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 de la 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 "user agent" (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 anterior podemos 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 menos trabajo 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 Windows siempre que tenga instalado el Framework. Veamos un ejemplo diferente, si se 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 organizadas porque 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 principal que 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, el Framework 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 código correspondiente se pueden introducir por herencia o composición. Se agregan nuevas 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 aplicaciones conozca los detalles internos del Framework, por lo cual son mucho más fáciles de 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, y algunas partes se implementan vía herencia o composición, y otras a través de 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 demasiado tiempo 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 POO los 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 compleja permitiendo 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.

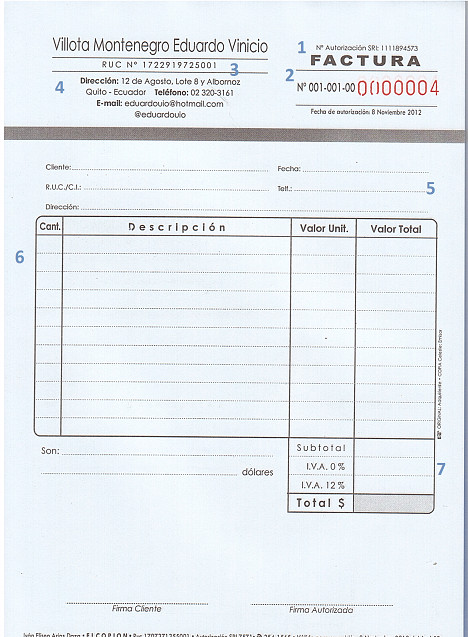
**Método:** Es un algoritmo asociado a un objeto que trabaja con sus propiedades, indica lo que el objeto está en capacidad de hacer.

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 estoy intentado crear 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.

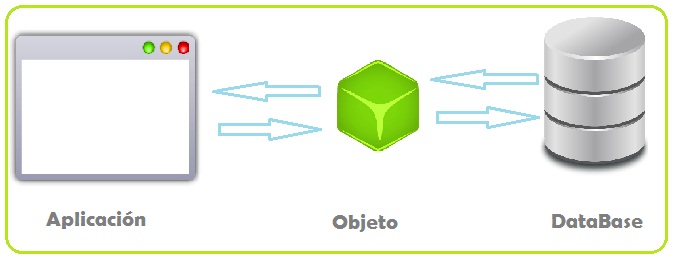
Además se muestra métodos como facturar que es crear una factura nueva, cobrar factura que es marcar a una factura como pagada para esto es necesario crear una propiedad en la que podamos guardar este estado, lo mismo sucede con abonar y anular factura.

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. Esta es la 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 desea heredar una 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 class ImplementaInterface implements InterfaceBaseDatos{

23

24 public function listarRegistros(){

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 clase ClaseNormal 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 dos puntos, 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 capítulo está 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]

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

Como todas las tecnologías en desarrollo HTML ha ido evolucionando con 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 las versiones 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:

* HTML Strict: 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, la etiqueta 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 <img src=”../img/miimagen.jpg”/>

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

6 <input name=”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 doctype que tenemos 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.

**2.4.2 Estructura**

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

1 <!DOCTYPE html>

2 <html lang="es">

3 <head>

4 <meta charset="utf-8">

5 <title>Mi títilo</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.

* + 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, se llaman así porque sirven para definir la estructura de la página pero 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.

**<html> esta**

**<meta>**

**<head>**

**<title>**

**<link>**

**<script>**

**<body>**

**<div>**

**<span>**

Etiquetas que le dan formato al texto:

<a>

<h1>

<p>

<br/>

<b>

<i>

<strong>

<img>

etiquetas para tablas

<img>

<table>

**2.4.3. Links**

**2.4.4. Capas**

**2.5. Formularios**

**2.7. Ejemplo Pagina Básica HTML**

**2.8. ¿Qué es HTML5?**

**2.8.1. Nuevas Etiquetas**

**2.9. Introducción a CSS**

**2.9.1. Origen CSS**

**2.9.2. Usos CSS**

**2.10. Introducción Básica A CSS**

**2.10.1. Sintaxis Básica**

**2.10.2. Selectores**

**2.10.3. Propiedades**

**2.10.4. Primeros Pasos**

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

**2.11. Introducción A Java Script**

**2.11.1. Origen Java Script**

**2.11.2. Usos Java Script**

**2.11.3. Lenguaje De Programación Java Script**

**2.11.4. Tipos De Datos**

**2.11.4.1. Numéricos**

**2.11.4.2. Cadenas**

**2.11.4.3. Booleanos**

**2.11.4.4. Nulos**

**2.11.4.5. Objeto**

**2.11.4.6. Arreglos**

**2.11.5. Sintaxis Básica Java Script**

**2.11.6. Operadores**

**2.11.7. Control De Flujo**

**2.11.7.1. Condicional IfElse**

**2.11.7.2. Condicional Switch**

**2.11.7.3. Bucles For**

**2.11.7.4. Bucles While**

**2.11.7.5. Clases Y Funciones En Java Script**

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

**2.13. Introducción A MySql**

**2.13.1. Comandos Básicos MySql**

**2.13.1.1.1. Comando SHOW**

**2.13.1.1.2. Crear Base De Datos Y Tablas**

**2.13.1.1.3. Alterar Bases De Datos**

**2.13.1.1.4. Consultas Con El Comando SELECT**

**2.13.1.1.4.1. Comprendiendo Al Comando SELECT**

**2.13.1.1.4.2. Consultas Básicas**

**2.13.1.1.5. Comando JOIN**

**2.13.1.1.5.1. Consultas Compuestas Con JOIN**

**2.13.1.1.6. Comando WHERE**

**2.13.1.1.6.1. Consultas Compuestas Con WHERE**

**2.13.1.2. Comandos Útiles MySql**

**2.13.1.3. Vistas**

**2.13.1.4. Procedimientos Almacenados**

**2.13.1.4.1. MySql Procedures**

**2.13.1.4.2. MySql Functions**

**2.13.1.4.3. MySql Triggers**

**2.14. Introducción a PHP**

**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 a perfección lo que es un CMS, con esto me refiero saber lo que es más allá de los conceptos empezaré citando la definición de Wikipedia la cual dice “*referencia wikipedia*”

Si vamos un poco masalla también podemos concluir que un cms gestiona no solamente los contenidos que se muetranal los usuarios y administradores sino que además es encargado de gestionar los cmponentes 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**](http://es.wikipedia.org/wiki/Internet)

**[2]**[**http://es.wikipedia.org/wiki/http**](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**](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

**[3]**http://mautime.files.wordpress.com/2010/02/4378559350\_d12ac7a766\_b.jpg?w=720&h=347

**6]php.net/🡪buscar la palabra class**

Pdf<http://observatoriodelacapacitacion.stps.gob.mx/oc/PDF/cursos_en_linea/POO_SE.pdf>

[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>

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

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