Convenciones:

* **Las referencias van entre corchetes y en una barra verde a la derecha**
* **El código fuente debe ir formateado y coloreado y con fuente monoespaciada**
* **Notas aparecen rodeada de dos líneas con letras azules**
* **Todas las clases tienen nombres con la primera en mayúsculas**
* **El nombre de las funciones o métodos siguen la regla de estilo camelCase a excepción los de los controladores**
* **Las observaciones dela vida real están en plomo y empiezan con la palabra Observacion**

**Temario.**

**1. Capítulo I**

En este capítulo he tratado de dar una introducción a los temas que sientan las bases para iniciar el desarrollo de CMS, he tratado de que sea lo más claro posible con ejemplos prácticos y completamente válidos, gran parte de este capítulo está dedicada meramente a los conceptos, espero sean de utilidad los conceptos.

**1.1. Definición Y Origen De Internet**

Antes de empezar a desarrollar el trabajo voy a citar la definición que Wikipedia da a la palabra Internet:

*[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….][1]*

En conclusión podríamos decir que todos los equipos que tiene acceso a Internet son parte de él, la forma que tiene de comunicarse con otros dispositivos en esta gran red es a través de protocolos el más usado es el WWW, también existe otro protocolo el cual es el más usado en la actualidad y es el HTTP, ambos protocolos realizar transferencia de HIPERTEXTO entre dos equipos o dispositivos, veamos las definiciones de los términos más importantes cuando se habla de Internet:

**SERVIDOR**

Equipo encargado de atender peticiones web y responderlas enviando Información a su solicitante a través de protocolos de comunicación.

**CLIENTE**

Equipo o Dispositivo con capacidad de interactuar con un servidor, enviando y recibiendo información con la ayuda de protocolos de comunicación, esta interacción es posible gracias a los navegadores web.

**NAVEGADOR WEB**

Es una aplicación que funciona en el dispositivo del cliente, como dice su nombre es un navegador, utiliza la red de Internet para encontrar contenidos y comunicarse con equipos remotos.La funcionalidad básica del navegador es permitir la visualización de contenidos.

**WWW**

Es una serie de protocolos que permiten él envió de información de un equipo a otro, fue el primer estándar creado para la transmisión de datos entre equipos, se necesita de un navegador web para que este muestre los contenidos de forma gráfica.

**HTTP**

Al ser estas definiciones técnicas citaré a una fuente muy fiable como es la gran biblioteca Wikipedia, la misma que dice:…

*[Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Al cliente que efectúa la petición (un navegador web o un spider) se lo conoce como "useragent" (agente del usuario). A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL). Los recursos pueden ser archivos, el resultado de la ejecución de un programa, una consulta a una base de datos, la traducción automática de un documento, etc.*

*HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de "sesión", y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado…] [2]*

Tal vez la definición este un poco enredada, pero podemos concluir diciendo que gracias a este protocolo los computadores se pueden conectar unos con otros compartiendo información, una pequeña desventaja es que no puede guardar el estado de la conexión pero para solucionar este inconveniente se crearon unos archivos que se almacenan en la máquina del cliente llamados cookies o galletas en español, la información que contienen las cookies es enviada dentro de las cabeceras HTTP a esto se le dio el nombre de sesión.

**HIPERTEXTO**

Es el texto que podemos ver en la pantalla de cualquier dispositivo electrónico. Un hipertexto consta de los siguientes elementos: nodos o secciones, enlaces o hipervínculos y anclajes.

Los nodos son las partes del hipertexto que contienen información accesible para el usuario. Los enlaces son las uniones o vínculos que se establecen entre nodos y facilitan la lectura secuencial o no secuencial por los nodos del documento. Los anclajes son los puntos de activación de los enlaces.

* + 1. **Orígenes**

EL origen de Internet se remota a los años 1960, cuando los computadores estaban aún en su “infancia” los mismos que no podían comunicarse con otros equipos ni se podía trabajar de manera colaborativa dentro de una organización, vista esta necesidad nace la primera red llamada ARPAnet(AdvancedResearchProjects Agency Network o Red de la Agencia para los Proyectos de Investigación Avanzada de los Estados Unidos), que nos legó el trazado de una red inicial de comunicaciones de alta velocidad a la cual fueron integrándose otras instituciones gubernamentales y redes académicas durante los años 70 [3].

* + 1. **Internet En La Actualidad**

Internet es hoy una necesidad más que un lujo, es una herramienta que nos ayuda a realizar todas las tareas ya sean productivas o de ocio, su difusión es tan grande que se lo usa para todo, esto es porque estamos en una era donde las comunicaciones son lo más importante, y el internet es uno de los medios más grandes y el más usado para llevarlas a acabo, si pensamos en las actividades que realizamos a diario estoy seguro que en la mayoría el internet está presente, ya sea para estudiar, para trabajar, pero si es una persona que no trabaja en una oficina o simplemente no usa un computador no significa que no use internet, el uso de internet no es solamente a través del computador o celular, cuantas veces vamos al banco o mejor aún quien no ha retirado dinero de un cajero electrónico, este dispositivo es un terminal o un cliente del servidor del banco y está conectado a través de internet.

Veamos otro ejemplo, para este ejemplo vamos a pensar en el gobierno antes todos los tramites se los realizaba de manera presencial en la institución, ahora en la mayoría de ellas muchos de sus trámites se los realiza por internet.

En realidad hay muchas cosas que llegan a nosotros a través del Internet y sin la necesidad de usar un computador. Lo que intento decir es que el Internet es un recurso al que le debemos dar su importancia y usarlo para nuestro beneficio, sacando el mayor provecho de él.

* 1. **Definición De Sitios Web, Su Estructura E Importancia**

Un sitio web es un conjunto de páginas web alojadas en un servidor web, por lo general las páginas del sitio tienen contenidos relacionados a un tema en particular o a un propósito, los contenidos están conectados a través de hiperenlaces que son los que guían al navegador para mostrarlos al usuario, a menudo se confunde sitio web con página web, la pagina no es más que un archivo con código HTML el cual es interpretado por el navegador y mostrado en forma clara al usuario, para dejar más claro el concepto veamos un ejemplo:

[www.google.com](http://www.google.com) es un sitio web el cual contiene una gran cantidad de páginas, si ingresamos al sitio nos muestra la página principal que es un formulario para introducir un texto, el cual nos ayuda a realizar búsquedas de algo en Internet.

Un sitio tiene los siguientes elementos:

* Dirección IP única, la de Google.com es 74.125.130.99
* Nombre de Dominio asignado a la IP www.google.com (dirección web)
* Estructura de árbol o jerárquica: A partir de una página de bienvenida o portal (raíz) se abren unas secciones (ramas) que a su vez contienen múltiples páginas web (hojas)
* Estructura lineal: a partir de una página de bienvenida o portal se suceden las siguientes páginas una tras otra como si se tratara de un libro.
* Estructura en red: Las páginas que forman el sitio web se enlazan unas con otras según sus contenidos en una especie de red en la que no se aprecia ningún tipo de jerarquía.

Existen dos tipos de Sitios Web:

**Sitios Estáticos**

Son sitios incapaces de interactuar con el usuario, sus contenidos se mantienen a menos que una persona con conocimientos de HTML los cambie manualmente, podemos decir que no tienen implementado un gestor de contenidos, en la actualidad existen muy pocos de estos sitios, aunque es frecuente encontrarlos en sitios web de empresas pequeñas, esto debido al costo que tiene implementar un sitio dinámico.

**Sitios Dinámicos**

Estos sitios aparecen gracias a la creación de programas informáticos desarrollados para el manejo de información del lado del servidor, antes del aparecimiento de estos lenguajes de programación, las páginas del sitio tenían que ser generadas manualmente por un webmaster, pero ahora gracias a la creación de los lenguajes de programación del lado del servidor, es posible crear una sola plantilla para mostrar los datos, la misma que está basada en HTML o XHTML dependiendo las necesidades del sitio, almacenar los datos en un sistema gestor de bases de datos o incluso en ficheros de texto plano (sin formato), con esta tecnología ya no es necesario cambiar a una página web desde su estructura HTML sino que se puede cambiar directamente los contenidos a través del sistema de administración de sitio, mismo que es capaz de realizar las tareas mecánicas necesarias para mostrar la información al cliente, brindando al sitio la capacidad de interactuar con los usuarios, esto ha dado origen a un sinfín de aplicaciones como son blogs, foros, tiendas online, salas de chat, etc.

Podemos concluir diciendo que en un sistema dinámico, las páginas web no existen realmente porque son el resultado del procesamiento de la petición del usuario, el cual fue formado luego de que el servidor buscara la información y le diera forma para que el navegador del usuario la reciba, pero esa instancia de la página no existe es solo el resultado del algoritmo asociado a esa petición, esta es la principal diferencia entre sitios estáticos y dinámicos en los estáticos la página que estamos viendo si existe y es exactamente como el navegador nos la muestra.

**Importancia de los Sitios WEB**

La importancia de los sitios web está en lo que las empresas y personas pueden hacer con ellos, existes sitios en los que se pueden realizar comercio Mercado Libre es un gran ejemplo en Latinoamérica, buscadores como Google, Yahoo, Bing, etc. Tenemos también los servicios de correo electrónico, servicios de comunicaciones, redes sociales. Todos estos tienen un sitio en internet cuentan con un nombre único y una página principal que en la que vemos cuando nos conectamos al sitio.

**1.2.1 Página Web**

Revisando la definición anteriorpodemos decir que página web es la unidad básica de un sitio web, es la encargada de mostrar los contenidos a los usuarios que la solicitan, estas no son más que archivos en formato HTML o XHTML los cuales son interpretados por el navegador del cliente y luego mostrados de manera clara al usuario, la página web es la única forma en la que es usuario entra en contacto con el servidor.

* 1. **Introducción A los Frameworks**
     1. **Introducción**

Los Frameworks son ambientes de trabajo y ejecución con un conjunto de bibliotecas orientadas a la reutilización de componentes de software para el desarrollo rápido de aplicaciones, la palabra Framework traducida al español significa Marco de Trabajo, el principal objetivo de estas herramientas es ayudar al desarrollador a identificar los requerimientos y prestaciones del software, que tratando con los tediosos detalles de bajo nivel, sientan una base para poder comenzar a trabajar en lugar de crearla nosotros mismos, por ende la persona que los utiliza puede generar más con menostrabajo y en tiempos más cortos, podemos decir que un Framework es un esqueleto o esquema del sistema en donde nosotros vamos implementando lo que necesitemos.

Un Framework es más que una herramienta que permite desarrollar un proyecto fácilmente, es un conjunto de elementos que ayudan a nuestra aplicación a ser robusta y compatible, ya que estos manejan todo lo que tiene que ver con la seguridad y compatibilidad, las aplicaciones desarrolladas con un Framework no tienen problemas de compatibilidad de un sistema operativo a otro (dependiendo del Framework que se use), por ejemplo una aplicación realizada en .Net que es el Framework de desarrollo de Microsoft, funciona de manera idéntica en cualquier sistema operativo Windowssiempreque tenga instalado el Framework. Veamos un ejemplo diferente, sise habla de una aplicación escrita en Java o Python las mismas pueden funcionar en más de un sistema operativo ya que el Framework es desarrollado para varios tipos de plataformas y equipos, lo que significa que una aplicación escrita es estos lenguajes puede funcionar de manera idéntica en Windows Linux MacUnix, etc.

Continuando con lo bueno de estas herramientas, podemos hablar de un beneficio más allá del código y del producto final, cuando un desarrollador o diseñador empieza a trabajar, toda la estructura está a su cargo, lo que quiere decir que son ellos los encargados de generar la estructura básica del sistema, la pueden crear como mejor les parezca de acuerdo a su criterio profesional, la desventaja de esto es que la mayoría de estas personas sobre todo los principiantes no conocen de patrones de diseño y de técnicas que hacen que una estructura sea funcional y que pueda crecer a futuro, los Frameworks ayudan a estas personas a ser organizadasporque les ayuda a entender la estructura de un sistema, nos enseña a dividir un problema en pequeñas partes fáciles de manejar, esta para mi es una de las ventajas más importantes que tienen los Frameworks.

La estructura de los Frameworks no es general, no existe un modelo genérico para un Framework esto depende de la entidad encargada de su desarrollo, a continuación vamos a revisar una imagen de la estructura de Framework principalque se van a usar en este proyecto.

CodeIgniter creado por ellislab.com [4].



Como podemos observar los contenidos son mostrados y solicitados desde una página principal llamada index.php los mismos que pasan por filtros de seguridad y optimización antes de llegar al controlador de la aplicación que sería el corazón del Framework y esta a su vez usa a los componentes creados por el desarrollador como son los controladores librerías etc., para responder las peticiones del cliente.

Después de tanto hablar de lo bueno que tienen los Frameworks ha llegado la hora de hablar de los aspectos negativos que tienen.

Una de las principales desventajas, es que la aplicación desarrollada con la ayuda de un Framework es dependiente de este, ya que como dijimos anteriormente un Framework brinda componentes para facilitar el desarrollo, lo que quiere decir que la aplicación usa esos componentes, por lo tanto hay que escribirla casi por completo, más de una persona puede decir. **¿Para qué cambiar de Framework?** La respuesta es sencilla,elFramework puede estar limitando la aplicación, puede ser que no nos deje desarrollar algún componente porque no fue diseñado para ello, aunque esto sucede rara vez, lo que si sucede es que el Framework consume recursos ya que controla la aplicación y la hace funcionar, para hacer que esto cambie el desarrollador puede generar la misma aplicación en un Framework que sea más óptimo, la forma de evitar esto es elegir a un Framework con el que se pueda trabajar a diferentes escalas.

Otra desventaja es que el desarrollador no se vale por sí mismo para resolver algunos problemas sobre todo de compatibilidad, lo que intento decir es que más de una ventaja se convierte en desventaja para el crecimiento del desarrollador, si la vemos desde un punto de vista crítico.

* + 1. **Historia**

Creo que si algo está claro es la razón por la que fueron creados, por lo que no ahondaremos en el tema, lo único que voy a decir es que los Frameworks funcionan y es esa la razón por la que existen, a continuación voy a mostrar un gráfico en el que se muestran la creación y evolución de algunos Frameworks a los largo del tiempo [5].



* + 1. **Tipos De Frameworks**

**Frameworks de Caja Blanca:**

La instanciación del Framework es posible a través de la creación de nuevas clases. Estas clases y el códigocorrespondiente se pueden introducir por herencia o composición. Se agregannuevas funcionalidades creando una subclase de una clase que ya existe en el Framework. Para usar Frameworks de caja blanca el desarrollador de aplicaciones debe conocer muy bien cómo funciona el Framework, este es el tipo de Framework que vamos a usar.

**Frameworks de Caja Negra:**

Producen instancias usando scripts de configuración del Framework, con los cuales se configura la aplicación final. Tienen la ventaja que no se requiere que el desarrollador de aplicacionesconozca los detalles internos del Framework, por lo cual son mucho más fácilesde usar.

**Frameworks de Caja Gris:**

La mayoría de los Frameworks son de Caja Gris,que son aquellos que contienen elementos de Caja Blanca y Caja Negra, yalgunas partes se implementan vía herencia o composición, y otras a travésde configuración de parámetros [6].

**1.3.4. ¿Por qué usar un Framework?**

Por qué esta en el motivo que impulso al ser humano a crearlos, gracias a la ayuda de estos podemos desarrollar aplicaciones de manera más eficiente y rápida, mi intención no es sobrevalorar a los Frameworks, sino de mostrar su valor real a la hora de llevar a cabo el desarrollo de un proyecto, los Frameworks nos ahorran trabajo que más que ser productivo es tedioso, que nos toma demasiadotiempo y esfuerzo, voy a listar una serie de las tareas que hacen por nosotros.

* Manejan la seguridad de la aplicación
* Administran eficientemente una base de datos
* Desarrollo con poco código
* Mejores tiempos de desarrollo
* Evita tareas repetitivas y básicas
* Manejan la interfaz de usuario
* Brindan métodos y funciones generales
* Se aprende patrones de diseño y estilos de programación
* Los productos son más eficientes

**1.4. Introducción A La Programación Orientada A Objetos (POO)**

La programación orientada a objetos (POO o OOP por sus siglas en inglés) es un paradigma es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se modelan a través de clases y objetos.La gracia de la POO es que hace que los problemas sean más sencillos, al permitir dividir el problema en objetos, de manera que cada objeto funciona de forma independiente, siendo este una parte del programa que posee sus datos y un funcionamiento propio, es decir un objeto está formado por datos a los que llamaremos propiedades y métodos los que también son llamados procedimientos, cuyas definiciones veremos más adelante.

Los conceptos de la programación orientada a objetos tienen origen en un lenguaje usado para hacer simulaciones de naves llamado Simula 67, creado por Ole-Johan Dahl y Kristen Nygaard, del Centro de Cómputo Noruego en Oslo. La idea surgió al agrupar los diversos tipos de naves en diversas clases de objetos, siendo responsable cada clase de objetos de definir sus propios datos y comportamientos. Fueron refinados más tarde en Smalltalk, diseñado para ser un sistema completamente dinámico en el cual los objetos se podrían crear y modificar "sobre la marcha" (en tiempo de ejecución) en lugar de tener un sistema basado en programas estáticos.

Con la aparición de la POO la informática hace grandes avances rediseñando el lenguaje de programación C para soportar objetos creando así C++, el cual fue aceptado y domino a los demás lenguajes gracias al surgimiento de las interfaces graficas de usuario, para las cuales la programación orientada a objetos está particularmente bien adaptada. En este caso, se habla también de programación dirigida por eventos como lo podemos apreciar en entornos de desarrollo de Microsoft como Visual Studio donde se puede generar aplicaciones dirigidas por eventos.

En nuestro caso es lenguaje que nos interesa es PHP el cual en su versión 5 se ha modificado; soporta una orientación completa a objetos, cumpliendo todas las características propias de la orientación a objetos, gracias a estas características podemos hablar de crear una aplicación con objetos los mismos que veremos más adelante[7].

**1.4.1. Elementos De La POO**

Hay más de un factor que determina la POOlos cuales la identifican de otros paradigmas revisemos los principales:

**Abstracción**

Aísla a un elemento de su contexto, define las características esenciales de un objeto incluyendo su comportamiento, en el sistema sirve como modelo abstracto que puede realizar trabajo,cambiar su estado y “comunicarse” con otros elementos sin mostrar sus características, con esto se quiere decir que es la vitalización de un objeto físico, luego de haberlo analizado,.

El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades del mundo real.

La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere resolver.

**Encapsulamiento**

Reúne al mismo nivel de abstracción a todos los elementos que pueden considerase pertenecientes a una misma entidad, la encapsulación es la creación del objeto, el cual se obtuvo de la abstracción del objeto real con sus propiedades y métodos.

**Modularidad**

Característica que permite dividir a una aplicación en partes pequeñas o varios módulos, las mismas que trabajan de forma independiente pero de manera colaborativa.

**Ocultación (Aislamiento)**

Las propiedades y métodos de los objetos están ocultas y son protegidas del exterior para que no sean modificadas por quienes no tengan acceso a ellas, los objetos exponen una interfaz a otros objetos de la misma clase en la que se especifica cómo se puede interactuar con él, el principio de ocultación también menciona que solamente los métodos propios del objeto pueden acceder a sus propiedades permitiéndoles modificarlas o publicarlas.

**Polimorfismo**

Es la capacidad que da a diferentes objetos del mismo tipo o instancias de la misma clase, la posibilidad de contar con métodos y propiedades con el mismo nombre sin que ello genere problemas entre un objeto y otro igual.

**Herencia**

Es una relación que existe entre dos o más clases, se denomina objeto padre al que posee los métodos y propiedades comunes, se les denomina hijos a los objetos que heredan dichos métodos y propiedades de las clases a las que pertenecen, las clases no están aisladas, sino que se relacionan entre si formando una jerarquía de clasificación, permitiendo compartir y extender las características de los objetos padres, la herencia ayuda mucho a la hora de implementar métodos generales para una serie de objetos, pongo como ejemplo la clase vehículo que tendría métodos como frenar y acelerar, un objeto bicicleta que hereda de vehículo esos métodos y además tenga los suyos propios como pedalear, que sería una implementación de método acelerar.

**Recolección de Basura**

Esta es una técnica que elimina de la memoria a todos los objetos que ya no se están utilizando, los lenguajes de programación tienen esta característica para liberar la memoria.

**1.4.2. Objeto**

Es una entidad que posee un conjunto de propiedades o atributos y de comportamientos o funcionalidades conocidos como métodos, estas entidades son una muestra de un objeto del mundo real, a esto se le llama abstracción ya que se toma un objeto del mundo real y se lo lleva a un objeto informático, el objeto es la parte más básica de la POO los objetos pueden estar formados por objetos, es decir los objetos pueden ser parte de estructuras más grande (Herencia) y complejapermitiendo crear aplicaciones más potentes, para concluir esta pequeña introducción definimos las características que todo objeto debe tener para ser funcional:

**Propiedad:** También conocidos como atributos, son variables que contienen datos asociados a un objeto, son las propiedades quienes guardan la información del objeto.

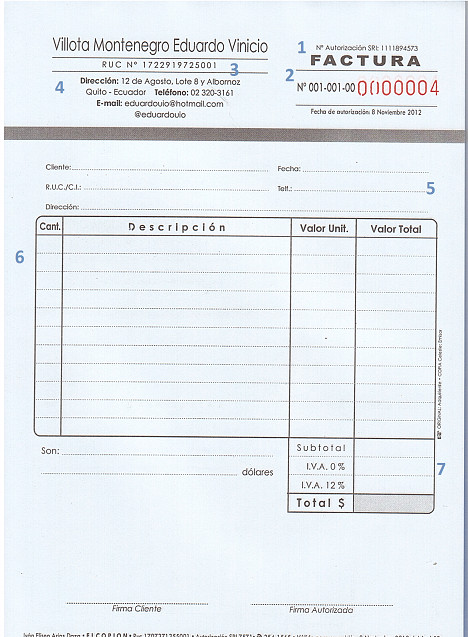
**Método:**Es un algoritmo asociado a un objeto que trabaja con sus propiedades, indica lo que el objeto está en capacidad de hacer.

Dentro de la programación un objeto es una instancia a una clase y la clase es la descripción de un objeto del mundo real, sea este real (una factura) o imaginario (una venta), veamos un ejemplo de objeto real e imaginario tomado de la vida real.

**Ejemplo de Objeto Real.**

Este ejemplo vamos a ver a una Factura como un objeto y lo llamo objeto real porque es algo que podemos tocar ver y documentar, antes de comenzar demos un vistazo a la siguiente imagen y veamos qué es lo que compone una factura.

Me tome la libertad de incluir números en las características principales de la factura el cual voy a revisar y comentar uno a uno.



1. Nombre del documento, este es el nombre que tiene le documento en este caso se llama factura, ya que existen otros tipos de comprobantes como Recibos o Notas de Venta
2. Numero de Factura, este es un número único de factura, es el que identifica a una factura de otra.
3. RUC es un numero único que identifica al emisor de las facturas
4. Información para contactarse con el emisor
5. Información del Cliente
6. Ítems de venta, contiene el bien o servicio comercializado entre el emisor y el cliente
7. Información de subtotales e impuestos

La observación a lo mejor no es profunda pero el objetivo no es confundir, sino aclarar lo que es un objeto, en este caso el objeto que estoyintentadocrear es una factura, es algo que podemos tocar, una factura es una venta o más bien es la representación de una, la venta es el resultado de llevar a cabo una transacción, en la que dos partes intercambian objetos de valores iguales (producto o servicio contra dinero u otro producto o servicio) este evento se registra con un documento llamado factura.

Vale la pena aclarar que más de una propiedad en esta factura puede ser considerado como objeto ya que cuenta con sus propias propiedades y métodos, pero primero vamos a describir al objeto factura, Cada elemento que podemos ver en el documento es una propiedad porque nos muestran información del objeto, los métodos son aquellas acciones que se hacen con la factura los nombres de las acciones la mayoría de veces son verbos terminador en ar, er e ir. El nombre del objeto es Factura

**Resumen de propiedades encontrados al objeto Factura**

* Numero factura
* RUC emisor
* Dirección del emisor
* Teléfono emisor
* Email emisor
* Nombre del cliente
* RUC/CI del cliente
* Fecha de facturación
* Teléfono del cliente
* Ítems de venta

Como podemos ver no es difícil saber lo que contiene cada una de las propiedades.

**Resumen de Métodos encontrados en el objeto Factura**

* Facturar
* Anular Factura
* Cobrar Factura
* Abonar Factura
* Subtotal
* IVA
* Total

Se preguntarán porque el subtotal, total e IVA están como métodos en lugar de propiedades, la respuesta es simple el subtotal es la suma del costo de los ítems de compra, el IVA es el cálculo del 12% del subtotal y el total es la suma del subtotal mas los impuestos, al ser estos datos que se pueden obtener mediante la implementación de un algoritmo no es necesario registrarlos como propiedades, esto es bueno ya que si en la factura, se elimina o aumenta ítems estos valores se cambian de manera automática sin que tengamos que llamar a la propiedad y asignarle un nuevo valor cada vez que se añada o elimine un ítem en la factura.

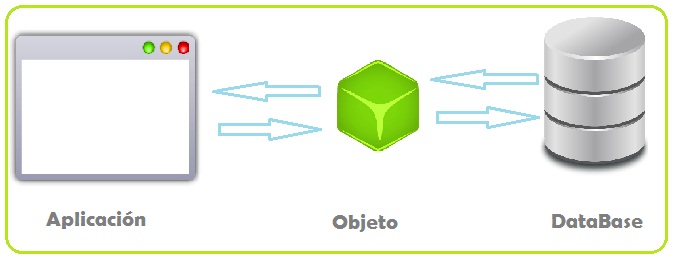
Además se muestra métodos como facturar que es crear una factura nueva, cobrar factura que es marcar a una factura como pagada para esto es necesario crear una propiedad en la que podamos guardar este estado, lo mismo sucede con abonar y anular factura.

Observación: Como se mencionó al principio de este artículo, el objeto factura puede estar formado por más objetos, es decir, cada propiedad puede ser un objeto ya que cuentan con sus propios métodos y propiedades, si tomamos al cliente como objeto ya no tendríamos que llenar la factura con todo los datos del cliente, sino que bastaría con buscar al cliente y asignarlo a una factura esta referenciaría de forma automática los datos del cliente a la factura, sucede exactamente lo mismo con los ítems de la factura la diferencia es que la factura soporta muchos ítems de facturación, dependiendo el formato, entonces es más fácil tomar a los ítems y referenciarlos a la factura con esto ya no habría la necesidad de estar ingresado la descripción o los valores en las líneas de los ítems de la factura.

**Ejemplo de Objeto Imaginario.**

Para este ejemplo quería mostrarles como se hace el análisis para crear un objeto imaginario básico, este trabajo no pretende ser un manual para nada de ningún tipo, pero. ¡Vamos adelante!

Pensemos en algo que todos lo que hemos programado necesitamos, un objeto que trabaje con la base de datos, las ventajas de esto, es que para trabajar con la BD solamente hace falta llamar al objeto e indicarle lo que tiene que hacer.Antes de continuar con le explicación veamos una imagen del objeto:



Al igual que con la factura vamos a revisar las características principales del objeto:

* Conectarse con la BD
* Enviar datos a la BD
* Traer datos de la BD
* Recibir órdenes y datos de la aplicación
* Enviar información a la aplicación

En el gráfico podemos ver una pequeña analogía del objeto, podemos ver que la aplicación y la base de datos nunca de conectan directamente, lo hacen a través de un objeto, esto facilita centralizar todas las operaciones contra la base de datos y filtrar todos los datos permitiendo crear un punto de control, a su vez la aplicación recibe datos que “entiende” porque el objeto los retorna de manera que la aplicación los pueda mostrar al usuario. Estaesla razón por la que no hay una conexión directa entre la aplicación y la base de datos (aunque podría existir).

**Resumen de Propiedades encontrados Objeto**

Para definir las propiedades del objeto primero tememos que ver sus características, las cuales listamos anteriormente.

Conectarse con la DB:Para realizar una conexión hace falta conocer algunos datos como son:

* Dirección del Servidor
* Puerto de Conexión
* Usuario de la BD
* Contraseña de la BD
* Nombre de la BD

Luego de saber dónde y cómo establecer una conexión con el servidor de bases de datos, empiezo a interactuar con él, analicemos el tipo de información que se va a enviar y recibir.

Se va a enviar cadenas de texto (consultas) y recibir datos, los datos que recibimos por lo general se retornan en forma de matriz o arreglo multidimensional.

* Consulta
* Resultados en matriz

La consulta es la información que la aplicación solicita al objeto y los resultados es lo que la aplicación recibe del objeto.

**Resumen de Métodos encontrados Objeto**

* **Conectar y desconectar del servidor**
* **Controlar consultas**
* **Leer información de la BD**
* **Enviar datos a la BD**

Observación: Si bien es cierto se puede conectar directamente la aplicación a la base de datos, no es recomendable ya que sería muy difícil mantener y controlar esa aplicación veamos el ¿por qué?

Cuando tenemos la gestión de la base de datos por separado es fácil cambiar de servidor cambiando algunos parámetros en la cadena de conexión, en lugar de editar cada línea en la que se llama al servidor, vamos más allá, si queremos poner un punto de control para depurar las consultas enviadas al servidor desde la aplicación podemos establecer filtros para que las entradas sean limpias y con esto evitar que alguien vea información sensible, veamos un ejemplo para que se entienda mejor:

En una aplicación web en la que el usuario se identifica con el sitio, ingresa la información para usuario y contraseña, si uno conecta directamente esta pantalla a la base de datos es decir armar la consulta con lo ingresado en el formulario, corre el riesgo de que el usuario ingrese código SQL y que la aplicación responda ese código como lo haría normalmente, a esto se le conoce como inyección SQL, con el objeto esto podría ser neutralizado ya que primero depuramos la cadena de consulta eliminando comandos que no deberían estar ahí o quitando caracteres de especiales.

**1.4.3. Clase**

El objeto es la representación de una entidad de la vida real sea esta física o abstracta, la clase es la representación del objeto frente al lenguaje de programación por parte del programador, también se puede considerar como una plantilla, como si fuese los planos de un auto antes de ser construido. Un objeto es la instancia de la clase, el tipo de objeto depende de la clase de la que se lo construyó, como vimos en los ejemplos anteriores se puede decir que la clase **Factura** será instanciada en un objeto tipo **Factura**, pasa lo mismo con el objeto Base Datos.

Al ser PHP el lenguaje dominante de este proyecto,veamos la definición que le da PHP a la palabra clase:

*“… La definición básica de clases comienza con la palabra clave* ***class****, seguido por un nombre de clase continuado por un par de llaves que encierran las definiciones de las propiedades y métodos pertenecientes a la clase. El nombre de la clase puede ser cualquier etiqueta valida que no sea una palabra reservada de PHP. Un nombre válido de clase comienza con una letra o guión bajo, seguido de la cantidad de letras, números o guiones bajos que sean…[6]”*

Ejemplo de clase válida en PHP:

1 <?php

2 // Forma de declarar una clase en PHP

3 classClaseNormal{

4 // Propiedades de la clase

5 public$Propiedad1\_;

6 public$Propiedad2\_;

7

8 //Métodos de la clase

9

10 publicfunctionprimerMetodo(){

11 // Algoritmo del método

12 }

13

14 publicfunctionsegundoMetodo(){

15 // Algoritmo del método

16 }

17 }

18

19 ?>

Las clases están formadas por componentes que describen las propiedades en variables y los métodos en funciones, en este caso todos los métodos y propiedades son públicos, veremos más adelante los tipos de ámbito que pueden tener estos componentes.

Como podemos ver la clase es la parte básica de los objetos, los objetos son construidos por los lenguajes, el programador solo diseña su estructura y le da funcionalidad.

Esta clase es la normal por decirlo de alguna forma ya que siempre genera un mismo tipo de objeto, más adelante veremos los tipos de clases que soporta PHP y las utilidades que tienen.

**1.4.3.1. Clase Instanciable**

Estas son clases normales, la definición de clase que vimos en el párrafo anterior aplica a este tipo de clase, por lo que no hace falta una nueva referencia.

**1.4.3.3. Clase Heredada**

Estas son conocidas también como clase hija o dependiente ya que es una extensión de una clase padre, esta cualidad dentro de la POO es conocida como **Herencia**, al ser una extensión, amplía la capacidad de la clase padre y tiene acceso a los métodos y propiedades que la clase padre permita ver, una a aclaración de esto se verá en el subcapítulo ámbito de los métodos y Propiedades.

Estas clases son muy útiles para implementar o sobre escribir métodos y propiedades de clases superiores, también son muy útiles a la hora de implementar clases abstractas.

Ejemplo de clase Heredada en PHP.

1 <?php

2 // Forma en la que se declara una clase heredada

3 classClaseHeredadaextendsClasePadre{

4 // Nuevas propiedades de la clase

5 public$PropiedadNueva1\_;

6 public$Propiedad2Nueva2\_;

7

8 //Métodos de la clase

9

10 publicfunctionnuevoMetodo(){

11 // Algoritmo del método

12 }

13

14 privatefunctionMetodoExistenteEnPadre(){

15 // Algoritmo del método

16 }

17 }

18

19 ?>

Se entiende que esta clase está extendiendo o heredando a la clase llamada **ClasePadre** y además está sobre escribiendo un método de la misma, la sobre escritura de métodos ocurre cuando un método de la clase hija lleva el mismo nombre de uno de su clase padre, esto lo podemos ver en el segundo método de la clase de ejemplo.La instancia de esta clase produce un objeto del tipo de la clase hija, con las características de la clase padre.

**1.4.3.2. Clase Abstracta**

Este tipo de clases sirve para definir estructuras, no pueden crear objetos por si mismas ya que para ser utilizadas deben ser implementadas por una clase Heredada,la cual redefine los métodos abstractos bajo el mismo ámbito aunque una clase abstracta en PHP si puede tener métodos definidos para que luego sean compartidos por las clases que hereden de esta.Los métodos abstractos son aquellos los que pueden ser solo declarados pero no implementados en la clase padre, por esta característica se dice que las clases abstractas crear plantillas o clases generales.

Ejemplo de clase abstracta:

1 <?php

2 // Forma en la que se declara una clase abstracta

3 abstractclassClaseAbstracta{

4 // Propiedades de la clase

5 public$PropiedadNueva1\_;

6 public$PropiedadNueva2\_;

7

8 //Los Métodos pueden o no tener un algoritmo

9

10 publicabstractfunctionnuevoMetodo(){

11 // Algoritmo del método

12 }

13 // método abstracto sin algoritmo, lo implementan las clases hijas

14 abstractprotectedfunctionMetodoExistenteEnPadre();

15 }

16

17?>

Esta es una clase abstracta que tiene un método definido y uno abstracto, el método abstracto como podemos ver solamente se encuentra definido, la clase hija tiene que volver a declarar este métodoe implementar su funcionamiento.

Observación: Imagino que se ha preguntado. ¿Qué uso le puedo dar a una clase como esta? La verdad aunque no lo parezca es muy útil, para demostrarlo veamos un ejemplo: Las clases abstractas sirven mucho como punto de control,hace ya algún tiempo me encontraba trabajando en una aplicación para control de pacientes, ya tenía implementadas todas las librerías de objetos que necesita al igual que el diseño visual de las vistas, tenía pantallas de pacientes, de especialidades, de doctores entre otras, como era una aplicación que trabajaba con una base de datos me vi en la necesidad de crear una sola clase abstracta llamada pantallas la misma que definía algunos métodos con su respectivo algoritmo y otros que eran redefinidos en las clases hijas, de esta manera logré centralizar la actividad de un usuario en mis sistema ya que la misma clase abstracta estaba siempre instanciada en la aplicación permitiéndome conocer todas las actividades que se realizaban con la base de datos y centralizando toda la actividad de esta en métodos definidos en la clase abstracta, la misma me sirvió como un punto de control también controlaba el acceso a las librerías ya que las clases heredaban todos sus métodos.

**1.4.3.4. Clase Interface**

Este tipo de clases es muy especial, en lo personal jamás he trabajado con interfaces pero la voy a mencionar ya que es un elemento de la POO, fuera de lo que se pueda imaginar no son clases encargadas de gestionar o crear interfaces visuales, sino más bien de generar clases genéricas que deben ser implementadas por las clases que la hereden, las que solamente la pueden implementar usando la palabra clave **implements**, las clases heredadas pueden implementar más de una interface a la vez, cosa que no es posible en la herencia normal de clases, ya que PHP soporta herencia de una clase a otra, si se deseaheredaruna clase heredada puede heredar a otra a esto se le llama herencia en cascada.

La Interfaz es similar a la clase abstracta, la diferencia es que en esta clase los métodos solo pueden ser abstractos y la forma en que se declara es diferente.

Ejemplo de interface:

1 <?php

2 /\*\*

3 \* Ejemplo de interface para una base de datos, en ella

4 \* se listan ejemplos de métodos abstractos, que se deben

5 \* implementar en las clases que la implementen.

6 \*/

7 interface InterfaceBaseDatos{

8

9 // se declaran los métodos necesarios

10 abstractpublicfunctionlistarRegistros();

11 abstractpublicfunctioneditarRegistro($tabla, $id\_registro);

12 abstractpublicfunctionBuscarRegistro($tabla, $id\_registro);

13 abstractpublicfunctioneliminarRegistro($tabla, $id\_registro);

14 abstractpublicfunctionbuscarRegistro($tabla, $condiciones);

15

16 }

17

18 /\*\*

19 \* Clase que implementa a la interfaz anterior, en esta

20 \* clase se redefinen los métodos y se los implementa

21 \*/

22 classImplementaInterfaceimplementsInterfaceBaseDatos{

23

24 publicfunctionlistarRegistros(){

25 //algoritmo del método

26 }

27

28 publicfunctioneditarRegistro($tabla, $id\_registro){

29 // algoritmo del método

30 }

31 ....

32 }

33 ?>

En el ejemplo podemos ver cómo se puede usar a una interface, podemos ver que las clases que heredan la interfaz están en la obligación de redefinir los métodos, esta es la razón por la que se llaman Interfaces, porque son una plantilla general.

**1.4.3.5. Clase Final**

Este tipo de clases son útiles si no queremos que una clase nunca pueda ser heredada, este tipo de clases se declara con la palabra clave **final** al inicio de la línea de código.

La palabra clave **final** también puede ser usada directamente en los métodos, esto en el caso de que solamente se desee asegurar algunos métodos, no se puede usar con las propiedades.

Ejemplo de clase final:

1 <?php

2 /\*\*

3 \* Se declara la clase final, se pueden definir propiedades

4 \* normalmente.

5 \*/

6 final classClaseFinal{

7 public $Nombre\_;

8

9 // todos los métodos de la clase son finales

10 publicfunctionmetodoFinal(){

11 // Algoritmo método final

12 }

13

14

15 }

16

17 classClaseNormal{

18 public $Nombre;

19

20 // este método no podrá ser modificado las clases que hereden esta clase

21 final functionmetodoFinal(){

22 // Algoritmo método final

23 }

24 }

25 ?>

Podemos ver en el ejemplo anterior a dos clases, una final y otra normal, todos los métodos de una clase final son abstractos de manera implícita, este es el caso de la primera clase, en la segunda clase la cual es una clase normal que se puede heredar, se define un método como final el mismo que no es heredado cuando se hereda la claseClaseNormal ya que este método está definido como final, esta no es la mejor manera de proteger métodos de las clases pero se puede hacer, una mejor manera de hacerlo se verá en el próximo capítulo.

**1.4.3.6 Resumen de clases**

Hasta aquí hemos visto los tipos de clases, ahora veamos un cuadro en el que se nos muestra que tipo de clase usar de acuerdo a la necesidad del momento.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Necesidad… | Instanciable | Abstracta | Final | Heredada | Interface |
| Crear una clase que pueda ser instanciada y/o heredada | X |  |  |  |  |
| Crear una clase, cuyo objeto guarda relación con los métodos y propiedades de otra clase |  |  |  | X |  |
| Crear una clase que solo sirva de modelo con la capacidad de definir métodos comunes, sin que pueda ser instanciada |  | X |  |  |  |
| Crear una clase que sirva solamente como modelo, que no se pueda instanciar ni definir ningún método |  |  |  |  | X |
| Crear una clase que necesite ser instanciada pero que no pueda ser heredada. |  |  | X |  |  |

Espero haber dejado claro los conceptos de los diferentes tipos de clases que existen, es bueno mencionar que estas clases son de la POO y no de PHP por lo que no es una pérdida de tiempo conocer sus conceptos ya que nos servirán cuando estemos trabajando con otro lenguaje [8].

**1.4.4. Ámbito de Métodos y Propiedades**

El ámbito de los métodos y propiedades también conocido como alcance, es por decirlo de alguna forma la visibilidad de los métodos y propiedades, esta es la forma en la que las clases hacen que sus método y propiedades sean visibles por las demás clases de la aplicación o si solo son visibles para sí mismas, hasta el momento conocemos los diferentes tipos de clases que existen dentro de la POO ahora veamos los tipos de métodos y propiedades que existen en las clases, su utilidad y la forma en la que se crean.

**1.4.4.1 Propiedades y métodos públicos**

Este tipo de métodos y propiedades son aquellos que pueden ser vistos por las demás clases de la aplicación.

También son públicas las propiedades definidas con la palabra clave “**var**”, este ámbito es ideal para trabajar con clases que controlan una capa inferior de clases o librerías.

**1.4.4.2 Propiedades y métodos protegidos**

Los métodos y propiedades protegidos solo pueden ser vistos por la clase a la que pertenece y por las clases que hereden la clase a la que pertenecen.

La ventaja es que solo las clases que queremos, pueden ver estos métodos y propiedades, esto nos puede servir a la hora de crear un objeto compuesto de varias clases, ya que solo el objeto tendría acceso a esas propiedades y métodos. Por ejemplo en un objeto que gestiona una tabla de base de datos heredaría de una clase superior las rutinas para consultar y escribir en la tabla, pero sería el único que tiene acceso a esos métodos.

**1.4.4.3 Propiedades y métodos privados**

Este tipo de métodos y propiedades solo pueden ser vistos y usados por la clase a la que pertenecen.

Es muy útil ocultar los métodos y las propiedades a las demás clases, ya que hay datos sensibles en las clases como la conexión a una base de datos, esta tiene la información del servidor, muchas veces en texto plano, por lo que es muy útil ocultar estas propiedades, además de esos datos tiene la conexión al servidor esta debe ser solamente manejada por la clase a la que pertenece.

**1.4.4.4 Propiedades y métodos estáticos**

Estos son métodos son públicos por defecto, pero las clases que la heredan no lo pueden modificar aunque si la pueden ver.

Lo que hace que estos métodos y propiedades sean útiles es la visibilidad y que no pueden ser modificados, hay algunas reglas para usar estos métodos , si el método es llamado desde una clase cualquiera se usa la palabra clave “self::” seguida por doble dospuntos, pero si es llamada desde una clase heredada se usa la palabra clase “parent::” seguida de doble dos puntos.

**1.4.4.5 Métodos Abstractos**

Este tipo de métodos solamente son la declaración de la función sin implementación, por esta razón no es necesario poner las llaves en la declaración del método y cada línea termina con “;” punto y coma.

**1.4.5. Resumen y ejemplo de Ámbito de Métodos y Propiedades**

Veamos un ejemplo completo de los tipos de métodos y propiedades en funcionamiento dentro de una clase real, que es la que gestiona una base de datos y otra que gestiona una tabla llamada clientes.

1 <?php

2 /\*\*

3 \* Se declara una clase abstracta para manejar la base de datos

4 \* se definen las propiedades necesarias para trabajar con la BD

5 \*/

6 abstractclassBaseDatos{

7

8 privatestatic$Servidor\_ = 'localhost';

9 privatestatic$Usuario\_ = 'root';

10 privatestatic$Contraseña\_ = 'miclave';

11 protected$NombreDb\_ = 'mibase';

12 protected$Consulta\_;

13 private$Conexion\_;

14 public$Resultado\_ = array();

15 var$Errores\_;

16

17 /\*\*

18 \* Método privado que conecta a la base de datos creando un objeto

19 \*/

20 privatefunctionconectarDb(){

21 $this->Conexion\_ = newmysqli(self::$Servidor\_, self::$Usuario\_,

22 self::$Contraseña\_, $this->NombreDb\_);

23 }

24

25

26 /\*\*

27 \* método privado encargado de cerrar la conexión con el servidor

28 \*/

29 privatefunctiondesconectarDb(){

30 $this->Conexion\_->close();

31 }

32 /\*\*

33 \* grupo de métodos abstractos encargados de manejar las tablas

34 \*/

35 abstractprotectedfunctionobtenerRegistros();

36 abstractprotectedfunctioncrearRegistro();

37 abstractprotectedfunctioneliminarRegistro();

38 abstractprotectedfunctionactualizarRegistro();

39

40 /\*\*

41 \* función pública encargada de mostrar un error de la BD

42 \*/

43 publicfunctionmostrarError(){

44 return$this->Errores\_;

45 }

46

47 /\*\*

48 \* Función encargada de mostrar los errores de la base de datos

49 \*/

50 protectedstaticfunctionmostrarSql(){

51 return$this->Errores\_;

52 }

53

54 /\*\*

55 \* Función encargada de consultar a la base de datos

56 \*/

57 protectedfunctionCosultar($sql){

58 if ($sql){

59 $this->Consulta\_ = $sql;

60 $this->conectarDb();

61 $this->Resultado\_= $this->Conexion\_->query($this->Consulta\_);

62 return$this->Resultado\_;

63 $this->desconectarDb();

64 }

65 }

66 }

67 ?>

Ahora veamos el código de la clase heredada llamada clientes.

1 <?php

2

3 require\_once'basedatos.php';

4 /\*\*

5 \* Clase heredad que implementa a la clase base datos

6 \*/

7 classClienteextendsBaseDatos{

8 private$Consulta\_;

9 private$Tabla\_ = 'clientes';

10

11 /\*\*

12 \* implementacion del primer metodoestatico

13 \*/

14 privatefunctionobtenerRegistros(){

15 $this->Consulta\_ = 'select \* from ' .$this->Tabla\_;

16 return$this->Cosultar($this->Consulta\_);

17 }

18

19 /\*\*

20 \* implementación del segundo métodoestático

21 \*/

22 publicfunctioncrearRegistro($datos){

23 $this->Consulta\_ = 'insert' .$this->Tabla\_ .$datos;

24 return$this->Cosultar($this->Consulta\_);

25 }

26

27 /\*\*

28 \* implementación del tercero método estático

29 \*/

30 protectedfunctioneliminarRegistro($id\_registro){

31 $this->Consulta\_ = 'delete ' .$this->Tabla\_ .'where id = ' . $id\_registro;

32 return$this->Cosultar($this->Consulta\_);

33 }

34

35 /\*\*

36 \* implementación del cuarto método estático

37 \*/

38 privatefunctionactualizarRegistro($datos){

39 $this->Consulta\_ = 'updtae' .$this->Tabla\_ .$datos;

40 return$this->Cosultar($this->Consulta\_);

41 }

42

43 // Métodos protegidos pueden ser vistos por la clase que herede esta clase

44 protectedfunctionmostrarSql(){

45 return$this->mostrarSql();

46 }

47

48 protectedfunctionmostratError(){

49 returnparent::Errores\_();

50 }

51

52

53 }

54 ?>

Antes de empezar a estudiar las dos clases cedas anteriormente (basedatos.php, clientes.php), quiero comunicar al lector que dichas clases no fueron diseñadas para funcionar de la manera más eficiente, de hecho ni siquiera la implementación de los métodos es la correcta, solamente están a modo de ejemplo para que se pueda observar con mejor detalle el ámbito de las propiedades y métodos, una vez aclarado este pequeño asunto, veamos qué es lo que tiene la primera clase.

Clase basedatos.php

Esta clase es la encargada de manejar las conexiones y tareas contra la base de datos de ahí su nombre, esta es una clase abstracta lo que indica que puede tener métodos abstractos, como podemos ver cuida de las propiedades más sensibles como son los datos de servidor de bases de datos reservándolos solo para ella, declara como privados los métodos de conexión a la base de datos y deja a como protegidos a los métodos de consulta ya que estos pueden ser ejecutados por las clases que lo hereden, tenemos un método estático que muestra los errores esto es porque este método no debe ser modificado por ninguna clase aunque puede ser heredado, tenemos métodos públicos que pueden ser vistos desde cualquier clase.

Mi intención no es entrar en detalle en las clases espero que el ejemplo se explique por sí mismo.

En la presente tabla podemos los ámbitos de los métodos y propiedades de una manera más gráfica.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Método/Propiedad | Publico | Protegido | Privado | Final | Estático | Abstracto |
| Que pueda ser visto desde donde sea | X |  |  |  |  |  |
| Que solamente pueda ser visto por la clase que herede la clase a la que pertenecen |  | X |  |  |  |  |
| Que solamente puedan ser vistos por la clase a la que pertenecen |  |  | X |  |  |  |
| Que no puedan ser heredados |  |  |  | X |  |  |
| Métodos que no puedan ser sobre escritos por las clases que lo hereden |  |  |  |  | X |  |
| Métodos declarados para ser implementados luego, en la clase que implemente la clase abstracta a la que pertenecen |  |  |  |  |  | X |

Al igual que con las clases, estos conceptos aplican a la POO, aunque se ha hecho referencia a PHP por ser el lenguaje más importante para este proyecto, los conceptos aplican a cualquier lenguaje que soporte POO.

**1.4.6. Evento Y Mensaje**

Esta es la forma en la que los objetos interactúan entre sí, haciendo posible el trabajo en conjunto para resolver un problema, veamos una definición exacta de los que es un evento y mensaje:

Evento: Es un suceso en el sistema

Mensaje: Es la comunicación del evento al objeto [8]

1. **Capítulo II**

Este capítuloestá dedicado a conceptos técnicos acerca de las tecnologías y herramientas usadas para crear el proyecto, vamos a ir poco a poco aprendiendo los conceptos básicos para adentrarse en el desarrollo web, con este capítulo se pretende explicar y comprender todo los referente a las tecnologías usadas para crear proyectos web, no vamos a hablar de ningún, nos vamos a referir a los lenguajes de programación y maquetación que vamos a usar para desarrollar en CMS.

* 1. **INTRODUCCIÓN A HTML**

HTML por sus siglas en inglés significa HyperText Markup Language lo que se traduce en: lenguaje de marcado de hipertexto, Este es un leguaje de etiquetas que mas se usa para elaborar paginas web, es HTML quien describe la estructura e informacion de la pagina en forma de texto dentro de un archivo con extencion “**.html”** aunque tambien es soportado en archivos de lenguajes de programacion web como PHP, .NET, JAVA, y todos los que soportan sintaxis enbebida en el codigo.

Este lenguaje se escribe en forma de etiquetas, dentro de HTML se puede incluir trozos de codigo de hojas de estilo en cascada CSS para hacer que la pagina se vea como el diseñador lo quiera y codigo de javascript para darle mayor funcionalidad.[9]

* 1. **Historia HTML**

El HTML se creó en un principio con objetivos divulgativos de información con texto y algunas imágenes. No se pensó que llegara a ser utilizado para crear área de ocio y consulta con carácter multimedia, de modo que, el HTML se creó sin dar respuesta a todos los posibles usos que se le iba a dar y a todos los colectivos de gente que lo utilizarían en un futuro. Sin embargo, pese a esta deficiente planificación, se han ido incorporando modificaciones con el tiempo, estos son los estándares del HTML. Numerosos estándares se han presentado ya. El HTML 4.01 es el último estándar a febrero de 2001. Actualización a mayo de 2005, en estos momentos está a punto de presentarse la versión 5 de HTML, de la que ya se tiene un borrador casi definitivo. [9]

La primera descripción de HTML disponible públicamente fue un documento llamado: HTML Tags (Etiquetas HTML), publicado por primera vez en Internet por Tim Berners-Lee en 1991.2 3 Describe 22 elementos que incluyen el diseño inicial y relativamente simple de HTML. Trece de estos elementos todavía existen en HTML. [10]

**¿Qué es la W3C?**

W3C es la abreviatura de World Wide Web Consortium, esta es un consorcio internacional que lleva adelante el desarrollo de los estándares web, es la encargada de manejar y desarrollar toda la tecnología web, las demás organizaciones tecnológicas siguen estos estándares a la hora de crear productos de software en base web, de esta manera se tiene un punto de control para las tecnologías web.

* 1. **Versiones Y Tipos De HTML**

Como todas las tecnologías en desarrollo HTML ha ido evolucionandocon el paso del tiempo ahora mismo se encuentra casi al final la elaboración de la versión 5 la cual trae novedades como geo-localización, escritura en disco, websokets, JavaScript CSS3, canvas, en realidad la última versión está orientada al aspecto multimedia y a intentar sacar del mercado a flash player el reproductor de video de Adobe, ya que HTML5 tiene esa capacidad pero hablaremos de ello más adelante. En eta tabla tomada de w3c (entidad encargada del desarrollo del estándar HTML) tenemos el listado oficial de lasversiones de HTML desde sus orígenes hasta la actualidad. [11]

|  |  |
| --- | --- |
| Versión | Año |
| HTML | 1991 |
| HTML+ | 1993 |
| HTML 2.0 | 1995 |
| HTML 3.2 | 1997 |
| HTML 4.01 | 1999 |
| XHTML 1.0 | 2000 |
| HTML5 | 2012 |
| XHTML5 | 2013 |

La razón por la que las versiones no son consecutivas es porque hay estándares que nuca fueron liberados.

Los tipos de HTML, que en realidad son las variantes que tiene el lenguaje para cada versión son:

* HTMLStrict: el normal, donde se pueden usar etiquetas HTML, pero no se aceptan etiquetas obsoletas.
* HTML Transitional: este permite usar todo tipo de etiquetas de todas las versiones de HTML, no es recomendable.
* HTML Frameset: Tiene soporte para frames, muy anticuado.[12]
  1. **Etiquetas**

Esta es la parte más básica de HTML las etiquetas son de dos tipos de apertura y cierre, el contenido de estas, va entre las etiquetas de apertura y cierre mismo que puede tener más etiquetas,laetiqueta de apertura está formada por el signo menor que “<” el nombre de la etiqueta en mayúsculas o minúsculas y el signo de mayor que “>”, la etiqueta de cierre están formadas por el signo menor que y el signo de barra inclinada “</” el nombre de la etiqueta en mayúsculas o minúsculas y menor que “>”, también existen etiquetas que no contienen ningún tipo de contenido las cuales se cierran en la misma línea como “<br/>” retorno de carro.

Ejemplos para etiquetas validas:

1 <!-- Esto es un comentario en HTML -->

2 <etiqueta> contenido <etiqueta/>

3 <ETIQUETA> contenido <ETIQUETA/>

4 <ETIQUETA> contenido <otraetiqueta>más contenido <br/></etiqueta></otraetiqueta>

Como podemos ver HTML nos deja crear las etiquetas a nuestro gusto, lo cual es bueno pero no tanto ya que si el programador o diseñador es desordenado puede generar un documento que sea muy difícil de leer, para evitar este inconveniente se crea XHTML el cual es más estricto, el cual no permite abrir y cerrar etiquetas que no estén escritas de igual manera y exige al programador que las etiquetas se cierren en el orden que fueron abiertas cosa que no lo hace el HTML normal como lo podemos ver en la línea 4, también existen técnicas de indentación para que el código sea más fácil de leer, esto no hace que funcione mejor solo lo hace más fácil de leer para el hombre, el navegador lo puede interpretar así este escrito en una sola línea, todo el código de la página.

Las etiquetas no solo son palabras entre símbolos también posen atributos los cuales son usados por otras tecnologías como CSS y JavaScript, veamos los principales atributos de una etiqueta:

**Class:**Atributo de que declara una clase, no como un objeto sino como un identificador se una para diferenciar a un grupo de etiquetas que forman parte de una sección de la página como por ejemplo el menú, este tipo de atributo es aprovechado por CSS para dar estilo y forma a las páginas web, tecnología que estudiaremos más adelante.

**Name:** Atributo que le da un nombre personalizado a una etiqueta, este atributo es usado para la gestión de formularios HTML, ya que es la forma en la que el servidor identifica los valores que se recibieron en el formulario usando los valores asociados a los nombres de los campos.

**Id:** Atributo que asigna un identificador a una etiqueta HTML, este tipo de atributo es más usado en JavaScript para darle funcionalidad a la página, por lo generar el Id debe ser único para cada etiqueta, aunque no es una regla.

**Style:** Este atributo nos ayuda a insertar código CSS dentro de una etiqueta, se usa para darle estilo a una etiqueta, aunque no es muy usado por lo difícil que es mantener la página, si se quiere cambiar el diseño habría que hacerlo a nivel de etiquetas, esto es muy complicado pero por suerte HTML permite ubicar este tipo de códigos en un archivo diferente y luego se lo enlaza con el HTML a través de la etiqueta LINK, de esta manera el mantenimiento es mucho más organizado y fácil.

**Href:** Atributo que indica una dirección web, por lo general este atributo se lo usa para armar enlaces con la etiqueta A.

**Src:** Atributo que indica la ubicación de un archivo dentro del sitio, este atributo es usado para indicar la ubicación de imágenes, hojas de estilo, y códigos JavaScript.

Los atributos van dentro de la etiqueta de apertura separada por un espacio del nombre de la etiqueta y con el signo “=” para asignar su contenido el cual va entre comillas dobles o sencillas.

Ejemplo de Atributos

1 <p class=”parrafo”> Contenidos </p>

2<p id=”identificador”> Contenidos </p>

3<p style=”color:7755A”> Contenidos </p>

4<imgsrc=”../img/miimagen.jpg”/>

5<ahref=”http://google.com”>Ir a Google </p>

6<inputname=”nombres” type=”text”>Ingrese su nombre:</input>

Que las clases se usen para CSS y los ID para JavaScripts no significa que sea una regla, se puede usar al revés o el mismo atributo por los dos, pero es preferible que se haga así para seguir los estándares, y por respeto a los profesionales que vengan después de nosotros.

* + 1. **Definición Documento**

Esta es un parte importante ya que de la definición de documento depende la interpretación que el navegador le dé al código que contiene el fichero de la página, a esto so lo conoce como doctype, veamos los tipos de definición de documento que tenemos para HTML5:

1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" <http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>">
2. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

1. <!DOCTYPE html>
2. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
3. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"http://www.w3.org/TR/html4/loose.dtd">

1. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

Estos tipos los vimos en el subcapítulo de versiones y tipos de HTML, pero en este subcapítulo, lo vemos de una manera más técnica, podemos ver al final del doctypequetenemos el tipo de estándar que se va a usar, el primero es normal, el segundo es transaccional, y así con los siguientes, nosotros vamos a usar el más sencillo de todos el ítem número 5.

* + 1. **Etiquetas Básicas HTML**

En la presente sección vamos a revisar las etiquetas más importantes de HTML, las etiquetas que se van a revisar son las necesarias para escribir una página, también se ha desarrollado el contenido para el manejo de formularios en HTML.

Etiquetas que definen la estructura de una página HTML, es un componente que los usuarios no pueden percibir, estas etiquetas son como la columnas de una casa están por ahí pero que uno difícilmente se las pueden ver a simple vista, son usadas por los navegadores para presentar los contenidos.

**<!DOCTYPE>** Esta etiqueta se usa para declara el tipo de estándar que usa el fichero HTML

**<html>**Esta es etiqueta contienen a todas las etiquetas de la pagina sus contenidos deben ir

dentro de ella.

**<head>**Etiqueta que contiene la información de la cabecera de la pagina, debe incluir un titulo para la pagina, puede contener referencias a archivos de hojas de estilo, scripts, meta información, y más.

**<meta>**Esta etiqueta se usa para mostrar datos de los datos conocidos como metadatos, los contenidos de estas etiquetas no se muestran al usuario pero son usados por los navegadores para conocer el sitio, estas etiquetas también se usan por los buscadores ya que contienen la información de las paginas, estas etiquetas siempre deben estar dentro de la etiqueta head los contenidos de estas etiquetas deben ser como un diccionario una palabra y un valor asignado por el signo igual “=”, también se usa para definir el conjunto de caracteres que usa la pagina.

**<title>**Etiqueta que contiene el titulo de la pagina, el titulo es el que aparece en la parta superior del navegador, por lo general es mostrado en la pestaña o en la barra de títulos de la ventana, esta etiqueta debe ir dentro del <head>.

**<link>**Etiqueta que contiene el enlace hacia archivos de hojas de estilo CSS, debe ir dentro del head,

**<style>** contiene código CSS dentro de la pagina HTML.

**<script>**Etiqueta que contiene un enlace a un archivo de JavaScript, también puede almacenar código entre las etiquetas de apertura y cierre, esta etiqueta puede ir dentro del head pero no es obligatorio, las buenas prácticas de desarrollo dicen que los enlaces a Java Script deben ir al final del código es decir antes del cierre de la etiqueta <html>, para que java script funcione la pagina debe estar completamente cargada, esa es la razón por la que se lo pone al final, y es bueno que mientras se carguen estos ficheros el usuario ya pueda ver el contenido de la pagina.

**<body>**Etiqueta que contiene las etiquetas del cuerpo de la pagina, es su contenido lo que el navegador le muestra a usuario.

**<div>**Esta es una etiqueta de división sirve para agrupar etiquetas, también se los conoce como capa, es usado para aplicar estilos a un grupo de etiquetas o para darles funcionalidad con java script, esta etiqueta debe ir dentro del <body>.

**<span>**Esta etiqueta funciona como la etiqueta <div>, pero a un nivel más bajo, la etiqueta <span> es usada para dar estilo a parte de un contenido, como poner fondo a una palabra, por lo general estas etiquetas tienen código CSS.

Hasta este momento hemos visto etiquetas que nos ayudan a formar la estructura de la página, para su correcta interpretación por el navegador, ahora vamos a estudiar las etiquetas que muestran información al usuario dando formato al texto y construyendo objetos visuales en la pantalla:

**<h1>** Etiqueta usada para mostrar textos con fuentes más grandes y con saltos de líneas, ideal para presentar el titulo de la pagina, esta etiqueta es muy importante para los buscadores ya que se usa como referencia en sus búsquedas, las reglas dicen que por ley debe haber una etiqueta <h1> en una página.

Esta etiqueta tiene unas variantes que van desde el <h1> hasta <h6> cada etiqueta tiene un nivel de importancia más baja y la forma en la que el usuario ve el contenido resalta del resto ya que estas etiquetas son usadas para títulos subtítulos y más.

**<a**> etiqueta usada para crear hipervínculos entre páginas, el atributo más importante de esta etiqueta en el href que indica el destino al que apunta el enlace, de forma predeterminada los enlaces son marcados de la siguiente forma en todos los navegadores:

* Un enlace no visitado está subrayado y azul
* Un enlace visitado está subrayado y púrpura
* Un vínculo activo está subrayado y el rojo

La forma visual de los enlaces puede ser modificada usando estilos CSS.

**<p>**Esta etiqueta es representada con espacios antes y después de su cierre, es usada para mostrar párrafos de contenidos.

**<br/>**Etiqueta que muestra un salto de línea, esta etiqueta no contiene atributos y se puede incluir texto o mas etiquetas dentro de ella, esta etiqueta no tiene etiqueta de cierre, se cierra en la misma etiqueta.

**<b>**Muestra el contenido de la etiqueta en negrita.

**<i>**Muestra el contenido de una etiqueta en itálica (tipografía inclinada)

**<strong>**Define un texto como importante y lo ubica con negrita.

**<img>**Define una imagen en una página, las imágenes no están insertadas, solamente están referenciadas o vinculadas a la pagina HTML, si se desea que una imagen actué como un enlace se coloca la etiqueta <img> dentro de la etiqueta <a>.

Esta etiqueta tiene dos atributos obligatorios src que indica la ubicación de la imagen y alt que indica el titulo de la imagen, este es usado por los buscadores, también es usado por los navegadores que muestra el nombre en lugar de la imagen cuando esta no se logra cargar.

Con las Etiquetas que acabamos de estudiar ya podemos crear una página de contenidos, básica pero funcional, ahora veamos las estructuras que sonposibles manejar con HTML

**Listas**

Las listas son un conjunto de valores o textos que en Word se conocen como viñetas para hacer una referencia, en HTML existen dos tipos de listas las ordenadas y sin ordenar.

1. **Listas ordenadas:** Son ítems que están entre etiquetas<ol>, el navegador las ordena sin importar el orden en el que estén escritas.
2. **Listas sin ordenar:** Son ítems que están entre etiquetas <ul>, el navegador las muestra a un usuario en el orden que están declaradas.

Los ítems de las listas se declaran usando la etiqueta <li> dentro de las etiquetas de listas ordenadas o desordenadas.

**Tablas**

Las tablas, por lo general se usan para mostrar información, antes se las usaba para diseñar los sitios esta práctica ya no es recomendable gracias a la aparición de etiquetas <div> y hojas de estilo CSS.

HTML define tablas usando la etiqueta <table>una tabla consiste en un elemento <table>y <tr> uno o más, <th> y elementos <td>.

La etiqueta<tr> define una fila de la tabla, la etiqueta<th> define un encabezado de la tabla, y la etiqueta<td> define una celda de tabla.Una tabla HTML más compleja puede incluir también <thead>, <tfoot> y elementos <tbody>.

La etiqueta <thead> define la cabecera de una tabla usada para mostrar los títulos de las columnas, la etiqueta <tfoot> define el pie de una tabla, se puede usar para mostrar información de totales o resúmenes, y <tbody> se usa para definir el cuerpo de la tabla que es en donde irían las etiquetas <tr> y <td>.

**Formularios**

Con la información adquirida en los subcapítulos anteriores podemos hacer que un sitio funcione, y que el usuario vea todo lo que le queremos mostrarle, ahora veamos cómo obtener información del usuario, esto se lo hace a través de formularios web, los formularios web son un conjunto de elementos donde el usuario puede elegir, marcar o ingresar datos, los cuales son enviados al servidor, y almacenados para luego ser revisados por el administrador del sitio.

Un formulario se declara usando la etiqueta <form> la cual tiene dos atributos muy importantes el “action” que indica a donde será enviado el formulario y el atributo “method” que es el que indica el método por el cual se envía el formulario, existen dos tipos de métodos, GET y POST, el primero envía los datos embebidos en la URL y el segundo en las cabeceras del http con este método incluso se puede cifrar los datos ingresados por el usuario.

La única forma de recibir información del usuario es a través de la etiqueta <input>, esta etiqueta tiene atributos muy importantes como son el “name” que indica el nombre del valor que recibe la etiqueta es obligatorio, este es usado por el servidor para acceder a sus valores, el atributo “value” que indica el contenido o el valor por defecto de la etiqueta imput, también se pueden incluir los atributos “class” y “id” para darle funcionalidad y estilo a la etiqueta, existe un atributo muy importante que tiene esta etiqueta que es el “type” especifica el tipo de <input> a usar, aquí tenemos un listado completo de los tipos de input que existen en este caso se va a incluir una imagen de lo que produce la etiqueta y el código que se necesita para generarlo:

**Button:** El navegador lo interpreta como botón, la función es solo la de mostrar un botón, pero la acción debe ser implementada con Java Script, el texto mostrado en el botón debe estar en el atributo “value”.

1 <inputtype="button"name="mi boton"id="mi\_id"value="texto de mi boton"/>



Genera:

**Checkbox:** El navegador lo interpreta como una caja tipo check para seleccionar opciones de un grupo de opciones, se puede predeterminar su valor con el atributo “checked = “checked””, lo que hará que aparezca marcado al cargar la pagina.

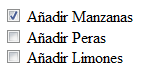
1 <inputtype="checkbox"name="mazanas"id="mi\_id"checked="checked"/>AñadirManzanas</input>

2 <br/>

3 <inputtype="checkbox"name="peras"id="mi\_id"/>AñadirPeras</input>

4 <br/>

5 <inputtype="checkbox"name="limones"id="mi\_id"/>AñadirLimones</input>



Genera:

Podemos ver que la primera opción tiene marcada por defecto el checkbox.

**File:** Atributo que sirve para adjuntar un fichero binario en el formulario, file solamente se puede usar con el método POST.

1 <inputtype="file"name="hoja\_de\_vida"id="input\_hoja\_vida"/>



Genera:

El botón es generad automáticamente por el navegador.

**Hidden:** Este atributo oculta el input por lo que el usuario no ve nada, se puede acceder a su valor desde el servidor, este tipo de campos se usan para colocar información que es de utilidad solamente para el desarrollador, el usuario no se entera de este tipo de campos, los datos de los campos pueden ser llenados con Java Script o enviados desde el servidor.

1 <inputtype="hidden"name="navegador"id="navegador"value="Firefox"/>

**Password:**Este tipo de campo oculta los caracteres que se ingresan en la caja de texto, se lo usa para el ingreso de contraseñas, el servidor accede a ellos directamente en texto plano.

1 ingresesucontraseña:

2 <input type="password" name ="pass" id ="pass"/>



Genera:

**Radio:** Genera un listado de opciones que A diferencia de checkbox en el que se podían seleccionar todas, aquí solo se puede seleccionar una.

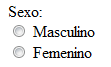
1 Sexo:

2 <br/>

3 <inputtype="radio"name="opcion"id="opcion1"Value="opcion 1"/>Masculino

4 <br>

5 <inputtype="radio"name="opcion"id="opcion2"Value="opcion 2"/>Femenino



Genera:

**Reset:**Elemento que limpia todas los inputs existentes en el formulario al que pertenece, el navegador genera un botón con el texto ingresado en el atributo “value” si se omite este atributo el navegador genera la palabra reset o restablecer dependiendo del idioma del navegador.

1 <inputtype="reset"name="opcion"id="opcion1"Value="Limpiar"/>



Genera:

**Submit:** Este atributo sirve para enviar el formulario, si no se especifica el texto del botón en el atributo “value” el navegador coloca enviar consulta.

1 <inputtype="submit"name="enviar"id="btn\_envio"Value="Enviar Informacion"/>

Genera:

**Text:** Esta propiedad convierte al input en una caja de texto en la que el usuario puede ingresar datos, se puede acompañar de “maxlen” que indica la cantidad máxima de caracteres aceptados, este tipo de input fue muy favorecido en HTML5 lo cual veremos más adelante, si en atributo “value” esta vacio aparecerá una caja de texto vacía pero sí queremos que aparezca un valor por defecto debemos poner ese valor en el atributo “value”.

1 IngresesuNombre:

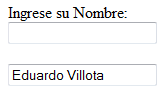
2 <br/>

3 <inputtype="text"name="nom\_user"id="nombre"Value=""/>

4 <br/>

5 <br/>

6 <inputtype="text"name="nom\_user"id="nombre"Value="Eduardo Villota"/>



Genera:

Hasta el momento hemos terminado con las etiquetas básicas necesarias para formar una pagina web, existen decenas de etiquetas, si es de interés de lector estudiar mas afondo las etiquetas puede ver la pagina de la W3C organismo encargado de llevar adelante esta tecnología, link citado en esta referencia. [13]

**2.4.2 EstructuraBásica de una Página**

Es la forma en la que se organizan los elementos dentro de un archivo HTML vamos a establecer la siguiente estructura básica:

1 <!DOCTYPEhtml>

2 <htmllang="es">

3 <head>

4 <metacharset="utf-8">

5 <title>Mitítilo</title>

6 </head>

7 <body>

8 permite el uso de caracteres como ñaño y barça y tíldes entre otros

9 </body>

10 </html>

Esta estructura es muy beneficiosa y es la más indicada cuando se inicia con HTML5, si se desea trabajar con otra versión del HTML lo único que cambia es el Doctype con cualquiera de los que vio en el subcapítulo anterior, la ventaja de esta estructura es que el navegador entiende que la pagina tiene contenidos en español gracias al atributo lang de HTML, también se establece un conjunto de caracteres para poder usar sin problemas los caracteres latinos y tildes al igual que otros caracteres, si el lector no conocía a HTML antes de la codificación Utf-8 tal vez no sabe de lo que le hablo, antes para poder poner un carácter especial hacia falta escribir códigos especiales para que el navegador los interprete y en su lugar ponga el carácter deseado veamos un ejemplo, antes de Utf8 para mostrar en pantalla una í (i tildada) había que hacer esto “**&iacute;”** estos códigos salían de una tabla de códigos, no vale la pena perder tiempo explicando algo que está obsoleto así que creo que es suficiente con la referencia que acabo de dar.

**2.7. Ejemplo Página Básica HTML**

Ha llegado la hora de entrar a la práctica, se terminaron de definir los conceptos básicos para poder elaborar una página HTML, veamos el código de una página y la imagen de lo que el navegador nos muestra.

1 <!DOCTYPE html>

2 <html lang=”es”>

3 <head>

4 <title>Mi Primera Página en HTML</title>

5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

6 <style type="text/css">

7 body {

8 background-color: #fff;

9 margin: 40px;

10 font: 13px/20px normal Helvetica, Arial, sans-serif;

11 color: #4F5155;

12 }

13 </style>

14 </head>

15 <body>

16 <h1>Contáctenos...</h1>

17 Si desea ponerse en contacto con nosotros por favor siga los soguiente pasos

18 <ul>

19 <li><a href="http://google.com">Registrese aquí</a></li>

20 <li><b>Ser mayor de 18 años</b></li>

21 <li><i>Vivir en Quito</i></li>

22 <li>Ser de Ecuador</li>

23 </ul>

24 <h3>Una vez cumplidos los requisitos anteriores

25 debe llenar los datos en el siguiente formulario:</h3>

26 <div class="formulario">

27 <form method="post" id="formulario" action="http://sln-ec.com/index.php/contactos/recibir/">

28 <fieldset>

29 <div class="control-group">

30 <div class="controls">

31 <labelclass="control-label">Nombres:</label>

32 <input class="input-xlarge focused" id="nombres" name="nombres" value="" type="text">

33 </div>

34 <div class="controls">

35 <label class="control-label">E-mail:</label>

36 <input id="email" name="email" value="" type="text">

37 </div>

38 <div class="controls">

39 <labelclass="control-label">Empresa:</label>

40 <input class="input-small" id="empresa" name="empresa" value="" type="text">

41 </div>

42 <div class="controls">

43 <labelclass="control-label">Telefono:</label>

44 <input class="input-small" id="telefono" name="telefono" value="" type="text">

45 </div>

46 <div class="controls">

47 <labelclass="control-label">Asunto:</label>

48 <input class="input-xlarge" id="asunto" name="asunto" value="" type="text">

49 </div>

50 <div class="controls">

51 <labelclass="control-label">Descripción:</label>

52 <textarea id="texto\_asunto" name="texto\_asunto" rows="4" columns="20"></textarea>

53 </div>

54 <div class="controls">

55 <div class="form-actions">

56 <button type="submit" class="btnbtn-info">EnviarInformación</button>

57 <button type="reset" class="btnbtn-warning">Reset</button>

58 </div>

59 <b>Dirección: </b>Av Cristóbal Colón y Av. Rio Amazonas Edificio Pepito Piso 35 <b>Of:</b> 1502<br>

60 <b>Telfonos:</b> +593 (2) 3444-563 +593 (2) 4332-961 <b>Fax:</b> 593 (2) 2341-578<br>

61 <b>Email:</b><a href="mailto:info@misitio.com" class="btnbtn-inverse"> info@misitio.com </a>

62 <a href="mailto:operaciones@misitio.com" class="btnbtn-inverse">operaciones@misitio.com </a>

63 </div>

64 </div></fieldset>

65 </form>

66 <h6>Gracias por comunicarse con nosotros, tendermos en cuenta su solicitud...</h6>

67 </div>

68 </body>

69 </html>

Esta página ha sido construida solamente con las etiquetas estudiadas en el subcapítulo anterior, se han agregado algunas etiquetas nuevas, pero su función es dar más semántica a l documento, en otras palabras lo vuelve más entendible mejorando su estructura, también usamos la estructura del ejemplo anterior. Veamos el resultado que produce:

****

Si desea se puede copiar el código a un archivo HTML y podrá ver mejor su resultado.

**2.8. ¿Qué es HTML5?**

HTML5 es el último estándar de HTML, aunque aúnestá en desarrollo por parte de la W3C ya se usa desde hace algunos años, en esta versión aparecieron nuevas etiquetas semánticas pero HTML5 no solo es una nueva versión de HTML sino que es la inclusión de nuevas tecnologías que dan a HTML5 una capacidad impresionante, podemos decir que HTML5 es igual a la unión de la última versión de HTML, con CSS3 y Java Script. Flash en especial ha sido usado en reemplazo de HTML para desarrollar aplicaciones web que superaran las habilidades de un navegador: Audio, video, webcams, micrófonos, datos binarios, animaciones vectoriales, componentes de interfaz complejos, entre muchas otras cosas. Ahora HTML5 es capaz de hacer esto sin necesidad de complementos y con una gran compatibilidad entre navegadores, un ejemplo de esto es el popular juego [angrybirds](https://www.google.com.ec/search?hl=es&client=firefox-a&hs=Jsk&tbo=d&rls=org.mozilla:es-ES:official&spell=1&q=angry+birds&sa=X&ei=H__LULXZMZL69gSBqYHQBw&ved=0CDEQvwUoAA)[14] Youtube también tiene una versión escrita en HTML5, la cual aún está en fase beta se la puede activar visitando la dirección <http://www.youtube.com/html5> y activando la versión de prueba.

HTML5 ha influido mucho al mundo de los móviles todos los sitios web optimizados para móviles están en HTML5, este es un listado de sitios que usan HTML5.

* Facebook
* Twitter
* Youtube
* Flikr
* Github
* Google
* Hotmail en su nueva versión llamada outlock.com
* Yahoo
* Etc.

Si tantas empresas alrededor del mundo lo están usando es porque funciona, se sorprenderá lo que se puede hacer con HTML5, entonces. ¿Por qué no empezamos a usarlo? No se debe tener miedo a los cambios, muchas veces traen cosas nuevas a nuestra vida, sin embargo es necesario un poco de dedicación por parte del programador para entender sus nuevos conceptos y los nuevos elementos.

**2.8.1. Nuevas Etiquetas**

Las principales etiquetas HTML5 nuevas no tienen una representación especial en pantalla. Todas se comportan como un <div> o un <span>. Pero cada una tiene un significado semántico superior a un simple <div> o <span> ayudan a definir de una manera más clara la estructura de la página, veamos el listado de las principales etiquetas:[15]

**<header>:** Se solía hacer cosas como<div id="header">para representar la cabecera de una pagina web. <header> está diseñada para reemplazar la necesidad de crear divs sin significado semántico.

**<hgroup>:** Muchos headers necesitan múltiples títulos, como un blog que tiene un titul. <hgroup> permite colocar un h1, h2 y h3 dentro del header sin afectar el SEO, permitiendo usar otro h1 en el sitio.En el HTML actual, sólo puedes usar h1 una vez por sitio o el h1 pierde prioridad de SEO.

**<nav>:** Igual que <header>, <nav> está diseñado para colocar en este lugar los menús de navegación principal. Se puede colocar cualquier etiqueta dentro, aunque lo recomendado es usar listas <ul>. Sólo se puede tener un <nav> por página.

**<section>:** Define un área de contenido única dentro del sitio. En un blog, sería la zona donde están todos los posts. En un video de youtube, habría un <section> para el video, uno para los datos del video, otro para la zona de comentarios.

**<article>:** Define zonas únicas de contenido independiente conocidas como artículos. En el home de un blog, cada post sería un article. En un post del blog, el post y cada uno de sus comentarios sería un <article>.

**<aside>:** Cualquier contenido que no esté relacionado con el objetivo primario de la página va en un <aside>. En un blog, obviamente el <aside> es la barra lateral de información, en algunos sitios se usa este lugar para poner links de contenidos relacionados. En el home de un periódico, puede ser el área de indicadores económicos o en youtube seria la lista de sugerencias de videos a la derecha.

**<footer>:**Compone el pie de página, en un sitio seria la información que siempre va en el pie de página como es la dirección y algunos enlaces de interés.

**<input>:** Esta etiqueta ha recibido nuevos atributos como placeholder, required yautofocus, el primer atributo coloca un texto en las cajas de texto que desaparece cuando se hace clicsobre la caja y reaparece cuando la caja queda vacía, el segundo atributo hace que el input sea obligatorio y no permite continuar sin que sea ingresada la información que necesita, el ultimo sirve para ubicar el cursor en el input deseado, solo puede haber uno por página, también se agregaron nuevos tipos del valores para el atributo “type”, los cuales hacen que el input sea másespecífico en lo que debe contener, como son números, fechas, email, entre otros, aquí tenemos un listado de los nuevos valores para el atributo“type”.

* **search:** coloca una “x” en la parte derecha de la caja de texto que borra el texto
* **tel:** Sirve para ingresar un número de teléfono, solo admite ingreso de números.
* **url:** Sirve para ingresar direcciones web
* **email:** Sirve para ingresar una dirección de correo, el navegador verifica que sea valida
* **datetime:** Sirve para ingresar textos formateados como fechas
* **date:** Ayuda a controlar el ingreso de fechas, dependiendo del navegador se muestra un calendario cuando el usuario da clic en la caja de texto.
* **Month:** Ayuda a ingresar un número del 1 al 12, o un nombre válido de un mes
* **Week:** Selecciona una semana de un calendario
* **Number:** Permite el ingreso de número, permite ingresar números científicos
* **Range:** Permite seleccionar un rango
* **Color:** Permite seleccionar un color de una tableta de colores

La mayoría de estos nuevos valores para el atributo “type” son aprovechados por los dispositivos móviles, ya que luego de leer el valor de “type” coloca al usuario un teclado másóptimo para facilitar el ingreso de la información, por ejemplo en el tipo email coloca en los caracteres necesarios para escribir una dirección de correo, si es un tipo date coloca un calendario en el que se elige una fecha. El objetivo de esta nueva tecnología es la de hacer más usables a los sitios web, se requiere de dedicación por parte del programador para entender todas sus características.

**2.8.2 Ejemplo pagina básica con HTML5**

Este es el ejemplo de un blog escrito en HTML5:

1 <!DOCTYPEhtml>

2 <html>

3 <head>

4 <title>Mi Primera Página en HTML</title>

5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

6 <style type="text/css">

7 body {

8 background-color: #fff;

9 margin: 40px;

10 font: 13px/20px normal Helvetica, Arial, sans-serif;

11 color: #4F5155;

12 }

13 </style>

14 </head>

15 <body>

16 <header>

17 <hgroup>

18 <h1>El blog de prueba</h1>

19 <h2>Este es un blog de prueba que muestra las sigularidades de HTML5</h2>

20 </hgroup>

21 </header>

22 <nav>

23 Aquí va la botonera de navegación

24 </nav>

25 <section>

26 <article>Aquí va un post, con su titulo en h2</article>

27 <article>Aquí va un post, con su titulo en h2</article>

28 <article>Aquí va un post, con su titulo en h2</article>

29 </section>

30 <aside>

31 Barra lateral con cosas que pueden ser de interes para el visitante,

32 como cuentas de twitter, facebook, posts viejos, etc.

33 </aside>

34 <footer>

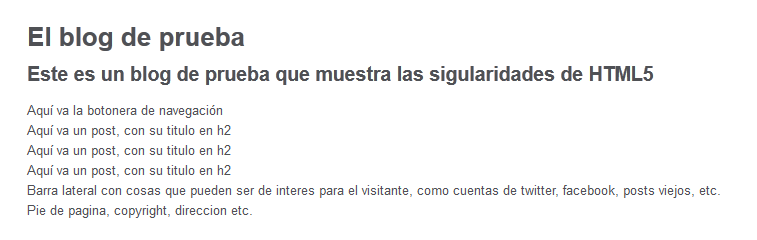
35 Pie de pagina, copyright, direccion etc.

36 </footer>

37 </body>

38 </html>

Este es el resultado que genera el código anterior:



Tal vez no era lo que esperábamos luego de escribir tantas etiquetas, pero lo bueno de todo esto es que el sitio está mejor estructurado y los buscadores lo toman en cuenta para mostrarlo en sus resultados, así que vale la pena ponerle el hombro a esta tecnología.

**2.9. Introducción a CSS**

CSS Cascading Style Sheet traducido al español significa **Hojas De Estilo En Cascada**, CSS es un lenguaje creado para controlar el aspecto y presentación de una página web, esta tecnología es imprescindible para crear una página web compleja, el principal objetivo de esta tecnología es separar el documento de su presentación.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML en cualquiera de la versiones existentes para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

CSS es parte del estándar HTML5, en este subcapítulo no vamos a dedicarnos a entender CSS como lo hicimos con HTML, ya que en nuestro proyecto vamos a usar un Framework CSS llamado Boostrapt el cual solamente trabaja con el atributo “class“ de las etiquetas HTML, mi intención con este subcapítulo es revisar lo necesario de CSS para hacer funcionar le Framework.

**2.9.1. Historia CSS**

CSS, se encuentra actualmente en la versión 3 de la cual solo se han publicado borradores, este estándar está en el mercado desde hace algunos años, aunque la versión final aún no ha sido liberada ya está soportado por la mayoría de los navegadores, la primera publicación del estándar fue liberado por la W3C en el año 1996, se creó por la necesidad de hacer que los sitios tengan una estructura visual, cosa que no es capaz de proveer HTML, vista esta necesidad se hicieron propuestas a la W3C esta entidad tomó lo mejor de dos estándares recibidos y publico CSS nivel 1 en 1996, desde entonces se continuo con el desarrollo de esta tecnología, en la actualidad CSS es capaz de realizar animaciones por hardware por solo mencionar una de sus capacidades, esta es una tecnología que cualquier desarrollador web debe conocer, y es imprescindible para un diseñador web.

**2.10.1. Sintaxis Básica**

Para escribir código CSS la sintaxis es muy simple, consiste en escribir el nombre de la etiqueta HTML a la que queremos aplicarle un estilo (esto afectará a todas la etiquetas que tengan ese nombre) o bien el nombre de la capa o bloque seguido de una llave “{“ y luego los estilos que queremos aplicar a esto se lo conoce como selectores a los cuales de dedicaremos tiempo en el siguiente subcapítulo, una vez que tenemos todo el código listo, cerramos las llaves“}” , cada línea de código termina con unto y coma “;”, los códigos CSS asignan valores a las propiedades asignadas a cada etiqueta, primero se escribe una propiedad y luego se le asigna un valor, para asignar valores en CSS no se usa el signo igual se usa los dos puntos “:”, este es un ejemplo de código CSS:

selector {

parametro:valor;

parametro:valor;

...

}

Como podemos ver el selector contienen entre llaves los atributos que se aplican al grupo de etiquetas que coincidan con este parámetro.

**2.10.2. Selectores**

Los selectores en CSS son los encargados de asignar estilos a una etiqueta o grupos de etiquetas, este es el nombre de una etiqueta, clase o id al que queremos asignar un estilo, los selectores se usan para identificar a las etiquetas, como sabemos todos las etiquetas HTML soportan los atributos “id” y “class” los cuales sirven para identificar y agrupar respectivamente y que cada etiqueta tiene sus propios atributos como la etiqueta “input”, CSS puede seleccionar un elemento de la páginahaciendo referencia a cualquier atributo, a continuación una tabla en las que se muestran los selectores más usados que soporta CSS tomado de la documentación oficial.[16]

|  |  |  |  |
| --- | --- | --- | --- |
| Selector | Ejemplo | Descripción | CSS |
| .class | .intro | Selecciona las etiquetas conclass="intro" | 1 |
| [#id](http://www.w3schools.com/cssref/sel_id.asp) | #firstname | Selecciona la etiqueta con id="firstname" | 1 |
| [\*](http://www.w3schools.com/cssref/sel_all.asp) | \* | Selecciona todos las etiquetas | 2 |
| [etiqueta](http://www.w3schools.com/cssref/sel_element.asp) | p | Selección todas las etiquetas <p> | 1 |
| [etiqueta,etiqueta](http://www.w3schools.com/cssref/sel_element_comma.asp) | div,p | Selecciona todas las etiquetas<div>y <p> | 1 |
| [etiquetaetiqueta](http://www.w3schools.com/cssref/sel_element_element.asp) | div p | Selecciona todas las etiquetas <p>que tengan una etiqueta dentro <div> | 1 |
| [etiqueta>etiqueta](http://www.w3schools.com/cssref/sel_element_gt.asp) | div>p | Selecciona todas las etiquetas <p>dentro de la etiquetas <div> | 2 |
| [etiqueta+etiqueta](http://www.w3schools.com/cssref/sel_element_pluss.asp) | div+p | Selecciona todas las etiquetas <p> que tienen un etiqueta<div>como siguiente elemento | 2 |
| [[atributo]](http://www.w3schools.com/cssref/sel_attribute.asp) | [target] | Selecciona etiquetas por un atributo | 2 |
| [[atributo=valor]](http://www.w3schools.com/cssref/sel_attribute_value.asp) | [target=\_blank] | Selecciona etiquetas con target="\_blank" | 2 |
| [:link](http://www.w3schools.com/cssref/sel_link.asp) | a:link | Selecciona link sin visitar | 1 |
| [:visited](http://www.w3schools.com/cssref/sel_visited.asp) | a:visited | Selecciona link visitados | 1 |
| [:active](http://www.w3schools.com/cssref/sel_active.asp) | a:active | Selecciona links activos | 1 |
| [:hover](http://www.w3schools.com/cssref/sel_hover.asp) | a:hover | Selecciona links que tengan el puntero del mouse encima | 1 |
| [:focus](http://www.w3schools.com/cssref/sel_focus.asp) | input:focus | Selecciona etiquetas imput con foco | 2 |
| [:first-line](http://www.w3schools.com/cssref/sel_firstline.asp) | p:first-line | Selecciona la primera etiqueta <p> del documento | 1 |
| [[attribute^=value]](http://www.w3schools.com/cssref/sel_attr_begin.asp) | a[src^="https"] | Selecciona etiquetas<a>donde src tenga como valor "https" | 3 |
| [:root](http://www.w3schools.com/cssref/sel_root.asp) | :root | Selecciona etiquetas de la raíz del documento | 3 |
| [:empty](http://www.w3schools.com/cssref/sel_empty.asp) | p:empty | Selecciona etiquetas<p>que estén vacías. | 3 |
| [:enabled](http://www.w3schools.com/cssref/sel_enabled.asp) | input:enabled | Selcciona etiquetas <input> disponibles | 3 |
| [:disabled](http://www.w3schools.com/cssref/sel_disabled.asp) | input:disabled | Seleccion etiquetas<input>deshabilitadas | 3 |
| [:checked](http://www.w3schools.com/cssref/sel_checked.asp) | input:checked | Selecciona etiquetas <input>tipo checkbox con check (signo de visto en el cuadro) | 3 |
| [:not(selector)](http://www.w3schools.com/cssref/sel_not.asp) | :not(p) | Selecciona todas las etiquetas menos las etiquetas <p> | 3 |
| [::selection](http://www.w3schools.com/cssref/sel_selection.asp) | ::selection | Selecciona las etiquetas seleccionadas por el usuario. | 3 |

Como podemos ver los selectores CSS pueden seleccionar lo que sea, debemos recordar que en esta tabla están los más usados. El Framework CSS que vamos a usar se llama Bootstraplos selectores más usados son “class” y el “id” por lo que no hace falta ahondar en el tema de selectores.

**2.10.3. Propiedades**

Las Propiedades son elementos CSS que toman valores que son asignados por el programador, estas propiedades son las encargadas de establecer efectos visuales, las propiedades CSS hacen cambios en la presentación de las etiquetas, a cada elemento o etiqueta de la página web se le pueden definir distintas propiedades las cuales, a su vez pueden tomar distintos valores. Así por ejemplo, una propiedad puede ser el color de fondo y los valores podrían ser, azul, verde, naranja, etc.

Estas propiedades pueden ser agrupadas para su fácil interpretación en estos tipos:

1. Propiedad color
2. Fuentes y Textos
3. Bordes
4. Márgenes
5. Propiedades para Listas
6. Tablas
7. Dimensiones
8. Efectos Visuales
9. Ubicación

**Propiedad de color**: Este tipo de propiedades son usadas para colorear la página.

|  |  |
| --- | --- |
| Propiedades | Descripción |
| background-color | Aplicar color de fondo a diferentes elementos. |
| background-image | Utilizar una imagen como fondo. |
| background-repeat | La imagen de fondo se repite. |
| background-attachment | Dejar fija la imagen de fondo. |
| background-position | Ubicar una imagen en un lugar determinado. |

**Propiedades de texto y fuentes**: En esta sección tenemos un listado de propiedades usadas para manejar los estilos de los textos.

|  |  |  |
| --- | --- | --- |
| Propiedades | | Descripción |
| text-indent | | Como dejar sangría en un texto. |
| text-align | | Centrar un texto. |
| text-decoration | | Definir un texto tachado o subrayado. |
| text-transform | | Convertir un texto a minúsculas o mayúsculas. |
| letter-spacing | | Controlar el espacio entre letras. |
| word-spacing | | Controlar el espacio entre palabras. |
| white-space | | Definir la ubicación del salto de línea. |
| color | | Colorear el texto a su gusto. |
| direction | | Definir el sentido de la escritura. |
| font-family | Definir un tipo de fuente. | |
| font-style | Escribir en itálica. | |
| font-variant | Variar la fuente a mayúsculas más pequeñas. | |
| font-weight | Como darle intensidad a la fuente. | |
| font-size | Definir el tamaño de la fuente. | |

**Propiedades para bordes**: Este tipo de propiedades trabajan con los bordes, se los usa para trabajar con etiquetas <div> o <table>, aunque se las puede usar en cualquier etiqueta.

|  |  |
| --- | --- |
| Propiedades | Descripción |
| border-width | Establecer el ancho del borde del elemento. |
| border-color | Colorear el borde. |
| border-style | Uso de bordes ocultos. |
| Uso de bordes punteados. |
| Uso de bordes a rayas. |
| Uso de bordes solidos. |
| Uso de bordes dobles. |
| Uso de bordes sombreados. |
| Uso de bordes en relieve. |
| Uso de bordes en recuadro. |
| Uso de bordes resaltados. |

**Propiedades para márgenes**: Establece márgenes en las etiquetas.

|  |  |
| --- | --- |
| Propiedades | Descripción |
| margin | Como dejar un márgen alrededor de un elemento con medidas. |
| margin | Como dejar un márgen alrededor de un elemento con porcentajes. |

**Propiedades para listas**: Ayudan a cambiar los marcadores de las listas, HTML solo soporta listas ordenadas (marcándolas con un número consecutivo a la izquierda) y desordenadas (marcándolas con un punto a la izquierda), con esas propiedades se puede cambiar ese marcador de acuerdo a las opciones listadas en la tabla.

|  |  |
| --- | --- |
| Propiedades | Descripción |
| list-style-type | Aplicar diferentes marcadores a los Ítems de una lista desordenada. |
| Aplicar números decimales y números romanos en listas ordenadas. |
| Aplicar letras, números y blancos para listas ordenadas. |

**Propiedades para tablas**: Trabaja con los bordes y el tamaño de las celdas de una tabla.

|  |  |
| --- | --- |
| Propiedades | Descripción |
| table-layout | Como controlar el tamaño de las celdas de una tabla. |
| border-collapse | Definir un borde separado o un borde de una línea fina en una tabla. |

**Propiedades para dimensiones**: Este tipo de propiedades nos ayuda a estructurar la página en cuanto a las dimensiones de los elementos

|  |  |
| --- | --- |
| Propiedades | Descripción |
| width | Defina el ancho de un párrafo a su gusto. |
| min-width | Defina un ancho mínimo para un párrafo. |
| max-width | Como definir un ancho máximo para un párrafo. |
| height | Como establecer la altura de un elemento. |
| min-height | Defina una altura mínimo para un elemento. |
| max-height | Defina una altura máxima para un párrafo. |
| line-height | Declare una separación entre líneas. |

**Propiedades para Efectos Visuales**: Estas propiedades ayudan a mostrar el contenido de diferentes maneras, nos ayuda a estructurar los contenidos de las etiquetas.

|  |  |
| --- | --- |
| Propiedades | Descripción |
| overflow | Insertar una barra de desplazamiento en caso que el contenido sea superior a la caja que lo contiene |
| clip | Mostrar parte de una imagen |
| display | Como convertir diferentes elementos en Ítems de una lista |
| visibility | Mostrar u ocultar un texto |

**Propiedades para Ubicación**: Dentro de la maquetación o construcción de un sitio estas propiedades nos ayudan mucho a la hora de colocar un elemento en un lugar de la pantalla, gracias a estas propiedades podemos hacer realidad los diseños creados en papel.

|  |  |
| --- | --- |
| Propiedades | Descripción |
| position | Posiciona un elemento dentro de la página por medio del método estático |
| Por medio del método relativo |
| Por medio del método absoluto |
| Por medio del método fijo |
| top right bottom left | Ubicar un elemento por medio de estas propiedades. |
| float | Posicionamiento flotante a la izquierda. |
| Posicionamiento flotante a la derecha |
| clear | No permite elementos flotantes a la izquierda. |
| No permite elementos flotantes a ambos lados |
| vertical-align | Alineación vertical del texto : Línea de referencia. |
| Por debajo |
| Por arriba |
| Por encima |
| Texto superior |
| Medio |
| Inferior |
| Texto inferior |
| Longitud |
| Porcentaje |
| z-index | Ubica una imagen en una capa inferior al texto. |

Estas son las propiedades más usadas a la hora de crear una página web, pero son solo una porción de la infinidad que existen si se desea conocer más visite la página de la W3C[16].

**2.10.5. Ejemplo De Una Página Con CSS**

Como vemos son demasiadas las posibilidades para crear un diseño único, además el nivel de personalización es increíblemente bueno, se puede aplicar un estilo a casi todas las etiquetas HTML.

Para este ejemplo quiero mostrar una página sin CSS y otra con CSS para que se pueda apreciar la diferencia que existe.

Veamos el código HTML de la pagina y luego vamos a ver el código CSS para esta página, cada tipo de código estará en un archivo diferente se usara la etiqueta <link> para conectar los dos archivos.

**Pagina HTML:**

1 <!DOCTYPE html>

2 <html lang="es">

3 <head>

4 <link rel="stylesheet" href="estilo.css"/>

5 <title>Quito</title>

6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

7 </head>

8 <body>

9 <div class="contenedor">

10 <h1 class="titulo">Quito Capital del Ecuador</h1>

11 <div class="articulo">

12 San Francisco de Quito, es la ciudad capital de la República

13 del Ecuador y también de la provincia de Pichincha, es la se

14 gunda ciudad más grande y poblada del Ecuador.<br/> Además es cab

15 ecera cantonal o distrital del Distrito Metropolitano de Qui

16 to.<p>Está ubicada sobre la hoya de Guayllabamba en las laderas

17 orientales del estratovolcán activo Pichincha, en la parte o

18 ccidental de los Andes. Se encuentra aproximadamente en las

19 coordenadas 0°13′23″S 78°30′45″O y su altitud promedio es de 2

20 800 msnm La ciudad está dividida en 32 parroquias, las cuales

21 se subdividen en barrios.</p><p>La fecha de su primera fundación

22 es incierta; los registros más antiguos se hallan en la hacien

23 da del Inga, sin embargo, se utiliza la conquista española de

24 la ciudad, el de diciembre de 1534, como su nacimiento</p>

25 </div>

26 </div>

27 </body>

28 </html>

**Código CSS:**

1 /\*\*

2 \* Inicion del archivo de CSS, CSS soporta comentarios al estilo de c/c++

3 \*/

4

5 body {

6 background-color:#e9e9e9;

7 }

8

9 .contenedor{

10 background-color:#cdcdcd;

11 border-radius: 50px;

12 min-height: 20px;

13 padding: 19px;

14 margin-bottom: 20px;

15 background-color: rgb(245, 245, 245);

16 border: 1px solid rgba(0, 0, 0, 0.05);

17 border-radius: 4px 4px 4px 4px;

18 box-shadow: 0px 1px 1px rgba(0, 0, 0, 4);

19 width: 600px;

20 margin-right: auto;

21 margin-left: auto;

22 }

23

24 .titulo{

25 margin: 12px 12px 14px 14px;

26 color:#5B5C5C

27 }

28

29 .articulo{

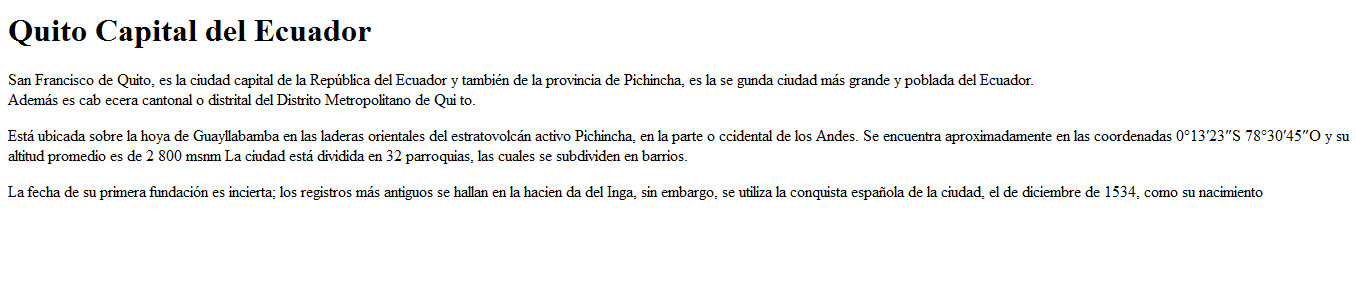
30 margin: 12px 12px 25px 14px;

31 color:#0A1F3A;

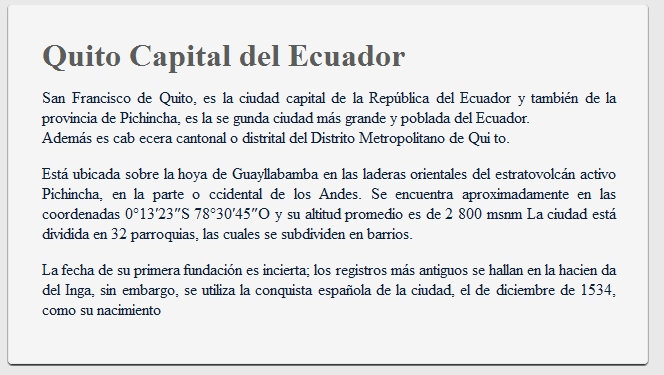
32 text-align:justify;

33 }

**Vista sin CSS:**

****

**Vista con CSS**

****

Lo admito no soy diseñador, pero es mucho lo que se puede hacer con CSS, el ejemplo que presente solo usa unas cuantas propiedades pero la verdad se ve mucho mejor con CSS que si él, si desea probar este código en el navegador debe crear dos archivos un HTML y otro CSS en la misma carpeta copiar los códigos a cada archivo y luego verificar que en la etiqueta <link> este el nombre del archivo CSS.

**2.11. Introducción A Java Script**

Este es un lenguaje de programación orientado a objetos que se ejecuta en la máquina del cliente, este lenguaje es usado principalmente para crear páginas web dinámicas. Paginas dinámicas son aquellas en las se incorpora efectos de animación o presentan contenidos sin la necesidad de recargar la página, también es capaz de mostrar ventanas, solicitar información al usuario, todo esto sin la necesidad de recargar la página de forma transparente al usuario. Java Script es un lenguaje de cliente interpretado, esto significa que no se debe compilar para que funcionen los programas creados con esta tecnología, ya que es ejecutado directamente por el intérprete que viene con el navegador web, y se lo llama de cliente porque los scripts “textos” son enviados desde el servidor hacia el cliente y este los ejecuta, esto ahorra recursos en el servidor y además permite un mayor control de la actividad del usuario en un sitio, ya que esta tecnología es capaz no solo de recibir información del servidor sino que también es capaz de enviarla, esto es muy común, se usa para conocer los hábitos de los usuarios, luego es usado para crear aplicaciones en las que los usuarios permanezcan más tiempo en el sitio.

Java Script es una de las tecnologías que forman parte de la nueva tecnología llamada HTML5, es quizás una de las más importantes del estándar, tiene elementos muy importantes y poderosos como video, audio y una de las mejores capacidades a mi manera de ver, es la capacidad **canvas** esta es una muy poderosa herramienta de HTML5 ya que con esto se puede hacer animaciones gráficos y un sinfín de experimentos muy interesantes que antes eran posibles solamente usando flash.

**2.11.1. Origen Java Script**

Co se mencionó en el artículo anterior, JavaScript es una tecnología que ahorra recursos del servidor, esta fue una de las principales razones por las que fue creada, todo empezó en los años 90 cuando las conexiones a internet no eran tan populares ni tan rápidas como ahora, la velocidad máxima registrada era de 28Kbps, antes presentar una imagen ya era mucho y tomaba muchos recursos, una de las principales necesidades fue identificada a la hora de validar la información enviada por el usuario al servidor, ya que este tenía que revisarlos antes de procesarla si algo estaba mal, en el mejor de los casos se enviaba los datos al cliente nuevamente para que sean corregidos por el usuario, en el peor de los casos el usuario tenía que ingresar la información de nuevo, además de los obvios problemas con la satisfacción de usuario era el desperdicio de recursos del servidor tratando información que no estaba bien digitada.

Para enfrentar estos inconvenientes **Brendan Erich** programador de la empresa Netscape creadora de un navegador web con el mismo nombre crea un lenguaje que se ejecuta en el navegador de usuario nombrado en un inicio como LiveScript.

Luego Netscape hace una alianza con Sun Microsystems para reescribir el lenguaje y se crea el primer estándar al que bautizan con el nombre de JavaScript, se usó ese nombre no porque tenga una similitud con el lenguaje desarrollado por Sun Microsystems Java, sino por la popularidad que tenía la palabra Java en aquel entonces.

Nota: Netscape es una empresa que ya no existe desapareció con la llegada de navegadores como Internet Explorer Firefox Opera entre otros, pero nos dejó a una de las más importantes tecnologías de hoy en día en la web. Sun Microsystems tampoco existe la misma fue vendida hace algunos años a Oracle, Oracle mantiene la mayoría de proyectos que Sun Microsystems llevaba a cabo como son el popular lenguaje de programación Java, su popular motor de base de datos MySql también retiro del mercado a muchos servicios y productos como el sistema operativo tipo UNIX llamado Solaris que era un sistema especial para servidores, este fue reemplazado por una distribución Linux llamada **Linux Umbrecable** la cual tiene mucha fuerza a la hora de trabajar en servidores dedicados.

**2.11.2. Usos Java Script**

Se puede decir que no hay un sitio web sin esta tecnología, ya que es muy importante no solo para mejorar la experiencia de usuario, sino también es importante para crear sitios funcionales y de calidad con animaciones y elementos interactivos en el sitio al ser parte de un estándar hace que un producto que se desarrolle con esta tecnología funciona igual en cualquier navegador y/o dispositivo desde el que se visualice una página web.

Este es un lenguaje al que muchas personas le dedican mucho trabajo, se han creado un sinfín de Frameworks de los cuales solo mencionare unos pocos

* JQuery
* JQueryUI
* Wijmo
* Motools
* KendoUI
* Bootrapt
* Boilerplate
* Etc.

Se puede ver que esta tecnología es muy utilizada por lo que no debemos dejarla pasar por alto, en esta sección vamos a estudiar un poco este lenguaje aunque no lo vamos a usar directamente en el proyecto ya que Boostrapt lo usa, pero es necesario su revisión para al menos conocer su funcionamiento.

¿Cómo usar JavaScript?

Al igual que con CSS existen dos formas de usar esta tecnología en una página web la primera es incrustando el códigos directamente en la página con la ayuda de la etiqueta **<script>**, también se los puede usar en archivos separados y fusionarlos con las páginas con la misma etiqueta <script> la única diferencia es que se debe llenar el atributo **“src”** con la dirección de la ubicación del archivo JavaScript, esto se debe hacer dentro de las etiquetas de la cabecera **<header>**.

**2.11.3. Sintaxis basica**

La sintaxis de JavaScript fue infuenciada por C y C++ lo que significa que cada línea de instrucción termina con un “;” punto y coma, este es un lenguaje orientado a objetos, veamos la forma de declarar alguas cosas muy útiles a la hora de programar.

**Comentarios**

Los comentarios en JavaScript funcionan de manera similar que en C y C++.

1 //comentario de una sola línea

2

3 /\*\*

4 \* Comentario

5 \* de

6 \* varias

7 \* líneas

8 \*/

**Anclar Java Script en HTML**

Como se mencionó en el artículo anterior existen dos formas de anclar archivos JavaScript a una página HTML veamos ambos.

1 <!DOCTYPE html>

2 <html>

3 <head>

4 <title>Página con JavaScript</title>

5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

**6 <script type="text/javascript">**

**7 alert('Código escrito en HTML directamente ')**

**8 </script>**

9 </head>

10 <body>

11 <div>Página con JavaScript</div>

12 </body>

13 </html>

Ahora veamos la segunda forma.

1 <!DOCTYPE html>

2 <html>

3 <head>

4 <title>Página con JavaScript</title>

5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

**6 <script type="text/javascript" src="nombre\_archivo.js"/>**

7 </head>

8 <body>

9 <div>Pagina Con Java Script</div>

10 </body>

11 </html>

Luego de ver estas dos alternativas queda una pregunta. ¿Cuál de las dos usar y cuando usarlas?, no tengo una respuesta exacta solamente diré que depende… Depende de las circunstancias y del uso que se le vaya a dar a la aplicación, si ponemos todos nuestros Scripts en archivos separados se pueden compartir con todas las páginas del sitio, optimizando recursos, si se necesita implementar una característica en especial para una página, se puede incluir el código directamente en la página.

**Variables**

Una variable es un espacio de memoria en donde se pueden almacenar datos, Las variables de JavaScript son definidas directamente, no hace falta escribir el tipo de dato de la variable como se lo hace en C/C++ sino que se escribe directamente el nombre de la variable, aunque también se puede definir una variable con la palabra clave **“var”,** la diferencia es quesi se las declara directamente deben ser utilizadas, con la palabra clave **“var”** se crean variables antes de su uso, variables vacías.

1 <!DOCTYPE html>

2 <html>

3 <head>

4 <title>Pagina con JavaScript</title>

5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

6 <script language="javascript">

7

8 //creo una variable

9 var x;

10 //x actualmente no tiene ningún valor

11 x = 25;

12 //ahora x tiene el valor numérico 25

13

14 //creo una segunda variable

15 var y = 230;

16 //he creado una variable y asignado un valor en un solo paso!!

17

18 //las variables guardan datos con los que puedo realizar operaciones

19 var suma;

20 suma = x + y;

21 //he creado la variable suma y he asignado la suma de x e y

22 alert(suma);

23 //he mostrado el valor de la variable suma

24

25 </script>

26 </head>

27 <body>

28 <div>Pagina Con Java Script</div>

29 </body>

30 </html>

Se puede apreciar que existen muchas formas de declarar una variable, el ámbito de las variables depende del lugar en donde son creadas, si una variable es creada dentro de una función esta es solamente vista por la función a la que pertenece, si se lo crea al inicio del Script puede ser vista por toda la aplicación.

**Funciones**

Las funciones son otro elemnto primordial en los lenguajes de programación, las funciones en JavaScript son declaradas con la palara clave **“function”** el cuerpo de la función va entre llaves.

1 function sumarValores(numa, numb) {

2 //cuerpo de la funcion

3 valor = (numa+numb);

4 return valor;

5 }

6

7 //invoco a la funcion

8 respuesta = sumarValores(20,98);

9

10 //muestro el resultado en un message box

11 alert('La suma es: ' , respuesta);

Para invocar a una función se usa el nombre de la función seguido de paréntesis en los que se incluyen parámetros en caso de que la función las requiera.

**Clases y Métodos**

En un inicio se mencionó que JavaScript es un lenguaje orientado a objetos por lo tanto soporta todas las características que revisamos en el capítulo de la POO, como sabemos las clases y los métodos son quizás lo más importante de este paradigma de programación, veamos cómo se implementan en JavaScript.

1 function MiClase (valor\_inicializacion){

2 //Inicializo las propiedades y métodos

3 this.miPropiedad = valor\_inicializacion

4 this.miMetodo = nombre\_de\_una\_funcion\_definida

5 }

6

7 //creo el objeto a partir de la clase

8 objeto = new MiClase();

9

10 //acceder a los metodos de un objeto

11 objeto.metodo(argumentos)

12

13 //accedemos a un propiedad del objeto

14 objeto.miPropiedad

Cuando se aprende a programar con un paradigma es menos difícil pasar de un parámetro a otro, ya que se puede usar los mismos algoritmos lo único que cambia es el lenguaje y esto es así por la peculiaridad de los lenguajes.

**2.11.4. Tipos De Datos**

JavaScript soporta todos los tipos de datos conocidos, a diferencia de C o C++ estos no están predefinidos, son asignados de manera automática por el intérprete a una variable, el programador solamente puede dar nombre a una variable, no puede especificar el tipo de dato que esta contendrá más adelante aunque hay una posibilidad que si lo permite, si desea que la variable tenga un tipo de dato determinado puede asignarle un valor en el momento de la definición, revisemos los tipos de datos más conocidos y útiles en JavaScript:

**2.11.4.1. Numéricos**

Este tipo de variables son usadas para almacenar números para luego poder usarlos en operaciones matemáticas, como hablamos JavaScript asigna el tipo de variable de acuerdo al valor que se le asigna la primera vez que se la usa puede tener valores enteros, flotantes o reales, luego de esto no puede cambiar el tipo de dato.

**2.11.4.2. Cadenas**

Las variables de este tipo se utilizan para almacenar caracteres y cadenas de caracteres, para hacer que una variable sea una cadena se le debe asignar el valor “Cualquier Texto” entre comillas dobles o sencillas las mismas que delimitan el contenido de la variable.

**2.11.4.3. Booleanos**

Este tipo de variable es quizás uno de los más primitivos pero aún es muy útil a la hora de desarrollar, como debe imaginarse el lector los valores que posee esta variable son “True” o “False” verdadero o falso

**2.11.4.4. Nulos**

En JavaScript todos las variables a las que no se le haya asignado un dato son del tipo NULL, este tipo de dato tambien es usado para resetear una variable como cuando en C hacemos **x = 0**

**2.11.4.5. Objeto**

Esta es la instanciación de una clase, y como dijimos en el capítulo de POO el tipo de dato del objeto depende de la clase, para crear un objeto en JavaScript se usa la palabra clava **new()** seguido por paréntesis en el cual se ponen valores de inicialización en caso de que la clase los requiera.

**2.11.4.6. Arreglo**

Los arreglos o colecciones de datos son usados para representar valores con algún parentesco como por ejemplo el listado de productos de una factura o una lista de tareas para determinada actividad, los arreglos en JavaScript se crean con las palabras claves **“new array()“** dentro del paréntesis puede ir un número que indica la longitud del arreglo, caso contrario la longitud es variable. Un ejemplo práctico seria almacenar en un arreglo los detalles de un producto como nombre, precio, marca, fecha elaboración, etc. Esto es posible ya que los arreglos pueden contener varios tipos de datos a la vez.

**2.11.4.7 Ejemplos De Tipo De Datos**

1 //uso de variables en JavaScript

2 // Numericos

3 var entero = 45;

4 var flotante = 45.09;

5 //una variable entera en cero

6 var entero\_cero = 0;

7 //una variable flotante en cero

8 var flotante\_cero = 0;

9 // Cadenas

10 //El numero esta como cadena, no funciona para operaciones matematicas

11 var cadena = '34';

12 var cadena2 = 'Esta es otra cadena';

13 var nombre = "asignacion con comillas dobles";

14 //tambien se puede hacer al revez comillas dobles en comillas simples

15 var concomillas = "este texto aparece en 'comillas simples'"

16

17 // Booleanos

18 var login = true;

19 var salir = false;

20

21 //Nulos

22

23 //por defecto esta variable es null

24 var nombre;

25 var apellido = null;

26

27 //Arreglos

28 //arreglo de longitud indefinida

29 var mi\_arreglo = new array();

30 // arreglo con logitud definida

31 var mi\_arreglo2 = new array(100);

32

33 // Objetos

34 //los arreglos son objetos

35 persona = new array(8);

36 //declaracion de un objeto

37 mi\_objeto = new objeto();

**2.11.6. Operadores**

Que sería de las variables si no se pudieran hacer operaciones con ellas los operadores nos permiten hacer toda clase de tareas con ellas, en la siguiente tabla vamos a ver los que son parte de JavaScript, para todos los ejemplo vamos a trabajar con la variable “x”.

|  |  |  |
| --- | --- | --- |
| Operador | Descripción | Ejemplo |
| = | Asignación (Binario) | x = “valor” |
| ++ | Incremento, aumenta en uno al valor, si va antes de la variable la variable es incrementada antes de ser usada, caso contrario es incrementada después. (Unitario se aplica a una sola variable) | x++ <o> ++x |
| -- | Decremento en uno, se aplica los mismos principios del incremento. (Unitario) | x -- <o> --x |
| + | Suma (binario se usa con dos variables) | x = a + b <o> x = d +9 |
| - | Resta (binario) | x = a - g <o> x = h - p |
| / | División (binario) | x = a / b <o> x = a/98 |
| \* | Multiplicación (binario) | x = a + b <o> x = a\*3 |
| % | Modulo (binario), toma el residuo de una división, su valor es cero cuando la división es exacta. | x = a % 2 |
| < | Mayor que (booleano resultado verdadero o falso) | 2 < 3 = true <o> 3 < 2 = false |
| > | Menor que (booleano) | 1 > 3 = true <o> 3 > 1 = false |
| == | Igualdad (booleano) | 1 == 1 = true <o> true == false = false |
| =! | Diferente de (booleano) | 3 != 3 = false <o> 3 != 4 = true |
| <> | Diferente de (booleano) | 3 <> 3 = false <o> 3 <> 4 = true |
| <= | Menor o igual que (booleano) | 1 <= 2 = true <o> 2 <= 1 = false |
| >= | Mayor o igual que (booleano) | 2 >= 1 = true <o> 1 >= 2 = false |
| ! | Negación (booleano) | !true = false <o> !false = true |
| && | Y (Booleano) | true && true = true  true && false = false  false && false = false |
| | | O (booleano) | true || false = true  false || true = true  true || true = true  false || false = false |

Estos fueron los operadores más usados a la hora de programar, los binarios son los que necesitan dos elementos para ser usados, los booleanos son aquellos que retornan un valor booleano sea verdadero o falso además son binarios, y los unitarios son aquellos que trabajan sobre una sola variable como los de incremento y decremento.

**2.11.7. Control De Flujo**

Sin las estructuras de control no es posible crear programas funcionales, gracias a estos operadores podemos optimizar operaciones repetitivas, y como dice su nombre controlar el flujo de las aplicaciones, estos flujos son controlados por los operadores vistos en el subcapítulo anterior. Estas sentencias trabajan con valores booleanos es decir true o false, si se cumple la condición se procede a realizar determinadas acciones de no cumplirse se ejecuta otras acciones o si se desea no se puede hacer nada cuando sea falso. A continuación se listan los más importantes.

**2.11.7.1. Estructura De Control If Else**

Esta es una de las estructuras más utilizadas en JavaScript al igual que en varios lenguajes de programación, esta es un estructura muy útil a la hora de tomar decisiones, si la condición se cumple se ejecutan un serie de instrucciones si no se cumple se ejecutan otras serie de instrucciones, este condicional puede ser anidado para cuando se desee tener un listado de opciones, aunque existe otra sentencia que maneja de mejor manera esas situaciones.

**Ejemplo:**

1 //estructura de control IF Else

2 if (nombre == 'Eduardo'){

3 alert('El usuario conectado es ' + nombre);

4 }else{

5 alert('El usuario conectado no es Eduardo');

6 }

Como podemos ver primero se analiza la condición y luego se ejecutan las instrucciones.

**2.11.7.2. Estructura De Control Switch**

Este es una estructura que permite listar una serie de condiciones, permite ejecutar determinadas líneas de código para cada opción en caso de que alguna sea verdadera, si se deseaba crear demasiadas condicione **if** anidadas se puede usar un solo **switch** para solucionar el problema.

**Ejemplo:**

1 //estructura de control IF Else

2 var dia = 'Lunes'

3 switch(dia){

4 case 'Lunes':

5 alert('Hoy es ' + dia + ' a trabajar!');

6 break;

7 case 'Martes':

8 alert('Hoy es ' + dia + ' sigue trabajando!');

9 break;

10 case 'Miercoles':

11 alert('Hoy es ' + dia + ' tienes pico y placa');

12 break;

13 case 'Jueves':

14 alert('Hoy es ' + dia + ' hay que revisar el trabajo');

15 break;

16 case 'Viernes':

17 alert('Hoy es ' + dia + ' último día de trabajo!.');

18 break;

19 case 'Sabado':

20 alert('Hoy es ' + dia + ' hay que arreglar la casa');

21 break;

22 case 'Domingo':

23 alert('Hoy es ' + dia + ' Ya es fin de semana a descansar');

24 break;

25 }

En esta estructura de control tenemos una sentencia muy peculiar **“break”**, como podemos ver al final de las instrucciones de cada caso tenemos la sentencia **“break”**, esta sentencia se encarga de romper la estructura, la rompe y continua analizando la siguiente condición, entonces cuando se ejecuta un case y este evalúa la condición como falso se ejecuta el **“break”** y se continua con la siguiente sentencia.

Esta sentencia no solamente es compatible con la estructura **“switch”** se puede usar con todas las estructuras de control, se dice que su uso de desaconsejable, pero la verdad durante mi vida profesional no he tenido problemas con estas sentencias.

**2.11.7.3. Estructura De Control For y For In**

Estas son estructuras repetitivas que se ejecuta siempre que la condición sea verdadera, dentro de la declaración de la estructura **“for”** se puede delimitar el rango de la estructura con expresiones matemáticas, se las usa para repetir varias veces un conjunto de sentencias, este tipo de estructura es más utilizada para recorrer vectores y matrices, tarea para la cual son muy buenos, aunque existe una estructura más óptima para este trabajo llamada **“for in”** esta estructura a diferencia de la anterior solamente puede ser usada para recorrer vectores, por ende no se necesita definir su rango, lo define automáticamente al recibir el vector a recorrer, veamos un ejemplo en el que primero se buscan a todos los números pares que existan entre cero y cien y luego a cada número se lo multiplica por diez, los números pares los vamos a definir con **“for”** y para multiplicarlos los vamos a recorrer con **“for in”**.

**Ejemplo:**

1 //estructura de control For y For in

2 var numeros\_pares = new array()

3 for(i=0;i<=100;i++){

4 if(i%2 == 0){

5 numeros\_pares(i)= (i) ;

6 }

7 }

8

9 for(numero in numeros\_pares){

10 alert('Par x 10 = : ' + (numero + 10) );

11 }

**2.11.7.4. Estructura De Control While**

Esta es una estructura de control similar al **for** con la diferencia de que este se ejecuta siempre que la condición sea verdadera, la diferencia con **for** es que **for** delimita automáticamente su rango, mientras que **while** se ejecuta siempre que la condición sea verdadera, nada impide al programador recorrer arreglos con **while**.

**Ejemplo:**

1 //estructura de control While

2 var suma = 0;

3

4 while (suma < 1000){

5 suma += parseInt(Math.random() \* 100);

6 document.write (suma + "<br>");

En este ejemplo sumamos números randonómicos hasta que suma sea menor que mil.

**2.11.7.5. Eventos en JavaScript**

Esta características es una de las más potentes de JavaScript, hasta este momento las podemos escribir un aplicación que se ejecuta de principio a fin, aunque se podría modificar este comportamiento con las estructuras de control no es suficientemente funcional, los eventos ayudan a que se dispare una parte de ese código, en lugar de ejecutar todo el script.

Con la ayuda de los eventos nuestros programas esperan a que es usuario haga algo en la página, algo como: pulsar una tecla, mover el mouse, pasar el mouse por un determinado lugar de la pantalla, dar clic en algún elemento de la página, enviar un formulario, llenar un campo, cerrar la ventana del navegador, etc. Inmediatamente después de ese evento la aplicación responde ejecutando código que se haya asociado a ese evento, a esas funciones se las conoce como manejadores de eventos o en inglés **“event handlers”**, generando un resultado y permitiendo interactuar con el usuario, alguna vez el lector puede haber visto en un sitio, que después de darle clic a cerrar ventana, antes de cerrar la ventana nos muestra un mensaje de si estamos seguros de salir del sitio, estoy seguro que sí.

A los eventos se los usa para detectar la actividad del usuario, con ella se puede detectar cuando el usuario mueve el mouse o cuando lo pasa por encima de algún elemento, y muchas cosas más que veremos en la tabla de eventos, lo que se hace es detectar las costumbres del usuario a lo hora de usar nuestro sitio web, para mejorar la funcionalidad del sitio de acuerdo a las estadísticas, por ejemplo si hay un botón que es pequeño que lleva a una sección x en nuestro sitio, y vemos que este es muy usado por los visitantes, podemos hacer que el botón sea más grande o ubicarlo en un lugar más visible, ese es solo un ejemplo de lo que se puede obtener con esta tecnología.

Tabla de eventos básicos soportados por JavaScript

|  |  |  |
| --- | --- | --- |
| Evento | Descripcion | Elementos para los que esta definido. |
| onblur | Deseleccionar el elemento. | <button>, <input>, <label>, <select>,<textarea>,<body> |
| onchange | Deseleccionar un elemento que se ha modificado. | <input>,<select>,<textarea> |
| onclick | Pinchar y soltar el ratón. | Todos los elementos |
| ondbclick | Pinchar dos veces seguidas con el ratón. | Todos los elementos |
| onfocus | Seleccionar un elemento | <button>,<input>,<label>,  <select>,<textarea>,<body> |
| onkeydown | Pulsar una tecla (sin soltar). | Elementos de formulario y <body> |
| onkeypress | Pulsar una tecla. | Elementos de formulario y <body> |
| onkeyup | Soltar una tecla. | Elementos de formulario y <body> |
| onload | La página se ha cargado completamente. | <body> |
| onmousedown | Pulsar sin soltar un botón del ratón | Todos los elementos |
| onmousemove | Mover el ratón. | Todos los elementos |
| onmouseout | El ratón sale del elemento (pasa por encima de otro elemento). | Todos los elementos |
| onmouseover | El ratón entra en un elemento (pasa por encima del elemento). | Todos los elementos |
| onmouseup | Soltar el botón que estaba pulsado del ratón. | Todos los elementos |
| onreset | Inicializar el formulario (borrar todos los datos ingresados en las cajas). | <from> |
| onresize | Se ha modificado el tamaño de la ventana del navegador. | <body> |
| onselect | Seleccionar un texto. | <input>,<textarea> |
| onsubmit | Enviar el formulario | <form> |
| onunload | Se abandona la página (por ejemplo cerrar el navegador) | <body> |

Ya sabemos los eventos más conocidos, ahora veamos cómo usarlos, no se estudiaran todos los eventos pero se establecerá un estándar para el uso de la mayoría de ellos, si el lector desea estudiarlos más afondo puede visitar la web oficial de la W3C <http://www.w3.org/>.

Como se expresó anteriormente las funciones encargadas de manejar los eventos se llaman **“event handler”** veamos los pasos a seguir para usar un evento.

1. Asignar el evento a una etiqueta
2. Crear una función de acuerdo a las necesidades
3. Asignar la función a ser invocada

Son solamente tres pasos a seguir, también se puede asignar el código directamente en la etiqueta, esto no es recomendable hacer esto porque hace que se genere un cód igo confuso y difícil de mantener, es más fácil mantener una aplicación donde todas las tecnologías usadas están en lugares diferentes. Veamos un ejemplo con más de un evento en la misma página.

1 <!DOCTYPE html>

2 <html>

3 <head>

4 <title>Pagina con JavaScript</title>

5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

6 <script language="javascript">

7

8 function mostrarMensaje(){

9 alert('este es un handler...')

10 }

11

12 function eventoInput(event){

13 var chCode = ('charCode' in event) ? event.charCode : event.keyCode;

14 alert ("el codigo de la tecla presionada es: " + chCode);

15 }

16 </script>

17 </head>

18 <body onload="mostrarMensaje();">

19 <div class="inicio" onclick="alert('Hola mundo...');">Pagina Con Java Script</div>

20 <p class="articulo" onmouseover="alert('EL mouse paso por esta capa');">

21 Pasa el mouse sobre este texto...

22 </p>

23 <form>

24 <input name="nombre" onkeypress="eventoInput(event)">

25 </form>

26 </body>

27 </html>

En el ejemplo podemos ver cómo implementar los métodos a través de manejadores de eventos y directamente.

**2.12. Importancia De Java Script En La Actualidad**

Luego de haber estudiado esta tecnología durante el desarrollo de este capítulo, podemos definir un listado de las utilidades más importantes.

* Crear paginas interactivas
* Generar reportes del uso de la página
* Hacer más usables las paginas
* Permite animar elementos de las paginas

**2.13. Introducción A MySql**

En toda aplicación es necesaria la presencia de una base de datos para gestionar la información y datos importantes de un sitio web, para este proyecto se ha considerado usar el motor de bases de datos llamado MySql, este es un motor muy popular y además es gratuito y de código abierto, se lo puede obtener desde <http://mysql.com> o empaquetado es una aplicación de servidor como WAMP o XAMPP, para este proyecto en particular se usará WAMP para el desarrollo.

Para usar el motor de bases de datos MySql es necesario utilizar un lenguaje llamado **SQL (Struct Query Languaje)** por sus siglas en inglés, su traducción al español es **Lenguaje Estructurado de Consultas**, y es así como se llaman a las sentencias que acepta el intérprete del motor de bases de datos, todo es una consulta o **query** en inglés, este lenguaje es implementado por la mayoría de motores de bases de datos, al ser un lenguaje también posee tipos de datos y estructuras de control, las estructuras de control se usan para crear operaciones avanzadas en las bases de datos, no se las va a revisar en este documento, ya que la base de datos de este proyecto es una estructura lo más simple posible.

Las sentencias o consultas deben ir separadas por un **“;”** un punto y coma esto se puede cambiar con la sentencia **delimiter**, los comentarios se escriben con doble guion **“--”** o con el signo de numeral **“#”** para comentarios de una sola línea o se puede usar **“/\*\* comentario varias líneas\*/”**.

**2.13.1. Motor De Bases De Datos**

Es una aplicación que es capaz de gestionar archivos con información, con o sin relación entre sí, gestiona no solo la información sino también a los usuarios que pueden tratar esa información al igual que las conexiones, sirve también como interprete para responder a las peticiones del usuario o aplicación a la que sirva, estas aplicaciones por lo general tienen implementado un servidor para responder a las consultas.

MySql es un gestor de bases de datos multihilo y multiusuario, es decir que puede gestionar varios usuarios y bases de datos a la vez. Este es un motor de código abierto y gratuito, es el más utilizado para gestionar información de sitios web.

Este gestor posee varios motores, el motor que vamos a usar se llama **innodb**, usaremos este motor ya que soporta bases de datos relacionales, no hay que conocer mucho sobre este motor de echo viene por defecto en las últimas versiones del gestor MySql, pero para usarlo explícitamente es necesario utilizar la sentencia **“engine=innodb;”** al terminar la declaración de la tabla.

**2.13.2. Columna**

La columna es la unidad básica de la estructura de una entidad o tabla, la columna es un elemento de una base de datos que posee un nombre y un tipo de dato asociado a ella, es el nombre de la columna a quien se llama para mostrar sus datos, así como para hacer referencias, restricciones, asociaciones o búsquedas.

**2.13.3. Tabla**

También es conocida como entidad, una tabla es una colección de columnas que almacenan información relacionada a una abstracción del mundo real, a esa abstracción se la conoce como entidad, por ejemplo si queremos almacenar la información de un cliente, creamos una tabla con columnas relacionadas a la información del cliente, columnas como nombre, apellido, dirección, etc. Las tablas también pueden estar relacionadas entre sí, continuando con el ejemplo del cliente se podría crear una nueva tabla llamada factura y relacionarla con cliente para almacenar las compras de los clientes valga la redundancia. Las relaciones se las hace utilizando a las columnas y aplicándoles restricciones.

**2.13.4. Vistas**

Se puede decir que las vistas son una tabla virtual, la vista está formada a partir de la unión de muchas tablas, la vista es un conjunto de sentencias que retornan un resultado, la utilidad de las vistas es combinar el resultado de las tablas para mostrar información más completa. Imaginemos que tenemos dos tablas, una de clientes y otra de ventas, con la ayuda de una vista se puede crear una tabla a la que se la llama vista, que nos muestra un detalle de todas las compras realizadas con ese cliente. Un ejemplo más claro se encuentra en el estudio de las sentencias SQL.

**2.13.5. Procedimiento Almacenado y Funciones**

Los procedimientos almacenados y las funciones son una mejora de las vistas, ya que están en capacidad de escribir en la base de datos, o de hacer cálculos a partir de información recibida llamar a otras funciones o procedimientos almacenados, las funciones y los procedimientos almacenados se los puede comparar con las funciones en los lenguajes de programación.

MySql soporta Procedures y Functions los dos pueden hacer lo mismo lo único que los diferencia es que los Procedures no retornan ningún valor, y las Functions tienen como obligación retornar un valor.

**2.13.6. Clave Primaria**

La clave primaria es una restricción aplicada a una columna, una tabla puede tener una sola clave primaria y puede estar formada por una o más columnas, el objetivo de la clave primaria es crear un índice para ordenar los datos de la tabla y evitar que se repita el valor de una columna en los registros, en el caso de estar formada por más de una columna se evita repetir la combinación.

Este tipo de columna son usadas para darle integridad a los datos de una tabla, con estos recursos podemos hacer que los datos almacenados en la tabla sea únicos para cada registro.

**2.13.7. Clave Foránea**

Esta es una restricción que se aplica a una o varias columnas, la razón de la existencia de estos elementos es la de permitir establecer relaciones entre entidades pertenecientes a una misma base de datos, los tipos de relaciones que estas claves soportan son:

* Uno a uno (1:1)

Esta es una relación en donde la clave foránea es una columna única.

* Uno a muchos (1:n)

Esta es una relación donde las claves de la tabla padre se repiten varias veces en la tabla hija

* Muchos a muchos (n:m)

Esta es una relación que se puede expresar pero no se puede implementar directamente, ya que este tipo de relaciones no son capaces de manejar los motores de bases de datos, para implementar este tipo de relación es necesario crear una tercera tabla con relación (1:n) uno a muchos a las tablas que tienen una relación muchos a muchos, rompiendo la relación muchos a muchos.

**2.13.8. Base De Datos**

Es el conjunto de tablas relacionadas o no, que pertenecen al mismo grupo, a este grupo se le llama base de datos, la base de datos tiene asignado usuarios los mismos que pueden trabajar con los datos que ella almacena.

**2.13.9 Tipos de Datos MySql**

Los tipos de datos en MySql se pueden establecer como tipo de columnas, ya que se le asigna a una columna un tipo de dato a soportar, y es este tipo de dato el único aceptado por la columna para cada registro.

Presento a continuación una tabla con los tipos de datos más usados en MySql, **n** significa que el tipo de dato acepta un entero como parámetro de longitud.

|  |  |  |
| --- | --- | --- |
| Tipo de Dato | Descripción | Ejemplo |
| char(n) | Contiene una cadena de longitud constante, es decir se debe cumplir con el límite para que el motor acepte el dato. | Código char(5) => ‘ABS09’ |
| varchar(n) | Contiene una cadena de longitud variable, el motor acepta el dato siempre y cuando no supere el límite. | Nombre varchar(60) => ‘Eduardo’ |
| int(n) | Contiene un entero de tamaño normal, aunque se puede establecer una longitud. | Metros int() => 652334 |
| tinyint | Es una variación de int se usa para almacenar enteros muy pequeño, usado para representar booleanos. | Soltero tinyint => 1 |
| boolean | Sinónimo semántico de tinyint |  |
| smallint | Sinónimo de int pero con un rango menor puede almacenar desde 0 a 65535. |  |
| mediumint | Sinónimo de int con mayor capacidad de small int. |  |
| float | Numero de punto flotante. | Precio float |
| decimal(n,n) | Tipo de dato que representa a un número, este tipo de dato permite especificar la cantidad de números enteros y la cantidad de decimales. | Decimal precio(4,2) |
| date | Sirve para almacenar fechas en formato (yyyy-mm-dd) | Facha date |
| datetime | Almacena un dato que contiene la fecha y hora de acuerdo al siguiente formato (yyyy-mm-dd hh:mm:ss) | Login datetime |
| timestamp | Es un alias de datetime, la diferencia es que se usa este tipo de dato para almacenar fechas automáticamente. | Login timestamp |
| time | Registra la hora de acuerdo al siguiente formato (hh:mm:ss) | Hora time |
| year | Registra un año | Año year |
| blob | Este tipo de dato permite almacenar información de forma binaria, como son, imágenes textos con formatos, y cualquier tipo de archivos, no es recomendable su uso ya que con el tiempo la base de datos se va a volver lenta. | Adjunto blob |
| text | Este es un tipo de dato en el que se almacena texto con una longitud hasta 65535 caracteres | descripción text |
| mediumtext | Almacena textos con una longitud máxima de 16777215 | Html mediumtext |

Como podemos ver los tipos de datos no son nada del otro mundo, MySql provee de tipos de datos similares pero con diferente capacidad para que el desarrollador optimice al máximo los recursos del servidor.

**2.13.7. Comandos Básicos MySql**

Las sentencias SQL en general se dividen en dos categorías en DML (Data Manipulation Language) y DDL (Data Definition Languaje), el primero sirve para crear modelos de datos y sus acciones son irreversibles, las sentencias del segundo grupo sirve para manipular los datos, las acciones hechas por las sentencias DDL se pueden deshacer aunque se necesita conocimiento más profundo del gestor.

Antes de hacer una revisión de los comandos quiero decir que los comandos o sentencias que se van a revisar son compatibles con la mayoría de los gestores de bases de datos, en realidad no se escriben exactamente igual aunque tampoco es que tengan muchas diferencias, lo que sí es igual es el comportamiento, esto ya que SQL es un estándar adoptado por todos los gestores de bases de datos SQL abiertos o cerrados gratuitos o de pago.

La técnica que voy a utilizar para explicar los comandos es la siguiente, primero voy a exponer la sintaxis básica de los comandos, en el momento en que se empiece a crear la base de datos para el proyecto podremos implementar a más de una sentencia.

**2.13.7.1. Comandos DDL (Lenguaje de definición de datos)**

Como su nombre lo indica, las sentencias que pertenecen a este grupo son aquellas que sirven para modelar, modificar o eliminar tablas y bases de datos, cada tipo de comando puede ser utilizado para más de una operación, pero cada una con su propio fin.

**2.13.7.1.2 Sentencias CREATE**

Este grupo de sentencias sirven para crear estructuras de datos ya sean tablas, vistas, bases de datos, funciones y procedimientos almacenados, veamos las capacidades de esta sentencia.

**2.13.7.1.2.1. CREATE TABLE**

Esta sentencia es usada para crear tablas bases de datos y vistas, a continuación mostramos la sintaxis que puede tener esta sentencias SQL.

1 CREATE [TEMPORARY] TABLE [IF NOT EXISTS] nombre\_tabla

2 [(denicion\_de\_columnas,...)]

3 [opciones\_tabla]

También se puede crear tablas temporales, este tipo de tablas son borradas de memoria luego de su uso, en la parte de opciones de tabla se puede especificar el juego de caracteres a utilizar y el motor de gestión de datos.

**2.13.7.1.2.2. CREATE VIEW**

Esta sentencia ayuda a crear una vista, como lo dijimos en un capitulo anterior las vistas son producto del resultado de un comando SELECT aplicado a una o varias tablas, la sentencia puede contener condiciones, sintaxis.

1 CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]

2 VIEW nombre\_vista[(columnas)]

3 AS sentencia\_select

4 [WITH [CASCADED | LOCAL] CHECK OPTION]]

Esta sentencia no solamente es capaz de crear vistas sino también de reemplazarlas o combinarlas aunque no es recomendable, es preferible definir el algoritmo nuevamente.

**2.13.7.1.2.3. CREATE PROCEDURE/CREATE FUNCTION**

Estas sentencias hacen prácticamente lo mismo, la diferencia es lo que pueden crear, la primera crea una rutina que no es capaz de devolver ningún valor, ideal para hacer trabajos de backup, la segunda genera rutinas que son capaces de devolver valores, ambas soportan parámetros de entrada, pero solo las funciones son capaces de devolver un valor, aunque a los dos se los conoce como procedimientos almacenados.

1 CREATE PROCEDURE nombre\_sp([parametros[,...]])

2 CREATE PROCEDUREy CREATE FUNCTION

3 [características...] cuerpo\_rutina

4 CREATE FUNCTION nombre\_sp([parametros[,...]])

5 RETURNS type

6 [características...] cuerpo\_rutina

7 parameter:

8 [ IN | OUT | INOUT ] nombre\_parametro type

9 type:

10 Any cualquier válido MySQL data type

11 características:

12 LANGUAGE SQL

13 | [NOT] DETERMINISTIC

14 | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }

15 | SQL SECURITY { DEFINER | INVOKER }

16 | COMMENT 'string'

17 cuerpo\_rutina:

18 procedimientos almacenados o comandos SQL válidos

Por defecto las rutinas pertenecen a la base de datos en la que fueron creadas, si ninguna base de datos está seleccionada el gestor nos arrojará un error y no permitirá crear la rutina, para invocar los procedimientos se usa la sentencia **CALL nombre\_sp** y para invocar las funciones solamente se escribe el nombre de la función y los parámetros.

**2.13.7.1.2.4. CREATE DATABASE/CREATE SCHEMA**

Estos dos comandos hacen lo mismo CREATE SCHEMA fue implementado para personas que vienen de otros motores de bases de datos, es una forma de hacer que los desarrolladores se adapten fácilmente.

Esta sentencia sirve para crear bases de datos, asignar el juego de caracteres a usar por defecto en las tablas, cuando se define el juego de caracteres en esta sentencia no hace falta definirlos en las tablas, si se cambia el juego de caracteres en la base de datos se lo cambia para todas las tablas.

1 CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] nombre\_db

2 Epecificaciones\_de\_creado:

3 [DEFAULT] CHARACTER SET juego\_caracteres

4 | [DEFAULT] COLLATE collation\_name nombre\_juego\_caracteres\_por\_defecto

**2.13.7.1.3 Sentencias ALTER**

Este grupo de sentencias son usadas para alterar las estructuras creadas con las sentencias créate, son capaces de aumentar o quitar atributos a los elementos.

**2.13.7.1.3.1. ALTER TABLE**

Esta sentencia permite cambiar la estructura de una tabla existente. Por ejemplo, se puede añadir o borrar columnas, crear o destruir índices, cambiar el tipo de columnas existentes, o renombrar columnas o a la misma tabla. Puede cambiar el comentario de la tabla y su tipo.

1 ALTER [IGNORE] TABLE nombre\_tabla

2 Especificación\_atelalter\_specification:

3 ADD [COLUMN] column\_definition[FIRST | AFTER col\_name]

4 | ADD [COLUMN] (definicion\_columna,...)

5 | ADD INDEX [nombre\_index] [tipo\_index] (nombre\_col\_index,...)

6 | ADD [CONSTRAINT [nombre\_clave]]

7 PRIMARY KEY [tipo\_index] (nombre\_col\_index,...)

8 | ADD [CONSTRAINT [nombre\_clave]]

9 UNIQUE [nombre\_index] [tipo\_index] (nombre\_col\_index,...)

10 | ADD [FULLTEXT|SPATIAL] [nombre\_col] (nombre\_col\_index,...)

11 | ADD [CONSTRAINT [nombre\_clave]]

12 FOREIGN KEY [nombre\_index] (nombre\_col\_index,...)

13 [referencia\_definición]

14 | ALTER [COLUMN] nombre\_col{SET DEFAULT literal| DROP DEFAULT}

15 | CHANGE [COLUMN] nombre\_col\_anterior definición\_col

16 [FIRST|AFTER nombre\_col]

17 | MODIFY [COLUMN] definicion\_col[FIRST | AFTER nombre\_col]

18 | DROP [COLUMN] nombre\_col

19 | DROP PRIMARY KEY

20 | DROP INDEX nombre\_index

21 | DROP FOREIGN KEY nombre\_clave

22 | DISABLE KEYS

23 | ENABLE KEYS

24 | RENAME [TO] nuevo\_nombre\_tabla

25 | ORDER BY nombre\_col

26 | CONVERT TO CHARACTER SET charset\_name[COLLATE nombre\_juego\_caracteres]

27 | [DEFAULT] CHARACTER SET charset\_name[COLLATE nombre\_juego\_caracteres]

28 | DISCARD TABLESPACE

29 | IMPORT TABLESPACE

**2.13.7.1.3.2. ALTER VIEW**

Esta sentencia no tiene la capacidad de modificar las vistas solamente es capaz de reemplazar una vista existente, como podemos recordar la vista es una tabla temporal que se obtiene del resultado del comando SELECT tomado de una o varias tablas, esta es una razón por la que no se la puede editar al mismo nivel de ALTER TABLE, solamente se reemplace el algoritmo.

1 CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]

2 VIEW nombre\_vista[(columnas)]

3 AS sentencia\_select

4 [WITH [CASCADED | LOCAL] CHECK OPTION]

**2.13.7.1.3.3. ALTER PROCEDURE/CREATE FUNCTION**

Al igual que con las vistas esta sentencia no tiene la capacidad de hacer cambios en el algoritmo solamente lo reemplaza por el nuevo.

1 ALTER {PROCEDURE | FUNCTION} sp\_name[characteristic...]

2 characteristic:

3 { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }

4 | SQL SECURITY { DEFINER | INVOKER }

5 | COMMENT 'string'

**2.13.7.1.4 Sentencias DROP**

Estas sentencias son utilizadas para destruir estructuras, son capaces de eliminar bases de datos índices y tablas.

1 -- Eliminar Base Datos

2 DROP {DATABASE | SCHEMA} [IF EXISTS] db\_name

3

4 -- Eliminar Indice

5 DROP INDEX index\_nameON tbl\_name

6

7 -- Eliminat Tabla

8 DROP [TEMPORARY] TABLE [IF EXISTS]

9 tbl\_name[, tbl\_name] ...

10 [RESTRICT | CASCADE]

Si se la usa para eliminar una base de datos, se eliminan todas las tablas con sus datos, si se elimina una tabla se elimina sus índices columnas y datos si se especifica el atributo CASCADE de la sentencia se eliminan también todas las tablas que heredan de ella (que tengan relaciones con la tabla a la que se intenta eliminar).

**2.13.7.2. Comandos DML (Lenguaje de Manipulación de datos)**

Este es un lenguaje que sirve para manipular los datos, las sentencias que pertenecen a este grupo están en capacidad de trabajar con los datos de forma directa, pueden escribir, editar, leer y actualizar los datos de una tabla.

**2.13.7.2.1 Sentencias SELECT**

Esta sentencia es usada para traer los datos de una o más tablas, es la única que permite traer datos, posee algunos elementos que la hacen muy potente y está en capacidad de elaborar consultas muy complejas aplicando lógica de algebra relacional.

1 SELECT

2 [ALL | DISTINCT | DISTINCTROW ]

3 [HIGH\_PRIORITY]

4 [STRAIGHT\_JOIN]

5 [SQL\_SMALL\_RESULT] [SQL\_BIG\_RESULT] [SQL\_BUFFER\_RESULT]

6 [SQL\_CACHE | SQL\_NO\_CACHE] [SQL\_CALC\_FOUND\_ROWS]select\_expr, ...

7 [INTO OUTFILE 'file\_name' export\_options

8 | INTO DUMPFILE 'file\_name']

9 [FROM tablas\_referenciadas

10 [WHERE definición\_condición]

11 [GROUP BY {nombre\_col| expr| posición}

12 [ASC | DESC], ... [WITH ROLLUP]]

13 [HAVING definición\_condición]

14 [ORDER BY {col\_name| expr| position}

15 [ASC | DESC] , ...]

16 [LIMIT {[offset,] row\_count| row\_countOFFSET offset}]

17 [PROCEDURE nombre\_sp(lista\_de\_argimentos)]

18 [FOR UPDATE | LOCK IN SHARE MODE]]

Como se puede apreciar esta sentencia posee una gran potencia, están en capacidad de realizar consultas para atender a una consulta superior a eso se le conoce como subconsultas, también pueden llamar a procedimientos almacenados y trabajar con esa información.

**2.13.7.2.1 Sentencias JOIN**

Esta sentencia es muy útil para obtener los datos de dos tablas, no trabaja por si sola debe ser implementada por un SELECT, UPDATE o DELETE, esta sentencia acepta condiciones para obtener los datos si se la omite obtiene un producto cruzado de las dos tablas.

1 tabla\_ref, tabla\_ref

2 tabla\_ref[INNER | CROSS] JOIN tabla\_ref[condición\_join]

3 tabla\_ref STRAIGHT\_JOIN tabla\_ref

4 tabla\_ref LEFT [OUTER] JOIN table\_reference condición\_join

5 tabla\_ref NATURAL [LEFT [OUTER]] JOIN table\_reference

6 { ON tabla\_ref LEFT OUTER JOIN tabla\_ref

7 ON expr\_condicional}

8 tabla\_ref RIGHT [OUTER] JOIN tabla\_ref condición\_join

9 tabla\_ref NATURAL [RIGHT [OUTER]] JOIN tabla\_ref

10

11 -- tabla\_ref se define como

12

13 tbl\_name[[AS] alias]

14 [[USE INDEX (key\_list)]

15 | [IGNORE INDEX (key\_list)]

16 | [FORCE INDEX (key\_list)]]

17

18 -- join\_conditionse define como:

19 ON conditional\_expr| USING (column\_list)

Hay “variantes “ de JOIN por decirlo de alguna forma, los menciono porque son muy útiles, me refiero a LEFT JOIN, RIGHT JOIN, CROSS JOIN, e INNER JOIN, la primera obtiene todos los registros de la tabla de la izquierda que estén en la derecha, la segunda hace lo mismo pero al en sentido contrario aunque su uso no es recomendado ya que el SQL no es portable ya que esta sentencia no es soportada por otros gestores de bases de datos, CROSS JOIN devuelve un producto cartesiano de las tablas, INNER JOIN retorna todos los registros que tienen una referencia en ambas tablas.

**2.13.7.2.1 Sentencias INSERT**

Esta sentencia es capaz de escribir en las tablas existentes, a cada conjunto de datos ingresados a una tabla se lo conoce con el nombre de registro, se debe especificar el nombre de las columnas en caso de no querer ingresar todos los datos del registro, los datos deben tener el mismo orden que la definición de los nombres de las columnas, el ingreso se lo hace por correspondencia de posición.

1 INSERT [LOW\_PRIORITY | DELAYED | HIGH\_PRIORITY] [IGNORE]

2 [INTO] nombre\_tabla[(nombre\_col,...)]

3 VALUES ({expr| DEFAULT},...),(...),...

4 [ ON DUPLICATE KEY UPDATE nombre\_col = expr, ... ]

5

6 -- También:

7 INSERT [LOW\_PRIORITY | DELAYED | HIGH\_PRIORITY] [IGNORE]

8 [INTO] nombre\_tabla

9 SET nombre\_col = {expr| DEFAULT}, ...

10 [ ON DUPLICATE KEY UPDATE nombre\_col = expr, ... ]

11

12 -- También:

13 INSERT [LOW\_PRIORITY | HIGH\_PRIORITY] [IGNORE]

14 [INTO] nombre\_tabla[(nombre\_col,...)]

15 SELECT ...

16 [ ON DUPLICATE KEY UPDATE nombre\_col = expr, ... ]

**2.13.7.2.1 Sentencias UPDATE**

Como su nombre lo indica sirve para actualizar los datos de una tabla, se vale de la cláusula SET para especificar las celdas o datos a modificar, esta sentencia recibe los nombres de las columnas y los datos nuevos a escribir, es necesario definir la condición la misma que se debe cumplir para que la sentencia se ejecute.

1 -- Sintaxis para una tabla:

2 UPDATE [LOW\_PRIORITY] [IGNORE] nombre\_tabla

3 SET nombre\_col1=expr1[, nombre\_col2=expr2...]

4 [WHERE definición\_condición]

5 [ORDER BY ...]

6 [LIMIT número\_entero]

7

8 -- Sintaxis para múltiples tablas:

9 UPDATE [LOW\_PRIORITY] [IGNORE] tablas\_referenciadas

10 SET nombre\_tabla2=expr1[, nombre\_tabla2=expr2...]

11 [WHERE definición\_condición]

**2.13.7.2.1 Sentencias DELETE**

Esta sentencia elimina los registros de una tabla, no puede eliminar un solo dato, elimina toda la fila de datos más conocida como registros, se le debe especificar una condición, puede eliminar datos de más de una tabla a la vez.

1 -- Sintaxis para una tabla:

2 DELETE [LOW\_PRIORITY] [QUICK] [IGNORE] FROM nombre\_tabla

3 [WHERE condición]

4 [ORDER BY ...]

5 [LIMIT número\_entero]

6

7 -- Sintaxis para múltiples tablas:

8 DELETE [LOW\_PRIORITY] [QUICK] [IGNORE]

9 nombre\_tabla[.\*] [, nombre\_tabla[.\*] ...]

10 FROM tablas\_referenciadas

11 [WHERE condición]

**2.13.8.1. Procedimientos Almacenados**

Los procedimientos almacenados pueden mejorar el rendimiento en aplicaciones de cliente, ya que se necesita enviar menos información entre el servidor y el cliente. Aunque se aumenta la carga de procesamiento del servidor de la base de datos ya que la mayoría del trabajo se realiza en la parte del servidor y no en equipo del cliente.

Los procedimientos almacenados permiten tener bibliotecas o funciones en el servidor de base de datos. Esta característica es compartida por los lenguajes de programación modernos que permiten este diseño interno, por ejemplo, usando clases. Usando estas características del lenguaje de programación de cliente es beneficioso para el programador, incluso fuera del entorno de la base de datos, se menciona lenguajes de programación porque algunos motores los soportan para ejecutar procedimientos almacenados, en MySql se pretende soportar PHP aunque aún no se lo ha implementado.

Algunas situaciones en que los procedimientos almacenados pueden ser particularmente útiles:

Cuando múltiples aplicaciones cliente se escriben en distintos lenguajes o funcionan en distintas plataformas, pero necesitan realizar la misma operación en la base de datos.

Cuando la seguridad es muy importante. Los bancos, por ejemplo, usan procedimientos almacenados para todas las operaciones comunes. Esto proporciona un entorno seguro y consistente, y los procedimientos pueden asegurar que cada operación se loguea (registro de actividades) apropiadamente.

En tal entorno, las aplicaciones y los usuarios no obtendrían ningún acceso directo a las tablas de base de datos, sólo pueden ejecutar algunos procedimientos almacenados, la forma de crearlos se los vio anteriormente, pero esta es una sección en la que se intenta ayudar el porqué de la existencia de estos elementos.

**2.13.8.2. MySql Triggers**

La palabra **“Trigger”** traducida al español significa **“Disparador”**, un disparador es un objeto con nombre alojado en una base de datos que se asocia con una tabla, y se activa cuando ocurre un evento en particular para esa tabla.

El disparador queda asociado a la tabla a través del nombre de esta. La misma que debe ser una tabla permanente no puede ser una tabla temporal ni tampoco una vista, porque los eventos a los que el disparador responde solamente están presentes en las tablas “reales y fijas”.

A los eventos se los conoce como momento de disparo, este es el momento en que el disparador entra en acción. Puede ser **BEFORE**(antes) o **AFTER**(después) del evento.

Las sentencias que tienen la capacidad de activar un disparador, pueden ser **INSERT**, **UPDATE**, o **DELETE**. Por ejemplo, un disparador **BEFORE** para sentencias **INSERT** podría utilizarse para validar los valores a insertar o un **AFTER** para **UPDATE** se podría usar para actualizar alguna vista.

No puede haber dos disparadores en una misma tabla que correspondan al mismo momento y sentencia. Por ejemplo, no se pueden tener dos disparadores **BEFORE UPDATE**. Pero sí es posible tener los disparadores **BEFORE UPDATE** y **BEFORE INSERT** o **BEFORE UPDATE** y **AFTER UPDATE**.

1 -- Sintaxis de CREATE TRIGGER

2 CREATE TRIGGER nombre\_disp momento\_disp evento\_disp

3 ON nombre\_tabla FOR EACH ROW sentencia\_disp

Como se puede apreciar en el ejercicio anterior **“sentencia\_disp”** es la sentencia que se ejecuta cuando se activa el disparador. Si se desean ejecutar múltiples sentencias, deben colocarse entre **BEGIN <sentencias SQL> END**, este es un constructor de sentencias compuestas, este constructor nos sirve para ejecutar varias sentencias como si fueran una sola, la forma de escribir el código es instrucción por instrucción y separando por punto y coma **“;”** a cada sentencia.

**2.14. Introducción a PHP**

**Este**

**2.14.1. Referencia Del Lenguaje**

**2.14.2. Sintaxis Básica**

**2.14.3. Tipos**

**2.14.3.1. Booleanos**

**2.14.3.2. Enteros**

**2.14.3.3. Números De Punto Flotante**

**2.14.3.4. Cadenas**

**2.14.3.5. Arrays**

**2.14.3.6. Objetos**

**2.14.4. Variables**

**2.14.5. Constantes**

**2.14.6. Expresiones**

**2.14.7. Operadores**

**2.14.8. Estructuras de Control**

**2.14.9. Funciones**

**2.14.10. Clases Y Objetos**

**2.14.10.1. Introducción**

**2.14.10.2. Lo básico**

**2.14.10.3. Propiedades**

**2.14.10.4. Constantes De Clases**

**2.14.10.5. Autocarga De Clases**

**2.14.10.6. Constructores Y Destructores**

**2.14.10.7. Visibilidad**

**2.14.10.8. Herencia De Objetos**

**3. Capítulo III**

**3.1. Recolección De Información**

**3.1.1. Identificar Fuentes De Información**

**3.2. Definición De Problema. ¿Qué Es Un CMS?**

Es necesario conocer lo más que se pueda lo que es un CMS, con esto me refiero saber lo que es más allá de los conceptos, su funcionalidad, su objetivo,empezaré citando la definición de Wikipedia la cual dice

“*referencia wikipedia*”

Esta definición no es tan clara como se podría desear pero su objetivo es dar a conocer lo que es un CMS, sabemos que un CMS gestiona no solamente los contenidos que se muestran a los usuarios y administradores sino que además es encargado de gestionar los componentes internos de la aplicación, veamos los todos y cada uno de los ítems que manejan los CMS.

* Maneja las Imágenes

Se las puede manejar almacenándolas directamente en el servidor a través de en el que está alojado el sitio con la ayuda de un Uploader o caso contrario en servidores externos y rescatando el link del fichero.

Nota: Uploader es una aplicación que gestiona las imágenes, además en la segunda opción podemos perder la imagen con el tiempo porque la persona dueño del sitio tiene potestad sobre ellas y las puede eliminar si lo considera necesario.

* Manejar Videos

EL soporte de videos esta dado simplemente con mostrar videos de servidores externos el único a soportar será Youtube

* Google CustomSearch (GCS)

Si pensamos un momento en el crecimiento del sitio, imaginemos por un momento que tenemos alojados más de cien artículos en más de diez categorías, sería muy difícil acceder a ellos solamente a través de los menús ya que estos serian muy extensos y además complicado para el usuario cuando este busque un contenido de su interés. Para enfrentar este inconveniente tenemos un producto de Google llamado Google CustomSearch, lo que hace este sistema es identificar todo el contenido del sitio y lo indexa a Google, una vez implementado en la pagina podemos hacer búsquedas en el sitio con la ayuda de Google sin la necesidad de programar un buscador propio, y con toda la capacidad que Google posee.

El servicio que se va a elegir es el gratuito, este nos muestra publicidad en las búsquedas delos usuariosperosi se la desea eliminar de los resultados se debe pagar por el servicio, es bueno mencionar que esto ayuda mucho en el posicionamiento en el buscador de Google, aunque no es el único buscador que existe.

Nota: cuando el usuario se siente confundido o simplemente no encuentra lo que busca con facilidad abandona el sitio, hay que recordar que no somos los únicos que ofrecemos ese tipo de información, ha millones de sitios con igual o mejor información, por ene es importante priorizar el acceso a los contenidos.

* Google Analytics (GA)

Una vez que se pone en funcionamiento el sitio es necesario conocer sus estadísticas, la intención de este trabajo no es de mencionar cuantos servicios de estadísticas existen ni cual es el mejor, sino de recomendar uno que sea bueno y de preferencia gratuito, dentro de esta recomendación esta Google Analitics o GA, este no es solamente un contador de visitas, ayuda a la toma de decisiones para el rumbo del proyecto, ya que nos da información de que es lo que la gente mas lee en nuestro sitio, cuanto tiempo le dedica al sitio, desde donde esta viendo el sitio, que sistema operativo usa, su navegador, que tipo de dispositivo usa ya sea teléfonos tabletas u otros. Es un servicio muy completo, lo recomiendo.

* Google Maps

Es un servicio de mapas, lo vamos a usar en lo mas básico que es mostrar la ubicación de la empresa en la pagina de contactos.

* SEO Básico
* SEO es una técnica que hace que los contenidos de nuestro sitio web puedan ser encontrados por los buscadores. En cuanto SEO se van a seguir algunos parámetros específicos en la publicación de los artículos, como mencioné en una nota Google no es el único buscador que existe, pero es el mas grande, por eso controla la mayoría de los parámetros que los desarrolladores de contenido deben tener en cuenta para ser “vistos” por Google, y al ser tan grande esta empresa hizo que sus competidores como Yahoo Bing y muchos mas siguieran esos estándares para indexar los contenidos a sus bases de datos, esto se profundizará mas en la sección correspondiente.
* Redes Sociales

En la actualidad las redes sociales son una muy buena herramienta para atraer a personas, con esto nuestro sitio tiene publicidad a través de terceros de forma gratuita, permitiendo que los cibernautas conozcan nuestro sitio, hay un dicho que dice “*si quieres vender la gente debe saber que existes”*, esta es un muy buena forma de lograrlo, existes tres principales redes sociales como son.

Twitter

Facebook

Google Plus

**3.2.1. Definición De CMS**

**3.2.2. Historia De CMS**

**3.2.3. Tipos De CMS**

**3.2.3.1. Abiertos**

**3.2.3.2. Cerrados**

**3.2.3.3. Comerciales**

**3.3. Ventajas**

**3.4. Desventajas**

**3.5. Estudio Del Funcionamiento De Un CMS**

**3.5.1. Análisis De Las Prestaciones**

**3.5.2. Reconocimiento De La Estructura De Un CMS**

**3.5.3. Manejo De La Información Del Sitio Por Parte Del CMS**

**3.5.4. Almacenamiento De Datos**

**3.5.5. Manipulación De Datos Por El Usuario**

**3.5.6. Flujo De Datos**

**3.5.7. Conclusiones**

**4. Capítulo IV**

**4.1. Definición De Las Prestaciones Del Sistema**

**4.1.1. Análisis De Requerimientos**

**4.1.1.1. Hardware**

**4.1.1.2. Software**

**4.1.1.3. Personal Humano**

**4.1.2. Definición De Funcionalidades Del Sistema**

**4.2. Creación Del Modelo De Base De Datos**

**4.2.1. Creando El Primer Modelo**

**4.2.2. Sometiendo A Pruebas Modelo**

**4.2.3. Definición De Funciones Y Vistas**

**4.3. Análisis Diseño Lógico Del Sistema**

**4.3.1. Casos De Uso**

**4.3.2. Crear Diagramas De Modelo UML**

**4.4. Elaboración Completa Del Sistema En Diagramas**

**4.4.1. Diagrama De Bases De Datos**

**4.4.2. Diagrama De Clases**

**4.5. Dividir La Estructura Del Sistema**

**4.5.1. Introducción A Patrones De Diseño**

**4.5.2. Estructura Del Sistema Con Boilerplate**

**5. Capítulo V**

**5.1. Análisis General A Los Frameworks CodeIgniter Y Bootstrap**

**5.1.1. Propósitos**

**5.1.2. Estructura**

**5.1.3. Requerimientos**

**5.2. Introducción A CodeIgniter PHP**

**5.2.1. Diagrama de Flujo de la Aplicación**

**5.2.2. Modelo Vista Controlador**

**5.2.3. Instrucciones De Instalación**

**5.2.4. Configuración**

**5.2.5. Primeros Pasos**

**5.2.6. Temas Generales**

**5.2.6.1. URLS De CodeIgniter**

**5.2.6.2. Modelos**

**5.2.6.3. Vistas**

**5.2.6.4. Controladores**

**5.2.6.5. Usar Las Bibliotecas De CodeIgniter**

**5.2.6.6. Crear Bibliotecas En CodeIgniter**

**5.2.6.7. Ruteo URI**

**5.2.6.8. Funciones Comunes**

**5.2.6.9. Manejo de Errores**

**5.2.6.10. Seguridad**

**5.2.6.11. Estilo y Sintaxis Generales**

**5.2.7. Clases de CodeIgniter**

**5.2.7.1. Clase Config**

**5.2.7.2. Clase Database**

**5.2.7.3. Clase Email**

**5.2.7.4. Clase Encrypt**

**5.2.7.5. Clase Input**

**5.2.7.6. Clase Pagination**

**5.2.7.7. Clase Session**

**5.2.7.8. Clase Upload**

**5.2.8. Helpers de CodeIgniter**

**5.2.8.1. Helper Download**

**5.2.8.2. Helper Form**

**5.2.8.3. Helper URL**

**5.3. IntroducciónA Bootstrap CSS**

**5.3.1. Como UsarBootstrap**

**5.3.2. Sintaxis De Bootstrap**

**5.3.3. Prestaciones Del Framework Bootstrap**

**5.3.4. Scaffolding**

**5.3.4.1. Grid System**

**5.3.4.2. Fluid Grid System**

**5.3.4.3. Layouts**

**5.3.5. Base CSS**

**5.3.5.1. Tables**

**5.3.5.2. Forms**

**5.3.5.3. Buttons**

**5.3.5.4. Images**

**5.3.5.5. Icons**

**5.3.6. Components**

**5.3.6.1. Dropdowns**

**5.3.6.2. Button Dropdowns**

**5.3.6.3. Nav**

**5.3.6.4. Navbar**

**5.3.6.5. Pagination**

**5.3.6.6. Labels**

**5.3.6.7. Alerts**

**5.3.7. Java Script**

**5.3.7.1. Modal**

**5.3.7.2. Tab**

**5.3.7.3. Tooltip**

**5.3.7.4. Collapse**

**6. Capítulo VI**

**6.1. Análisis De Interfaz De Usuario**

**6.1.1. Interfaz de Administrador**

**6.1.2. Interfaz de Usuario**

**6.2. Construcción De Interfaces**

**6.2.1. Construcción De Interfaz Administrador Con Bootstrap**

**6.2.2. Construcción De Interfaz Usuario Con Bootstrap**

**6.3. Pruebas De Interfaces**

**6.3.1. Implementación De Modelos**

**6.3.2. Implementación Modelo De Base Datos**

**6.3.3. Implementación Modelo De Clases**

**6.3.4. Implementación Diagrama General Del Sistema**

**6.4. Pruebas Del Sistema**

**6.5. Correcciones Al Sistema**

**6.6. Publicación Del Sistema**

**7. Capítulo VII**

**7.1. Conclusiones**

**7.2. Recomendaciones**

**7.3. Bibliografía**

**Referencias de este libro**

[1]<http://es.wikipedia.org/wiki/Internet>

[2]<http://es.wikipedia.org/wiki/http>

[3] dibujo de la aprne de wikipedia

[4]http://ellislab.com/codeigniter/user-guide/overview/appflow.html

[5]<http://es.wikipedia.org/wiki/programacion-orientada-a-objetos>

[6]Van Gurp J. y Bosch J. 2001. Implementation and Evolution of Object-Oriented Frameworks: concepts & guidelines. Software: Practice andExperience. pp. 277-300

[7]

[8]http://eugeniabahit.blogspot.com/p/poo-y-mvc-en-php.html

[9]http://www.desarrolloweb.com/articulos/que-es-html.html

[10][http://es.wikipedia.org/wiki/HTML#Historia\_de\_HTML](http://es.wikipedia.org/wiki/HTML" \l "Historia_de_HTML)

[11]http://www.w3schools.com/html/html\_intro.asp

[12]<http://techtastico.com/post/%C2%BFcuales-son-las-versiones-del-html/>

[13] <http://www.w3schools.com>

[14] www.angrybirds.com

[15]http://www.cristalab.com/tutoriales/introduccion-a-html5-c92171l/

http://www.sineyaculacionprecoz.net/durar-mas-en-la-cama-25-posiciones-nuevas-para-el-sexo/

[16]http://www.w3schools.com/cssref/css\_selectors.asp