache no podrá iniciarse. Para solucionar este inconveniente, existen varias opciones: cambiar los puertos, desactivar (desinstalar) el otro servidor, o detener el servicio correspondiente al otro servidor e iniciar Apache.
- Para abrir el Panel de Control de XAMPP debe verificar primero que éste no haya sido iniciado, lo cual puede comprobar en el área de notificación observando el icono respectivo. En caso de no haber sido iniciado y dependiendo de las opciones especificadas en la instalación, podrá acceder por el menú *Inicio / Programas / Apache Friends / XAMPP / XAMPP Control Panel*.

#### **PUBLICAR PÁGINAS EN EL SERVIDOR**

##### **¿Dónde se ejecuta el código de PHP?**

PHP se ejecuta en el servidor antes que la página sea enviada al usuario que realizó la petición. Esto incluye los siguientes pasos:

- El usuario pulsa sobre un enlace solicitando un documento
- Llega la solicitud al servidor y localiza el documento. Por la extensión del archivo se determina que es un archivo que contiene código PHP y lanza el intérprete.
- El intérprete lanza el script solicitado y genera un resultado (habitualmente una página HTML) que se devuelve al servidor para que éste a su vez lo transfiera al cliente.
- Se visualiza el documento en el navegador del usuario.

#### **Trabajo con PHP**

El código PHP se crea en un documento con extensión `.php` y puede escribirse en cualquier editor de texto (Dreamweaver, Texpad, Notepad++, Crimson, etc.). Los archivos que se vayan a publicar en el servidor deben guardarse en una carpeta específica, dependiendo del paquete instalado para PHP, esto incluye los archivos `.html`, imágenes, etc. Con la instalación de XAMPP, los archivos y carpetas deben guardarse en la siguiente ruta (que se considera la raíz del servidor o el Document Root):

`C:\xampp\htdocs\`

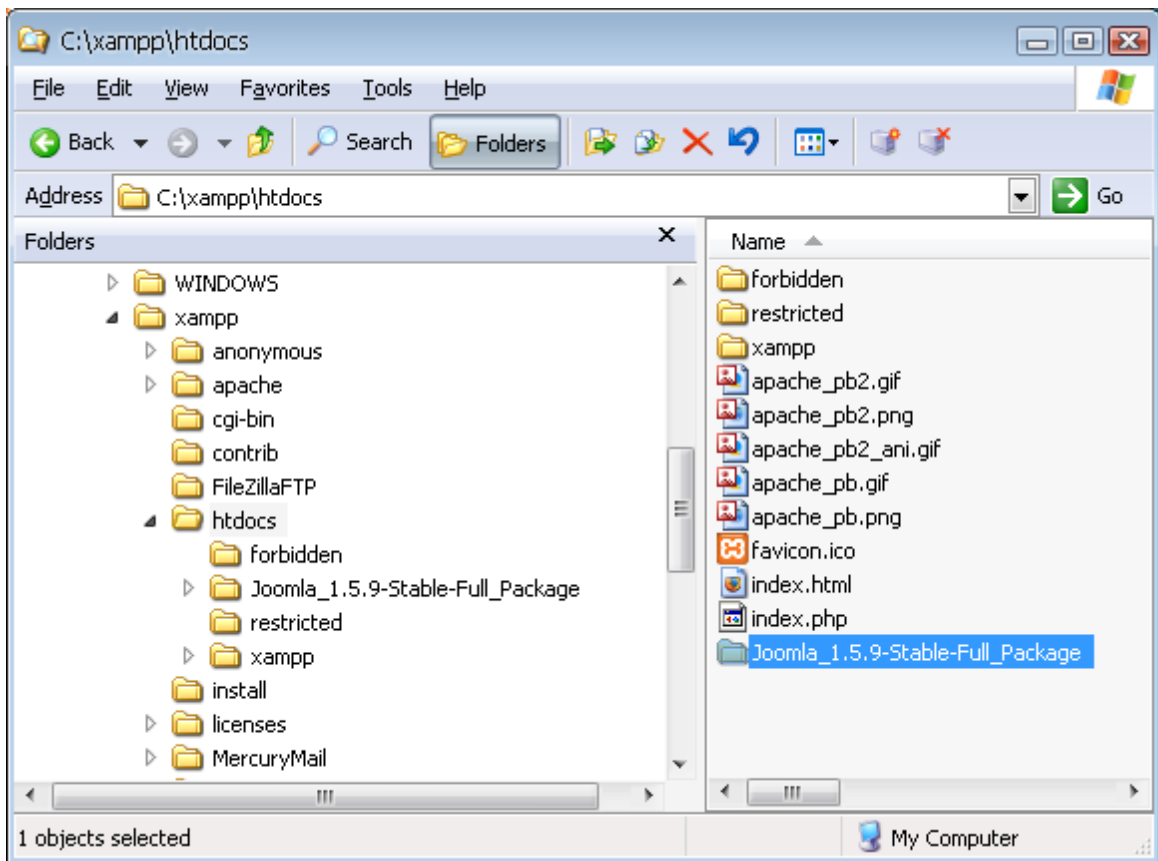


Figura 21. Subárbol de directorios de xampp/htdocs visualizado en el Explorador de Windows

Se recomienda crear carpetas y subcarpetas en la raíz del servidor. Así, por ejemplo para los que desean explorar administradores de contenidos (CMS) , y desean instalar un sitio *Joomla!* en su servidor, deben descomprimir el archivo descargado de la red dentro de alguna carpeta (como por ejemplo misitiojoomla) que se encuentre en la carpeta *htdocs*. Y luego, proseguir con los pasos de instalación; o también si desea crear su propio sitio PHP desde el principio, es muy importante crear una carpeta que identifique dicho sitio.

Publicar una página html en el servidor

Antes de publicar una página en el servidor Apache, hay que verificar que está funcionando, es decir, que se ha iniciado el servicio respectivo. Para esto podemos utilizar la opción indicada mas arriba con el botón Admin de Apache en el panel de control de XAMPP, o ingresar en la barra de direcciones del navegador la siguiente URL:

*<http://localhost/>*

Veremos la página principal de XAMPP, la cual es similar a la que muestra la siguiente figura:



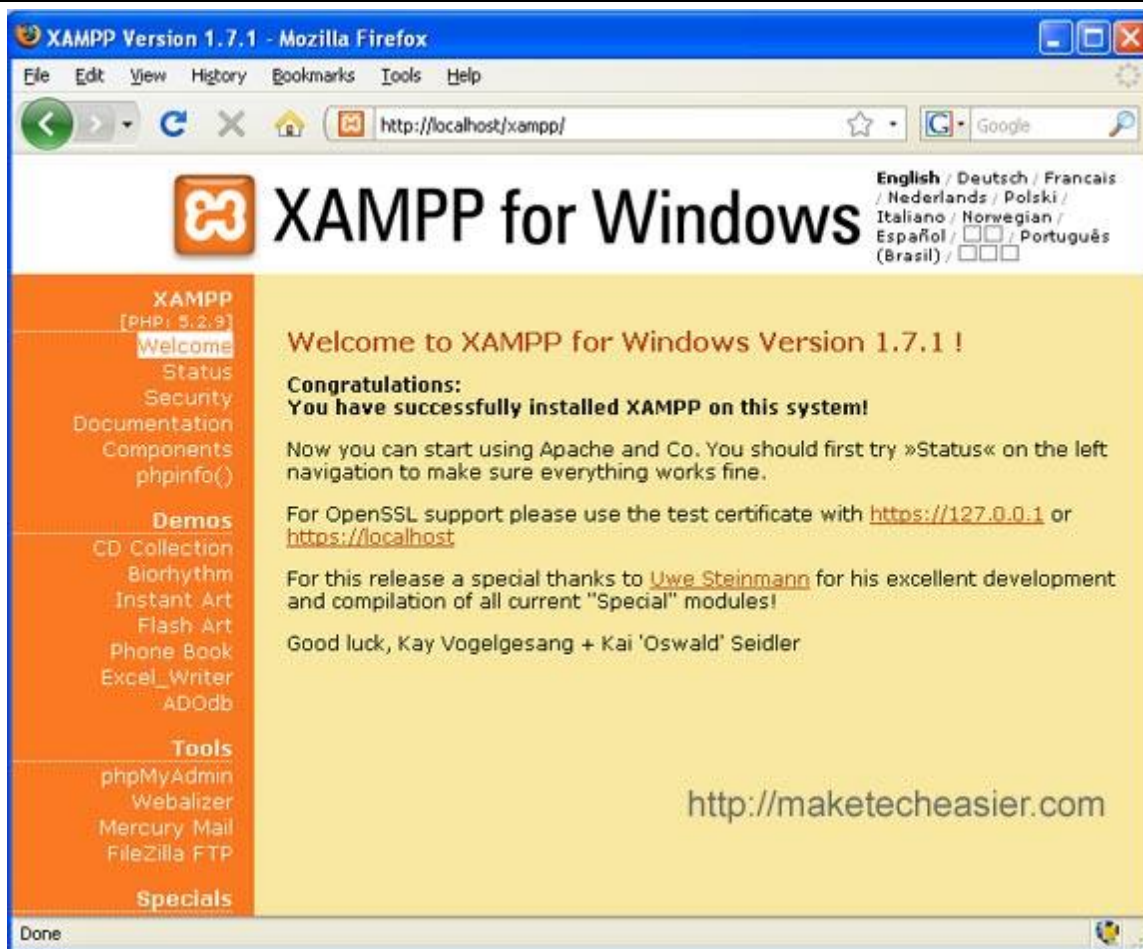


Figura 22. Página principal de Localhost de XAMPP 1.7.1, en inglés y visualizada en Firefox

Una vez haya verificado el estado de Apache y tenga el (los) archivo(s) en la carpeta del servidor Web puede digitar en la barra de direcciones del navegador algo similar a esto:  
*[http://localhost/\[nombre\\_carpeta\]/archivo.html](http://localhost/[nombre_carpeta]/archivo.html)*

## Ejemplos

1.  
<http://localhost/micarpeta/miarchivo1.html>
2.  
<http://localhost/archivo2.html>
3.  
<http://localhost/misitiojoomla/index.php>

## Ejemplo

El siguiente código pertenece a una página PHP que implementa algunas instrucciones del lenguaje. Este archivo debe guardarse con extensión .php. Más adelante se explican las diferentes instrucciones que aparecen en él.

```
<html>
<head>
<title></title>
</head>
<body bgcolor="#ffffff" text="#000000" >
<center>
<h3>ciclo while</h3>
</center>
<?
$i=1;
while($i<=10)
{
    if($i % 2==0)
        echo "$i es par <br>";
    else
        echo "$i es impar <br>";
    if ($i==5)
    {
        break;
        echo "Esto no se ejecuta";
    }
    i++;
}
echo "<br>Primera instrucción después del bucle";
?>
</body>
</html>
```



## DESARROLLO DE PÁGINAS WEB DINÁMICAS CON PHP

### CLASE No. 9

#### ESTRUCTURA GENERAL DE UNA PÁGINA PHP

Una página PHP es similar a una página HTML, con la diferencia de que la extensión del archivo es .php y que en algunas partes del código se puede encontrar la *etiqueta php* que especifica que en ese punto de la página se aplicará código php. A continuación se describe dicha etiqueta.

#### Etiqueta PHP

Utilizamos `<?...?>`, para abrir y cerrar el modo PHP, es decir, el código que esté entre este par de etiquetas se considera código PHP:

```
<? Abrir modo php  
?> cerrar modo php
```

También es común encontrar la siguiente forma de la etiqueta:

```
<?php Abrir modo php  
?> cerrar modo php
```

#### SALIDA DE INFORMACIÓN EN PHP

Las instrucciones `echo` y `print` se utilizan para imprimir algo en pantalla (cadenas, números, etc.), esto es, permiten enviar contenido al navegador Web.

#### Ejemplos

El código html se mezclará con el código php, en un archivo .php. Se muestran dos formas de enviar contenido al navegador con PHP (tenga presente que el código php está embebido dentro del html, pero se ejecuta en el servidor):

1.

```
<html>  
<head>  
<title>ejemplo</title>  
</head>  
<body>  
<? echo "Esto es PHP dentro de HTML"; ?>  
<? print "Esto es PHP dentro de HTML, otra forma de imprimir"; ?>  
</body>  
</html>
```

2.

```
<html>  
<head>  
<title>ejemplo</title>  
</head>  
<body>
```

```
<?
echo "<font color='red'><b>Esto es PHP</b></font><br>";
print "Esto es PHP dentro de HTML, otra forma de imprimir";
?>
</body>
</html>
```

### Notas

- Observe como se puede manipular HTML desde PHP en la instrucción *echo* (o *print*), siendo el servidor web el encargado de enviar el contenido generado a través de HTTP.
- Cada instrucción en PHP debe terminar en punto y coma (;).

### TIPOS DE DATOS EN PHP

Los tipos de cada variable en PHP no están tan claros como en el lenguaje C. El intérprete asigna el tipo de una variable según el uso que se esté haciendo de ella.

Para asignar un tipo fijo a una variable se utiliza la función `settype()`. Los tipos son: Enteros, Flotantes, String, Arrays, Objetos.

#### Enteros

Permite almacenar números sin decimales.

#### Flotantes

Este tipo de dato es utilizado para guardar números que pueden o no tener una parte decimal, tal como 3.141592, 2.8182, etc.

#### String

Las cadenas pueden estar delimitadas por comillas (") o apóstrofes ('). Si la cadena está delimitada por comillas dobles, cualquier variable incluida dentro de ella será sustituida por su valor. Para especificar el carácter comillas (") se escapará con el carácter backslash (\).

#### Booleanos

PHP permite trabajar con variables lógicas (booleanas) que admiten uno de dos valores: *true* y *false*, que a su vez son constantes del lenguaje.

#### Arrays

Los Arrays en PHP se pueden utilizar tanto como Arrays indexados o como Arrays asociativos. Los Arrays de una sola dirección, pueden ser tanto escalares como asociativos. En realidad no existen ninguna diferencia entre ellos. Las funciones que se utilizan para crear Arrays de este tipo son `list()` o `array()`.

En el caso de que no se especifique el índice en un array, el elemento que se asigna se añade al final.

#### Objetos

Para inicializar un objeto se utiliza el método *new*, y para acceder a cada uno de sus métodos se utiliza el operador *->*

## VARIABLES EN PHP

En PHP las variables se representan con un signo de pesos (\$) seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas y no debe tener caracteres especiales (a excepción del guión bajo \_ que es permitido). PHP no exige la declaración de variables.

### Ejemplo

```
<html>
<head>
<title>ejemplo</title>
</head>
<body>
<?
$variable1="Programación para la web: ";
$variable2='HTML, PHP Y MySQL';
echo $variable1 . " " . $variable2;
?>
</body>
</html>
```

### Notas

- Observe el uso del carácter punto (.) utilizado para concatenar cadenas.
- Las cadenas de caracteres se encierran entre comillas o apóstrofes.

## MANEJO DE COMENTARIOS

Un comentario es una instrucción que es ignorada por el intérprete a la hora de publicar la página; permiten hacer documentaciones o explicaciones de partes del código, o “tapar” algo que no queremos que se ejecute pero que no deseamos eliminar. En PHP los comentarios pueden realizarse de dos formas:

```
// Esto es un comentario.
/*
```

```
Esto es otro comentario. Puede utilizarse cuando el comentario ocupa varias líneas de código.
*/
```

## OPERADORES

PHP cuenta con los siguientes operadores:

### Operadores aritméticos

Utilizados para realizar cálculos matemáticos

Nombre Operador	Símbolo	Prioridad
Negación unaria	-	Alta
Multiplicación	*	Media
División	/	
Módulo	%	
Suma	+	Baja

Resta	-	
-------	---	--

Tabla 8. Operadores aritméticos en PHP

#### Notas

- La prioridad se refiere al orden en que los operadores se efectúan en una expresión aritmética: los de mayor prioridad se efectúan primero.
- Las operaciones encerradas entre paréntesis se efectúan primero, por lo que tienen mayor prioridad.

#### Operadores relacionales

Permiten realizar comparaciones entre dos expresiones, devolviendo un valor booleano (true o false); esto son:

Nombre Operador	Símbolo	Prioridad
Igual	==	Alta
Idéntico	=== Devuelve true si las dos expresiones son iguales y del mismo tipo de dato	
Diferente	!=	
Mayor que	>	Media
Menor que	<	
Mayor o igual que	>=	Baja
Menor o igual que	<=	

Tabla 9. Operadores relacionales o de comparación en PHP

#### Operadores lógicos o booleanos

Permiten conectar expresiones de comparación y realizar operaciones lógicas. El valor devuelto (true o false) depende del conectivo lógico utilizado, según las leyes del álgebra proposicional y booleana; estos son:

Nombre Operador	Símbolo		Prioridad
Negación	!		Alta
Conjunción	&&	and	Media
Disyunción		or	Baja

Disyunción exclusiva	<i>xor</i> (da como resultado <i>true</i> , si una de las expresiones es <i>true</i> , si ambas son verdaderas o ambas son falsas, devolverá <i>false</i> ).	
----------------------	--	--

Tabla 10. Operadores lógicos o booleanos en PHP

### Operadores aritméticos/asignación

PHP, al igual que Javascript y otros lenguajes, implementa varios operadores que abrevian la escritura normal de expresiones aritméticas, veamos en la siguiente tabla.

Nombre Operador	Símbolo	Ejemplo
Incremento en una unidad	++	Ejemplo: Escribir: a++ Equivale a escribir: a=a+1
Decremento en una unidad	--	Igual ocurre con el operador --
Acumulador de sumas	+=	Ejemplo: Escribir: s+=3
Acumulador de restas	-=	Equivale a escribir: s=s+3
Acumulador de productos	*=	De manera similar se comportan los operadores
“Acumulador” de divisiones	/=	-= *=
“Acumulador” de módulos	%=	/= %=
Inicialización (asignación simple)	=	x=0

Tabla 11. Operadores aritméticos/asignación en PHP

### ESTRUCTURAS DE CONTROL

En los lenguajes de programación, las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa. Con ellas se puede:

De acuerdo a una condición, ejecutar un grupo u otro de sentencias.

Ejecutar un grupo de sentencias un número determinado de veces  
Interrumpir la ejecución normal del programa

Todas las estructuras de control tienen un único punto de entrada y un único punto de salida. Las estructuras de control se puede clasificar en: decisión, iteración y de control avanzadas.

### Condicionales

Se utilizan para tomar decisiones a partir de valores booleanos obtenidos de la comparación de expresiones lógicas. Veamos como se implementan en PHP:

#### Condicional simple y condicional compuesto

Instrucción if - else

Evalúa una determinada condición o expresión de comparación, en caso de ser verdadera, se ejecuta un bloque de instrucciones. Si dicha condición no se cumple, esto es, es falsa, entonces ninguna de las instrucciones es ejecutada a no ser que se especifique la cláusula *else*.

Sintaxis

```
if (<expresión_de_comparación>)  
{  
    bloque de instrucciones php si expresión_de_comparación es verdadera;  
}  
[else  
{  
    bloque de instrucciones php si expresión_de_comparación es falsa;  
}]
```

#### Ejemplo

```
<html>  
<head>  
<title></title>  
</head>  
<body>  
<?  
$valor1=5;  
$valor2=7;  
  
if ($valor1>$valor2)  
{  
    $resultado = $valor1+$valor2;  
}  
else  
{  
    $resultado = $valor2-$valor1;  
}  
?>  
<table width="200" border="1">  
<tr>
```

```
<td><? echo "La operación dio como resultado <b>$resultado</b>"; ?></td>
</tr>
</table>
</body>
</html>
```

### Nota

Observe el uso de llaves { } para encerrar ciertas partes del código, éstas nos indican inicio y fin de una estructura o bloque. Cuando la estructura de control que requiere llaves solo contiene una instrucción, puede omitirse su uso.

### Selector múltiple o estructura caso

Permite la ejecución de un bloque de instrucciones en función del valor que tome una expresión. Esta es la Estructura Caso o el Selector Múltiple utilizado en lógica de programación como alternativa al condicional simple cuando se tienen más de dos salidas lógicas.

### Instrucción switch<sup>11</sup>

Sintaxis

```
switch (<expresión>)
{
    case valor1:
        Bloque de instrucciones valor 1;
        break;

    case valor 2:
        Bloque de instrucciones valor 2;
        break;

    ...
    case valor N:
        Bloque de instrucciones valorN;
        break;

    [default:
        Bloque de instrucciones por defecto;]
}
```

### Ejemplo

Este ejercicio está conformado por dos archivos. Uno guardado con extensión HTML que contendrá un formulario y el otro con extensión PHP donde se procesarán los datos del formulario. El ejemplo también ilustrará la forma de enviar datos por POST a una página PHP.

Este archivo debe guardarlo con extensión .html (formulario.html)

```
<html>
```

<sup>11</sup> La instrucción break utilizada en cada caso (case) se comenta en la subsección Bifurcación de control, que se encuentra mas adelante.

```
<head><title>Alternativas</title></head>
<body>
<h2>Introducción de datos</h2>
<form name="form1" id="form1" action="formulario.php" method="post" >
  <table border=1>
    <tr>
      <td>Nombre:</td>
      <td><input type="text" name="txtnom" id="txtom" size="10" maxlength="20" /></td>
    </tr>
    <tr>
      <td>Apellido:</td>
      <td><input type="text" name="txtape" id="txtape" size="10" maxlength="20" /></td>
    </tr>
    <tr>
      <td><input type="submit" value="Enviar" /></td>
      <td><input type="reset" value="Restablecer" /></td>
    </tr>
  </table>
</form>
</body>
</html>
```

### Nota

Observe que el método *action* llama al archivo formulario.php mediante el botón *Submit* y los datos del formulario son enviados a dicha página mediante el método *POST*. Desde PHP accedemos a dichos elementos utilizando el atributo *name* de cada objeto creado, tal y como se muestra a continuación.

Ahora escriba el siguiente código y guárdelo con el nombre de formulario.php

```
<html>
<head><title>Alternativas</title></head>
<body>
<?php
switch ($_POST["txtnom"])
{
  case "JORGE":
    $fecha = date("d-m-y");
    echo "hola " . $_POST["txtnom"] . " hoy es $fecha";
    break;

  case "ANA":
    echo "hola " . $_POST["txtnom"] . " bienvenida<br />";
    echo "Tu apellido es " . $_POST["txtape"] . "<br />";
    break;

  case "PEDRO":
    echo "Hola " . $_POST["txtnom"] . " ya era hora";
    break;
}
```



```
default:
    echo "No estás autorizado " . $_POST["txtnom"] . " " . $_POST["txtape"];
}

?>
</body>
</html>
```

### Notas

- Si un usuario escribe en el formulario HTML uno de los nombres: JORGE, ANA o PEDRO, aparecerá un mensaje de bienvenida acompañado del nombre que suministró, en caso contrario aparecerá un texto que dice "No estás autorizado".
- Observe la manera como se accede a los objetos del formulario: \$\_POST['nombre\_objeto'].

### Ciclos

Los ciclos o bucles son estructuras de control que repiten un grupo de instrucciones mientras se cumpla una condición. PHP cuenta con tres tipos de ciclos.

#### Ciclo while

Permite la repetición de un bloque de instrucciones un determinado número de veces de acuerdo a una condición: si ésta es verdadera, las instrucciones del ciclo se repiten, en caso contrario finaliza la ejecución de éste y continúa con la siguiente instrucción después del ciclo. Es posible que las sentencias del bucle no se lleguen a ejecutar nunca, ya que antes de proceder a interpretar la primera instrucción se evalúa la condición, y si ésta resulta ser falsa, no entrará en las instrucciones del bloque.

#### Sintaxis

```
while (<condición>)
{
    Bloque de instrucciones;
}
```

#### Ejemplo

```
<html>
<head>
<title> Ciclos </title>
</head>
<body>
<h3>Ciclo while</h3>
<?
$i=1;
while($i<=10)
{
    if($i % 2==0)
        echo "$i es par <br />";
```

```
else
    echo "$i es impar <br />";

    i++;
}
echo "<br />Primera instrucción después del bucle";
?>
</body>
</html>
```

### Ciclo do while

Es similar al ciclo while, con la diferencia de que la condición se evalúa al final del ciclo, garantizando que las instrucciones dentro de él se ejecutarán por lo menos una vez.

Sintaxis

**do**

```
{
    Bloque de instrucciones;
} while (<condición>);
```

### Ejemplo

```
<html>
<head>
<title> Ciclos </title>
</head>
<body>
<h3>Ciclo do while</h3>
<?
$i=1;
do
{
    if($i % 2==0)
        echo "$i es par <br>";
    else
        echo "$i es impar <br>";
    i++;
} while($i<=10);
echo "<br />El valor de i es $i";
?>
</body>
</html>
```

### Ciclo for

Tiene como fin repetir un bloque de instrucciones mientras cumpla una condición preestablecida. Se deben indicar tres parámetros: La condición que determina si se debe seguir ejecutando o no el bucle (condición), una condición que vaya haciendo cambiar algún parámetro que varíe el

cumplimiento de la condición anterior (actualización) y por supuesto, una expresión que determine cuál es la situación de partida en el cumplimiento de dicha condición (inicialización).

Sintaxis

```
for (<inicialización>; <condición>; <actualización>)  
{  
    Bloque de instrucciones;  
}
```

## Ejemplos

1. Imprimir varias líneas aplicando formato.

```
<html>  
<head>  
<title> Ciclos </title>  
</head>  
<body>  
<h3>Ciclo for</h3>  
  
<?php  
    for ($i = 1; $i <= 10; $i++)  
    {  
        print "Línea número: <b>$i</b> <br />";  
    }  
?>  
</body>  
</html>
```

2. Este ejemplo crea una tabla dinámicamente. Cuando hablamos de tabla dinámica, significa que se crea en tiempo de ejecución y no en tiempo de diseño.

```
<html>  
<head>  
<title> Ciclos </title>  
</head>  
<body>  
<table border="1">  
    <tr>  
        <th>Número </th>  
        <th>Cuadrado </th>  
    </tr>  
<?php  
    for ($i = 1; $i <= 100; $i++)  
    {  
        $c=$i * $i;  
        print "<tr>  
            <td>$i </td>
```

```
        <td>$c </td>
    </tr>";
}
?>
</table>
</body>
</html>
```

### BIFURCACIÓN DE CONTROL

La Bifurcación de control está conformada por un grupo de instrucciones que permite hacer saltos sobre ciertas partes del código de un programa, interrumpiendo su flujo normal. PHP maneja las mismas sentencias que utiliza lenguaje C, a excepción de la instrucción *goto*; estas son:

#### Instrucción **exit**

Interrumpe el Script que se esté ejecutando. Cancela la ejecución del código de la página. Puede utilizarse así: `exit()`, o sin los paréntesis: `exit`.

#### Ejemplo

```
<html>
<head>
<title>Bifurcación de control</title>
</head>
<body>
<font color="#ff0000"><b>
<?
echo "Esto es PHP";
exit();
?>
</b></font><br>
<?php
print "Esto es PHP (este mensaje no aparece)";
?>
</body>
</html>
```

#### Instrucción **break**

Permite romper o finalizar la estructura de control donde se encuentre pasando a la instrucción que le sigue. En particular, si la instrucción se encuentra en un ciclo, lo finaliza sin importar la condición de control y continúa con la instrucción siguiente al ciclo. (El ejemplo de esta instrucción se aplicó en la subsección de la sentencia *switch*. Allí es necesario utilizarla, ya que cuando se evalúa un caso, si no se encuentra la sentencia *break*, inmediatamente continúa con las instrucciones del siguiente caso y así sucesivamente).

#### Instrucción **continue**

Utilizada en los ciclos, hace que salte el resto de instrucciones de este pasando a la siguiente iteración.

#### Ejemplo

```
<?php
$i = 0;
while ($i <= 10)
{
    $i++;
    if ($i == 4)
        continue;
    print "$i <br />";
}
?>
```

### **Instrucción return**

Devuelve el control del programa a una función o porción de programa que llama; a su vez puede devolver un valor a dicha función. En la subsección de funciones definidas por el usuario se ilustrará el uso de esta instrucción.

### **PASO DE PARÁMETROS POR URL**

En internet es muy común el paso de parámetros por URL (Uniform Resource Locator). Su sintaxis es:

*[http://nombre\\_pagina.php?parametro1=valor1&parametro2=valor2&...&parametroN=valorN](http://nombre_pagina.php?parametro1=valor1&parametro2=valor2&...&parametroN=valorN)*

Con esto logramos pasar valores de variables de una página a otra.

### **Ejemplos**

<http://localhost/hojavidaphp?cedula=123>

<http://www.misitio.com/paginas/hojavidaphp?ced=123&nom=Pedro>

En el primer caso quiere decir que cargará una página llamada `hojavidaphp` y que cuando esta se ejecute tendrá una variable en memoria llamada `cedula` con un valor 123. En el segundo, se pasan dos parámetros por la URL: `ced` y `nom`, con los valores 123 y Pedro respectivamente; observe que varios parámetros se separan con un ampersand (&)

### **Nota**

Las variables pasadas por URL se envían por (el método) GET, a diferencia de los datos pasados por formulario que se envían por (el método) POST.

### **Ejemplo**

El siguiente ejemplo nos crea una tabla con dos columnas y con el número de filas que se especifique como parámetro en la URL. Guarde el archivo con el nombre de `parametros.php`.

```
<html>
<head>
<title>Parámetros URL</title>
```

```
</head>
<body>
<h1 align="center">Creación de una tabla a partir de dos parámetros</h1>
<table width="200" border="1">
  <tr>
    <th>Número</th>
    <th>Número * Dato</th>
  </tr>
  <?php
  if(!isset($_GET['valor']))
  {
    echo "No escribió ningún valor!!!";
    exit;
  }
  for ($i = 1; $i <= $_GET['valor']; $i++)
  {
  ?>
    <tr>
      <td><? print $i; ?> </td>
      <td><? print $i * $_GET['dato']; ?></td>
    </tr>
  <?
  }
  ?>
</table>
</body>
</html>
```

Ejecute este ejemplo escribiendo en la barra del navegador:

<http://localhost/micarpeta/parametros.php?valor=25&dato=15>

Ejecútelo varias veces cambiando los valores de los parámetros y observando el resultado en el navegador.

### **Nota (Acercas de los atributos Action y Method de los formularios)**

#### **Atributo Action:**

Indica el programa que se encargará de tratar los datos del formulario. Este programa debe encontrarse en el servidor y estar escrito en algún lenguaje de programación. A este programa se le pasarán como parámetros los datos introducidos en el formulario y retornará un código HTML que se mostrará tras procesar el formulario. A este tipo de programas se les llama cgi-bin.

#### **Atributo Method:**

Indica el protocolo usado para el envío de los datos. Con POST envía los datos en la entrada estándar del programa que trata el formulario y con GET los datos se pasan por parámetro, en la línea de comandos, al programa. El usar uno u otro método vendrá determinado por cómo son tratados los parámetros en el formulario en el script PHP (CGI-BIN). El método de uso será de acuerdo a las necesidades.

## **FUNCIONES PARA VALIDAR DATOS EN PHP**

PHP cuenta con dos funciones para validar datos. Estas son `isset()` y `empty()`.

### **Función `isset()`**

En php la función `isset()` nos permite determinar si una variable concreta tiene ya un valor asignado o no. Esto es especialmente útil en procesos de depuración de scripts. Durante la corrección de los mismos se puede comprobar si hay contenido en una variable para rastrear posibles errores. La función `isset()` recibe como argumento el nombre de una variable y devuelve un valor de tipo booleano, es decir `true` (verdadero) si la variable ha sido definida o `false` (falso) en caso contrario.

### **Función `empty()`**

Tiene un efecto similar a `isset()`, pero con el resultado opuesto, es decir devuelve un valor booleano igual a `true` si la variable no ha recibido todavía ningún valor; en caso contrario devuelve el resultado `false`.

## **MANIPULACIÓN DE CADENAS DE CARACTERES**

Dado el uso del lenguaje PHP, el tratamiento de cadenas es muy importante. Existen bastantes funciones para el manejo de cadenas, a continuación se explican algunas.

### **`strlen(cadena)`**

Nos devuelve el número de caracteres de una cadena.

### **`split(separador,cadena)`**

Divide una cadena en varias usando un carácter separador.

### **`sprintf(cadena de formato, var1, var2...)`**

Formatea una cadena de texto al igual que `printf` pero el resultado es devuelto como una cadena.

### **`substr(cadena, inicio, longitud)`**

Devuelve una subcadena de otra, empezando por *inicio* y de longitud *longitud*.

### **`chop(cadena)`**

Elimina los saltos de línea y los espacios finales de una cadena.

### **`strpos(cadena1, cadena2)`**

Busca la *cadena2* dentro de *cadena1* indicándonos la posición en la que se encuentra.

### **`str_replace(cadena1, cadena2, texto)`**

Reemplaza *cadena1* por *cadena2* en *texto*.

### **Concatenación de cadenas: *cadena1 . cadena2 . ... . cadenaN***

Une, junta o pega *cadena1* con *cadena2*, *cadena3*, etc., hasta *cadenaN*. Para concatenar utilizamos el operador punto (`.`)

### **`strtolower`, `strtoupper`, `ucwords` (*cadena*)**

Convierte todos los caracteres a minúsculas o a mayúsculas. En el último caso sólo pone mayúsculas la primera letra de cada palabra.

### **trim, ltrim, rtrim (cadena)**

Devuelve una cadena con los espacios eliminados del principio y final de la misma (trim), sólo del principio (ltrim) o sólo del final (rtrim).

### **strstr, strstr, strstr (cadena1, cadena2)**

Devuelven todo el contenido de *cadena1* desde que encuentran por primera vez *cadena2* hasta el final. Los dos primeros empiezan a buscar por el principio, pero el segundo busca sin diferenciar entre mayúsculas o minúsculas. En el tercer caso, la búsqueda comienza por el final.

#### **Ejemplo**

```
$s = ucwords ("hola pepe");  
//Resultado: $s = "Hola Pepe";
```

#### **Ejemplo**

```
$s = strstr ("hola pepe", "e");  
//Resultado: $s = "epe";
```

#### **Ejemplo**

```
<html>  
<head>  
  <title>Ejemplo de cadenas en PHP</title>  
</head>  
<body>  
<?php  
  echo "Longitud de 12345 es " . strlen("12345") . "<br/>";  
  $palabras=split(" ", "Esto es una prueba");  
  echo "<br/>palabras es: $palabras <br/>";  
  for($i=0; $palabras[$i]; $i++)  
    echo $palabras[$i] . "<br/>";  
  
  $resultado=sprintf("8x5 = %d <br>",8*5);  
  echo $resultado. "<br>";  
  echo substr("Devuelve una subcadena de otra",9,3) . "<br /><br />";  
  if (chop("Cadena \n\n ") == "Cadena")  
    echo "Iguales<br/><br/>";  
  
  echo strpos("Busca la palabra dentro de la frase", "palabra") . "<br /><br />";  
  echo str_replace("verde","rojo","Un pez de color verde, como verde es la hierba.") . "<br/>";  
?>  
</body>  
</html>
```

## **MANIPULACIÓN DE FECHAS Y HORAS**

PHP cuenta a su vez con una serie de funciones para manipular fechas y horas.

### **Funciones para el manejo de fechas y horas**



**checkdate**

Valida una fecha u hora.

**date**

Da formato a la fecha/hora local.

**getdate**

Obtiene información de fecha y hora.

**gettimeofday**

Obtiene la hora actual.

**gmdate**

Da formato a una fecha/hora GMT/CUT.

**gmmktime**

Obtiene el valor timestamp UNIX de una fecha GMT.

**gmstrftime**

Da formato a una fecha/hora GMT/CUT según las convenciones locales.

**localtime**

Obtiene la hora local.

**microtime**

Devuelve el valor timestamp UNIX actual con microsegundos.

**mktime**

Obtiene el timestamp UNIX de una fecha.

**strftime**

Da formato a la hora o fecha local de acuerdo con las convenciones locales.

**strtotime**

Procesa cualquier descripción textual de fecha/hora en inglés convirtiéndola en una timestamp de UNIX.

**time**

Devuelve el timestamp UNIX actual, que es el valor por defecto, obteniendo la hora local.

**Configurar la fecha y hora local**

Al tratar con la fecha y hora, es posible que nos aparezca una que no corresponde a nuestro huso horario, por lo que será necesario configurar la hora local en nuestro servidor. Para ello utilizamos la siguiente instrucción antes de utilizar y mostrar fechas y horas en nuestras páginas.

```
date_default_timezone_set("Continente/Ciudad");
```

**Ejemplo**

```
date_default_timezone_set("América/Bogotá");
```

### Formatos de fecha y hora de la función date

La función date da formato a una hora o fecha local; sigue esta sintaxis de definición:

*string date (string \$formato [, int \$marca\_de\_tiempo ] )*

Devuelve una cadena con formato de acuerdo a la cadena de formato dada usando el entero *marca\_de\_tiempo* entregado o la hora actual si no se da una marca de tiempo. En otras palabras, *marca\_de\_tiempo* es opcional y su valor predeterminado es el valor de *time()*. Permite aplicar diversos formatos para obtener fechas y horas, se muestran en la siguiente tabla.

Caracteres que son reconocidos en la cadena del parámetro <i>formato</i>		
Caracter de <i>formato</i>	Descripción	Valores de ejemplo devueltos
<i>Día</i>	---	---
<i>d</i>	Día del mes, 2 dígitos con ceros iniciales	01 a 31
<i>D</i>	Una representación textual de un día, tres letras	Mon a Sun
<i>j</i>	Día del mes sin ceros iniciales	1 a 31
<i>l</i> ('L' minúscula)	Una representación textual completa del día de la semana	Sunday a Saturday
<i>N</i>	Representación numérica ISO-8601 del día de la semana (agregado in PHP 5.1.0)	1 (para Lunes) a 7 (para Domingo)
<i>S</i>	Sufijo ordinal en inglés del día del mes, 2 caracteres	st, nd, rd o th. Funciona bien con <i>j</i>
<i>w</i>	Representación numérica del día de la semana	0 (para el Domingo) a 6 (para el Sábado)
<i>z</i>	El día del año (comenzando en 0)	0 a 365
<i>Semana</i>	---	---
<i>W</i>	Número de la semana del año ISO-8601, las semanas comienzan en Lunes (agregado en PHP 4.1.0)	Ejemplo: 42 (la 42va semana del año)
<i>Mes</i>	---	---
<i>F</i>	Una representación textual completa de un mes, como January o March	January a December
<i>m</i>	Representación numérica de un mes, con ceros iniciales	01 a 12
<i>M</i>	Una representación textual corta de un mes, tres letras	Jan a Dec
<i>n</i>	Representación numérica de un mes, sin ceros iniciales	1 a 12
<i>t</i>	Número de días en el mes dado	28 a 31
<i>Año</i>	---	---
<i>L</i>	Indica si es un año bisiesto	1 si es un año bisiesto, 0 de lo contrario.
<i>o</i>	Número de año ISO-8601. Este es el mismo valor que Y, excepto que si el número de semana ISO ( <i>W</i> ) pertenece al	Ejemplos: 1999 o 2003

**ESCUELA DE INFORMÁTICA**  
**TÉCNICO EN DESARROLLO DE SOFTWARE**  
**SUBMÓDULO PROGRAMACIÓN PARA LA WEB I**



	año previo o siguiente, ese año será usado en su lugar. (agregado en PHP 5.1.0)	
<i>Y</i>	Una representación numérica completa de un año, 4 dígitos	Ejemplos: 1999 o 2003
<i>y</i>	Una representación de dos dígitos de un año	Ejemplos: 99 o 03
<i>Hora</i>	---	---
<i>a</i>	Ante meridiano y Post meridiano en minúsculas	<i>am</i> o <i>pm</i>
<i>A</i>	Ante meridiano y Post meridiano en mayúsculas	<i>AM</i> o <i>PM</i>
<i>B</i>	Hora Swatch Internet	000 a 999
<i>g</i>	formato de 12-horas de una hora sin ceros iniciales	1 a 12
<i>G</i>	formato de 24-horas de una hora sin ceros iniciales	0 a 23
<i>h</i>	formato de 12-horas de una hora con ceros iniciales	01 a 12
<i>H</i>	formato de 24-horas de una hora con ceros iniciales	00 a 23
<i>i</i>	Minutos con ceros iniciales	00 a 59
<i>s</i>	Segundos, con ceros iniciales	00 a 59
<i>u</i>	Milisegundos (agregado en PHP 5.2.2)	Ejemplo: 54321
<i>Zona horaria</i>	---	---
<i>e</i>	Identificador de zona horaria (agregado en PHP 5.1.0)	Ejemplos: <i>UTC</i> , <i>GMT</i> , <i>Atlantic/Azores</i>
<i>l</i> (i mayúscula)	Indica si la fecha están en hora de ahorro de luz diurna	1 si es Hora de Ahorro de Luz Diurna, 0 de lo contrario.
<i>O</i>	Diferencia con la hora Greenwich (GMT) en horas	Ejemplo: +0200
<i>P</i>	Diferencia con la hora Greenwich (GMT) con dos-puntos entre las horas y los minutos (agregada en PHP 5.1.3)	Ejemplo: +02:00
<i>T</i>	Abreviación de zona horaria	Ejemplos: <i>EST</i> , <i>MDT</i> ...
<i>Z</i>	Desplazamiento de la zona horaria en segundos. El desplazamiento para zonas horarias al oeste de UTC es siempre negativo, y el de aquellas al este de UTC es siempre positivo.	-43200 a 50400
<i>Fecha/Hora Completa</i>	---	---
<i>c</i>	Fecha ISO 8601 (agregada en PHP 5)	2004-02-12T15:19:21+00:00
<i>r</i>	Fecha en formato » RFC 2822	Ejemplo: <i>Thu, 21 Dec 2000 16:01:07 +0200</i>
<i>U</i>	Segundos desde el Epoch Unix (January 1 1970 00:00:00 GMT)	Vea también time()

Tabla 12. Caracteres que son reconocidos en la cadena del parámetro formato

### Nota

Los caracteres no reconocidos en la cadena de formato serán impresos como son. El formato Z siempre devuelve 0 cuando se usa gmdate().

### Ejemplo

Se ilustrará a continuación el uso de la función date con diferentes formatos pasados como parámetro.

```
<?php
// definir la zona horaria predeterminada a usar. Disponible desde PHP 5.1
date_default_timezone_set('América/Bogotá');

// Imprime algo como: Monday
echo date("l");

// Imprime algo como: Monday 15th of August 2005 03:12:46 PM
echo date('l dS \of F Y h:i:s A');

// Imprime: July 1, 2000 is on a Saturday
echo "July 1, 2000 is on a " . date("l", mktime(0, 0, 0, 7, 1, 2000));

/* usar las constantes en el parámetro formato */
// imprime algo como: Mon, 15 Aug 2005 15:12:46 UTC
echo date(DATE_RFC822);

// imprime algo como: 2000-07-01T00:00:00+00:00
echo date(DATE_ATOM, mktime(0, 0, 0, 7, 1, 2000));
?>
```

### ARREGLOS

Se define a un arreglo como un grupo de elementos relacionados entre sí por medio de índices. Los arreglos pueden ser de una o más dimensiones, los de una dimensión, son llamados comúnmente *vectores* y los de dos dimensiones se conocen como *tablas* o *matrices*.

A diferencia con el lenguaje C, en PHP, un vector puede tener elementos de distintos tipos. Para hacer referencia a un elemento del vector, se utiliza un índice, que indica la dirección en donde se encuentra un determinado valor. El índice en un arreglo comienza generalmente en cero, sin embargo el índice de un vector no necesariamente debe ser un número entero, sino que también puede ser un texto.

Algunas funciones para la manipulación de arreglos se indican a continuación:

**array** -- Crear una matriz

**arsort** -- Ordena una matriz en orden inverso y mantiene la asociación de índices

**asort** -- Ordena una matriz y mantiene la asociación de índices

**compact** -- Crea una matriz que contiene variables y sus valores

**count** -- Cuenta los elementos de una variable

**current** -- Devuelve el elemento actual de una matriz

**each** -- Devuelve el siguiente par clave/valor de una matriz  
**end** -- Mueve el puntero interno de una tabla al último elemento  
**extract** -- Importa variables a la tabla de símbolos desde una matriz  
**in\_array** -- Devuelve TRUE si un valor está en una matriz  
**key** -- Obtiene una clave de una matriz asociativa  
**krsort** -- Ordena una matriz por clave en orden inverso  
**ksort** -- Ordena una matriz por clave  
**list** -- Asigna variables como si fueran una matriz  
**next** -- Avanza el puntero interno de una matriz  
**pos** -- Obtiene el elemento actual de una matriz  
**prev** -- Rebobina el puntero interno de una matriz  
**rango** -- Crea una matriz que contiene un rango de enteros  
**reset** -- Fija el puntero interno de una matriz a su primer elemento  
**rsort** -- Ordena una matriz en orden inverso  
**shuffle** -- Mezcla una matriz  
**sizeof** -- Obtiene el número de elementos de una matriz  
**sort** -- Ordena una matriz  
**uasort** -- Ordena una matriz mediante una función de comparación definida por el usuario y mantiene la asociación de índices  
**uksort** -- Ordena una matriz por claves mediante una función definida por el usuario  
**usort** -- Ordena una matriz por valores mediante una función definida por el usuario

## Ejemplos

### 1. Vector con datos homogéneos

```
<html>
<head>
<title>Arreglos</title>
</head>
<body>
<?
    for($i=0; $i<10; $i++)
    {
        $vec[$i]=$i+1;
    }

    $s=0;
    for($i=0; $i<10; $i++)
    {
        $s=$s+$vec[$i];
    }
    echo "la suma es: $s";
    echo "<br>Total elementos del vector:" . count($vec);
?>
</body>
</html>
```

### 2. Vector con datos heterogéneos

```
<html>
<head><title> Ejemplo 2 </title></head>
<body>
<?php // Inicializacion del Vector
$Empleado[0] = 4371;
$Empleado[1] = "Pedro Martínez";
$Empleado[2] = "27.643.742";
$Empleado[3] = 1429000.54;
$Empleado[4] = "Arquitecto";
// Impresion del vector
echo ("Código: " . $Empleado[0] . "<br/>");
echo ("Nombre: " . $Empleado[1] . "<br/>");
echo ("Cédula : " . $Empleado[2] . "<br/>");
echo ("Sueldo: " . $Empleado[3] . "<br/>");
echo ("Profesion: " . $Empleado[4] . "<br/>");
?>
</body>
</html>
```

### 3. Matrices

```
<html>
<head><title>Matrices</title>
</head>
<body>
<?php
$mat = array();
for($i=0;$i<10;$i++)
{
    for($j=0;$j<10;$j++)
    {
        $mat[$i][$j]=$i*$j;
    }
}

echo "<center><table border='1'><caption align='top'><b>Elementos de la matriz</b></caption>";

for($i=0;$i<10;$i++)
{
    echo "<tr>";
    for($j=0;$j<10;$j++)
    {
        echo "<td width='30'>" . $mat[$i][$j] . "</td>";
    }
    echo "</tr>";
}

echo "</table></center>";
```

```
?>  
</body>  
</html>
```

## CLASE No. 10

### FUNCIONES MATEMÁTICAS

PHP cuenta con un número significativo de funciones para el tratamiento de expresiones matemáticas complejas, estas se enuncian a continuación.

#### Funciones para cálculos matemáticos en PHP

Estas son algunas funciones disponibles en PHP para realizar cálculos matemáticos.

**abs** — Valor absoluto  
**acos** — Arco coseno  
**acosh** — Arco coseno hiperbólico  
**asin** — Arco seno  
**asinh** — Arco seno hiperbólico  
**atan2** — Arco tangente de dos variables  
**atan** — Arco tangente  
**atanh** — Arco tangente hiperbólica  
**base\_convert** — Convertir un número entre bases arbitrarias  
**bindec** — Binario a decimal  
**ceil** — Redondear fracciones hacia arriba  
**cos** — Coseno  
**cosh** — Coseno hiperbólico  
**decbin** — Decimal a binario  
**dechex** — Decimal a hexadecimal  
**decoct** — Decimal a octal  
**deg2rad** — Convierte el número en grados a su equivalente en radianes  
**exp** — Calcula la exponencial de e  
**expm1** — Devuelve  $\exp(\text{numero})-1$ , calculado de tal forma que no pierde precisión incluso cuando el valor del numero se aproxima a cero.  
**floor** — Redondear fracciones hacia abajo  
**fmod** — Devuelve el residuo de punto flotante (módulo) de la división de los argumentos  
**getrandmax** — Mostrar el mayor valor aleatorio posible  
**hexdec** — Hexadecimal a decimal  
**hypot** — Calcula la longitud de la hipotenusa de un triángulo de ángulo recto  
**is\_finite** — Encuentra si un valor es un número finito legal  
**is\_infinite** — Encuentra si un valor es infinito  
**is\_nan** — Encuentra si un valor no es un número  
**lcg\_value** — Generador lineal congruente combinado  
**log10** — Logaritmo en base 10  
**log1p** — Devuelve  $\log(1 + \text{numero})$ , calculado de tal forma que no pierde precisión incluso cuando el valor del numero se aproxima a cero.

**log** — Logaritmo natural  
**max** — Encontrar el valor más alto  
**min** — Encontrar el valor más bajo  
**mt\_getrandmax** — Mostrar el mayor valor aleatorio posible  
**mt\_rand** — Genera un mejor número entero aleatorio  
**mt\_srand** — Genera un mejor número entero aleatorio a partir de una semilla  
**octdec** — Octal a decimal  
**pi** — Obtener valor de pi  
**pow** — Expresión exponencial  
**rad2deg** — Convierte el número en radianes a su equivalente en grados  
**rand** — Genera un número entero aleatorio  
**round** — Redondea un float  
**sin** — Seno  
**sinh** — Seno hiperbólico  
**sqrt** — Raíz cuadrada  
**srand** — Genera un número entero aleatorio a partir de una semilla  
**tan** — Tangente  
**tanh** — Tangente hiperbólica

**Ejemplo:**

Ilustración de algunas funciones matemáticas con el lenguaje PHP.

**funciones\_matematicas.php**

```
<html>
<head>
<title>Funciones matemáticas</title>
</head>
<body>
<h2>Funciones matemáticas</h2>
<?php
    $x=3.141592;
    $r=sin($x);
    print ("sen($x)=$r<br/>");
    $r=cos($x);
    print ("cos($x)=$r<br/>");
    $r=floor($x);
    print ("Redondeo por debajo: floor($x)=$r<br/>");
    $r=ceil($x);
    print ("Redondeo por encima: ceil($x)=$r<br/>");

?>
</body>
</html>
```

**Números aleatorios**

**Números aleatorios. Generación de números aleatorios**

Un número aleatorio es un número generado al azar de un grupo, todos con la misma probabilidad de ser seleccionados; un ejemplo es la lotería donde todos los que compran el billete tienen la



misma probabilidad de ganar (o perder) y dichos números son generados aleatoriamente, es decir, al azar. PHP dispone de una serie de funciones para generar números aleatorios. La forma más básica de generar un número aleatorio en PHP consiste en dos pasos:

```
//alimentamos el generador de aleatorios  
srand (time());  
//generamos un número aleatorio  
$numero_aleatorio = rand(1,100);
```

Como vemos, en el primer paso se utiliza la función `srand()` para alimentar la semilla de generación de números aleatorios. Este paso es necesario sólo en versiones anteriores a PHP 4.2.0, pues a partir de esta versión este paso se hace automáticamente. A la función `srand()` hace falta enviarle un valor para alimentar la semilla. Nosotros enviamos lo que devuelve `time()`, que es un timestamp con el número de segundos desde el inicio de 1970.

Luego generamos un número aleatorio con la función `rand()` que recibe un par de valores opcionalmente, que son el mínimo y el máximo de los números aleatorios generados. En el caso anterior se consigue un número aleatorio entre 1 y 100, incluyendo estos dos valores entre los posibles.

Si no se indica nada a `rand()`, el valor mínimo será cero. El valor máximo depende de la plataforma donde se esté ejecutando PHP, por ejemplo en Windows el valor máximo sería 32786. Si queremos asegurarnos que este valor máximo sea mayor, entonces conviene definir los valores máximo y mínimo al llamar a la función. Pueden generarse números aleatorios decimales usando una división:  $\$r=100/\text{rand}(1,100)$ , en este caso solo habrían 100 opciones; si quiere aumentar el número de posibles valores, se puede aumentar el valor del denominador  $\$r=1000/\text{rand}(1,1000)$ ;

Generación de números aleatorios con `mt_rand()`

PHP tiene otras funciones para generar números aleatorios, aparte de las mencionadas, que utilizan unos algoritmos mejorados para conseguir números al azar.

La función de PHP `mt_rand()` genera aleatorios con un algoritmo que es en promedio 4 veces más rápido que el algoritmo que utiliza `rand()`.

El uso de `mt_rand()` es similar:

```
//alimentamos el generador de aleatorios  
mt_srand (time());  
//generamos un número aleatorio  
$numero_aleatorio = mt_rand(0,5);
```

Primero se debe utilizar `mt_srand()` para empezar la generación de números aleatorios con una semilla. Pero este paso a partir de PHP 4.2.0 no es necesario, porque se hace automáticamente.

Luego se generan los números aleatorios con `mt_rand()`, a la que le pasamos el rango de valores que queremos obtener, con los parámetros mínimo y máximo. En nuestro ejemplo obtendremos valores aleatorios entre el 0 y el 5.

**Ejemplo:**

Este ejemplo se compone de dos páginas: la primera, aleatorios.html contiene un formulario donde se pide que especifique el total de elementos para un vector, el cual guardará números aleatorios entre 1 y 50 generados en la segunda página, aleatorios.php. La página html contiene una validación con Javascript para que el usuario ingrese datos numéricos.

#### **aleatorios.html**

```
<html>
<head>
<title>Números aleatorios</title>
<script language="javascript" type="text/javascript">
function validar()
{
    if (!isNaN(document.getElementById("te").value))
        return (true);
    else
        return(false);
}
</script>
</head>
<body>
<h2>introducción de datos</h2>
<form name="form1" id="form1" action=" aleatorios.php" method="post" onsubmit="return
validar()">
<table border="1">
<tr>
<td width="127">Total elementos:</td>
<td width="77"><input name="te" type="text" id="te" size="10" maxlength="4"></td>
</tr>
<tr>
<td colspan="2">
<input type="submit" value="Generar aleatorios" >
<input type="reset" value="Restablecer" >
</td>
</tr>
</table>
</form>
</body>
</html>
```

#### **aleatorios.php**

```
<html>
<head>
<title>Arreglos y números aleatorios</title>
</head>
<body>
<table border="1">
<tr>
<td>
```

```
srand (time()); //generamos la semilla

for($i=0; $i<$_POST['te']; $i++)
{
    $vec[$i]=rand(1,50);//generamos el número aleatorio
?>
    <td><? print $vec[$i]; ?></td>
<?
}
?>
</tr>
</table>
</body>
</html>
```

#### Nota

Observe el manejador de evento *onsubmit* en la etiqueta de formulario: Si la función Javascript devuelve verdadero (true), los datos se envían a la página PHP especificada en el atributo action al presionar el botón *submit* del formulario, de lo contrario éstos no se enviarán.

#### FUNCIONES DEFINIDAS POR EL USUARIO

Al desarrollar aplicaciones nos vemos en la necesidad de utilizar la misma porción de código en varias ocasiones; para optimizar la programación y evitar tener que repetir segmentos de código, los lenguajes implementan los subprogramas, secuencias de código a las que podemos invocar pasándoles de 0 a N parámetros y que tras ejecutar una serie de operaciones, nos pueden devolver 0, 1 o mas valores. Normalmente, cuando devuelven un valor se les llama *funciones*, y en los demás casos, *procedimientos*. PHP trata todos los subprogramas como funciones y con algunos ajustes pueden simularse procedimientos, incluso para que nos devuelvan mas de un valor.

En PHP, como en cualquier otro lenguaje de alto nivel, existen muchas funciones ya implementadas, pero en esta subsección vamos a ver como crear funciones e implementarlas en los sitios web.

Para definir funciones, utilizamos la palabra reservada *function*, a continuación el nombre de la función y entre paréntesis los parámetros que recibirá. En el caso de que devuelva la función algún valor, se utiliza la sentencia *return* seguida de dicho valor.

#### Declaración de prototipos de función en PHP

El cuerpo de una función en PHP es como se muestra en la siguiente sintaxis:

```
function <nombre_función>([Lista_de_argumentos])
{
    Cuerpo de la función;
    ...

    [return <valor_de_retorno>;]
}
```

### Nota

Como buena técnica, se recomienda definir el prototipo de las funciones en las primeras líneas de código, antes de que estas sean invocadas o escribirlas en archivos independientes (.php) que luego puede incluir en las páginas requeridas con una de las siguientes instrucciones:

```
include("nombre_archivo_a_incluir.php");  
require("nombre_archivo_a_incluir.php");
```

Así, puede crear librerías propias que contengan diversas funciones (puede tener varias funciones en un mismo archivo, pero es conveniente que estén agrupadas por temáticas).

### Pasar parámetros por valor o por referencia

Por defecto, los parámetros de una función se pasan por valor, de manera que, al cambiar el valor de un parámetro dentro de la función, no se ve modificado fuera de ella. Para permitir que dichos cambios se vean reflejados fuera de la función, hay que pasar los parámetros por referencia.

Para conseguir que un parámetro de una función siempre se pase por referencia, hay que anteponer un ampersand (&) al nombre del parámetro en la definición de la función. En los ejemplos se ilustrará como hacer esto.

Ejemplos

#### 1. Funciones implementando cadenas.

```
<html>  
<head><title>Funciones</title></head>  
<body>  
<?  
function escribe_separa($cadena)  
{  
    for ($i=0;$i<strlen($cadena);$i++)  
    {  
        echo $cadena[$i];  
        if ($i<strlen($cadena)-1)  
            echo "-";  
    }  
}  
escribe_separa ("hola");  
echo "<p>";  
escribe_separa ("Texto más largo, a ver lo que hace");  
echo "</p>";  
>  
</body>  
</html>
```

#### 2. Sumar dos números utilizando una función

```
<html>  
<head>  
    <title>Funciones</title>
```

```
</head>
<body>

<?
function sumar_numeros($a, $b) //$a y $b son parámetros formales
{
    $s=$a+$b;
    return $s;
}

$suma=sumar_numeros (5,8); //5 y 8 son parámetros actuales
echo "<p>La suma es $suma</p>";
?>
</body>
</html>
```

3. Este ejemplo utiliza dos archivos .php: el primero es la página que invoca una función que se encuentra en el segundo archivo.

Llame esta primera página: ejemplo\_funciones\_inclusion\_archivos.php

```
<!-- ejemplo_funciones_inclusion_archivos.php-->
<html>
<head>
    <title>Funciones</title>
</head>
<body>
include("ejemplo_funciones_archivo_externo.php");
$suma=sumar_numeros (5,8);
echo "<p>La suma es $suma</p>";
?>
</body>
</html>
```

Llame esta segunda página: ejemplo\_funciones\_archivo\_externo.php

```
<?
function sumar_numeros($a, $b)
{
    $s=$a+$b;
    return $s;
}
?>
```

**Nota**

Observe que una vez incluido el archivo donde se encuentran las funciones, podemos entonces invocarlas en la página. Observe también que el archivo donde están las funciones están limpios de HTML, pues dicho código se encuentra en la página que hace el llamado a la función.

4. Paso de parámetros por valor y por referencia.

```
<html>
<head>
  <title>Funciones</title>
</head>

<body>

<?
function sumar_numeros($a, &$b) //$a y $b son parámetros formales
{
  $s = $a+$b;
  $a = 10;
  $b = -9;
  return $s;
}

$x=7;
$y=5
$suma=sumar_numeros ($x,$y); //$x e y son parámetros actuales
echo "<p>La suma es $suma</p>";
echo "<p>Valor de x: $x</p>";
echo "<p>Valor de y: $y</p>";
?>
</body>
</html>
```

En este ejemplo, la variable \$x se pasa por valor, mientras que la variable \$y se pasa por referencia.

**BASES DE DATOS EN PHP. MYSQL**

**CLASE No. 11**



Figura 23. Caricatura de un archivador con un ratón (mouse) conectado representando una base de datos

**CONCEPTOS GENERALES SOBRE BASES DE DATOS**

Una base de datos es un “almacén” que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrarla y utilizarla fácilmente.

El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California, USA. Una **base de datos** se puede definir como un conjunto de información relacionada que se encuentra agrupada ó estructurada.

Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Cada base de datos se compone de una o más tablas (entidades) que guardan un conjunto de datos. Cada tabla tiene una o más **columnas** (campos o atributos) y **filas** (**registros** o **tuplas**) conformadas por un conjunto de campos. Los campos guardan una parte de la información sobre cada elemento que queramos guardar o procesar en la tabla, cada fila de la tabla conforma un registro, que representa el elemento a guardar o a procesar.

**SISTEMAS DE GESTIÓN DE BASE DE DATOS (SGBD)**

Antes del surgimiento de las bases de datos, la información de las empresas se procesaba en archivos independientes, diseminados a lo largo de toda la organización, originando problemas de redundancia, inconsistencias, dispersión, inseguridad e incorrección. Veamos en que consisten estos problemas:

- Redundancia: Los datos dispersos en múltiples archivos.
- Inconsistencia: Diversas copias de los mismos datos no concuerdan entre si.
- Dispersión: La información se encuentra esparcida en diversos lugares físicos. No existe un control centralizado de la información.
- Inseguridad: Usuarios no autorizados pueden tener acceso a los datos.
- Incorrección: Los datos deben ser íntegros, es decir deben concordar con la realidad.

Para dar solución a dichos problemas, surgen los Sistemas de Gestión de Bases de Datos (SGBD). Los Sistemas de Gestión de Base de Datos (en inglés DBMS: DataBase Management System) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Algunos de los objetivos que cumple un DBMS son:

- Disminuir redundancia
- Evitar inconsistencias
- Evitar la dispersión (acceso concurrente)
- Aumentar la seguridad
- Mantener la integridad de los datos
- Proporcionar visión abstracta de los datos

### **LENGUAJE ESTRUCTURADO DE CONSULTA SQL**

SQL (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional de tuplas, permitiendo lanzar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre la misma. Existen básicamente tres tipos de consultas en SQL: Consultas de selección, funciones de agregación y las consultas de acción.

#### **Consultas de selección**

Permiten extraer (filtrar) información de las tablas de acuerdo a unos criterios especificados. La sintaxis general es:

```
select <lista_de_campos>  
from <lista_de_tablas>  
[where <expresión_de_comparación>]
```

#### **Funciones de agregación**

Permiten sacar resúmenes de los datos contenidos en las tablas; estas son:

##### **sum(campo)**

Suma todos los valores que cumplan una determinada condición.

##### **avg(campo)**

Devuelve el promedio de todos los valores que cumplan una determinada condición.

##### **count(campo)**

Cuenta todos los datos que cumplan con el criterio especificado.

##### **max(campo)**

Devuelve el mayor dato que cumpla una determinada condición.

##### **min(campo)**

Devuelve el menor dato que cumpla una determinada condición.



### **Consultas de acción**

Actúan sobre la base de datos, permitiendo alterar la información así como la estructura misma de la base de datos. Algunas sentencias para ejecutar consultas de acción son: *insert into*, *delete from*, *update set*, *create*, *alter*, etc. Algunas de estas instrucciones serán ilustradas en los ejemplos.

### **MYSQL**

Como se comentó anteriormente, MySQL es un potente motor que trabaja como sistema de gestión de bases de datos relacionales, muy común en ambientes web: actualmente (13 de Febrero de 2013) se encuentra en la versión 5.6.10. Para gestionar las bases de datos MySQL, pueden utilizarse diferentes aplicaciones, muchas de ellas de descarga gratuita de la red; una de las mas conocidas es *PHPMyAdmin* (mencionada más arriba), una aplicación web creada en PHP, con una interfaz muy intuitiva que facilita bastante la tarea de los diseñadores de bases de datos; otra herramienta interesante es *MySQL Front*, una aplicación de escritorio que ofrece un entorno de desarrollo integrado (IDE) agradable y fácil de manejar; también se pueden manipular bases de datos de MySQL desde la consola, por ejemplo en Windows entramos a la siguiente carpeta:

```
c:\xampp\mysql\bin>
```

### **Ejemplo**

Estando sobre dicha carpeta, ingresamos la siguiente orden:

```
c:\xampp\mysql\bin>mysql -h localhost -u root -p
```

A continuación especificamos el password o contraseña de MySQL. El prompt del sistema se mostrará así:

```
mysql>
```

La siguiente figura muestra también el estado de editor de línea de comandos (shell) cuando nos conectamos a MySQL:

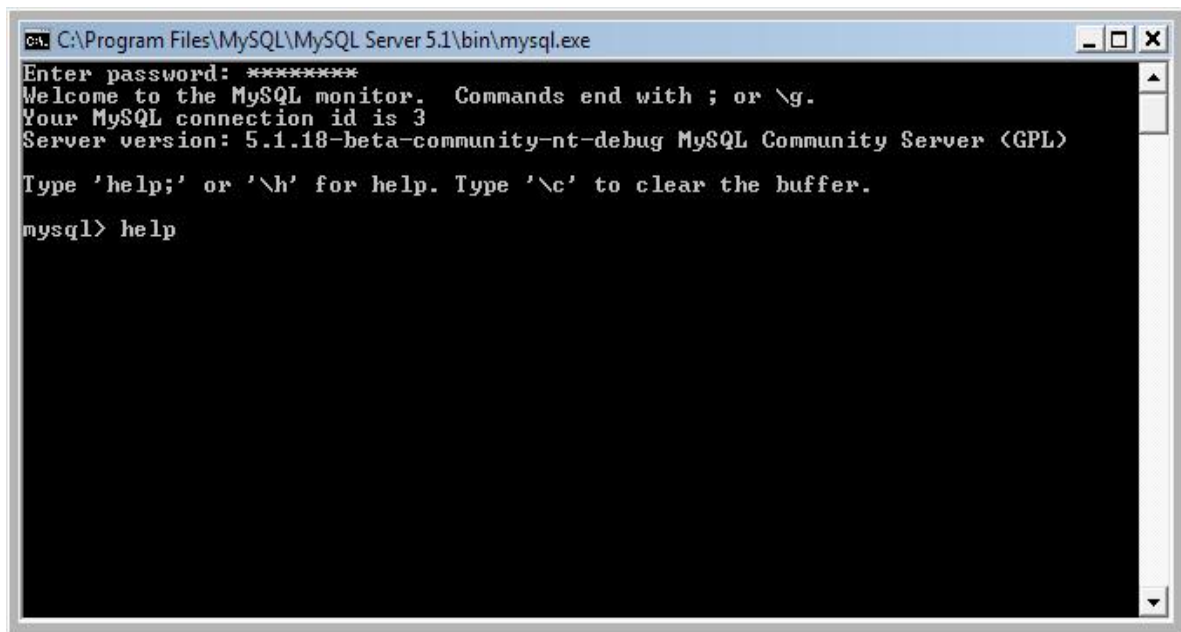


Figura 24. Editor de línea de comandos (shell, DOS) de Windows. El indicador del sistema (prompt) muestra que se está ejecutando MySQL

Ahora podemos acceder a nuestras bases de datos desde la consola.

### Ejemplo

Veamos algunas órdenes que pueden ingresarse desde la consola:

1. Mostrar las bases de datos que hay en MySQL

```
mysql>show databases;
```

2. Seleccionar una base de datos

```
mysql>use ejemplo;
```

3. Mostrar las tablas de una base de datos

```
mysql>show tables;
```

4. Ejecutar una consulta

```
mysql>select * from tabla1;
```

5. Ver la estructura de una tabla

```
mysql>describe tabla1;
```

6. Cerrar la conexión con el servidor MySQL

```
mysql>quit;
```

### Nota

Las bases de datos MySQL se almacenan en la carpeta: *c:\xampp\mysql\data* para el caso de XAMPP bajo Windows, y generalmente la ruta *mysql\data* es la misma en otras distribuciones.

### **Tipos de datos en MySQL**

Después de la fase de diseño de una base de datos, y una vez se ha realizado el paso a tablas del mismo, es necesario crear las entidades correspondientes dentro de la base de datos. Para cada campo de cada una de las tablas, es necesario determinar el tipo de datos que contiene, para de esa forma ajustar el diseño de la base de datos, y conseguir un almacenamiento óptimo con la menor utilización de espacio. A continuación se describe cada uno de los tipos de datos que puede tener un campo en Mysql.

Los tipos de datos que puede haber en un campo, se pueden agrupar en tres grandes grupos:

- Tipos numéricos
- Tipos de Fecha y Hora
- Tipos de Cadena

#### **Tipos numéricos**

Existen tipos de datos numéricos, que se pueden dividir en dos grandes grupos, los que están en coma flotante (con decimales) y los que no.

**TinyInt:** Es un número entero con o sin signo. Con signo el rango de valores válidos va desde -128 a 127. Sin signo, el rango de valores es de 0 a 255.

**Bit ó Bool:** Un número entero que puede ser 0 ó 1.

**SmallInt:** Número entero con o sin signo. Con signo el rango de valores va desde -32768 a 32767. Sin signo, el rango de valores es de 0 a 65535.

**MediumInt:** Número entero con o sin signo. Con signo el rango de valores va desde -8.388.608 a 8.388.607. Sin signo el rango va desde 0 a 16777215.

**Integer, Int:** Número entero con o sin signo. Con signo el rango de valores va desde -2147483648 a 2147483647. Sin signo el rango va desde 0 a 429.4967.295.

**BigInt:** Número entero con o sin signo. Con signo el rango de valores va desde -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807. Sin signo el rango va desde 0 a 18.446.744.073.709.551.615.

**Float:** Número pequeño en coma flotante de precisión simple. Los valores válidos van desde -3.402823466E+38 a -1.175494351E-38, 0 y desde 1.175494351E-38 a 3.402823466E+38.

**xReal, Double:** Número en coma flotante de precisión doble. Los valores permitidos van desde -1.7976931348623157E+308 a -2.2250738585072014E-308, 0 y desde 2.2250738585072014E-308 a 1.7976931348623157E+308.

**Decimal, Dec, Numeric:** Número en coma flotante desempaquetado. El número se almacena como una cadena.

Tipo de Campo	Tamaño de Almacenamiento
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT	4 bytes
INTEGER	4 bytes
BIGINT	8 bytes
FLOAT(X)	4 ú 8 bytes
FLOAT	4 bytes
DOUBLE	8 bytes
DOUBLE PRECISION	8 bytes
REAL	8 bytes
DECIMAL(M,D)	M+2 bytes sí D > 0, M+1 bytes sí D = 0
NUMERIC(M,D)	M+2 bytes if D > 0, M+1 bytes if D = 0

Tabla 13. Tipos de datos numéricos en MySQL y su tamaño de almacenamiento

### Tipos de fecha y hora

A la hora de almacenar fechas, hay que tener en cuenta que MySQL no comprueba de una manera estricta si una fecha es válida o no. Simplemente comprueba que el mes esté comprendido entre 0 y 12 y que el día esté comprendido entre 0 y 31.

**Date:** Tipo fecha, almacena una fecha. El rango de valores va desde el 1 de enero del 1001 al 31 de diciembre de 9999. El formato de almacenamiento es de año-mes-día (aaaa/mm/dd).

**DateTime:** Combinación de fecha y hora. El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. El formato de almacenamiento es de año-mes-día horas:minutos:segundos (aaaa/mm/dd hh:mm:ss).

**TimeStamp:** Combinación de fecha y hora. El rango va desde el 1 de enero de 1970 al año 2037. El formato de almacenamiento depende del tamaño del campo:

Tamaño	Formato
14	AñoMesDiaHoraMinutoSegundo aaaammddhhmmss
12	AñoMesDiaHoraMinutoSegundo aammddhhmmss
8	ñoMesDia aaaammdd
6	AñoMesDia aammdd

4	AñoMes aamm
2	Año aa

Tabla 14. Tamaños y formatos de fecha en MySQL para el tipo timestamp

**Time:** Almacena una hora. El rango de horas va desde -838 horas, 59 minutos y 59 segundos a 838, 59 minutos y 59 segundos. El formato de almacenamiento es de 'HH:MM:SS'.

**Year:** Almacena un año. El rango de valores permitidos va desde el año 1901 al año 2155. El campo puede tener tamaño dos o tamaño 4 dependiendo de si queremos almacenar el año con dos o cuatro dígitos.

Tipo de Campo	Tamaño de Almacenamiento
DATE	3 bytes
DATETIME	8 bytes
TIMESTAMP	4 bytes
TIME	3 bytes
YEAR	1 byte

Tabla 15. Tipos de datos de fecha y hora en MySQL

### Tipos de cadena

Es importante conocer cual es el tipo de dato apropiado a utilizar cuando se tratan cadenas para aprovechar al máximo los recursos y definir adecuadamente cada campo.

**Char(n):** Almacena una cadena de longitud fija. La cadena podrá contener desde 0 a 255 caracteres.

**VarChar(n):** almacena una cadena de longitud variable. La cadena podrá contener desde 0 a 255 caracteres.

Dentro de los tipos de cadena se pueden distinguir otros dos subtipos, los tipo Text y los tipo BLOB (Binary large Object).

La diferencia entre un tipo y otro es el tratamiento que reciben a la hora de realizar ordenamientos y comparaciones. Mientras que el tipo text se ordena sin tener en cuenta las mayúsculas y las minúsculas, el tipo BLOB se ordena teniéndolas en cuenta.

Los tipos BLOB se utilizan para almacenar datos binarios como pueden ser ficheros.

**TinyText y TinyBlob:** Columna con una longitud máxima de 255 caracteres.

**Blob y Text:** Un texto con un máximo de 65535 caracteres.

**MediumBlob y MediumText:** Un texto con un máximo de 16.777.215 caracteres.

**LongBlob y LongText:** Un texto con un máximo de caracteres 4.294.967.295. Hay que tener en cuenta que debido a los protocolos de comunicación los paquetes pueden tener un máximo de 16 Mb.

**Enum:** Campo que puede tener un único valor de una lista que se especifica. El tipo Enum acepta hasta 65535 valores distintos.

**Set:** Un campo que puede contener ninguno, uno o varios valores de una lista. La lista puede tener un máximo de 64 valores.

Tipo de campo	Tamaño de Almacenamiento
CHAR(n)	n bytes
VARCHAR(n)	n +1 bytes
TINYBLOB, TINYTEXT	Longitud+1 bytes
BLOB, TEXT	Longitud +2 bytes
MEDIUMBLOB, MEDIUMTEXT	Longitud +3 bytes
LOB, LONGTEXT	Longitud +4 bytes
ENUM('value1','value2',...)	1 ó dos bytes dependiendo del número de valores
SET('value1','value2',...)	1, 2, 3, 4 ó 8 bytes, dependiendo del número de valores

Tabla 16. Tipos de datos de fecha en MySQL y su tamaño de almacenamiento

Diferencia de almacenamiento entre los tipos Char y VarChar

Valor	CHAR(4)	Almacenamiento	VARCHAR(4)	Almacenamiento
"	"	4 bytes	"	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

Tabla 17. Diferencia de almacenamiento entre los tipos Char y VarChar

### Claves foráneas en MySQL

Estrictamente hablando, para que un campo sea una clave foránea, éste necesita ser definido como tal al momento de crear una tabla. Se pueden definir claves foráneas en cualquier tipo de tabla de MySQL, pero únicamente tienen sentido cuando se usan tablas del tipo InnoDB.

A partir de la versión 3.23.43b, se pueden definir restricciones de claves foráneas con el uso de

tablas InnoDB. InnoDB es el primer tipo de tabla que permite definir estas restricciones para garantizar la integridad de los datos.

Para trabajar con claves foráneas, necesitamos hacer lo siguiente:

Crear ambas tablas del tipo InnoDB.

Usar la sintaxis `FOREIGN KEY(campo_fk) REFERENCES nombre_tabla (nombre_campo)`

Crear un índice en el campo que ha sido declarado clave foránea.

InnoDB no crea de manera automática índices en las claves foráneas o en las claves referenciadas, así que debemos crearlos de manera explícita. Los índices son necesarios para que la verificación de las claves foráneas sea más rápida. A continuación se muestra como definir dos tablas con una clave foránea, para establecer la relación: “un cliente puede estar en una o varias ventas”.

### Ejemplo

```
-- Tabla cliente
CREATE TABLE cliente
(
    id_cliente INT NOT NULL,
    nombre VARCHAR(30),
    PRIMARY KEY (id_cliente)
) TYPE = INNODB;

-- Tabla venta
CREATE TABLE venta
(
    id_factura INT NOT NULL PRIMARY KEY,
    id_cliente INT NOT NULL,
    cantidad INT,
    INDEX (id_cliente),
    FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente)
    ON DELETE CASCADE
    ON UPDATE CASCADE
) TYPE = INNODB;
```

La sintaxis completa de una restricción de clave foránea es la siguiente:

```
[CONSTRAINT símbolo] FOREIGN KEY (nombre_columna, ...)
    REFERENCES nombre_tabla (nombre_columna, ...)
    [ON DELETE {CASCADE | SET NULL | NO ACTION
        | RESTRICT}]
    [ON UPDATE {CASCADE | SET NULL | NO ACTION
        | RESTRICT}]
```

Las columnas correspondientes en la clave foránea y en la clave referenciada deben tener tipos de datos similares para que puedan ser comparadas sin la necesidad de hacer una conversión de

tipos. El tamaño y el signo de los tipos enteros debe ser el mismo. En las columnas de tipo caracter, el tamaño no tiene que ser el mismo necesariamente.

## CLASE No. 12

### PHPMYADMIN

Realizaremos ahora una introducción al gestor de bases de datos MySQL, *phpMyAdmin*, una aplicación web escrita en PHP para procesar la información guardada en este motor de base de datos.

Para entrar a *phpMyAdmin* nos aseguramos que el servidor de bases de datos MySQL está iniciado. Luego, en el navegador, ingresamos la siguiente url:  
<http://localhost/phpmyadmin>

El enlace respectivo también se encuentra en la página principal de Localhost. Para ingresar remotamente, lo debe hacer desde el cPanel de su Hosting, allí encuentra la opción para abrir phpMyAdmin.

Al ingresar la URL, nos encontraremos con una página similar a la que muestra la siguiente figura:

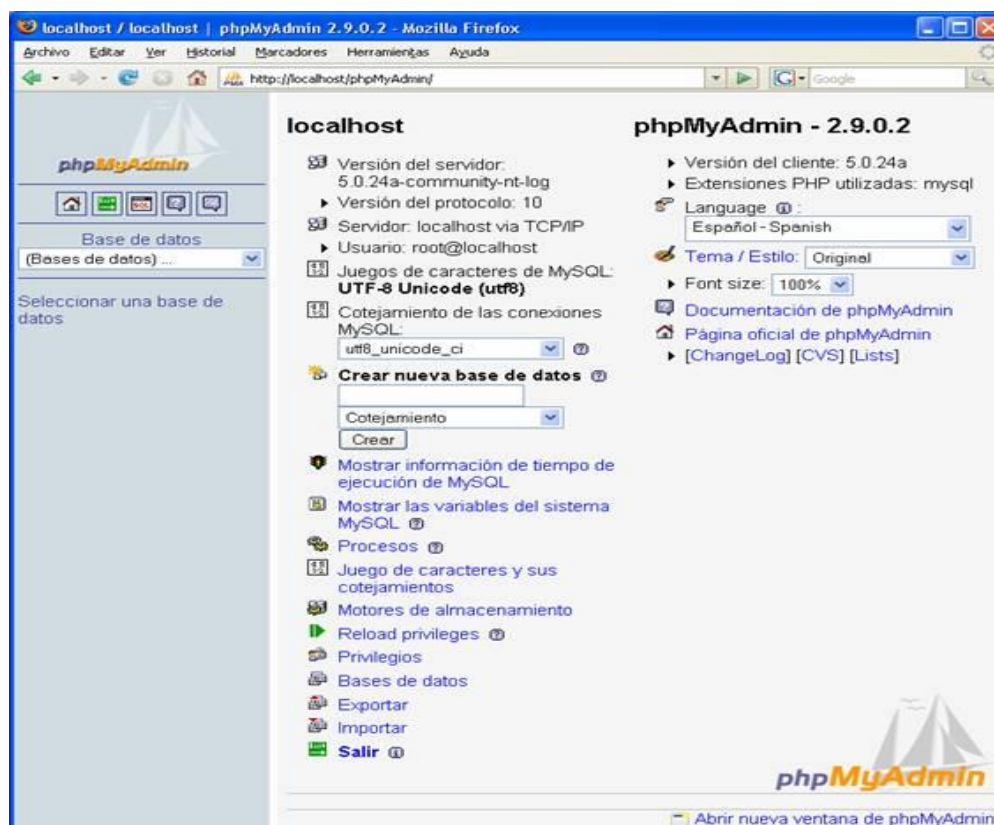


Figura 25. Vista en Firefox de la página principal de phpMyAdmin (versión 2.9.0.2) en Localhost



Si miran arriba a la izquierda veremos un icono con una casita que nos permitirá regresar a esta página inicial del phpMyAdmin.

Debajo dice *Bases de Datos*, ahí veremos la lista de bases de datos que hay en nuestro PC en MySQL (o en el hosting), con esta lista podemos seleccionar cualquiera de ellas para trabajar.

Ejemplo. Crear la base de datos “inmobiliaria”

En este caso no tenemos aún ninguna base de datos creada para nuestra Inmobiliaria, por eso vamos a ir al centro de la página y donde dice: *Crear nueva Base de Datos*, escribiremos “inmobiliaria” y damos clic en *Crear*. De esta forma logramos tener la base de datos Inmobiliaria. Ésta aparecerá en la parte izquierda, como indica la siguiente figura y en la parte central nos sugiere ya Crear las Tablas que contendrá esta base de datos (Propiedades y Ciudades).



Figura 26. Vista de la base de datos inmobiliaria

Ahora, en donde dice *Crear nueva Tabla en la Base de Datos inmobiliaria*, escribiremos de nombre “propiedades” y en número de campos 20 (no importa que sobren, se pueden ignorar después), y le damos clic en *Continuar*.

Nos aparecerá una serie de 20 cajas de texto para ingresar el Nombre de cada Campo, el Tipo de dato del Campo y la Longitud del campo.

Las bases de datos, para clasificar la información, permiten indicarle que tipo de datos contendrá cada campo, a esto se le llama tipo del campo, y dependerá si es un valor numérico sin

decimales, un valor numérico con decimales, un valor de texto corto, un valor de mucho texto, o una fecha, veremos en cada caso el tipo apropiado.

Debemos ingresar los campos como indica la siguiente figura, con sus nombres y tipos; es importante tener en cuenta si usamos mayúsculas y minúsculas puede traer problemas, ya que PHP lo diferenciará. Se recomienda por tanto escribir en minúscula para facilitar las tareas de programación.



Campo	Tipo ?	Longitud/Valores
id	INT	
titulo	VARCHAR	255
mtscuadrados	VARCHAR	255
banos	VARCHAR	255
habitaciones	VARCHAR	255
idciudad	INT	
precioventa	DOUBLE	
preciorenta	DOUBLE	
tipoinmueble	INT	
descripcion	TEXT	
fechaingreso	DATE	

Figura 27. Vista de la Estructura de la tabla propiedades de la base de datos inmobiliaria

Luego de ingresar todos los campos, damos clic en el botón Grabar, esto creará la tabla “propiedades” con los campos correspondientes.

Como puede verse, para el campo id, se utilizó el tipo int, que indica un valor numérico sin decimales, esto me permitirá identificar cada Propiedad.

En los campos titulo, mtscuadrados, banos, habitaciones, utilizamos varchar con un largo de 255, esto permitirá números, letras o símbolos hasta un ancho máximo de 255 caracteres, es algo bastante, pero el control del largo también se puede llevar a cabo en las cajas de texto de los formularios al ingresar cada propiedad, ya que el tipo varchar, a pesar de tener una longitud de 255, prevé ese máximo pero no lo ocupa todo. Si adentro guardamos solo un título “casa en la playa”, gastará solo 16 caracteres, que es lo que contiene dicho título (cadena), dado que el tipo

varchar es un tipo dinámico y ocupa memoria en caso de requerirla, a diferencia de el tipo char que ocupa exactamente el espacio que se le haya indicado, pues este tipo es estático.

Observe también que no se escribió “baños” sino banos para no utilizar la “n” (símbolos), y tampoco se usaron espacios en blanco.

Otra cosa es que si bien habitaciones contendrá un número, no utilizamos INT, si no VARCHAR (como regla: si un campo recibe números y no se utiliza para búsquedas directas u operaciones matemáticas se debe definir como tipo VARCHAR).

En precioventa y preciorenta se utiliza el tipo double, que permite guardar números con decimales.

En tipoinmueble, se usó el tipo numérico int para guardar en él, un 1 para casas en Venta, 2 para casas en Renta y un 3 si es ambos casos, o sea, cada tipo de inmueble tendrá ese código numérico que lo identifica.

En descripcion, se utilizó el tipo text, ya que es un texto amplio que puede ser de 10 o 100 o 500, o más caracteres, por tanto excede el máximo de un VARCHAR que son 255, y se aplica mejor un campo TEXT, que es virtualmente ilimitado (generalmente permiten hasta 64000 caracteres).

Por último, en el campo fechaingreso, donde se guardará la fecha en que esa propiedad se ingresa al sistema, empleamos el tipo date, que permite guardar una fecha.

Crear el índice para la tabla Propiedades

Como dijimos, el id, será el campo que identifica la propiedad, a estos campos usados para identificar registros, se le llama índice. Un índice permite acelerar las búsquedas cuando hay muchos registros (para nuestro caso, si hubiesen muchas Propiedades en la Tabla).

Si prestamos atencion, abajo dice: “¡No se ha definido el índice!”, bien, hagámoslo. Para esto, donde dice Crear un índice en 1 columna(s), damos clic en el botón Continuar.

	Campo	Tipo	Cotejamiento	Atributos	Nulo
<input type="checkbox"/>	id	int(11)			No
<input type="checkbox"/>	título	varchar(255)	utf8_general_ci		No
<input type="checkbox"/>	mtscuadrados	varchar(255)	utf8_general_ci		No
<input type="checkbox"/>	banos	varchar(255)	utf8_general_ci		No
<input type="checkbox"/>	habitaciones	varchar(255)	utf8_general_ci		No
<input type="checkbox"/>	idciudad	int(11)			No
<input type="checkbox"/>	precioventa	double			No
<input type="checkbox"/>	precioenta	double			No
<input type="checkbox"/>	tipoinmueble	int(11)			No
<input type="checkbox"/>	descripcion	text	utf8_general_ci		No
<input type="checkbox"/>	fechaingreso	date			No

Marcar todos/as /  Desmarcar todos Para los elementos marcados:

1 campo(s) ☒ Al final de la tabla ☐ Al comienzo

Después de

---

**Índices:**

¡No se ha definido el índice!

Crear un índice en  columna(s)

Espacio utilizado	
Tipo	Uso
<b>Datos</b>	0 Bytes
<b>Índice</b>	0 Bytes
<b>Total</b>	0 Bytes

Figura 28. Advertencia por no especificar un índice para la tabla

En la siguiente página, seleccionamos como campo, id, y verificamos que en Tipo de Índice, aparezca PRIMARY, ya que con este campo se harán la mayoría de búsquedas y además nos permitirá distinguir un registro de cualquier otro, definiendo así la clave principal de la tabla; por último damos clic en Grabar.

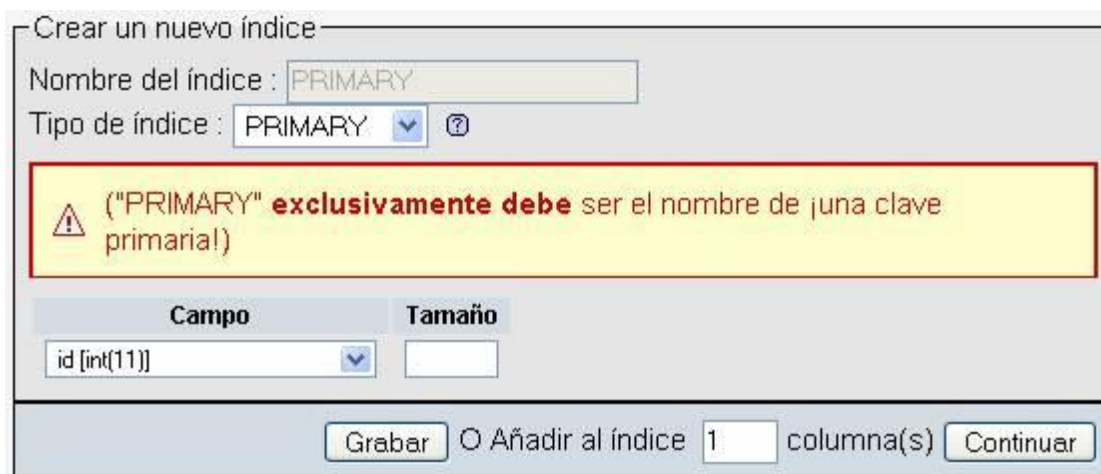


Figura 29. Creación de un índice

Veremos entonces que el campo id queda subrayado (I en la figura) indicando que es un índice principal, también abajo en la Lista de Índices (II en la figura) está el índice PRIMARY para el campo id.

Ese número de propiedad, podríamos ingresarlo a mano, 1, 2, 3, etc., o también podemos hacer que sea un número automático, es decir, que se autoincremente. Para esto, vamos a dar clic en el icono Cambiar (el lápiz de la marca III en la figura).



Figura 30. Vista estructura de la tabla propiedades para su modificación



Y en la siguiente página, como indica la siguiente figura, seleccionamos como parámetro en la propiedad Extra, el valor auto\_increment, de manera que ese número aumente solo, de forma automática y consecutiva.



Campo	Tipo ?	Longitud/Valores*1	Extra
id	INT	11	auto_increment

Figura 31. Establecer un campo entero (int) como autoincremental

Por último le damos clic al botón Grabar.

De esta forma tenemos lista nuestra primera tabla, llamada “propiedades”, en nuestra base de datos llamada “inmobiliaria”.

Crear la tabla “Ciudades”

Ahora, de la misma forma, crearemos la tabla “ciudades”, de paso repasamos un poco lo que realizamos con la tabla “propiedades.”

Damos un clic en la base de datos “inmobiliaria”, que vemos a la izquierda (I en la figura).



phpMyAdmin

Base de datos: inmobiliaria (1)

inmobiliaria (1) ← I

propiedades

Tabla: propiedades | Acción: [iconos] | Registros: 0 | Tipo: MyISAM

1 tabla(s) | Número de filas: 0 | MyISAM

Marcar todos/as / Desmarcar todos

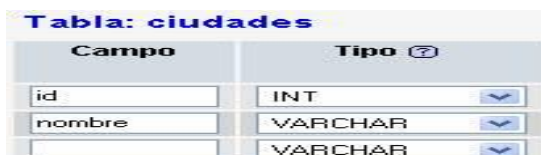
Vista de impresión | Diccionario de datos

Crear nueva tabla en la base de datos inmobiliaria

Nombre: [input] ← III | Número de campos: [input]

Figura 32. Creación de la tabla ciudades en la base de datos inmobiliaria

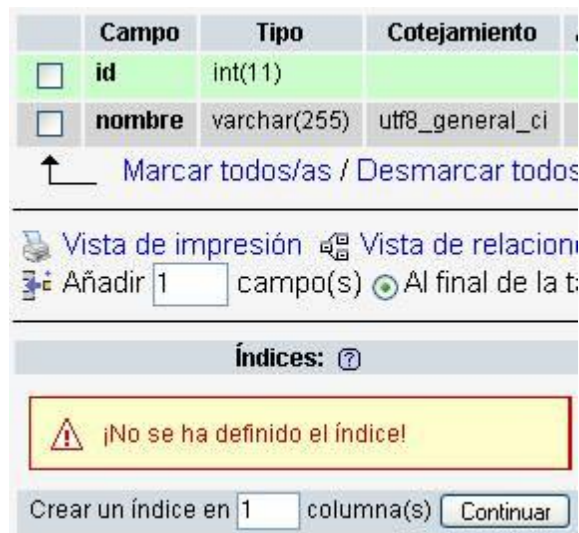
Luego escribimos el nombre de la tabla a crear, en este caso “ciudades” (II en la figura), y la cantidad de campos (III en la figura), por ejemplo 10, y damos Continuar.



Campo	Tipo ?
id	INT
nombre	VARCHAR
	VARCHAR

Figura 33. Especificación de campos en la tabla ciudades

Esta tabla, como vemos en la figura, tendrá solo 2 campos, que son id, de tipo int y el campo nombre (de la ciudad) de tipo varchar, de 255 de largo. Presionamos luego clic en el botón Grabar.



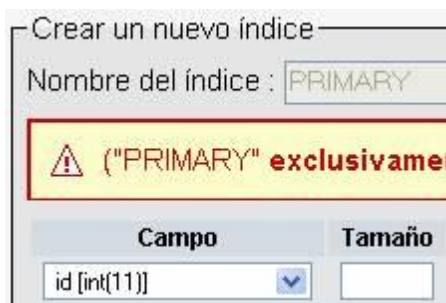
	Campo	Tipo	Cotejamiento
<input type="checkbox"/>	id	int(11)	
<input type="checkbox"/>	nombre	varchar(255)	utf8_general_ci

**Índices:**

¡No se ha definido el índice!

Figura 34. Advertencia por no especificar un índice para la tabla

Nos mostrará, que “¡No se ha definido índice!”; para crear el índice para el campo id, damos clic en Continuar.



**Crear un nuevo índice**

Nombre del índice: PRIMARY

(!) ("PRIMARY" exclusivamente para campos PRIMARY)

Campo	Tamaño
id [int(11)]	

Figura 35. Creación de un índice

Indicamos que el índice será para el campo id, de tipo PRIMARY y damos clic en el botón Grabar.

Seleccionamos cambiar (en el lapicito bajo Acción), para indicar nuevamente que el campo id, será un campo que autoIncremental, tal como hicimos en la tabla propiedades.

Y con esto tendríamos listas las 2 tablas para la base de datos inmobiliaria.

Vamos a seleccionar el icono de la casita para ir a la página principal de phpMyAdmin y escogemos nuevamente la base de datos “inmobiliaria” y ahí veremos una pantalla similar a esta:



Figura 36. Vista de la base de datos “inmobiliaria”

Aquí podemos ver las dos tablas, “ciudades” y “propiedades”, que son las que componen nuestra base de datos !inmobiliaria!

Insertar registros en una tabla

Insertar registros, significa agregar datos a la tabla, en este caso lo haremos en la tabla “ciudades”.

Para esto ,a la izquierda, vamos a dar clic, en “ciudades”.

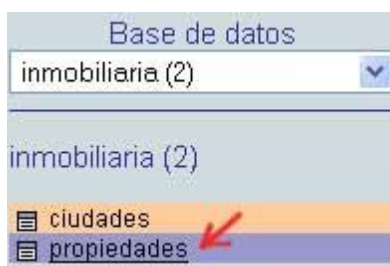


Figura 37. Tablas de la base de datos “inmobiliaria”

En la parte derecha de la siguiente figura, con la estructura de la tabla ciudades, la cual recordemos, tiene dos campos (I en la figura), id y nombre. También podemos ver que nuestra Tabla, tiene cero filas o registros (II en la figura). Pero a nosotros nos interesa la opción de Insertar Registros, que la realizamos presionando en el link Insertar (III en la figura) en la parte superior.



Figura 38. Vista de la estructura de la tabla “ciudades”

Esta opción permite agregar registros en la tabla actual (que es “ciudades”) y al seleccionarla podremos, como se muestra en la siguiente figura, escribir uno o dos registros. No es necesario llenar los 2, bastaría con ingresar una sola ciudad. Pero en este caso indicé “Madrid” y “Montevideo”, así ingresamos dos registros en vez de uno.



Figura 39. Vista Insertar de la tabla “ciudades”. Aquí puede ingresar los registros de la tabla

El número de id no se ingresa, pues se indicó al crear la tabla como un autonumérico, o sea que incrementa automáticamente cada vez que ingresamos algo. Bien, luego de ingresado, presionamos clic en Continuar (I en la figura). Al presionar el botón Continuar, volveremos a la lista de datos de la tabla, o quedara listo para insertar mas datos, esto depende del cuadro



desplegable que dice “Volver” (I en la figura). Si damos clic en la lista desplegable de “Volver” veremos las opciones que indica la siguiente figura:

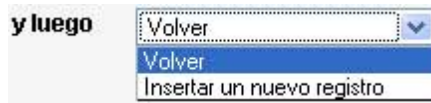


Figura 40. Opciones del cuadro de lista “y luego”

Las dos opciones, son: Volver, que regresa a la lista de registros e Insertar un nuevo registro, que vuelve a esta página de ingreso para seguir agregando más registros. De esta forma podemos ingresar tantas ciudades como queremos.

Ver la lista de registros o ver la estructura de la tabla

En la parte superior, veremos dos opciones: Examinar y Estructura, como indica la figura siguiente:



Figura 41. Enlaces para ir a las vistas Examinar y Estructura

Estructura: muestra, como vimos hace unos momentos, los campos de la tabla, los índices y el número de filas ingresadas.

Examinar: muestra la lista de registros ingresados, en nuestro caso, serán las ciudades, como muestra la figura:



Figura 42. Vista “Examinar” de la tabla “ciudades”

Editar y eliminar registros

En la figura anterior vemos como se muestran las ciudades ingresadas, cada una con el nombre que le dimos, y se ve también que fueron numeradas automáticamente en el id, ya que como recordamos, este es autonumérico.

Vemos también que en cada registro, hay dos iconos, un lápiz y una cruz (I en la figura).

El lápiz, permite modificar o editar ese registro, por ejemplo para cambiar a mayúsculas de la primera letra de cada ciudad

La cruz (ó X) permite eliminar el registro (ciudad) que quiera, basta con dar un clic en el icono X y luego en la opción Confirmar.

En algunos casos, puede ser necesario editar o eliminar, más de un registro, para esto tenemos los Cuadros de Selección (II en la figura), donde se podrán elegir cuales registros vamos a eliminar o editar. Luego de seleccionados los registros con un simple clic, para editar o eliminarlos, vamos a los iconos de la parte derecha (III en la figura), que nos permiten justamente, editar o eliminar los registros seleccionados.

Si queremos seleccionar todos los registros damos clic en en vínculo “Marcar todos/as”.

Si queremos deseleccionar todos los registros damos clic en “Desmarcar todos”.

Insertar nuevas propiedades

Como estamos en la tabla “ciudades”, debemos cambiarnos, para esto vamos a dar clic a la derecha en propiedades, donde indica la figura.

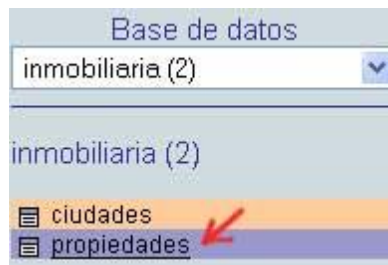


Figura 43. Tablas de la base de datos “inmobiliaria”: Seleccionar tabla “propiedades”

Al dar clic, veremos la estructura de la tabla “propiedades”, y tal como hicimos con las ciudades, daremos clic en Insertar.

**ESCUELA DE INFORMÁTICA**  
**TÉCNICO EN DESARROLLO DE SOFTWARE**  
**SUBMÓDULO PROGRAMACIÓN PARA LA WEB I**



Servidor: localhost Base de datos: inmobiliaria Tabla: propiedades

Examinar Estructura SQL Buscar **Insertar** Exportar Importar Operaciones Vaciar Eliminar

Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra	Acción
<input type="checkbox"/> id	int(11)			No		auto_increment	[iconos]
<input type="checkbox"/> titulo	varchar(255)	utf8_general_ci		No			[iconos]
<input type="checkbox"/> mtscuadrados	varchar(255)	utf8_general_ci		No			[iconos]
<input type="checkbox"/> banos	varchar(255)	utf8_general_ci		No			[iconos]
<input type="checkbox"/> habitaciones	varchar(255)	utf8_general_ci		No			[iconos]
<input type="checkbox"/> idciudad	int(11)			No			[iconos]
<input type="checkbox"/> precioventa	double			No			[iconos]
<input type="checkbox"/> preciorenta	double			No			[iconos]
<input type="checkbox"/> tipoinmueble	int(11)			No			[iconos]
<input type="checkbox"/> descripcion	text	utf8_general_ci		No			[iconos]
<input type="checkbox"/> fechaingreso	date			No			[iconos]

Marcar todas/as / Desmarcar todos Para los elementos que están marcados:

Vista de impresión Vista de relaciones Planteamiento de la estructura de tabla

Añadir 1 campo(s) Al final de la tabla Al comienzo de la tabla Después de id Continuar

Índices:				Espacio utilizado		Estadísticas de la fila	
Nombre de la clave	Tipo	Cardinalidad	Acción	Campo	Tipo	Uso	Enunciado
PRIMARY	PRIMARY	0	[iconos]	id	Datos	0 Bytes	Formato
Crear un índice en 1 columna(s) Continuar				Índice	1,024 Bytes	Cotejamiento	utf8_general_ci
				Total	1,024 Bytes	Filas	0
						Próxima Autoindex	1
						Creación	22-02-2007 a las 03:23:06
						Última actualización	22-02-2007 a las 03:23:06

Figura 44. Vista de la estructura de la tabla “propiedades”

En la opción Insertar, aparecerá la ficha de edición, donde ingresaremos la información, como por ejemplo la de la figura que se muestra:

Campo	Tipo	Función	Nulo	Valor
id	int(11)	[dropdown]		[input]
titulo	varchar(255)	[dropdown]		Casa en la Playa
mtscuadrados	varchar(255)	[dropdown]		50
banos	varchar(255)	[dropdown]		2
habitaciones	varchar(255)	[dropdown]		4
idciudad	int(11)	[dropdown]		2 <span style="color: red;">(I)</span>
precioventa	double	[dropdown]		100000
preciorenta	double	[dropdown]		1500
tipoinmueble	int(11)	[dropdown]		3 <span style="color: red;">(II)</span>
descripcion	text	[dropdown]		Casas en la Playa, a 200 mts del mar, excelente <u>ubicacion</u> y condiciones.
fechaingreso	date	[dropdown]		2007-02-26 <span style="color: red;">(III)</span>

Figura 45. Vista “Insertar” de la tabla “propiedades” con valores ingresados

Acá debemos tener en cuenta (I en la figura), que en el campo idciudad, debemos ingresar el número identificador de la ciudad, tal y como se especificó en la tabla “ciudades”, en este caso

se ha puesto el 2, por tanto corresponde a la ciudad montevideo. Lo que estamos haciendo, es establecer una relación “lógica” de los registros que están en la tabla “ciudades” con los registros que se encuentran en la tabla “propiedades”, según la regla “en una ciudad se pueden tener varias propiedades”.

En el campo tiponinmueble (II en la figura) veremos un 3, que significa que la casa está tanto para venta como para renta (recuerden que 1 es venta, y 2 es renta). Esto lo convenimos así para facilitar el sistema.

Por ultimo el campo fecha debe ser escrito en ese formato año-mes-día (aaaa-mm-dd): cuatro dígitos del año, guión, 2 dígitos del mes, guión, 2 dígitos del día, o damos clic en el icono calendario (III en la figura) y seleccionamos con otro(s) clic(s), la fecha correspondiente.

Terminamos el ingreso con el botón Continuar, recordando que podemos volver a ingresar otros registros cuando queramos, o eliminar y/o editar como explicamos anteriormente.

### CLASE No. 13

#### **FUNCIONES DE PHP PARA ACCEDER A MYSQL**

Veamos algunas funciones de PHP que utilizaremos en el siguiente ejemplo para acceder a las bases de datos de MySQL.

##### **Función `mysql_connect (nombre_servidor, usuario_bd, contraseña)`**

Abre o reutiliza una conexión a un servidor MySQL. Deben especificarse tres argumentos: el nombre del servidor, el usuario de la base de datos y la contraseña de este usuario. Esta función retorna la cadena de conexión al servidor de MySQL.

##### **Función `mysql_select_db (nombre_bd, cadena_conexión)`**

Establece la base de datos activa en el servidor asociado con el identificador de enlace especificado. Cada llamada posterior a la función `mysql_query()` será ejecutada en la base de datos activa.

##### **Función `mysql_query (cadena_consulta, cadena_conexión)`**

Envía una única consulta (múltiples consultas no están soportadas) a la base de datos actualmente activa en el servidor asociado con la cadena de conexión (*link\_identifier*).

##### **Funciones `mysql_num_rows (resultado_consulta)` y `mysql_numrows (resultado_consulta)`**

Recuperan el número de filas de una consulta tipo SELECT (a este se le conoce como un *resultset* -conjunto de resultados-. En otros lenguajes estos elementos se consideran objetos y pueden encontrarse con nombres como *recordset* -conjunto de registros-). Este comando es únicamente válido para sentencias como SELECT o SHOW que retornan un *resultset* real. Para recuperar el número de filas afectadas por consultas INSERT, UPDATE, REPLACE o DELETE, use `mysql_affected_rows()`.

### **Función `mysql_fetch_array` (resultado\_consulta)**

Recupera una fila de resultado como un array (arreglo) asociativo, un array numérico o como ambos, es decir, guarda en un arreglo los registros devueltos en una consulta tipo SELECT. Si no quedan más filas, la función devuelve falso.

### **Función `mysql_free_result` (resultado\_consulta)**

Libera toda la memoria asociada con el identificador del resultado de la consulta. Solo necesita ser llamada si hay preocupación por la cantidad de memoria que está siendo usada por las consultas que devuelven conjuntos de resultados grandes. Toda la memoria de resultados asociados se liberará automáticamente al finalizar la ejecución del script.

### **Función `mysql_close` (cadena\_conexión)**

Cierra la conexión no continua al servidor de MySQL que es asociada con el identificador de enlace especificado. Si el *link\_identifier* (cadena\_conexión) no es especificado, el último enlace abierto es usado. Frecuentemente no es necesario usar esta función, ya que los enlaces abiertos no continuos son automáticamente cerrados al final de la ejecución del script.

## **EJEMPLO SOBRE BASES DE DATOS CON PHP**

Desarrollaremos un sitio Web conformado por varias páginas que nos permitirán mostrar y manipular información de la base de datos. Se trabajará con una tabla llamada *tblusuario* que tiene los campos *usr* (*varchar(30)* y *clave principal*), *nombre* (*varchar (50)*), *correo* (*varchar (70)*) *clave* (*varchar (100)*). El nombre de la base de datos es *dbusuarios*, la cual debe ser creada en MySQL.

### **Ejemplo:** conexion.php

Esta página permite la conexión con el servidor de bases de datos y con la base de datos misma. Esta página no se ejecuta por si sola, debe ser invocada por otra; ésta contiene una función que devuelve un valor que nos permite saber si se estableció o no la conexión con la base de datos.

```
<?php
//Función conectar
function conectar()
{
    if (!($link=mysql_connect("localhost","root","")))
    {
        echo "Error. No puede conectarse al servidor de base de datos, intente más tarde.";
        exit();
    }
    if (!mysql_select_db("dbusuarios",$link))
    {
        echo "Error. Hay un problema al intentar conectar la base de datos especificada o ésta no
existe.";
        exit();
    }
    return $link;
}
?>
```

### **Ejemplo:** mostrar\_usuarios.php

Esta página muestra el listado general de usuarios registrados. La página muestra unos vínculos perforados, los cuales se crean dinámicamente al igual que la tabla HTML que muestra los datos.

```
<html>
<head>
<title>Mostrar usuarios</title>
</head>
<body>
<h1 align="center">Listado General de Usuarios</h1>
<?php
include("conexion.php");
$link=conectar();
$sql=mysql_query("select * from tblusuario",$link);
$numreg=mysql_numrows($sql);
?>
<center>
<table border="1" cellspacing="1" cellpadding="1">
<tr bgcolor="#CCCCCC">
<th width="58">Usuario</th>
<th width="69">Nombre</th>
<th width="66">Correo</th>
<th width="32">Modificar</th>
<th width="32">Eliminar</th>
</tr>
<caption align="bottom">
Total de registros:
<?
echo $numreg;
?>
</caption>
<?php
while($fila = mysql_fetch_array($sql))
{
printf("<tr>
<td>%s</td>
<td>%s</td>
<td>%s</td>
<td>
<a href=\"modificar.php?usr=%s\">Modificar</a>
</td>
<td>
<a href=\"eliminar.php?usr=%s\" onclick=\"return confirm('¿Está seguro de eliminar este
registro?')\" title=\"Elimina a %s\">Eliminar</a>
</td>
</tr>",
$fila ["usr"], $fila ["nombre"], $fila ["correo"], $fila ["usr"], $fila ["usr"], $fila ["nombre"]);
}
mysql_free_result($sql);
mysql_close($link);
```



```
?>
</table>
</center>
</body>
</html>
```

**Ejemplo:** agregar.html

Esta página contiene un formulario que permite el ingreso de los datos del usuario. La página también contiene una validación con Javascript que verifica que los datos estén bien ingresados; esta validación se encuentra en un archivo js, que se supone, está en una subcarpeta del sitio llamada "scripts". Los datos serán guardados mediante una página llamada guardar.php

```
<html>
<head>
<title>Agregar usuarios</title>
<script language="javascript" type="text/javascript" src="scripts/validar.js">
</script>
</head>

<body>
<h1 align="center">USUARIOS</h1>
<hr>
<form name="form1" id="form1" method="post" action="guardar.php" onSubmit="return
validar_datos()">
<center>
<table width="427" border="0">
<tr>
<td>Usuario</td>
<td><input name="txtusr" type="text" id="txtusr" maxlength="30"></td>
</tr>
<tr>
<td>Nombre</td>
<td><input name="txtnom" type="text" id="txtnom" maxlength="50"></td>
</tr>
<tr>
<td>Correo</td>
<td><input name="txtcorreo" type="text" id="txtcorreo" maxlength="70"></td>
</tr>
<tr>
<td>Contraseña</td>
<td><input name="txtclave" type="password" id="txtclave" maxlength="30"></td>
</tr>
<tr>
<td>Confirmar Contraseña</td>
<td><input name="txtclave1" type="password" id="txtclave1" maxlength="30"></td>
</tr>
<tr>
<td colspan="2">
```

```
<input type="submit" value="Guardar datos">
    &nbsp;
    <input type="reset" value="Restablecer">
</td>
</tr>
</table>
</center>
</form>
</body>
</html>
```

**Ejemplo:** validar.js

Esta página contiene la validación del formulario que permite el ingreso de los datos del usuario.

```
function validar_datos()
{
    var usr, nom, correo, clave, clave1, sw;
    sw=false; //Supuesto: datos mal ingresados
    usr=document.getElementById("txtusr").value;
    nom=document.getElementById("txtnom").value;
    correo=document.getElementById("txtcorreo").value;
    clave=document.getElementById("txtclave").value;
    clave1=document.getElementById("txtclave1").value;

    if (usr!="" && usr.length>=3 && usr.indexOf(" ",0)==-1))
    {
        if (nom!="" && nom.length>=5)
        {
            if (/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/ .test(correo))
            {
                if (clave!="" && clave.length>=3 && clave.indexOf(" ",0)==-1))
                {
                    if (clave1==clave2)
                    {
                        sw=true; //Los datos están bien ingresados
                    }
                    else
                    {
                        alert("La contraseña y su confirmación no coinciden");
                        document.getElementById("txtclave1").focus();
                    }
                }
            }
            else
            {
                alert("Ingrese una contraseña válida");
                document.getElementById("txtclave").focus();
            }
        }
    }
    else
    {

```



```
        {
            alert("Ingrese una dirección de correo válida");
            document.getElementById("txtcorreo").focus();
        }
    }
    else
    {
        alert("Ingrese el nombre");
        document.getElementById("txtnom").focus();
    }
}
else
{
    alert("Nombre de usuario no válido");
    document.getElementById("txtusr").focus();
}
return sw;
}
```

#### CLASE No. 14

##### **Ejemplo:** guardar.php

Esta página guarda los datos pasados del formulario de la página agregar.html

```
<html>
<head>
<title>Guardar datos</title>
</head>
<body>
<?php
//Guardar información en la bases de datos
function validar_existencia($usr, $link)
{
    $sql = mysql_query("select usr from tblusuario where usr = '$usr' ", $link);
    if(mysql_num_rows($sql)==0)
        $sw=true;
    else
        $sw=false;
    return $sw;
}
include("conexion.php");
$link=conectar();
$usr=$_POST['txtusr'];
$nom=$_POST['txtnom'];
$correo=$_POST['txtcorreo'];
```

```
$clave=$_POST['txtclave'];
//Validamos si el usuario no existe en la base de datos
if(validar_existencia($usr, $link))
{
    mysql_query("insert into tblusuario (usr,nombre,correo,clave) values ('$usr', '$nom', '$correo',
md5('$clave'))", $link);
    $men = "Se guardó el registro en la base de datos";
    $url = "mostrar_usuarios.php";
}
else
{
    $men = "El usuario $usr ya se encuentra registrado en la base de datos";
    $url = "agregar.html";
}
print "<script>
    alert('$men');
    parent.location.href = '$url';
</script>";
mysql_close($link);
?>
</body>
</html>
```

**Ejemplo:** buscar.php

Esta página contiene un formulario donde especificamos un texto de búsqueda. El fin es mostrar el dato o datos que coincidan con la expresión de búsqueda. Utilizaremos como modelo la página mostrar\_usuarios.php con un formulario al inicio; la consulta mostrará los usuarios que coincidan con la cadena o con parte de ella, la cual será especificada en un cuadro de texto.

```
<html>
<head>
<title>Mostrar usuarios</title>
</head>
<body>
<h1 align="center">Listado General de Usuarios</h1>
<form name="f1" id="f1" action="buscar.php" method="post">
<b>Ingrese el nombre de usuario o parte de él para realiza la búsqueda</b>
<br/>
<input type="text" name="txtusr" id="txtusr"/>
<input type="submit" value="Buscar"/>
</form>

<?php
If (isset($_POST['txtusr']))
{
    $usr=$_POST['txtusr'];
    include("conexion.php");
    $link=conectar();
```

```
$sql=mysql_query("select * from tblusuario where usr like '%$usr%'", $link);
$nreg=mysql_numrows($sql);
?>
<center>
<table border="1" cellspacing="1" cellpadding="1">
<tr bgcolor="#CCCCCC">
<th width="58">Usuario</th>
<th width="69">Nombre</th>
<th width="66">Correo</th>
<th width="32">Modificar</th>
<th width="32">Eliminar</th>
</tr>
<caption align="bottom">
Total de registros:
<?
    echo $nreg;
?>
</caption>
<?php
while($fila = mysql_fetch_array($sql))
{
printf("<tr>
    <td>%s</td>
    <td>%s</td>
    <td>%s</td>
    <td>
        <a href='\"modificar.php?usr=%s\"'>Modificar</a>
    </td>
    <td>
        <a href='\"eliminar.php?usr=%s\"' onclick='\"return confirm('¿Está seguro de eliminar este
registro?')\" title='\"Elimina a %s\"'>Eliminar</a>
    </td>
</tr>",
$fila ["usr"], $fila ["nombre"], $fila ["correo"], $fila ["usr"], $fila ["usr"], $fila ["nombre"]);
}
mysql_free_result($sql);
mysql_close($link);
?>
</table>
</center>
<?
}
?>
</body>
</html>
```

**Ejemplo:** eliminar.php

Esta página recibe un parámetro de la página mostrar\_usuarios.php y elimina el registro respectivo de la tabla.

```
<html>
<head>
<title>Eliminación de usuarios</title>
</head>
<body>
<?php
include("conexion.php");
$link=conectar();
$usr=$_GET['usr'];
mysql_query("delete from tblusuario where usr='$usr'", $link);

print "<script>
    alert('Se eliminó el registro en la base de datos');
    window.location = 'mostrar_usuarios.php';
</script>";

mysql_close($link);
?>
</body>
</html>
```

**Ejemplo:** modificar.php

Esta página recibe un parámetro de la página mostrar\_usuarios.php y ofrece la interfaz para modificar el registro respectivo de la tabla. Mostrará los campos Usuario, Nombre y Correo y no dejará modificar el usuario. Para la clave se puede crear una página similar que permita modificarla.

```
<html>
<head>
<title>Modificación de usuarios</title>
</head>
<body>
<?php
include("conexion.php");
$link=conectar();
$usr=$_GET['usr'];
$sql=mysql_query("select * from tblusuario where usr='$usr'", $link);
$fila=mysql_fetch_array($sql);
?>
<h2>Modificación de usuario</h2>
<br/>
<form name="f1" id="f1" action="actualizar.php" method="post">
Usuario
<?php
    printf("<input type='text' name='txtusr' id='txtusr' readonly='readonly' value='%s'/>", $fila["usr"]);
?>
<br/>
Nombre
```

```
<?php
    printf("<input type='text' name='txtnom' id='txtnom' maxlength='50' value='%s' />", $fila["nom"]);
?>
<br/>
Correo
<?php
    printf("<input type='text' name='txtcorreo' id='txtcorreo' maxlength='70' value='%s' />",
    $fila["correo"]);
?>
<br/>
<input type='submit' value='Guardar cambios' />
<input type='reset' value='Restablecer' />

</form>
</body>
</html>
```

**Ejemplo:** actualizar.php

Esta página recibe el formulario de la página modificar.php y actualiza el registro respectivo de la tabla.

```
<html>
<head>
<title>Actualización de usuarios</title>
</head>
<body>
<?php
include("conexion.php");
$link=conectar();
$usr=$_POST['txtusr'];
$nom=$_POST['txtnom'];
$correo=$_POST['txtcorreo'];

mysql_query("update tblusuario set nom='$nom', correo='$correo' where usr='$usr'", $link);

echo "<script>
    alert('Se actualizó el registro en la base de datos');
    window.location = 'mostrar_usuarios.php';
</script>";

mysql_close($link);
?>
</body>
</html>
```

## FUNCIONALIDADES DE PHP

### CLASE No. 15

PHP cuenta con una serie de poderosas características que permiten acceder a recursos del servidor web, a los clientes y a otros tipos de servidores. Algunas de estas funcionalidades se enuncian a continuación.

#### SESIONES

En PHP, una sesión es la secuencia de páginas que un usuario visita en un sitio Web, desde que entra, hasta que lo abandona. Los navegadores muestran este comportamiento observando los botones “Anterior”, “Siguiente”, donde va “recordando” las páginas y sitios que visitamos. Las sesiones generalmente tienen asociado un nombre o identificador que permite implementar características de seguridad en los sitios web, por ejemplo, los sistemas de cuentas (de correo u otra) permiten ver cierta información dependiendo de los parámetros que suministre un usuario (nombre de usuario, contraseña, etc.); una vez se ingresa a la parte deseada, se crea una sesión que tendrá un nombre acorde a lo suministrado por éste. Estas sesiones generalmente tienen una opción para destruirla (Cerrar, Salir, Logout, etc.), de tal manera que el navegador posteriormente no “recuerde” nada y de esta manera otros usuarios no puedan ver nuestra información.

Las sesiones entonces nos servirán para almacenar información que se memorizará durante toda la visita de un usuario a un sitio web. Para cada usuario, PHP genera internamente un identificador de sesión único, que sirve para saber las variables de sesión que pertenecen a cada usuario. Para conservar el identificador de sesión durante toda la visita de un usuario al sitio, PHP almacena la variable de sesión en una cookie, o bien la propaga a través de la URL. Esto se puede configurar desde el archivo php.ini.

El término sesión en PHP (*session* en inglés), se aplica a la secuencia de navegación; para establecer sesiones en nuestros ejemplos crearemos un identificador único que asignamos a cada una de estas sesiones de navegación. A este identificador de sesión se le denomina comúnmente como la *sesión*.

El proceso en los lenguajes de programación con el tratamiento de sesiones sigue esta secuencia:

- ¿Existe una sesión?
- Si existe, se retoma
- Si no existe, se crea una nueva
- Se genera un identificador único para la sesión

Para que no perder el hilo de la navegación del usuario, se debe asociar la sesión a todas las URLs y acciones de formulario.

Veamos ahora algunas funciones para el tratamiento de las sesiones en PHP:

### **Función session\_start()**

Al manejar sesiones en PHP debemos incluir la función session\_start() ya sea para retomar la sesión o para crearla. Esta función debe especificarse al inicio de todo el código, por encima de las cabeceras, de lo contrario no funcionará.

### **Función session\_destroy()**

Esta función destruye la sesión actual. Al hacer esto, la navegación que traía el usuario se pierde, por lo que tendrá que acceder con las URLs necesarias y con las acciones de formulario que requiera nuevamente.

### **session\_id()**

Devuelve el identificador de la sesión.

### **Ejemplo: manejo de sesiones**

A continuación se creará un sistema para iniciar sesión utilizando la base de datos *bdusuarios*. El usuario deberá ingresar uno de los nombres de usuario y contraseña que se encuentran almacenadas en la tabla *tblusuario*. Si logra iniciar la sesión, el sistema lo redireccionará a una página donde solo se podrá acceder identificándose.

### **Ejemplo: ingreso\_usuario.html**

Esta página proporciona un formulario para que el usuario suministre los datos de inicio de sesión; usuario y contraseña y lo enviará a una página que valida que la información exista en la base de datos.

```
<html>
<head>
<title>Iniciar Sesión</title>
</head>
<body>
<h1 align="center">USUARIOS</h1>
<hr>
<form name="form1" id="form1" method="post" action="validar_ingreso.php">
  <center>
    <p>Iniciar sesión </p>
    <table border="0">
      <tr>
        <td>Usuario</td>
        <td><input name="txtusr" type="text" id="txtusr"></td>
      </tr>
      <tr>
        <td>Contrase&ntilde;a</td>
        <td><input name="txtclave" type="password" id="txtclave"></td>
      </tr>
      <tr>
        <td colspan="2">
          <input type="submit" value="Ingresar">
          &nbsp;
          <input type="reset" value="Restablecer">
        </td>
      </tr>
    </table>
  </center>
</form>
```

```
</tr>
</table>
</center>
</form>
</body>
</html>
```

**Ejemplo:** validar\_ingreso.php

Esta página recibe la información del formulario de inicio de sesión y verificará que la información suministrada se encuentre almacenada en la base de datos. Redireccionará al usuario adecuadamente a una de dos páginas, dependiendo de si suministró los datos correctos o no.

```
<html>
<head>
<title>Validar ingreso</title>
</head>
<body>
<?
include("conexion.php");
$link=conectar();
$usr=$_POST['txtusr'];
$clave=$_POST['txtclave'];
$sql=mysql_query("select * from tblusuario where usr='$usr' and clave=md5('$clave')",$link);
$nreg=mysql_num_rows($sql);
if ($nreg>0)
{
    $mensaje="$usr ha iniciado sesión";
    $url="inicio_sesion.php?usr=$usr";
}
else
{
    $mensaje="Usuario o contraseña incorrectos. Sesión no iniciada";
    $url="ingreso_usuario.html";
}
echo "<script type='text/javascript'>
    alert('$mensaje');
    window.location='$url';
</script>";

mysql_free_result($sql);
mysql_close($link);
?>

</body>
</html>
```

**Ejemplo:** inicio\_sesion.php

Esta página es la que crea la sesión y le asigna el nombre “usuario” con el valor que se ingrese en el campo respectivo del formulario. Luego de hacerlo, redirecciona el control a otra página.



```
<?
session_start();
?>
<html>
<head>
<title>Iniciar sesión</title>
</head>
<body>
<?
$_SESSION["usuario"] = $_GET['usr'];
$url="mostrar_usuarios.php";
print "<script type='text/javascript'>
        parent.location.href='$url';
        </script>";
?>
</body>
</html>
```

**Ejemplo:** cerrar\_sesion.php

Esta página destruye la sesión actual. Esto es necesario para no dejar la sesión disponible a otros usuarios, en particular, si se está implementando un control de acceso a las páginas.

```
<?
session_start();
?>
<html>
<head>
<title>Cerrar sesión</title>
</head>
<body>
<?
session_destroy();
$url="ingreso_usuario.html";
echo "<script type='text/javascript'>
        alert('La sesión ha finalizado');
        window.location='$url';
        </script>";
?>
</body>
</html>
```

**Ejemplo:** mostrar\_usuario\_sesiones.php

Vamos a realizar unos cambios en la página mostrar\_usuarios.php, de tal manera que solo puedan acceder a esta página los usuarios que hayan iniciado sesión, y que además muestre el usuario que inició sesión. Con esto, estamos añadiendo características de seguridad a esta página, aunque cabe anotar, que para configurar aspectos de seguridad más avanzados, será necesario escribir algo más de código. Por tanto, para todas las páginas que requieran de un

control de acceso, se deben realizar algunos cambios en el código de cada una, tal y como se ilustra aquí.

```
<?
session_start();
?>

<html>
<head>
<title>Mostrar usuarios</title>
</head>
<body>
<?php
    if (!isset($_SESSION["usuario"]))
    {
        print "<script type='text/javascript'>
            alert('Debe iniciar sesión para acceder a esta página');
            parent.location.href='ingreso_usuario.html';
        </script>";
        exit();
    }
?>
<p align="right">
Usuario Conectado:
<span style="color:green;">
<?php
    print $_SESSION["usuario"];
?>
</span>
</p>
<p align="right">
<a href="cerrar_sesion.php">Cerrar Sesión</a>
</p>

<h1 align="center">Listado General de Usuarios</h1>
<?php
include("conexion.php");
$link=conectar();
$sql=mysql_query("select * from tblusuario",$link);
$nreg=mysql_numrows($sql);
?>
<center>
<table border="1" cellspacing="1" cellpadding="1">
<tr bgcolor="#CCCCCC">
<th width="58">Usuario</th>
<th width="69">Nombre</th>
<th width="66">Correo</th>
<th width="32">Modificar</th>
<th width="32">Eliminar</th>
```

```
</tr>
<caption align="bottom">
Total de registros:
<?
    echo $nreg;
?>
</caption>
<?php
while($fila = mysql_fetch_array($sql))
{
printf("<tr>
    <td>%s</td>
    <td>%s</td>
    <td>%s</td>
    <td>
        <a href=\"modificar.php?usr=%s\">Modificar</a>
    </td>
    <td>
        <a href=\"eliminar.php?usr=%s\" onclick=\"return confirm('¿Está seguro de eliminar este
registro?')\" title=\"Elimina a %s\">Eliminar</a>
    </td>
</tr>",
$fila ["usr"], $fila ["nombre"], $fila ["correo"], $fila ["usr"], $fila ["usr"], $fila ["nombre"]);
}
mysql_free_result($sql);
mysql_close($link);
?>
</table>
</center>
</body>
</html>
```

## CLASE No. 16

### PHP FILE UPLOADS

En PHP tenemos muchas funcionalidades desarrolladas desde el principio y sin necesidad de instalar ningún añadido en nuestro servidor. Es el caso de subir archivos a un servidor Web por HTTP y a través de una página con un formulario, donde se permite seleccionar el archivo que queremos cargar de nuestro disco duro.

El ejemplo se encuentra bien documentado en muchas de páginas para desarrolladores, sin ir más lejos en la página de la propia tecnología: <http://www.php.net/manual/es/features.file-upload.php>. En este caso intentamos ir un poco más allá, realizando un par de comprobaciones al subir el fichero (archivo) y combinando en el mismo formulario campos de tipo *file* y tipo *text*.

**Ejemplo: subir imágenes al servidor**

Este ejemplo se compone de dos páginas, una que ofrece un formulario al usuario para seleccionar el archivo a subir y otra que realiza la subida al servidor. A continuación se describen los pasos para crear un sistema que sube imágenes al servidor (aunque con esta funcionalidad de PHP se pueden subir al servidor diferentes tipos de archivos).

**Ejemplo: formulario\_subir.php**

Formulario para seleccionar los archivos a subir. Es un formulario cualquiera, pero tiene una serie de particularidades y campos **file**, que no solemos utilizar habitualmente.

```
<html>
<head>
<title>Subir archivos</title>
</head>
<body>
<form action="subirarchivo.php" method="post" enctype="multipart/form-data" name="f1" id="f1">
  <input type="hidden" name="MAX_FILE_SIZE" id="MAX_FILE_SIZE" value="1000000"/>
  <br/>
  <br/>
  <b>Subir un nuevo archivo al servidor</b>
  <br/>
  <input name="archivousuario" id="archivousuario" type="file"/>
  <br/>
  <input type="submit" value="Subir archivo"/>
</form>
</body>
</html>
```

Para empezar vemos que se ha colocado un nuevo atributo en el formulario: `enctype="multipart/form-data"`, necesario para subir en un mismo formulario datos y archivos.

También tenemos el campo *hidden* `MAX_FILE_SIZE`, que sirve para indicar el tamaño en bytes de los archivos a subir. Este campo algunos navegadores no tienen porqué entenderlo o hacerle caso. Además, es fácil saltarse esa protección, por lo que deberemos en las propias páginas PHP comprobar que el archivo tenga el tamaño que deseamos. Por último, tenemos el campo tipo *file*, donde se seleccionará el archivo a subir.

**Página que sube los archivos**

Esta página debe hacer las comprobaciones necesarias para saber si las características del archivo a subir son las que deseamos y realizar la copia del archivo en un directorio del servidor.

Para hacer las comprobaciones, PHP nos crea una serie de variables que podemos acceder con la información del archivo enviado. Estas se indican a continuación.

`$_FILES['archivousuario']['name']`

El nombre original del archivo en la máquina cliente.

`$_FILES['archivousuario']['type']`

El tipo mime del archivo (si el navegador lo proporciona). Un ejemplo podría ser "image/gif".

`$_FILES['archivousuario']['size']`

El tamaño en bytes del fichero recibido.

`$_FILES['archivousuario']['tmp_name']`

El nombre del archivo temporal que se utiliza para almacenar en el servidor el archivo recibido.

**Ejemplo:** subirarchivo.php

Esta página se encarga de llevar el archivo hasta el servidor. Se supone que existe en el sitio una subcarpeta con el nombre de imágenes, allí se enviará la imagen que se cargue.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Subir archivo</title>
</head>
<body>
<?
//Datos del archivo. La imagen se cargará en una subcarpeta llamada "imágenes"
/*En este ejemplo vamos a utilizar el arreglo $_FILES[], pero es muy probable que en la
documentación en Internet encuentre el arreglo $HTTP_POST_FILES para realizar lo mismo;
algunas versiones de PHP no admiten este segundo arreglo, y por otro lado, $_FILES es una
implementación más reciente*/
$nombre_archivo = "imagenes/" . $_FILES['archivousuario']['name'];
$tipo_archivo = $_FILES['archivousuario']['type'];
$tamano_archivo = $_FILES['archivousuario']['size'];
//compruebo si las características del archivo son las que deseo
if (!(strpos($tipo_archivo, "gif") || strpos($tipo_archivo, "jpeg")) && ($tamano_archivo <=
1000000)))
{
    $mensaje= "La extensión o el tamaño de los archivos no es correcta. Puede subir archivos .gif
o .jpg, con un tamaño máximo de 1MB";
}
else
{
    if (move_uploaded_file($_FILES['archivousuario']['tmp_name'], $nombre_archivo))
    {
        $mensaje= "El archivo ha sido cargado correctamente.";
    }
    else
    {
        $mensaje= "Ocurrió algún error al subir el archivo. No pudo cargarse.";
    }
}

echo "<script type='text/javascript'>
    alert('$mensaje');
    window.location='formulario_subir.php';
```

```
</script>";  
?>  
</body>  
</html>
```

Se recogen los datos necesarios del archivo, como su nombre, extensión y tamaño y se pasa a comprobar que la extensión sea .gif o .jpg y que el tamaño sea menor o igual a 100000 Bytes (100KB).

Si el archivo tiene las características deseadas, se puede subir al servidor. Para ello se utiliza la función *move\_uploaded\_file()*, que recibe el nombre del archivo temporal que se desea subir y el nombre del archivo que se desea dar.

Cuando se sube el archivo, el servidor lo copia en una localización temporal para que seamos nosotros los que elijamos la posición definitiva donde queremos que se almacene. Si no lo copiamos a ningún sitio, después de la ejecución de la página, se borra de su localización temporal.

La función *move\_uploaded\_file()* se utiliza para mover el archivo a la posición definitiva. Recibe por un lado el nombre temporal del fichero y por otro el nombre que deseamos asignarle definitivamente y, si se desea, la ruta para llegar al directorio donde queremos guardarlo. En el caso del ejemplo se indica el subdirectorío y el nombre del archivo, en caso de no indicar el subdirectorío, el archivo se subirá al mismo directorio donde están las páginas PHP que hacen el upload. Esta función devuelve un valor booleano que indica si hubo o no éxito al subir el archivo.

#### **Nota**

Es importante señalar que el upload de archivos es un proceso crítico que puede dar lugar a errores y agujeros de seguridad. Por ejemplo, si los directorios destino están protegidos contra escritura, nos dará un error. Podemos ver los errores comunes relatados en el sitio de PHP (<http://www.php.net>).

### **CLASE No. 17**

#### **ENVÍO DE CORREO ELECTRÓNICO CON PHP**

Enviar un e-mail con PHP es muy sencillo, tan solo tenemos que utilizar la función *mail()*. Pero cuando escribimos código HTML en el cuerpo del mensaje, este lo recibimos como texto y no como una página Web, como querríamos. Esto tiene fácil solución, solo necesitamos añadir la cabecera "Content-type: text/html" en el e-mail y el código que enviemos se interpretará como HTML. Veamos con un ejemplo:

#### **Ejemplo**

Crearemos un sistema para el envío de correo electrónico desde una página web.

#### **Ejemplo**

```
<?php  
$codigohtml = '
```

```
<html>
<head>
<title>E-Mail HTML</title>
</head>
<body>
<a href="http://www.innovasistemas.org">Innovasistemas</a>
</body>';
$email = 'micorreo@nose.tal';
$asunto = 'E-Mail HTML';
$cabeceras = "Content-type: text/html\r\n";
mail($email,$asunto,$codigohtml,$cabeceras);
?>
```

De esta forma, los e-mails que enviemos se verán como una página Web. En las cabeceras podemos añadir otras cosas, como por ejemplo si queremos especificar quien envía el e-mail haremos:

```
$cabeceras = "From: direccion@email.dom\r\nContent-type: text/html\r\n";
```

#### **Nota**

Cuando publique (ejecute) esta página localmente en Windows (y quizá en otros sistemas operativos), es posible que tenga que configurar un servidor SMTP, como Mercury Mail (que viene con XAMPP) u otro, para poder enviar correos. En la red encontrará documentos que le ayudarán al respecto. A continuación se proporciona una forma de configurar Mercury Mail para Windows.

### **Configurar Mercury Mail/32 de XAMPP para Windows para enviar correos externos**

Con los siguientes pasos configuraremos Mercury Mail/32 que viene con XAMPP para Windows para enviar correos electrónicos (e-mails) a correos externos desde el servidor local. Es necesario disponer de una cuenta en Gmail.

1. Iniciemos el Mercury/32 desde el panel de control de XAMPP y presionamos clic en el botón *Admin*. Se iniciará el panel de control de Mercury/32.
2. Vamos a la opción *Configuration/Protocol Modules* y desactivamos “*MercuryB HTTP web server*” y “*Mercury IMAP4rev1 server*”. Para mandar e-mails a correos externos desactivamos “*MercuryE SMTP end-to-end delivery client*”, y, en cambio, activamos “*MercuryC SMTP relaying client*”. Damos clic al botón *Ok* y reiniciamos Mercury.
3. Volvemos a la consola de Mercury y vamos a *Configuration/Mercury core module*, y nos ubicamos en la pestaña *General*. En “*Internet name for this system*” ponemos el dominio que tenemos en nuestro servidor, ya sea *localhost*, u otro. Los otros campos están ya configurados, sólo tenemos que desactivar todos las casillas (*checkbox*) de abajo menos “*Send copies of all errors to postmaster*”. Vamos a la pestaña “*Local domains*” y añadimos el *Internet name* (para la mayoría de usuarios, que cuentan con un servidor *localhost*, ya estará añadido). Damos clic en el botón *Ok*.
4. Vamos ahora a configurar el *SMTP* para los e-mails salientes en la opción *Configuration/MercuryS SMTP Server*. En la pestaña *General*, en el campo “*Announce myself as*”

ponemos el nombre que nos salga: "XX SMTP", o el que se desee. Comprobamos que el *TCP/IP port* está a 25, que es el del SMTP. En "*IP interface to use*" ponemos 127.0.0.1. Ahora limitaremos el acceso al servidor a sólo la máquina local de la siguiente forma: En la pestaña *Connection control* damos clic al botón *Add restriction* y ponemos 127.0.0.1 to 127.0.0.1. Comprobamos que está activos *Allow Connection* y dejamos todos las casillas (checkbox) desactivados. En la pestaña *Connection Control* desactivamos *Do not Permit SMTP relaying to non-local mail*. Y damos clic en el botón *OK*.

5. Configuraremos el *POP3* de Mercury en *Configuration/MercuryP POP3 Server*. En la pestaña *General* comprobamos el que *TCP port* es 110 y la "*IP interface to use*" es 127.0.0.1. Vamos a *Connection Control* y añadimos la misma restricción que en el anterior punto, sólo para nuestra máquina local de la misma forma. Damos clic al botón *Ok*.

6. Hay que configurar el cliente del SMTP de Mercury en *Configuration/MercuryC SMTP Client*. Para mandar e-mails al exterior necesitamos los datos de un correo exterior. Utilizaremos los datos de una cuenta de *gmail* del SMTP para correos salientes. En "*Smart host name*" ponemos *smtp.gmail.com*. Para el puerto elegiremos el 587. Luego elegimos *STARTTLS* que es lo que soporta *gmail*. En "*Login username*" ponemos la cuenta de correo *gmail*, y en "*Password*" la contraseña respectiva. Damos clic en el botón *Ok*.

7. *Configuration/Manage local users*: comprobamos que tenemos los usuarios *Admin* y *postmaster* con permisos de administrador.

8. Con Mercury no hace falta configurar mas aspectos, ahora es necesario modificar el archivo *php.ini* que se encuentra en *xamplapachelbin*. Nos dirigimos a *[mail function]* y comprobamos que los siguientes datos están así: *SMTP = localhost*, *smtp\_port = 25* y añadimos la siguiente línea: *sendmail\_from = postmaster@localhost* (o descomentamos la instrucción existente y la cambiamos por estos datos). Guardamos y reiniciamos el servidor web *Apache*.

9. Comprobar el envío de correo: en Mercury, en la opción *File/Send mail message* enviamos un e-mail a un correo externo. Para comprobar desde PHP, creamos un archivo .php con la función *mail()* como se mostró anteriormente.

## CLASE No. 18

### EJERCICIOS PROPUESTOS

A continuación se presentan ejercicios diversos para que desarrolle utilizando lo visto sobre PHP y bases de datos con MySQL. Muchos de los ejemplos resueltos y ejercicios propuestos en la sección DHTML (página 103) pueden también pueden ser desarrollados con PHP y/o combinar ambas tecnologías.

1. Cargar varias noticias de forma aleatoria en una página. Las noticias pueden estar almacenadas en una base de datos
2. Crear un contador en la página de las visitas o recargas realizadas.
3. Crear una clase que dibuje círculos.
4. Muestre el día, fecha y la hora en español en una página. Por ejemplo así: Martes, 16 de Agosto de 2011.



5. Finalizar una sesión de usuario después de cierto tiempo de inactividad.
6. Crear una galería dinámica de imágenes. Estas deben ser subidas por los usuarios al servidor, almacenando su ruta en una base de datos y luego mostradas en una galería, la cual debe ofrecer efectos dinámicos con DHTML, como poder hacer un zoom, entre otras funciones.
7. Crear una paginación donde el usuario pueda desplazarse entre contenidos de una base de datos.
8. Mostrar banners de publicidad en una página aleatoriamente. Estos deben encontrarse almacenados en una base de datos, y mediante efectos DHTML, mostrarse en la página
9. Implemente un log de transacciones en el sistema de usuarios indicando la operación que realiza un usuario conectado, con fecha y hora.
10. Modificar la contraseña de un usuario conectado. El formulario debe pedir contraseña actual, nueva contraseña y confirmación de la nueva contraseña.
11. Cree el sistema de usuarios desarrollado en los ejemplos anteriores utilizando clases.
12. Problemas concretos sobre bases de datos con PHP. En estos debe aplicar también los conceptos sobre sesiones, PHPFileUpload, correo electrónico, manejo de hosting y DHTML. Se recomienda hacer un desarrollo por etapas

**Problema 1.** Sistema para el manejo de cuentas bancarias: BANCO

Se tienen las siguientes tablas para manejar información bancaria:

cliente (identificacion, nombre, direccion, telefono, correo, sexo, estado\_civil)

cuenta (numero\_cuenta, fecha\_apertura, sucursal\_ciudad, tipo, saldo, identificacion)

transacción (numero\_transaccion, tipo, fecha\_hora, valor, sucursal\_ciudad, numero\_cuenta)

**Reglas y observaciones:**

- Un cliente puede tener una o varias cuentas y una cuenta pertenece a un solo cliente
- En una cuenta puede haber una o muchas transacciones y una transacción pertenece a una cuenta
- Las claves principales están indicadas por los campos que se encuentran subrayados y las foráneas por los que se encuentran con doble subrayado
- Defina apropiadamente cada campo y cree las respectivas relaciones entre las tablas
- Ingrese datos de prueba en cada tabla

**Problema 2.** Sistema para el manejo de estudiantes: MATRÍCULAS

Se tienen las siguientes tablas para manejar información de registro estudiantil: matrículas y notas:

estudiante (identificacion, nombre, direccion, telefono, correo, sexo, estado\_civil)

asignatura (codigo\_asignatura, nombre, intensidad\_horaria, objetivo, nivel)

calificacion (consecutivo\_calificacion, identificacion, codigo\_asignatura, fecha\_informe, nota)

**Reglas y observaciones:**

- Un estudiante puede ver una o varias asignaturas y una asignatura puede ser vista por uno o varios estudiantes
- La tabla “calificacion” soluciona el inconveniente de la relación entre las tablas “estudiante” y “asignatura”: un estudiante obtiene una o varias calificaciones; en una asignatura se generan una o varias notas. Una calificación pertenece a un estudiante de alguna asignatura

- Las claves principales están indicadas por los campos que se encuentran subrayados y las foráneas por los que se encuentran con doble subrayado
- Defina apropiadamente cada campo y cree las respectivas relaciones entre las tablas
- Ingrese datos de prueba en cada tabla

**Problema 3.** Sistema para el manejo de sitios turísticos: TURISMO

Se tienen las siguientes tablas para manejar información relacionada con países y destinos turísticos (en este problema puede incluir otra tabla para almacenar imágenes acordes a los destinos turísticos):

país (codigo\_pais, nombre, continente, idioma, moneda, capital)

departamento (codigo\_departamento, nombre, capital, sitios\_interes, geografia, codigo\_pais)

ciudad (codigo\_ciudad, nombre, sitios\_interes, clima, es\_capital, codigo\_departamento)

**Reglas y observaciones:**

- Un país puede tener una o varios departamentos y un departamento pertenece a un solo país
- En un departamento puede haber una o muchas ciudades y una ciudad pertenece a un departamento
- Las claves principales están indicadas por los campos que se encuentran subrayados y las foráneas por los que se encuentran con doble subrayado
- Defina apropiadamente cada campo y cree las respectivas relaciones entre las tablas
- Ingrese datos de prueba en cada tabla

**Problema 4.** Sistema para el manejo de pagos: PRÉSTAMOS

Se tienen las siguientes tablas para manejar información de préstamos:

cliente (identificacion, nombre, direccion, telefono, correo, sexo, estado\_civil)

prestamo (numero\_prestamo, detalle, fecha\_prestamo, fecha\_plazo, valor\_prestamo, saldo\_actual, identificacion)

abono (numero\_abono, fecha\_hora, valor, numero\_prestamo)

**Reglas y observaciones:**

- Un cliente puede realizar uno o varios préstamos y un préstamo es para un solo cliente
- En un préstamo puede haber uno muchos abonos y un abono se realiza para un préstamo
- Las claves principales están indicadas por los campos que se encuentran subrayados y las foráneas por los que se encuentran con doble subrayado
- Defina apropiadamente cada campo y cree las respectivas relaciones entre las tablas
- Ingrese datos de prueba en cada tabla

**Problema 5.** Sistema para el manejo de foros sencillos: FOROS

Se tienen las siguientes tablas para manejar información de foros:

usuario (identificacion, nombre, direccion, telefono, correo, sexo, estado\_civil, foto)

tema (codigo\_tema, detalle, fecha\_publicacion, fecha\_cierre,)

comentario (numero\_comentario, fecha\_hora, descripcion, codigo\_tema, identificacion)

**Reglas y observaciones:**

- Un usuario puede comentar uno o varios temas y un tema puede ser comentado por uno o varios usuarios
- Un tema cerrado no se puede comentar

- Las claves principales están indicadas por los campos que se encuentran subrayados y las foráneas por los que se encuentran con doble subrayado
- Defina apropiadamente cada campo y cree las respectivas relaciones entre las tablas
- Ingrese datos de prueba en cada tabla

**Problema 6.** Sistema para el manejo de facturación: FACTURAS

Se tienen las siguientes tablas para manejar información de facturas:

cliente (identificacion, nombre, direccion, telefono, correo, sexo, estado\_civil, foto)

producto (codigo\_producto, detalle, precio, existencia)

factura (numero\_factura, fecha\_hora, identificacion)

detalle\_factura (numero\_detalle, fecha\_hora, cantidad, valor, codigo\_producto, numero\_factura.)

**Reglas y observaciones:**

- Un cliente puede adquirir uno o varios productos y un producto puede ser adquirido por uno o varios clientes
- No se pueden vender producto cuya existencia sea cero.
- Las claves principales están indicadas por los campos que se encuentran subrayados y las foráneas por los que se encuentran con doble subrayado
- Defina apropiadamente cada campo y cree las respectivas relaciones entre las tablas
- Ingrese datos de prueba en cada tabla

**Notas (para todos los problemas de bases datos)**

- Aplique validación de datos con Javascript.
- Aplique formato al sitio utilizando estilos CSS.
- Cree las secciones de Administrador (back-end) y de usuario (front-end). Realice las operaciones de SQL apropiada para páginas de usuario, de administración y para informes apropiados en cada sistema.
- Valide la integridad de los datos.
- El sitio debe quedar montado en un hosting gratuito. Debe consultar como realizar esto (recuerde que el hosting debe tener soporte para PHP)<sup>12</sup>.
- Cada sitio debe tener implementado el envío de correo electrónico cuando se encuentre en el área de administración, ya sea para enviar correos individuales o correo masivo (debe utilizar la función mail() explicada en la Guía Académica y consultar como configurar el servidor SMTP Mercury Mail que viene con XAMPP u otro sistema similar para realizar dicha tarea en Windows).
- Todos los sistemas deben tener implementado el módulo de usuarios incluyendo el inicio de sesión (login) para realizar tareas de administración.
- El sitio debe permitir que el usuario suba su imagen una vez inicie sesión; la imagen y el nombre de usuario conectado deben mostrarse cuando éste ingrese al sistema. (Para subir imágenes, debe implementar el ejemplo que se encuentra en la Guía Académica, en la sección PHPFileUploads).

---

<sup>12</sup> Algunos hosting gratis que ofrecen alojamiento de páginas PHP y bases de datos MySQL son: <http://www.260mb.com>, <http://www.110mb.com>, <http://www.zobyhost.com>, <http://www.freewebhostingarea.com>, <http://www.miarroba.com> y <http://www.000webhost.com> entre otros.

---

**BIBLIOGRAFÍA Y CIBERGRAFÍA**

**TEXTOS**

MONTOYA MONTOYA, Jaime Eduardo. Guía Académica Programación para la Web. Medellín: Cesde, Semestre 1-2012. 195p.

GUTIÉRREZ, Abraham; BRAVO, Ginés. PHP 5 a través de ejemplos. México D.F.: Alfaomega RA-MA, 2005.

LÓPEZ QUIJADO, José. Domine PHP MySQL: programación dinámica en el lado del servidor. México, Alfaomega RA-MA, 2007

GIL RUBIO, Francisco Javier; TEJEDOR CERBEL; Jorge A. y YAGUE PANADERO, Agustín. Creación de sitios web con PHP 4. Medelin: McGraw-Hill, 2001. 547 p. 1 Cd-rom.

ZAKAS, Nicholas C. JavaScript para desarrolladores web. Madrid: Anaya Multimedia, 2005.

DANESH, Arman; DÍAZ MENA, Jorge Iván y GARCÍA, María Margarita. Aprendiendo Javascript en una semana. España: Prentice Hall Hispanoamericana, s.a., 1996. 543 p.

FRENTZEN, Jeff. Superutilidades para Javascript. Santafé de Bogotá: McGraw-Hill, c1999. 1 Cd-rom.

KORTH, Henry F.; SILBERSCHATZ, Abraham. Fundamentos de Bases de Datos. Madrid: McGraw-Hill, Segunda edición, 1992.

**ENLACES EN INTERNET**

Manual de PHP. En internet: <http://www.php.net/manual/es/index.php>

MySQL 5.0 Reference Manual. En internet: <http://dev.mysql.com/doc/refman/5.0/es/index.html>

Desarrollo Web.com. Tú mejor ayuda para aprender a hacer webs. En internet: <http://www.desarrolloweb.com>

Wikipedia, la enciclopedia libre. PHP. En internet: <http://es.wikipedia.org/wiki/PHP>

Wikipedia, la enciclopedia libre. MySQL. En internet: <http://es.wikipedia.org/wiki/MySQL>

Wikipedia, la enciclopedia libre. HTML. En internet: <http://es.wikipedia.org/wiki/HTML>

Wikipedia, la enciclopedia libre. DHTML. En internet: <http://es.wikipedia.org/wiki/DHTML>

Wikipedia, la enciclopedia libre. Tim Barners-Lee. En internet: [http://es.wikipedia.org/wiki/Tim\\_Barners-Lee](http://es.wikipedia.org/wiki/Tim_Barners-Lee)

Wikipedia, la enciclopedia libre. En internet: <http://es.wikipedia.org/wiki/ICANN>

Wikipedia, la enciclopedia libre. Internet. En internet: <http://es.wikipedia.org/wiki/Internet>

Wikipedia, la enciclopedia libre. World Wide Web. En internet: [http://es.wikipedia.org/wiki/World\\_Wide\\_Web](http://es.wikipedia.org/wiki/World_Wide_Web)

WebEstilo. Usabilidad, programación y mucho más. En internet: <http://www.webestilo.com>

WebTaller. El portal del Webmaster. En internet: <http://www.webtaller.com>

Programación Web. Programación y diseño de páginas web. En internet: <http://www.programacionweb.net>

Programación Fácil. En internet: <http://www.programacionfacil.com>

Tejedores del Web. En internet: <http://www.tejedoresdelweb.com>

Programatium. Comunidad de programación por y para programadores. En internet: <http://www.programatium.com>

Programación y diseño web. Todo sobre programación web & más. En internet: <http://www.todo-programacion.com.ar/>

La Web del Programador: Comunidad de Programadores. En internet: <http://www.lawebdelprogramador.com>

Manual de PHP. En internet: <http://www.manualdephp.es>

Manual de PHP. En internet: <http://www.manualdephp.com>

El Guru Programador: Actualidad, Programación y Desarrollo Web. En internet: <http://www.elguruprogramador.com.ar>

Tutorial de PHP. En internet: <http://flanagan.ugr.es/php/>

Manual de PHP 4. En internet: <http://www.laigu.net/ManualPHP.html>

Tutorial de PHP. En internet: <http://www.tutorialesfacil.com.ar/php/>

Online Language Dictionaries. En internet: <http://www.wordreference.com/es/index.htm>

Estadísticas históricas y de uso diario. En internet: <http://news.netcraft.com>

Wikipedia, la enciclopedia libre. Correo Yahoo! En internet: [http://es.wikipedia.org/wiki/Correo\\_Yahoo!](http://es.wikipedia.org/wiki/Correo_Yahoo!)

Wikipedia, la enciclopedia libre. GNU. En internet: <http://es.wikipedia.org/wiki/GNU>

Wikipedia, la enciclopedia libre. GNU - Licencia Pública General. En internet:  
[http://es.wikipedia.org/wiki/GNU\\_General\\_Public\\_License](http://es.wikipedia.org/wiki/GNU_General_Public_License)

Wikipedia, la enciclopedia libre. W3C. En internet: <http://www.w3.org/>

W3C. En internet: <http://www.w3c.org/> (W3 España)

W3C. En internet: <http://validator.w3.org/>

Wikipedia, la enciclopedia libre. World Wide Web Consortium. En internet:  
[http://es.wikipedia.org/wiki/World\\_Wide\\_Web\\_Consortium](http://es.wikipedia.org/wiki/World_Wide_Web_Consortium)

Wikipedia, la enciclopedia libre. WYSIWYG. En internet: <http://es.wikipedia.org/wiki/WYSIWYG>

En internet: <http://www.htmlquick.com/es/tutorials/forms.html>

En internet: <http://es.kioskea.net/contents/html/htmlform.php3>

En internet: <http://webusable.com/CharactersTable.htm>

En internet: <http://webusable.com/CharsExtendedTable.htm>

En internet: <http://html.conclase.net/w3c/html401-es/struct/lists.html>

En internet: <http://goliatenterrado.es/2009/03/03/configurar-el-mercury32-del-xampp-para-enviar-correos-externos/>

En internet: <http://www.hackingballz.com/herramientas/manual-oficial-de-php/function.date.html>

En internet: <http://www.tuxi.com.ar/2007/06/18/javascript-validar-e-mail-con-expresiones-regulares/>