

# **INSTITUTO SUPERIOR TECNOLÓGICO PARA EL DESARROLLO ISPADE**

**CREACIÓN DE UN DE UN CMS CON CODEIGNITER Y  
BOOTSTRAP**

**TRABAJO DE FIN DE CARRERA PREVIO A LA OBTENCIÓN DEL  
TÍTULO DE TÉCNICO EN ANALISIS DE SISTEMAS  
VILLOTA MONTENEGRO EDUARDO VINICIO**

**DIRECTOR: ING. ARMANDO GUILCAPI**

**QUITO, NOVIEMBRE 2013**

## **DECLARACIÓN**

Yo, Villota Montenegro Eduardo Vinicio, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ninguna calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración, cedo mis derechos de propiedad intelectual correspondientes a este trabajo, al Instituto Superior Tecnológico Para el Desarrollo ISPADE, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por

La normatividad institucional vigente.

Atentamente:

Eduardo Villota

172291972-5

## **CERTIFICACIÓN**

Certifico que el presente trabajo fue desarrollado por el Sr. Villota Montenegro Eduardo Vinicio, bajo mi supervisión.

Ing. Armando Guilcapi

**DIRECTOR DEL TRABAJO DE TITULACIÓN**

## **AGRADECIMIENTO**

Este trabajo no hubiera sido posible sin la ayuda primero de Dios, agradezco a mi familia por permitirme tomar el tiempo necesario para poder llevar el proyecto adelante soportando mis ausencias, agradezco también a mi tutor designado Ing. Armando Guilcapi quién me ha sabido aconsejar para que el proyecto logre el éxito deseado y conseguir que funcione de la mejor manera.

Un agradecimiento especial para todos mis profesores, es de ellos de quienes he aprendido lo aplicado en el presente documento, a mis compañeros de aula mismos que supieron brindar su ayuda y compañerismo.

Mi paso por el ISPADE fue una gran experiencia gracias a todas las personas que hacen parte de su dirección.

## **DEDICATORIA**

Este trabajo va dedicado de especial manera a mi padre ya que sin la educación impartida por él nada de esto fuera posible, se lo dedico a mi madre por las enseñanzas y la paciencia para conmigo, aunque ya no esté conmigo sigo sintiendo su presencia, a mi hijo y mi esposa que son la razón de mi trabajo, por ellos hago el esfuerzo de salir adelante y conseguir logros que ayuden a la estabilidad y felicidad de mi hogar, a mis compañeros que me supieron dar una mano cuando se los pedí.

## Índice de contenido

DECLARACIÓN.....	2
CERTIFICACIÓN.....	3
AGRADECIMIENTO.....	4
DEDICATORIA.....	5
RESUMEN DEL PROYECTO.....	12
PRESENTACIÓN.....	13
CAPÍTULO 1 INTRODUCCIÓN.....	14
1.1. PLANTEAMIENTO DEL PROBLEMA.....	14
1.1.1. OBJETIVO GENERAL.....	14
1.1.2. OBJETIVOS ESPECÍFICOS.....	15
1.1.3. JUSTIFICACIÓN DEL TRABAJO.....	15
1.1.4. JUSTIFICACIÓN METODOLÓGICA.....	16
1.2. DEFINICIÓN Y ORIGEN DE INTERNET.....	17
1.2.1. INTERNET EN LA ACTUALIDAD.....	17
CAPÍTULO 2 CMS Y FRAMEWORKS.....	19
2.1. DEFINICIÓN DE SITIOS WEB, SU ESTRUCTURA E IMPORTANCIA.....	19
2.1.1. PÁGINA WEB.....	19
2.1.1.1. Sitios estáticos.....	20
2.1.1.2. Sitios dinámicos.....	20
2.1.1.3. Importancia de los sitios web.....	21
2.2 INTRODUCCIÓN A LOS FRAMEWORKS.....	21
2.2.1. HISTORIA.....	21
2.2.2. TIPOS DE FRAMEWORKS.....	22
2.2.2.1. Frameworks de caja blanca.....	22
2.2.2.2. Frameworks de caja negra.....	22
2.2.2.3. Frameworks de caja gris.....	22
2.2.3. ¿POR QUÉ USAR UN FRAMEWORK?.....	23
CAPÍTULO 3 POO.....	24
3.1. INTRODUCCIÓN A LA POO.....	24
3.2. OBJETO.....	24
3.2.1. EJEMPLO DE OBJETO REAL.....	24
3.2.2. EJEMPLO DE OBJETO IMAGINARIO.....	27
3.3. TIPOS DE DATOS.....	30
3.4 OPERADORES.....	30
3.4.1. ESTRUCTURAS DE CONTROL.....	30
3.5. INTRODUCCION A PHP.....	31
3.5.1. Referencia del lenguaje.....	31
3.6. PROGRAMACIÓN ORIENTADA A OBJETOS EN PHP.....	32
3.6.1. CLASE.....	32

3.6.2. CLASE HEREDADA.....	33
3.6.3. MÉTODOS.....	34
3.7. PROPIEDADES.....	34
3.7.1. ÁMBITOS DE PROPIEDADES Y MÉTODOS EN PHP.....	34
CAPÍTULO 4 TECNOLOGÍAS PROYECTO.....	36
4.1. INTRODUCCIÓN A HTML.....	36
4.1.1. DEFINICIÓN DOCUMENTO.....	36
4.1.2. ETIQUETAS.....	37
4.2. INTRODUCCIÓN A CSS.....	38
4.2.1. SINTAXIS BÁSICA.....	38
4.2.2. SELECTORES.....	38
4.3 INTRODUCCIÓN A JAVA SCRIPT.....	40
4.3.1. SINTAXIS BÁSICA.....	40
4.3.2. IMPORTANCIA DE JAVA SCRIPT EN LA ACTUALIDAD.....	40
4.4. INTRODUCCIÓN A MYSQL.....	41
4.4.1. BASES DE DATOS.....	41
4.4.1.1. Tabla.....	41
4.4.1.2. Columna.....	42
4.4.2. TIPOS DE DATOS MYSQL.....	42
4.4.3. DISEÑO DE BASES DE DATOS MODELO ENTIDAD RELACIÓN.....	42
4.4.3.1. Modelo entidad relación.....	43
4.4.3.2. Clave primaria.....	43
4.4.3.3. Clave foránea.....	43
4.5. INTRODUCCIÓN A CODEIGNITER PHP.....	44
4.5.1. MODELO VISTA CONTROLADOR.....	45
4.5.1.1. Modelo.....	45
4.5.1.2. Vista.....	45
4.5.1.3. Controlador.....	45
4.5.2. DISTRIBUCIÓN DIRECTORIOS CODEINGITER.....	45
4.5.3. INSTRUCCIONES DE INSTALACIÓN.....	46
4.5.4. CONFIGURACIÓN.....	46
4.5.5. URLS DE CODEIGNITER.....	47
4.6. INTRODUCCIÓN A BOOTSTRAP CSS.....	48
CAPITULO DESARROLLO BASE DE DATOS.....	50
5.1 RECOLECCIÓN DE INFORMACIÓN.....	50
5.1.1. IDENTIFICAR FUENTES DE INFORMACIÓN.....	50
5.1.2. DEFINICIÓN DE PROBLEMA. ¿QUÉ ES UN CMS?.....	50
5.1.3. HISTORIA DE CMS.....	51
5.1.3.1. TIPOS DE CMS.....	52
5.1.3.1.1. CMS abiertos.....	52
5.1.3.1.2. CMS cerrados.....	52
5.1.3.1.3. CMS comerciales.....	52

5.1.3.1.4. Ventajas CMS.....	53
5.1.3.1.5. Desventajas CMS.....	53
5.1.3.2. Estudio del funcionamiento de un CMS.....	53
5.1.3.3. Diseño CMS.....	55
5.1.3.4. Análisis de las prestaciones CMS.....	56
5.1.4. ALMACENAMIENTO DE DATOS.....	56
5.1.4.1. Manipulación de datos por el administrador.....	57
5.1.4.2. Flujo de datos.....	57
5.2. DEFINICIÓN DE LAS PRESTACIONES DEL SISTEMA.....	57
5.2.1 PRESTACIONES DEL SISTEMA.....	57
5.2.2. ANÁLISIS DE REQUERIMIENTOS.....	58
5.2.2.1. Requisitos de hardware.....	58
5.2.2.2. Requisitos de software.....	58
5.2.2.3. Personal humano.....	59
5.2.2.4. Almacenamiento de información.....	59
5.3. DEFINICIÓN ESTRUCTURA DEL SITIO.....	59
5.3.1. GESTIÓN DE PANTALLAS.....	61
5.3.2. GESTIÓN DE URL'S.....	62
5.3.3. PAGINAS CMS.....	63
5.3.4. ARTÍCULOS CMS.....	63
5.3.5. IMÁGENES CMS.....	63
5.3.6. GESTIÓN DE BASE DE DATOS CMS.....	65
5.3.7. GESTIÓN DE ARTÍCULOS POR PÁGINAS (PAGINACIÓN).....	65
5.3.8. ADMINISTRACIÓN DE USUARIOS.....	66
5.3.9. GESTIÓN DE MENÚS.....	66
5.3.9.1. El foco de los menús.....	66
5.3.9.2. Barra de menú lateral.....	66
5.4. CREACIÓN DEL MODELO DE BASE DE DATOS.....	67
5.4.1. CREANDO EL PRIMER MODELO.....	67
5.4.2. DICCIONARIO DE DATOS.....	67
5.4.2.1. Entidad page.....	67
5.4.2.2. Entidad article.....	69
5.4.2.3. Entidad form.....	70
5.4.2.4. Entidad user.....	72
5.5. DEFINICIÓN DE FUNCIONES Y VISTAS.....	73
5.5.1. VISTA PARA PRESENTAR UN LISTADO DE ARTÍCULOS.....	74
5.5.2. VISTA TABLÓN.....	75
5.4.3. DIAGRAMA DE BASES DE DATOS.....	75
5.4.4. SOMETIENDO A PRUEBAS EL MODELO.....	75
CAPÍTULO 6 IMPLEMENTACIÓN DE CMS.....	76
6.1. ANÁLISIS DE INTERFAZ DE USUARIO.....	76
6.1.1. INTERFAZ DE USUARIO.....	76



6.1.2. DISEÑO DE INTERFAZ DE INICIO.....	77
6.1.3. DISEÑO DE INTERFAZ DE CONTENIDOS.....	78
6.1.4. DISEÑO DE INTERFAZ DE CONTACTOS.....	79
6.1.5. INTERFAZ DE ADMINISTRADOR.....	80
6.1.6. DISEÑO INTERFAZ DE LOGIN.....	81
6.1.7. DEFINICIÓN ESTRUCTURA DE VISTAS.....	81
6.2. VISTAS TEMINADAS.....	82
6.2.1. PRESENTACIÓN DE CONTENIDOS.....	82
6.2.2. PÁGINA DE INICIO.....	83
6.2.3. ESTRUCTURA VISTAS ADMINISTRADOR.....	83
6.2.4. VISTA ADMINISTRADOR.....	84
6.2.4.1. Vista login.....	84
6.2.4.2. Presentación de contenidos administrador.....	85
6.3. ANÁLISIS DISEÑO LÓGICO DEL SISTEMA.....	85
6.3.1. DIAGRAMAS DE MODELO UML.....	85
6.3.1.1. Casos de uso aplicación web.....	86
6.3.1.2. Caso de uso interacción del usuario con el sitio web.....	87
6.3.1.3. Caso de uso interacción del usuario con el formulario de contactos.....	88
6.3.1.4. Diagrama de clases sitio.....	89
6.3.1.5. Casos de uso interacción del usuario con el sitio web administrador....	90
6.3.1.6. Caso de uso identificación de usuario.....	91
6.3.1.7. Caso de uso crear artículo.....	92
6.3.1.8. Caso de uso editar artículo.....	93
6.3.1.9. Caso de uso eliminar artículo.....	94
6.3.1.10 diagrama de clases administrador.....	95
6.3.2 PRUEBAS DEL SISTEMA.....	96
ANEXOS.....	97
7.1. PUBLICACIÓN DEL SISTEMA.....	98
7.1.2. CONFIGURACIONES INICIALES.....	98
7.1.3. CREAR SUBDOMINIO.....	100
7.1.4. SUBIENDO SITIO WEB.....	101
7.1.5. SUBIR Y CONFIGURAR BASE DE DATOS.....	103
7.2.1. COMO USAR BOOTSTRAP.....	106
7.2.2. SINTAXIS DE BOOTSTRAP.....	108
7.2.3. PRESTACIONES DEL FRAMEWORK BOOTSTRAP.....	108
7.2.3.1. Scaffolding.....	109
7.2.3.2. Fluid grid system.....	109
7.2.3.3. Layouts.....	110
7.2.3.4. Base css.....	112
7.2.3.5. Formularios con Bootstrap.....	112
7.2.3.5.1. Buttons De Bootstrap.....	113
7.2.3.5.2. Iconos En Bootstrap.....	114

7.2.3.7. Componentes de Bootstrap.....	115
7.2.3.7.1. Button Dropdowns.....	115
7.2.3.7.2. Navbar Bootstrap.....	116
7.2.3.7.3. Pagination Bootstrap.....	117
7.2.3.7.4. Labels Bootstrap.....	118
7.2.3.7.5. Alertas Bootstrap.....	119
7.2.4. COMPONENTES JAVA SCRIPT EN BOOTSTRAP.....	119
7.2.4.1. Modal Bootstrap.....	119
7.2.4.2. Tabs Bootstrap.....	120
7.2.4.3. Collapse Bootstrap.....	121
CONCLUSIONES.....	123
RECOMENDACIONES .....	124
BIBLIOGRAFÍA .....	125

## Índice de tablas:

Selectores CSS: .....	39
Atributos entidad page:.....	68
Índices entidad page: .....	68
Atributos entidad article: .....	69
Índices entidad article:.....	70
Atributos entidad form: .....	70
Índices entidad form: .....	72
Atributos entidad user: .....	72
Índices entidad User: .....	73
Entidad tipo vista para página: .....	73
Entidad tipo vista para tablón: .....	74
Descripción escenario interacciones sitio usuario: .....	87
Descripción escenario formulario de contactos:.....	88
Descripción escenario administrador web:.....	90
Descripción escenario identificación de usuario; .....	91
Descripción escenario creación de artículo: .....	92
Descripción de escenario edición de artículo: .....	93
Descripción escenario eliminar artículo: .....	94
Estilos de botones en Bootstrap:.....	113
Estilos labels en Bootstrap: .....	118

## RESUMEN DEL PROYECTO

Este proyecto no solo está enfocado a producir una aplicación sencilla, el resultado final del trabajo no es algo complejo, a simple vista podría decepcionar al lector, el enfoque principal de este trabajo fue describir el camino correcto para conseguir un producto de calidad, el enfoque del desarrollo fue escribir una sola vez el código, optimizar el desarrollo usando herramientas de libre acceso evitando “inventar” el agua tibia, se busca demostrar que se puede desarrollar un proyecto complejo en poco tiempo.

EL proyecto esta creado usando programación orientada a objetos y está dividido en tres secciones llamadas capas, cada capa fue documentada, las capas trabajan en conjunto y fueron creadas para facilitar la comprensión del funcionamiento del sitio.

Se usó herramientas como CodeIgniter el cual es uno de los mejores Frameworks de PHP para construir le backend y para el frontend se usó Bootstrap el Framework oficial del Twitter, no fue fácil el camino pero el resultado es bueno y lo mejor de todo, funciona.

## PRESENTACIÓN

IanCMS es un gestor de contenidos que incluye un sitio web y una aplicación que gestiona los contenidos programados en PHP, los contenidos están en una base de datos MySQL para llevar a cabo el desarrollo del proyecto se usó las siguientes herramientas libres multiplataforma

Para la elaboración del proyecto se usó herramientas y tecnologías multiplataforma listadas a continuación:

- Motor de bases de datos MySQL Community Server disponible en <http://dev.mysql.com/downloads/mysql/>
- MySQL Workbench IDE de desarrollo oficial de MySQL disponible en <http://dev.mysql.com/downloads/tools/workbench/>
- PHP Lenguaje de programación disponible en <http://php.net>
- Apache servidor web disponible en <http://httpd.apache.org/>
- SublimeText2 excelente editor de código disponible en <http://www.sublimetext.com/>
- Libre Office suite ofimática usada para redactar este documento <http://es.libreoffice.org/>
- ArgoUML herramienta usada para crear los diagramas disponible en <http://argouml.tigris.org/>
- Navegador Firefox disponible en <http://www.mozilla.org/es-ES/firefox/fx/>
- HTMLtoPHP herramienta para dar formato a los códigos disponible en <http://tohtml.com/php/>
- CodeIgniter: Herramienta principal de desarrollo para el back end disponible en <http://ellislab.com/codeigniter>
- Bootstrap: Herramienta usada en la creación de las pantallas y funcionalidades el front end disponible en: <https://github.com/twbs/bootstrap>

# CAPÍTULO 1 INTRODUCCIÓN.

## 1.1. PLANTEAMIENTO DEL PROBLEMA

Desde hace mucho tiempo no basta con crear sitios web dinámicos, capaces de interactuar con el usuario, sino que además estos mismos, deben ser capaces de interactuar con el administrador, editores, participantes y demás roles. El problema a la hora de mantener el flujo de contenidos en un sitio web, está en que la persona que lo mantiene debe tener conocimientos de programación e informática, si pesamos un momento en la cantidad de sitios web que existen y la temática de cada uno de ellos, como sitios de ventas, de belleza, de cocina, hogar por solo mencionar a algunos, no todas las empresas o personas están en capacidad de mantener trabajando a un informático para sí, ya que es muy necesario a una persona con esos conocimientos.

Este no es un problema nuevo, tiene su origen con el aparecimiento del Internet el cual dio origen a los sitios web, visto la necesidad de mantener un sitio y economizar recursos, se crean los **CMS** (Content System Management) lo que se conoce como Gestor de Contenidos traducido al español, de lo que se encargan estas herramientas es de manejar los contenidos de manera transparente al usuario, de esta forma las entidades de una organización pueden administrar el sitio sin la necesidad de conocer nada de desarrollo web. Esto no es una herramienta nueva y existen muchos CMS en el mercado ya sean libres, gratuitos o de pago, cada uno con su filosofía y sus tecnologías, los gestores de contenidos están siempre alojados en el sitio web, y nos ayudan que sea soportado por todos los navegadores web, es el encargado de gestionar los contenidos y resolver problemas de compatibilidad con cada navegador (esto solo si el diseñador del sitio le dio esa capacidad)

### 1.1.1. OBJETIVO GENERAL

Implementar un CMS (Content System Management) práctico, sencillo y fácil de usar con herramientas libres, capaz de manejar el sitio de una persona o empresa

sin la necesidad de conocimientos avanzados de informática.

### **1.1.2. OBJETIVOS ESPECÍFICOS**

- Definir el Funcionamiento de un CMS
- Implementar el uso de Frameworks, para un desarrollo rápido y robusto
- Implementar modelos de desarrollo como MVC (modelo vista controlador)
- Crear una interfaz independiente del contenido y viceversa
- Conocer Tecnologías nuevas como HTML5 CSS3 y Java Script

### **1.1.3. JUSTIFICACIÓN DEL TRABAJO**

Para vivir en Internet o incluso de él y avanzar conforme a los requerimientos no solo de los clientes sino de las autoridades, es preciso responder con prontitud y certeza a sus demandas, con esto quiero decir:

Que aunque se tuviese a un desarrollador web propio, este no está en la capacidad de responder inmediatamente todos los requerimientos, por ejemplo: Si en un sitio web dedicado al comercio electrónico formado por varias personas, si alguien deseara cambiar la descripción de un producto o agregar imágenes, no lo puede hacer directamente, tendría que buscar al profesional encargado, y aún para el desarrollador por mas experimentado y hábil que sea, va a tomar tiempo en resolverlo, además no siempre va a estar disponible y aunque exista más de un desarrollador trabajando en turnos rotativos, estos solo pueden atender una petición a la vez eso significa que están en capacidad de atender a una persona por turno, imaginemos el tiempo que tomaría editar y publicar 300 productos, además hay que considerar los costos ya que un solo desarrollador no es suficiente, entonces fuese más fácil si el mismo usuario puede modificar las características de su publicación cuando lo quiera desde donde sea y a la hora que fuese, ya que es sistema estaría trabajando 24/7 simultáneamente para todos, sin la necesidad del desarrollador, sin embargo no pretendo desvalorizar al desarrollador o empresas dedicadas a este negocio ya que es indispensable que alguna entidad le brinde soporte, ya que nada es infalible sobre todo las cosas

hechas por el hombre.

Concluyo recalcando que un sistema CMS(Content System Management) es una muy buena solución para que personas que no conocen de desarrollo web puedan publicar contenidos de cualquier índole en Internet, creando de esta forma un Internet más accesible para todos, y que las personas que visitan el sitio puedan recibir esa información conforme los deseos y gustos de autor.

#### **1.1.4. JUSTIFICACIÓN METODOLÓGICA**

El fin de este trabajo es crear un producto óptimo, que responda a las necesidades de la entidad que lo vaya a usar, para descubrir las necesidades de esa entidad nos sujetaremos a la definición de un CMS (Content System Management) y a las prestaciones que este debe tener, y de los diseños de interfaz a los que el administrador, usuario y demás se van a enfrentar durante el uso del sistema, por esa razón deben ser amigables y fáciles de usar, los pasos a seguir son los siguientes:

- Investigar el funcionamiento de un CMS a detalle
- Establecer un modelo de bases de datos que responda al análisis anterior
- Comprobar modelo de base de datos y realizar las correcciones pertinentes
- Gráfica borrador del funcionamiento del sistema, implementando todas las prestaciones que este contenga.
- Análisis por separado de las funcionalidades del sistema, junto con documentación y diagramas, como resultado tendremos el problema dividido en entidades independientes que trabajan entre sí.
- Gráfica Final del funcionamiento del sistema
- Elaboración de Estructura informática del sistema, como lo son patrones de diseño, arquitectura informática, convenciones de programación.
- Comprobar que la estructura sea eficiente y robusta, con esto lo que está buscando que la aplicación funcione en todos los navegadores, Sistema Operativo o plataforma.



- Implementar las funcionalidades de sistema dividiéndolas en módulos o clases en programación orientada a objetos, esto se llevará a cabo con la ayuda de CodeIgniter.
- Se trabaja en el diseño de la interfaz de administrador y usuario implementado a Bootstrap Frameworks CSS
- Se combina el diseño gráfico con el diseño lógico del sistema y se implementa la aplicación.
- Se realizan pruebas de funcionamiento a todas las prestaciones de la aplicación al igual que a la interfaz de usuario ya que esta debe ser lo más sencilla posible.
- Se sube la aplicación a un servidor web, en mi caso estará alojado en <http://ispade.liposerve.com>

## 1.2. DEFINICIÓN Y ORIGEN DE INTERNET

Un buen inicio sería 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. (<https://es.wikipedia.org/wiki/Internet> julio-2013).

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 usados al inicio es el WWW, existe otro protocolo el cual es uno de los más usados en la actualidad y es el HTTP, ambos protocolos realizar transferencia de HIPERTEXTO entre dos equipos o dispositivos.

### 1.2.1. 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 cabo, 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 quién 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.

## **CAPÍTULO 2 CMS Y FRAMEWORKS.**

### **2.1. DEFINICIÓN DE SITIOS WEB, SU ESTRUCTURA E IMPORTANCIA**

Un sitio web es un conjunto de páginas HTML alojadas en un servidor web, 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 página 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 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](http://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.

#### **2.1.1. PÁGINA WEB**

Es el elemento más básico de un sitio web, está formado principalmente por código HTML, CSS, y JavaScript, en el caso de sitios dinámicos es el resultado del procesamiento de una petición de usuario, en un sitio estático es el archivo

definido por el creador del sitio.

La estructura más básica de una página web está basada en etiquetas, código de una página sencilla:

```
<html>
  <head>
    <title>Mi primer sitio Web</title>
  </head>
  <body>
    <p>Hola mundo</p>
  </body>
</html>
```

Todo el contenido de una página web va entre las etiquetas `<html>` `</html>` estas etiquetas es el inicio de la estructura de una página, una página es esta compuesta por cabecera, cuerpo y pie de página.

#### **2.1.1.1. 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, o al desconocimiento del administrador ya que existen herramientas que permiten crear un sitio dinámico de forma fácil y personalizable.

#### **2.1.1.2. Sitios dinámicos**

Estos sitios aparecen gracias a la creación de herramientas informáticas desarrolladas para el manejo de información del lado del servidor, con el apareamiento de los lenguajes de programación para servidores, se desarrollaron aplicaciones que facilitaban la creación de sitios webs dinámicos complejos de una manera rápida y con menos conocimientos técnicos, esto para facilidad de muchos desarrolladores, aunque también permitieron crear páginas tan útiles como buscadores, blogs, redes sociales entre otros.

Para concluir diré que un sitio dinámico es aquel que tiene la capacidad de responder a un evento ocasionado por un ente ajeno a sí mismo, tiene la capacidad de procesar datos y transfórmalos en información.

### **2.1.1.3. Importancia de los sitios web**

La importancia de los sitios web está en lo que las empresas y personas pueden hacer con ellos, existen sitios de comercio, entretenimiento, publicidad, encuentros, buscadores.

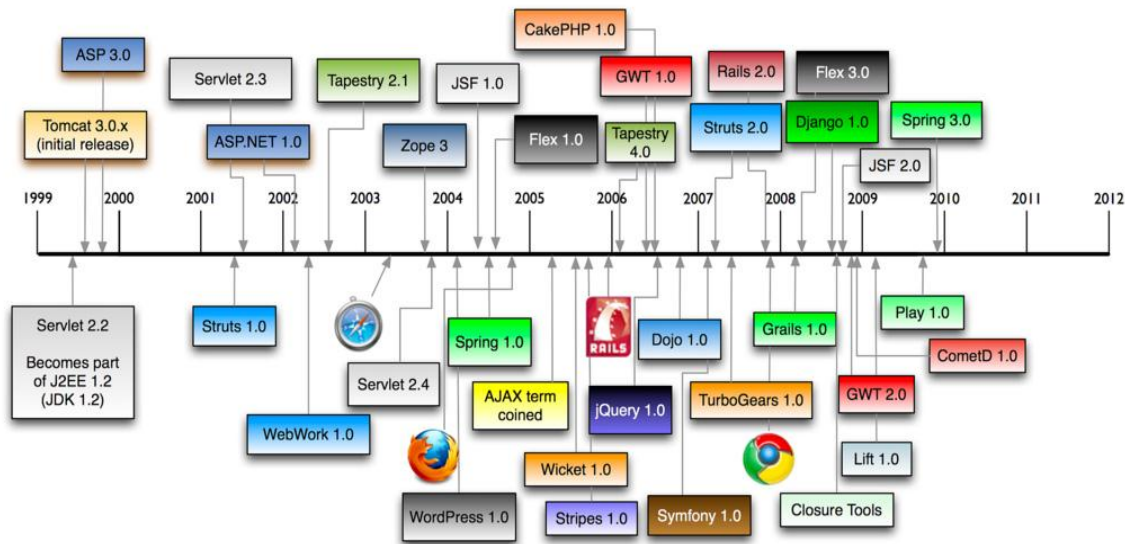
Los sitios web es uno de los muchos canales que existen para comunicarse con el cliente y dar a conocer no solo nuestros servicios sino también nuestras necesidades, el internet es parte importante de las empresas.

## **2.2 INTRODUCCIÓN A LOS FRAMEWORKS**

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 Frameworks 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 menos tiempo, podemos decir que un Framework es un esqueleto o esquema del sistema en donde se implementa una aplicación.

### **2.2.1. 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.



(<http://www.mareosdeungeek.es/wp-content/uploads/2010/02/web-frameworks.jpg>  
julio-2013)

## 2.2. 2. TIPOS DE FRAMEWORKS

### 2.2.2.1. 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.

### 2.2.2.2. 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.

### 2.2.2.3. 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.

### 2.2.3. ¿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
- Manejan la interfaz de usuario
- Brindan métodos y funciones generales
- Se aprende patrones de diseño y estilos de programación
- Los productos son más eficientes

## CAPÍTULO 3 POO.

### 3.1. INTRODUCCIÓN A LA POO

La programación orientada a objetos (POO o OOP por sus siglas en inglés) es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se representan a través de objetos a esta acción se la conoce como modelar.

### 3.2. 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 modela su comportamiento, 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.

#### 3.2.1. 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.

- Nombre del documento, este es el nombre que tiene el documento en este caso se llama factura, ya que existen otros tipos de comprobantes como Recibos o Notas de Venta
- Numero de Factura, este es un número único de factura, es el que identifica a una factura de otra.
- RUC es un numero único que identifica al emisor de las facturas
- Información para contactarse con el emisor
- Información del Cliente
- Ítems de venta, contienen el bien o servicio comercializado entre el emisor y el cliente
- 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 intentando 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 terminados 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, lo más común es encontrarnos con que el cliente es también considerado como un objeto, ya que tiene sus propios métodos y propiedades, en este caso tomamos algunas de sus propiedades y las plasmamos en el objeto factura, pero en realidad se hace referencia al objeto completo.

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.

### **3.2.2. 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 son elementos que un programador debe conocer.

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, es una entidad que trabaja directamente con la base de datos y sirve de enlace para los demás niveles de la aplicación brindando seguridad a los datos.

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 se 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 en el Objeto:**

Para definir las propiedades del objeto primero tenemos 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.

### **3.3. TIPOS DE DATOS**

La parte más importante de la programación en si son las variables, todos los lenguajes de programación sin excepción manejan variables, el contenido de las variables se lo conoce como dato, los datos tienen varios niveles de complejidad pero los más básicos son:

1. Numéricos
2. Texto
3. Booleanos

### **3.4 OPERADORES**

Que sería de las variables si no se pudieran realizar nada con ellas, los operadores nos permiten hacer toda clase de tareas con ellas, operaciones matemáticas, compararlas, asignarlas, o limpiarlas.

Todas las acciones posibles matemáticas y lógicas son el fruto de la acción de un operador PHP soporta todos los operadores existentes en los lenguajes de programación tradicionales.

#### **3.4.1. ESTRUCTURAS DE CONTROL**

Las estructura de control, ayudan a definir el flujo del programa, estos flujos son controlados por los operadores lógicos vistos en el capítulo anterior, gracias a las estructuras podemos optimizar operaciones repetitivas o recorrer arreglos, el control de las estructuras en su mayoría está definido por verdadero y falso, es decir que se ejecutan determinadas sentencias mientras se cumpla una condición, las estructuras de control son diferentes en los lenguajes pero el uso es el mismo.

### 3.5. INTRODUCCION A PHP

Esta es quizás una de las más importantes tecnologías que se usan en este proyecto, es parte central del proyecto, si la base de datos es el alma de toda aplicación este lenguaje hace que se genere el cuerpo de la aplicación, con la ayuda de este lenguaje se creará un aplicación que estará encargada de gestionar la información ingresada por el usuario y la contenida en la base de datos, se puede decir que con este lenguaje se va a crear el motor que hace que el sistema funcione, es esta una razón lo suficientemente grande como para ponerle especial atención a sus conceptos.

Lo que en realidad se hace es crear un conjunto de scripts (textos) que trabajan juntos, mismos que forman una aplicación que ayuda al usuario y administradores a usar la base de datos, pero la diferencia es que ese uso esta dictado por los algoritmos que la componen, si pensamos un segundo en la aplicación sabemos que toda la información se va a almacenar en la base de datos incluyendo la estructura del sitio (en determinadas ocasiones), esta aplicación se encarga de convertir datos en información útil al usuario.

#### 3.5.1. Referencia del lenguaje

Empecemos por definir la palabra PHP (acrónimo de *PHP: Hypertext Preprocessor*) se lo llama acrónimo porque hace referencia a sí mismo en su definición, este es un lenguaje de programación interpretado es decir que no se compila el código para poder usarlo.

PHP es un lenguaje de programación muy popular en la actualidad soporta orientación a objetos, desde la versión 5, se lo puede obtener desde <http://php.net> o incluido en un paquete como XAMP o WAMP, una de las ventajas de este lenguaje es su comunidad y su documentación, ya que la documentación es muy completa y está traducida a varios idiomas entre ellos el español, y la comunidad es muy grande, es un buen lenguaje ara iniciarse en el mundo web.

### 3.6. PROGRAMACIÓN ORIENTADA A OBJETOS EN PHP

PHP soporta la POO desde la quinta versión, en la que aplica mediante técnicas de programación los conceptos de la POO.

#### 3.6.1. 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, para una mejor comprensión se puede imaginar a la clase como si fuese los planos de un objeto antes de ser construido. El objeto es la construcción derivada de una clase, a esto se le conoce con el nombre de instancia, sin importar el número de veces que se instancie una clase cada uno de los objetos es independiente y único, no compartes métodos ni propiedades aunque los mismos estén presentes en cada uno.

Revisando el manual oficial de PHP tenemos la siguiente definición para 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 válida 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...”*  
( <http://www.php.net/manual/es/language.oop5.basic.php> julio-2013).

Ejemplo de clase válida en PHP:

```
<?php
class Valor{
    public $Resultado_;
    protected function Inicio(){
        $this->Resultado_ = (rand(23,1000));
        return $this->Resultado_;
```



```

    }
}

$numero = new Valor();
print $numero->Inicio();
?>

```

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

Las clases no son de un solo tipo, PHP soporta varios tipos de clases, mismas que tienen aplicaciones diferentes.

### 3.6.2. CLASE HEREDADA

Estas son conocidas también como clase hija o dependiente, 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 le permita ver, una aclaración de esto se verá en el subcapítulo ámbito de los métodos y Propiedades.

Para definir una clase heredada se usa la palabra reservada “class” seguido por un nombre válido a continuación la palabra “extends” y el nombre de la clase de la que se hereda, la herencia se aplica de uno a uno no es posible heredar de varias clases, en caso de ser necesario implementar a más de una clase se aplica herencia en cascada, imaginemos que tenemos tres clases “a”, “b” y “c”, por algún motivo queremos heredar de “a” y “b” a “c”, no se puede aplicar herencia de manera simultánea primero se debe heredar de “a” a “b” y luego de “b” a “c”.

Ejemplo de clase Heredada en PHP.

```

<?php

class Edificacion{

    //clase que representa una edificación

}

```

```

class Casa extends Edificacion{

    //tiene acceso a las propiedades y métodos
    //de Edificación

}

//objeto con las propiedades y métodos de la clase
// Casa

$micasa = new Casa();

?>

```

En este caso el objeto “**micasa**” es la representación de la clase casa la misma que tiene acceso a los métodos y propiedades de edificación.

### 3.6.3. MÉTODOS

Es el algoritmo asociado a una clase en forma de función PHP, son los métodos los que definen la capacidad de una clase, una función se define con la palabra clave “función” seguido por un nombre válido de función, para llamar a un método dentro de la clase es necesario el uso de la palabra clave “this” seguido de los símbolos “->” guión mayor que, los métodos pueden tener una visibilidad.

## 3.7. PROPIEDADES

Las propiedades son el componente de la clase que almacena la información del objeto, también son los elementos con los que los métodos de la clase trabajan, una propiedad en PHP se define luego de la declaración de la clase fuera de cualquier función, las variables definidas en el cuerpo del método solo existen cuando se convoca al método y solo él método al que pertenece puede trabajar con esa información.

Las propiedades se las define con la palabra clave “var” seguido por la variable PHP válida es decir precedida del signo de dólar \$, las propiedades al igual que los métodos tienen una visibilidad.

### 3.7.1. ÁMBITOS DE PROPIEDADES Y MÉTODOS EN PHP

En PHP al igual que en otros lenguajes de programación incorporan ámbitos y un

alcance para las propiedades y métodos, una de las principales razones es proteger los datos de una clase de los demás componentes de la aplicación, también ayudan a optimizar el uso de memoria y capacidad de procesamiento, los métodos y propiedades puede ser:

- públicos
- privados
- estáticos
- protegidos

## CAPÍTULO 4 TECNOLOGÍAS PROYECTO.

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 lo 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.

### 4.1. 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 lenguaje de etiquetas usado para elaborar páginas web, HTML describe la estructura e información de la página en forma de texto dentro de un archivo con extensión “.html”, HTML no es un lenguaje de programación es una tecnología que permite estructurar una página web y los contenidos de la misma, en el mundo web a HTML se lo conoce como lenguaje semántico. HTML es la base de la web moderna junto a otras tecnologías como CSS y Java Script que son las que le dan estilo y funcionalidad a los sitios web, el control del desarrollo de estas tecnologías lo lleva la WW3C.

HTML es un lenguaje de etiquetas que interpreta el navegador produciendo un resultado visual la mayoría de veces, para mejorar la apariencia de las páginas HTML existe una tecnología de hojas de estilo llamada CSS.

#### 4.1.1. DEFINICIÓN DOCUMENTO

Como se ha venido hablando existen varios estándares de HTML, cuando el diseñador de la página empieza a escribir su estructura debe especificar el estándar a usar para esto se lo especificar con la ayuda del “**DOCTYPE**”, veamos los tipos de definición de documento que tenemos para HTML5:

```
<!DOCTYPE html>
```

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

En el listado anterior tenemos la definición de para los estándares 4 y 5 de HTML, para definir el estándar usado es necesario escribir la palabra **"DOCTYPE"** antes de **"html"** luego la palabra PUBLIC la especificación detallada de la versión la rama de esa versión se está **(transitional,frameset,strict)** y luego de dos barras inclinadas la abreviado del lenguaje a usar, en el caso de los ejemplos anteriores el español conocido por la abreviación internacional ES.

#### 4.1.2. ETIQUETAS

Esta es la parte más básica de HTML las etiquetas son de dos tipos de apertura y cierre, todo lo que va entre las etiquetas es el contenido este contenido puede ser texto u otras etiquetas que también pueden contener texto o etiquetas, por esta razón al documento se lo suele llamar árbol de etiquetas.

La forma de escribir una etiqueta es con el símbolo 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:

```
<html>
<!-- Comentario en HTML -->
<!--Apertura de etiqueta head-->
<head>
    <!--Contenido de head-->
    <title></title>
</head>
<!--Cierre de la etiqueta head-->
<body>
    <!--Contenido de body-->
</body>
</html>
```

En el ejemplo anterior se describe la estructura básica de una página HTML, tenemos a las etiquetas más importantes de una página web, la etiqueta “<html>” es la que contiene a todas las etiquetas de la página sin ellas no existe la página web, la etiqueta “<head>” no solo sirve para darle un título a la página sino también para adjuntar hojas de estilo y scripts Java Script este último tipo de ficheros es recomendable adjuntarlos al final de la página antes del cierre de la etiqueta “<html>” esto ayuda a que el sitio se cargue más rápido.

Existe un estándar llamado XHTML este es un estándar muy estricto, una de las restricciones que tiene es que no permite abrir y cerrar etiquetas que no estén escritas de igual manera (solo en mayúsculas o minúsculas), exige al programador que las etiquetas se cierren en el orden que fueron abiertas.

## 4.2. INTRODUCCIÓN A CSS

CSS “**Cascading Style Sheet**” hojas de estilo en cascada, es una tecnología usada para maquetar sitios web, con CSS se manipula la apariencia y posición de las etiquetas HTML, esta herramienta permite crear sitios web estéticos, trabaja con colores, sombras, degradados, etc.

### 4.2.1. SINTAXIS BÁSICA

La sintaxis de CSS es simple, es necesario un selector al que se le van a aplicar estilo, el conjunto de sentencias de estilo van entre llaves para aplicar un estilo basta con escribir el nombre del estilo seguido por el símbolo dos puntos “:” y el valor del atributo terminando la línea en punto y coma “;”, CSS soporta comentarios de C/C++.

### 4.2.2. SELECTORES

Son los selectores quienes eligen el destino de los estilos, con un selector se elige a las etiquetas que se desee aplicar estilos, los selectores CSS permiten

seleccionar cualquier elemento de la página y en cualquier instante, si una etiqueta cambia un atributo o estado, el selector aplica el estilo de manera inmediata.

### Selectores CSS:

Selector	Ejemplo	Descripción	CSS
<code>.class</code>	<code>.intro</code>	Selecciona las etiquetas con <code>class="intro"</code>	1
<a href="#"><u>#id</u></a>	<code>#firstname</code>	Selecciona la etiqueta con <code>id="firstname"</code>	1
<a href="#"><u>*</u></a>	<code>*</code>	Selecciona todas las etiquetas	2
<a href="#"><u>etiqueta</u></a>	<code>p</code>	Selección todas las etiquetas <code>&lt;p&gt;</code>	1
<a href="#"><u>etiqueta,etiqueta</u></a>	<code>div,p</code>	Selecciona todas las etiquetas <code>&lt;div&gt;</code> y <code>&lt;p&gt;</code>	1
<a href="#"><u>Etiqueta.etiqueta</u></a>	<code>div p</code>	Selecciona todas las etiquetas <code>&lt;p&gt;</code> que tengan una etiqueta dentro <code>&lt;div&gt;</code>	1
<a href="#"><u>etiqueta&gt;etiqueta</u></a>	<code>div&gt;p</code>	Selecciona todas las etiquetas <code>&lt;p&gt;</code> dentro de la etiquetas <code>&lt;div&gt;</code>	2
<a href="#"><u>etiqueta+etiqueta</u></a>	<code>div+p</code>	Selecciona todas las etiquetas <code>&lt;p&gt;</code> que tienen un etiqueta <code>&lt;div&gt;</code> como siguiente elemento	2
<a href="#"><u>:link</u></a>	<code>a:link</code>	Selecciona link sin visitar	1
<a href="#"><u>:visited</u></a>	<code>a:visited</code>	Selecciona link visitados	1
<a href="#"><u>:active</u></a>	<code>a:active</code>	Selecciona links activos	1
<a href="#"><u>:root</u></a>	<code>:root</code>	Selecciona etiquetas de la raíz del documento	3
<a href="#"><u>::selection</u></a>	<code>::selection</code>	Selecciona las etiquetas seleccionadas por el usuario.	3

## 4.3 INTRODUCCIÓN A JAVA SCRIPT

Este es un lenguaje de programación interpretado que se ejecuta en el navegador del cliente, al inicio su principal funcionalidad fue la de verificar los datos que el usuario enviaba al servidor, en la actualidad esta tecnología es usada para crear sitios interactivos, juegos, animaciones, etc. Es una tecnología muy importante en la web creó el concepto de tiempo real en la web, un ejemplo son las notificaciones de las redes sociales, Java Script no tiene nada que ver con el lenguaje de programación Java de Oracle, la sintaxis de JavaScript es similar a C/C++, la extensión de los ficheros con JavaScript es “.js”.

### 4.3.1. SINTAXIS BÁSICA

La sintaxis está basada en C/C++, al igual que CSS JavaScript también posee selectores, dispone de eventos lo que permite ejecutar determinadas líneas de código cuando un evento en especial suceda en el sitio, estos eventos son carga del sitio, movimientos de cursor, clics, etc. JavaScript es un lenguaje funcional es decir soporta orientación a objetos, funciones, tiene funciones nativas aplicadas a manejo de cadenas, manejo de funciones matemáticas.

Los comentarios soportados en JavaScript son los mismos de C/C++, las sentencias terminan en un punto y coma.

### 4.3.2. IMPORTANCIA DE JAVA SCRIPT EN LA ACTUALIDAD

Sin la ayuda de esta tecnología sería imposible la existencia de sitios como Amazon, Facebook, mercado libre, google por solo mencionar a una ínfima minoría, en el ultimo estándar de HTML JavaScript recibe una atención especial tiene la capacidad de crear juegos reproducir vídeos, y se puede crear animaciones de muy alta calidad con un bajo coste de tráfico, ya que todo se ejecuta en el cliente.



## 4.4. INTRODUCCIÓN A MYSQL

MySQL es una marca registrada por MySQL AB subsidiaria de Sun Microsystems adquirida posteriormente por Oracle en el año 2009. Todas las aplicaciones necesitan almacenar datos, las bases de datos son muy usadas, por la facilidad que brindan al momento de almacenar y recupera la información, dejando el manejo de ficheros a la aplicación de base de datos, estas aplicaciones son conocidas como motores de bases de datos.

MySQL es un motor de bases de datos multiusuario con licenciamiento dual, si se desea implementar MySQL en proyectos abiertos se usa la versión libre, para proyectos con carácter comercial se usa la versión de pago, en el nuestro caso usaremos la libre.

Se puede obtener el ejecutable de MySQL de la página oficial <http://mysql.com> , existen soluciones que ya traen instalado MySQL en entornos de desarrollo y producción como WAMP o XAMP, estas soluciones ofrecen un entorno completo incluyendo un servidor web y el intérprete de PHP. Si la necesidad es solo instalar MySQL se lo puede hacer obteniendo el binario.

### 4.4.1. BASES DE DATOS

Es el conjunto de datos de varios tipos, organizados e interrelacionados almacenados generalmente en estructuras que se guardan en algún medio.

Es necesario conocer algunos elementos que forman parte de una base de datos para aplicar conceptos más complejos como consultas.

#### 4.4.1.1. Tabla

Una tabla es una colección de columnas cuyos registros almacenan información creada a partir de una abstracción del mundo real, a esa abstracción se la conoce como entidad dentro POO, la tabla debe tener un nombre y al menos una columna.

#### 4.4.1.2. Columna

La columna es la unidad básica de la estructura de una tabla, posee un nombre y un tipo de dato, es el nombre de la columna a quien se referencia para realizar consultas.

#### 4.4.2. TIPOS DE DATOS MYSQL

Los tipos de datos en MySQL indican la información que se puede asignar a una columna, este tipo de dato el único aceptado por la columna para cada registro, los tipos de datos soportados por MySQL, son los mismos estudiados en el capítulo de POO, al ser un motor de datos es necesario optimizar recursos, por esta razón existen muchos tipos de datos similares, la diferencia es los recursos que usan.

En el caso de las variables que tienen un sufijo “n” significa que aceptan como parámetro un entero que especifica la longitud.

Dato es una palabra ambigua para no crear una nueva definición citare un fragmento de un libro llamado **Mysql con clase** que dice:

Debe cumplir algunas condiciones, por ejemplo, debe permanecer en el tiempo. En ese sentido, estrictamente hablando, una edad no es un dato, ya que varía con el tiempo. El dato sería la fecha de nacimiento, y la edad se calcula a partir de ese dato y de la fecha actual. Además, debe tener un significado, y debe ser manipulable mediante operadores: comparaciones, sumas, restas, etc. (por supuesto, no todos los datos admiten todos los operadores).(tomado de <http://mysql.conclase.net/curso/index.php?cap=001> # julio-2013)

Es una excelente definición, razón por la que la cito textualmente.

#### 4.4.3. DISEÑO DE BASES DE DATOS MODELO ENTIDAD RELACIÓN

Este es un proceso en el que se traslada un problema del mundo a una base de datos, a este procedimiento se le llama modelado, para poder realizar el modelado

es necesario comprender el problema de la mejor forma posible y aplicar conceptos que permiten crear bases de datos funcionales, estos conceptos reciben el nombre de normalización.

#### **4.4.3.1. Modelo entidad relación**

Este modelo toma un problema de la vida real y lo transforma en entidades relacionadas entre sí. Estas entidades pueden ser tablas, vistas, procedimientos, funciones, una consulta, una entidad es la representación de un objeto de la vida real, este modelo ayuda a comprender de mejor manera el problema y forma la base de sistema.

- La asociación que existe entre las entidades es conocida como relación.
- La unión de estos dos conceptos forman un modelo entidad relación.

#### **4.4.3.2. Clave primaria**

Es común encontrarse con claves primarias dentro de una tabla de bases de datos, existen motores que no permiten crear una tabla si el desarrollador no especifica una. La clave primaria es aquella que identifica un registro en la tabla.

#### **4.4.3.3. Clave foránea**

Permite 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 también clave principal.

- Uno a muchos (1:n)

Esta es una relación donde las claves de la entidad padre tienen más de una correspondencia en la entidad hija.

- Muchos a muchos (n:m)

Esta es una relación que se puede expresar pero no se puede implementar directamente, porque no asegura integridad en los datos, lógicamente no es posible crear una relación de este tipo entre dos entidades porque no existe un punto de control, es necesario crear una nueva entidad que maneje esa relación, la nueva

entidad tiene una relación de 1:n con las dos entidades implicadas y registra las correspondencias.

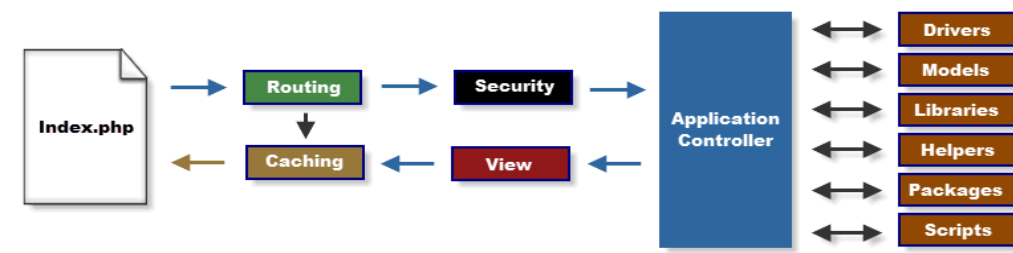
## 4.5. INTRODUCCIÓN A CODEIGNITER PHP

CodeIgniter es un Framework para desarrollo rápido de aplicaciones web con orientación a objetos usando el lenguaje de programación PHP, es un Framework muy ligero y flexible permite crear aplicaciones muy potentes con poco código, CodeIgniter permite al desarrollador enfocarse en el producto final, en lugar de programar librerías, dispone de características para todo tipo de actividades como manejo de archivos, sesiones, base de datos, entre otros.

CodeIgniter es un Framework muy sencillo de usar posee una curva de aprendizaje corta, brinda al desarrollador la facilidad de usar o no las librerías, se puede crear librerías y helpers o extender el core del Framework, es uno de los Frameworks aceptados por los creadores de PHP.

Diagrama de Flujo de la Aplicación

CodeIgniter ofrece un diagrama que especifica su funcionamiento para una mejor comprensión del desarrollador.



La parte que controla el flujo de la aplicación es el “**index.php**” este solicita un recurso a través de URL, se comprueba si la solicitud afecta la seguridad del sistema, en caso de ser una solicitud normal es enviada al controlador de la aplicación, es el controlador quien trabaja con los demás elementos del Framework, se considera a los controladores el núcleo de la aplicación .

#### 4.5.1. MODELO VISTA CONTROLADOR

Este es un patrón de diseño usado para crear software, este patrón divide a una aplicación en tres secciones conocidas como capas:

##### 4.5.1.1. Modelo

En esta capa se trabaja con los datos, se crean objetos y métodos que representan a la base de datos, trabajan sobre ella, es la única capa con acceso directo a la fuente de datos en la aplicación, representa a la base de datos y pasa la información a la capa superior.

##### 4.5.1.2. Vista

Esta capa contiene la estructura visual de la aplicación llamadas plantillas, las plantillas están construidas con código HTML para la parte estática (estructura) y PHP para la parte dinámica (Información).

##### 4.5.1.3. Controlador

Es la parte central del sitio, trabaja entre el modelo y las vistas, es el encargado de generar la página web, el controlador es quien manipula al modelo para obtener datos a través solicitudes, se toman los datos y se los fusiona con las vistas, generando una página con información útil al usuario.

#### 4.5.2. DISTRIBUCIÓN DIRECTORIOS CODEIGNITER

Los directorios en CodeIgniter son tres especificados de la siguiente manera:

- **application:** Lugar donde se guarda el código fuente de la aplicación creada por el desarrollador.
- **system:** directorio donde se encuentra alojado el código del Framework o núcleo (core en inglés), no se debe modificar ninguno de sus contenidos, a menos que se desee agregar nuevas funcionalidades al Framework.
- **user\_guide:** Guía de usuario en inglés, se puede eliminar este directorio sin que ello represente problemas al sistema.

Además posee un archivo con el texto de la licencia del Framework y el fichero de

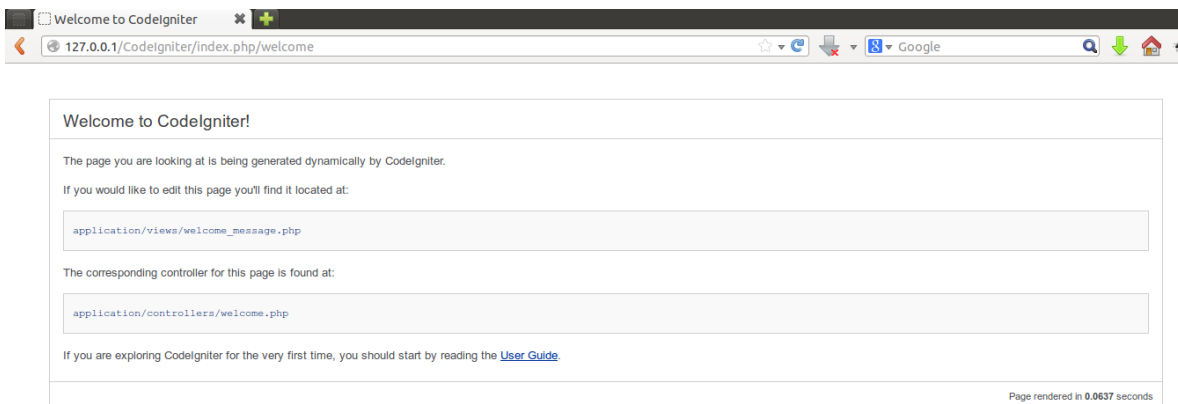
inicio llamado “index.php” es el encargado de hacer funcionar el sitio.

### 4.5.3. INSTRUCCIONES DE INSTALACIÓN

CodeIgniter necesita que el servidor en el que este instalado PHP por ser el lenguaje que lo compone, un motor de base de datos (opcional), y una aplicación de servicio HTTP como apache, con estos requisitos CodeIgniter es muy fácil instalar y usar, basta con descargar el .zip de la aplicación y descomprimirlo en el directorio en el que se almacenan los sitios web generalmente depende de la solución que se instale en el servidor, se dispone un guía de instalación en los anexos de este documento.

Una vez que se tiene el fichero descomprimido en el directorio correspondiente podemos acceder a él desde un navegador escribiendo la siguiente ruta [http://127.0.0.1/nombre\\_carpeta](http://127.0.0.1/nombre_carpeta) esta ruta es la dirección IP del computador y el nombre de la carpeta en la que se extrajo el fichero .zip, aparecerá en la pantalla del navegador lo siguiente:

### 4.5.4. CONFIGURACIÓN



Se puede apreciar que CodeIgniter no necesita de ninguna configuración inicial para funcionar.

Se recomienda revisar los ficheros de configuración ubicado en “**application/config**” los más importantes de este directorio son:

- **autoload:** En este fichero se especifican elementos que deben estar

siempre presente, es muy útil para cargar de manera automática ficheros de configuración, librerías, modelos y helpers.

- **config:** Fichero de configuración de seguridad, rutas, nombres de sesiones página principal, lenguaje, juego de caracteres entre otros.
- **database:** En este fichero se encuentra un conjunto de opciones que permiten a la aplicación conectarse con la mayoría de motores de bases de datos.
- **routes:** Fichero en el que se puede definir rutas, especifica rutas para controlador principal y la página de contenido no encontrado.

Adicional CodeIgniter permite realizar cambios en los nombres de los directorios tanto de la aplicación como del núcleo del Framework, en caso de cambiar el nombre de los directorios se debe modificar el valor asignados a las variables “**\$system\_path**” y “**\$application\_folder**” del fichero “**index.php**” por el nuevo nombre del directorio.

El desarrollador debe conocer el funcionamiento al menos de los conceptos básicos del Framework a continuación se describen los principales.

#### 4.5.5. URLS DE CODEIGNITER

Las URL's CodeIgniter están estructuradas en segmentos URI, están diseñadas para ser amigables al usuario y buscadores, el funcionamiento básico está dado por el patrón modelo-vista-controlador iniciando con el nombre del dominio seguido del nombre del controlador y el nombre de método que atiende la petición seguido por los argumentos del método.

Sintaxis:

`http://sitio.com/[clase-controlador]/[método-controlador]/[argumentos]`

Ejemplos:

`http://sitio.com/`

`http://sitio.com/index.php/inicio/`

[http://sitio.com/index.php/noticias/mi\\_noticia/](http://sitio.com/index.php/noticias/mi_noticia/)

<http://sitio.com/index.php/zapatos/buscar/color/negro/talla/40>

El primer ejemplo ejecuta el controlador por defecto del sitio, este controlador ejecuta el método “**index**” este método se ejecutará siempre y cuando no se especifique un nombre de método distinto, en el segundo se especifica el nombre de un controlador, el tercero adicional ejecuta un método específico, en el último ejemplo la URL especifica el método (buscar) de un controlador (zapatos) le pasa los argumentos color negro y talla 40.

## 4.6. INTRODUCCIÓN A BOOTSTRAP CSS

Bootstrap es el encargado de darle un estilo visual a nuestro sitio, haciéndolo elegante y dotándolo de funcionalidad, Bootstrap es muy sencillo de usar y no requiere de conocimientos en CSS avanzados permite crear páginas de forma rápida, tan solo se debe seleccionar los elementos que deseamos incorporarlos al sitio, Bootstrap cuenta con elementos para realizar lo siguiente:

- Estructura de la página (layouts)
- Imágenes
- Menús
- Botones
- Labels
- Formularios
- Tablas
- Iconos
- Cuadros de diálogo
- Ventanas Emergentes



- Animaciones
- Sliders
- Etc.

En el presente documento se le hace una breve referencia a las características de este Framework CSS

## CAPITULO 5 DESARROLLO BASE DE DATOS

### 5.1 RECOLECCIÓN DE INFORMACIÓN

Para entender el problema es necesario adquirir conocimientos en el área, luego de tener la suficiente información se puede plantear el problema de una mejor manera y con un enfoque práctico.

#### 5.1.1. IDENTIFICAR FUENTES DE INFORMACIÓN

Antes de buscar fuentes de información se debe conocer lo que es, según Wikipedia la información es:

“En sentido general, la información es un conjunto organizado de datos procesados, que constituyen un mensaje que cambia el estado de conocimiento del sujeto o sistema que recibe dicho mensaje...”

(<http://es.wikipedia.org/wiki/Informaci%C3%B3n> Agosto-2013)

Entonces la información son contenidos textuales de un elemento o ente que se puede estudiar, es indispensable ubicar fuentes de información verás.

Para identificar estas fuentes se puede basar en la calificación que los demás lectores dan al artículo en el caso de ser fuentes de libre acceso pero con procedencia no especificada como lo son los artículos de Wikipedia, es recomendable acceder a la información pública de fuentes de información pertenecientes a organizaciones privadas, o incluso a la información con un fin comercial como los libros o revistas.

#### 5.1.2. DEFINICIÓN DE PROBLEMA. ¿QUÉ ES UN CMS?

La traducción literal de **CMS (Content System Manager)** es **Sistema de Gestión de Contenidos**.

Esta definición no es tan clara como se podría desear pero su objetivo es dar a conocer lo que es un CMS, un CSM es un sistema que controla los contenidos y recursos de un sitio, un sitio web por lo general tiene incorporado un CMS, este

para ser el núcleo del sitio o página de internet, la página web como tal es el resultado del pre procesamiento de datos, para entenderlo se lo puede comparar con un libro, el libro es el gestos de páginas y la página es una sección del libro pero no es el libro en sí.

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, el CMS guarda las rutas y las enlaza a las páginas para que el usuario las pueda ver.

- Gestión de Ficheros:

Gestiona todos los ficheros del sitio, los componentes de la página los archivos de conexión, la gestión de ficheros se encarga de mostrar a la aplicación en donde y como llegar a los archivos y componentes de la aplicación.

- Gestión de base de datos

Permite conectarse, obtener y escribir datos en la base de datos.

- Gestión de Información:

Después de obtener los datos los procesa y convierte en información estructurada de fácil comprensión para el ser humano.

- Gestión de usuarios:

Para un CMS no todos los usuarios son los mimos, posee métodos para identificar los usuarios, por ejemplo el usuario administrador posee un acceso diferente a la aplicación.

### **5.1.3. HISTORIA DE CMS**

La aparición del primer CMS como tal, data a 1995 cuando el sitio de tecnología CNET (<http://www.cnet.com/>) publico un sistema que manejaba sus contenidos,

para esa época los sitios web no eran más que pantallas estáticas enlazadas entre sí, el sector en impulsar el desarrollo de los CMS son las empresas privadas de comunicación, revistas, tiendas, es decir cualquier ente que necesitara modificar sus sitios de forma simple y rápida.

#### **5.1.3.1. TIPOS DE CMS**

Los CMS crearon un mercado, existen grandes corporaciones que patrocinan sistemas de este tipo, así como sociedades sin fines de lucro o personas con el suficiente intelecto y voluntad.

##### ***5.1.3.1.1. CMS abiertos***

Existe una gran confusión, mucha gente piensa que los CMS de tecnologías abiertas son gratuitos y libres, la verdad no siempre un producto abierto es gratuito, la razón por la que los propietarios publican su código y permiten modificarlo, es para que más desarrolladores apoyen el proyecto. En el caso de estos sistemas existen varios modelos de negocio, existen CMS que son abiertos, libres y gratuitos, la empresa no obtiene beneficios desarrollando el sistema, los obtiene por soporte o venta de plugins que son funcionalidades extras al sistema.

##### ***5.1.3.1.2. CMS cerrados***

Los CMS de este tipo no siempre son comerciales, aunque es su fin, en lo que respecta a sistemas web no existen binarios de una página web todo es código, por esta razón el cliente siempre obtiene el código fuente del sistema, ya que lo necesita para que el servidor lo interprete.

Se conoce como tecnología cerrada a aquella que por alguna razón legal o física no permitan realizar cambios en el código del sistema.

##### ***5.1.3.1.3. CMS comerciales***

Estos CMS pueden ser además de cualquier tipo anterior, pero su uso está condicionado a un licenciamiento por parte del usuario final, la ventaja de esté es que el software viene con garantías de funcionalidad, los dos anteriores no siempre.

#### **5.1.3.1.4. Ventajas CMS**

Los CMS representan muchas ventajas, mismas que lo convierten en una herramienta indispensable cuando de sitios web se habla, por la facilidad de gestionar una página, gracias a estas herramientas la web está más cerca de las personas.

#### **5.1.3.1.5. Desventajas CMS**

Las desventajas dependen de la herramienta seleccionada, para evitar inconvenientes a futuro se recomienda estudiar atentamente las alternativas, todo depende de las necesidades del sitio, no siempre un CMS de pago es la mejor opción, sucede lo mismo con los libres, en ocasiones un desarrollador opta por el CMS con más fama en lugar de mirar la calidad de su comunidad, algunos tips para ser una buena elección:

- El CMS debe ser de fácil comprensión
- Soporte por parte de alguna entidad
- Que tenga un desarrollo constante
- No sea demasiado complicado
- Que tenga documentación
- Análisis de requerimientos

#### **5.1.3.2. Estudio del funcionamiento de un CMS**

Para determinar el funcionamiento de CMS es necesario un estudio a profundidad del problema.

Un CMS maneja la información protegiéndola de los usuarios no autorizados, transforma datos que carecen de un sentido propio de una base de datos en información útil y fácil de comprender, además gestiona los componentes visuales del sitio es decir, luego de obtener y procesar los datos, toma los elementos necesarios para generar una página, corvina estos dos resultados y devuelve una página web de fácil comprensión, se puede incluir al sistema CMS herramientas externas para mejorar el control como:

- 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 estadísticos existen ni cuál es el mejor, sino de recomendar uno que sea bueno y de preferencia gratuito, dentro de esta recomendación esta Google Analytics 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 más lee en nuestro sitio, cuánto tiempo le dedica al sitio, desde donde está viendo el sitio, que sistema operativo usa, su navegador, que tipo de dispositivo usa ya sea teléfonos tabletas u otros.

- Google Custom Search (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, haciendo que el usuario necesite más tiempo para encontrar algo de su interés. Para enfrentar este inconveniente tenemos un producto de Google llamado Google Custom Search, lo que hace este sistema es identificar todo el contenido del sitio y lo indexa a Google, una vez implementado en la página 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 de forma gratuita.

Las dos herramientas anteriores ahorran tiempo y trabajo al desarrollo del sitio, ya que son dos cosas muy importantes en las que el desarrollador no se preocupa, permitiéndole enfocarse en los demás elementos del sitio.

En un gráfico sencillo se describe el funcionamiento de un CMS:



El CMS actúa como un agente de control entre elementos de una entidad, analizando el gráfico anterior podemos ver que el CMS posee una comunicación bilateral entre algunos entes y unilateral con otros, esto es debido a la importancia de los usuarios al sistema y sus tareas, un CMS es un sistema de gestión apoyado por una serie de usuarios a distintos niveles.

### 5.1.3.3. Diseño CMS

Un CMS permite a los diseñadores alterar la presentación visual del sitio permitiéndole manipular sus plantillas.

**Generadores De Contenido:** Conocidos también como data entry son los encargados de ingresar información al sitio, este tipo de información no está necesariamente verificada, por lo general aquí existen más usuarios.

**Editores:** Los Editores son los encargados de revisar la información ingresada por los generadores de contenidos, verificarla de ser posible y realizar correcciones gramáticas en los textos.

**Colaboradores:** Son un tipo de usuario de un nivel superior al de los visitantes, son entidades con una serie de permisos y tareas asignadas en el sistema.

**Visitantes:** Los visitantes son los usuarios que consumen el contenido del sitio, lo único que estos pueden hacer es solicitar información, no poseen una interacción bidireccional con el servidor.

#### **5.1.3.4. Análisis de las prestaciones CMS**

Con el conocimiento necesario se establece las características del sistema a desarrollar, estas dependen de las necesidades del usuario final o cliente, se recomienda realizar este análisis junto al cliente, para que el producto final sea de su agrado, tomando en cuenta las variables:

- Costos
- Tiempos
- Requerimientos físicos
- Requerimientos humanos

Las características del sistema se deben enfocar en ser las más simples y elementales para el entorno.

#### **5.1.4. ALMACENAMIENTO DE DATOS**

La información del sitio se divide en dos tipos, la información basada en textos como son los contenidos de los artículos y la información basada en objetos como son las imágenes y archivos PDF, los datos que se puedan almacenar en forma de texto se los aislara en una base de datos relacional, los datos que se compongan de elementos “físicos” se almacenan en un directorio del servidor, y se guarda el nombre y la ubicación junto a los datos de texto para luego relacionarlos.

Este CMS está creado para administrar artículos, es decir que se pueden crear, editar y eliminar artículos, los artículos poseen texto y en algunos casos otros elementos como imágenes o vídeos, también se toma en cuenta que es un único usuario el que tiene el control del sitio, los demás serán solamente usuarios visitantes.

En conclusión el almacenamiento de datos se divide en dos siendo la base de datos la que registra el nombre y la ubicación de los datos “físicos”



#### **5.1.4.1. Manipulación de datos por el administrador**

Al administrador posee la capacidad de administrar los contenidos del sitio creándolos, modificándoles o dándoles de baja, este usuario es incapaz de acceder a la estructura del sistema, no puede realizar cambios en el funcionamiento del sistema ni de mejorar las existentes, el administrador está en capacidad de trabajar con la información del sitio.

#### **5.1.4.2. Flujo de datos**

El flujo de datos se origina en el servidor precisamente en la base de datos y en los ficheros que contienen información, los datos son transformados en información por parte de CMS y presentados al usuario, el recorrido de los mismos empieza como datos brutos y termina como información útil.

### **5.2. DEFINICIÓN DE LAS PRESTACIONES DEL SISTEMA**

En este se inicia con el desarrollo de la aplicación en sí, se trabaja con metodologías y siguiendo patrones de diseño de software para que el producto obtenido al final sea del agrado del cliente.

#### **5.2.1 PRESTACIONES DEL SISTEMA**

Para la definición de las características del sistema en este caso depende de el alcance que deseemos implementar él, se definen características básicas de acuerdo al tiempo disponible para el desarrollo de la aplicación.

- Almacenamiento de información
- Gestión de base de datos
- Gestión de ficheros
- Gestión de pantallas
- Gestión de artículos por páginas (paginación)

- Administración de usuarios
- Gestión de URL's
- Gestión de Menús
- Gestión de artículos

Estas características son básicas en el funcionamiento de un CMS tradicional cada una de ellas son analizadas por separado en el siguiente subcapítulo.

### **5.2.2. ANÁLISIS DE REQUERIMIENTOS**

Una vez obtenido un listado de lo que se desea que el sistema pueda hacer es necesario analizar cada característica por separado antes de implementar.

#### **5.2.2.1. Requisitos de hardware**

Se entiende que el sistema trabajará en un servidor web en un dominio existente o nuevo, en un inicio se necesita de un servidor con transferencia mensual ilimitada al menos unos 5GB de espacio en disco, conforme evolucione el proyecto se analizará implementar una mejora en el servidor, si en el servidor que se desea implementar el sistema funciona algún otro tipo de aplicación es necesario examinar la misma para definir si ese servidor es el que se necesita.

#### **5.2.2.2. Requisitos de software**

En cuanto al Software se requiere usar las mismas tecnologías y herramientas usadas para crear el sistema.

- Motor de bases de datos MySQL versión 5.0
- Un servidor HTTP apache versión 2.0
- Un servidor FTP para subir los ficheros del sistema cualquier versión
- Un intérprete PHP versión 5.5
- Bootstrap CSS
- CodeIgniter

#### **5.2.2.3. Personal humano**

en cuanto a los requerimientos de personal humano se necesita una persona que sea capaz de trabajar con un computador debe poseer conocimientos de informática.

#### **5.2.2.4. Almacenamiento de información**

Para almacenar la información se posee de varios métodos, en este caso se opta por dos métodos uno para los contenidos que se puedan expresar en forma de texto y otro para los contenidos que no.

En el caso de los contenidos que se puedan ser almacenados en forma de texto se usará un modelo relacional de base de datos, en este se guarda todos los datos del sitio.

Para el caso en el que los datos sean ficheros estos serán almacenados en un directorio en el servidor, el CMS debe tomar la ruta del directorio el nombre del fichero con su extensión y guardar esos datos en la base de datos, para luego relacionar los contenidos.

### **5.3. DEFINICIÓN ESTRUCTURA DEL SITIO**

Luego de definir las especificaciones es necesario estructurar el sitio, definir los componentes del sitio, para aclarar la idea listemos las páginas.

- Pagina de Inicio
- Pagina de Nosotros
- Pagina de Noticias
- Pagina de Servicios
- Pagina de Contactos

**Página inicio:** Esta página posee un resumen de los artículos o contenidos más importantes del sitio, en la pantalla de inicio se presenta un resumen del artículo y un enlace que lleva al artículo, posee además una sección dinámica para presentar contenidos de forma dinámica, esto es conocido como slider.

**Página nosotros:** En esta sección se presentan artículos referentes a la entidad a la que pertenece el sitio.

**Página noticias:** Esta sección contiene una serie de artículos de noticias.

**Página servicios:** En esta sección se presentan artículos relacionados a los servicios prestados por la entidad.

**Página contactos:** Una página con información de contacto, dirección de la entidad a la que pertenece el sitio y un formulario web para que el cliente pueda hacer llegar un mensaje a la entidad.

Analizando las páginas se definen tres tipos de páginas.

- Página de Inicio
- Presentación de contenidos
- Formulario de contactos

La página de inicio presenta un diseño diferente al de las demás páginas del sitio, esta página establece una base para la creación de las siguientes, la estructura de la página de inicio es:

- Un menú de las páginas de sitio
- Un slider para presentar novedades
- Una fila con los tres artículos más importantes
- Un pie de página

Las páginas de noticias, nosotros y servicios tienen una estructura similar las tres presentan artículos de forma ordenada, un análisis de los elementos.

- Un menú de las páginas del sitio
- Un listado de artículos
- Un pie de página

La página de contactos posee:

- Un menú de las páginas del sitio
- Un artículo de presentación

- Un formulario web
- Un pie de página

Luego del análisis de los componentes de las páginas se definen tres tipos de páginas, a excepción de la página de inicio, se puede agregar a las páginas un menú lateral con un listado de los 10 artículos más leídos de esa página.

### **5.3.1. GESTIÓN DE PANTALLAS**

De los tres tipos de página se puede obtener los elementos que forman la página web y aislarlos en ficheros separados para luego armarlos como si se tratara de un rompecabezas definamos los elementos presentes que forman parte de una página.

- Artículos inicio
- cabecera
- slider
- formulario de contactos
- menú
- menú lateral
- modal
- paginación
- pie
- presentación

los artículos de inicio tienen una presentación diferente, porque el diseño es diferente.

La ventaja de dividir las plantillas es que se centraliza el código escribiéndolo una sola vez es la aplicación quien lo repite en las páginas, si se desea modificar alguna parte del sistema de plantillas basta con editar el archivo deseado y el resultado es igual para todas las páginas.

**Artículos:** los artículos son la forma de mostrar información al usuario, para hacer esta experiencia más rica mostrados el texto en una ventana emergente a la que llamaremos pantalla de presentación de contenidos, los artículos se presentan empotrados en la página en ventanas emergentes tipo modal.

En el anterior subcapítulo dividimos las páginas en secciones de código será la sección modal la encargada de presentar los artículos, los artículos están formados por tres componentes:

- Un título
- Un elemento multimedia como imágenes o vídeos
- El cuerpo de artículo

Los artículos se almacenan en la base de datos la forma de direccionarlos es a través de su ID el cual se usa en la URL como parámetro de una función.

### 5.3.2. GESTIÓN DE URL'S

La gestión de URL's es una de las partes más importantes del sistema, estas son las encargadas de ubicar un contenido en el sitio, existe una relación de uno a uno entre un contenido y una URL, cuando uno de los dos no existe el CMS retorna un error 404 de página no encontrada.

De preferencia las URL's deben ser fáciles de recordar, escribir y leer esto ayuda a que los buscadores encuentren y den preferencia a nuestros contenidos por sobre los demás, las URL gestionan los siguientes elementos del sitio.

- Hojas de Estilo
- Ficheros de código PHP
- Ficheros de código JavaScript
- Ruta de imágenes
- Dirección de una página
- Nombre del controlador

- Parámetros de funciones

cada uno de esos elementos posee una ruta única, se debe establecer un estándar para su manejo.

Todos los componentes mencionados estarán en un mismo servidor, por lo que la URL base será el nombre del dominio <http://midominio.com/> esta indica en donde están los elementos del sitio, la estructura permite ubicar los elementos en directorios separados para una mejor organización se usa un directorio para el código, otro para las imágenes, un directorio para las hojas de estilo y uno para los ficheros de código JavaScript, de ser necesario se creará una estructura interna de directorios dentro de los ya mencionados.

### 5.3.3. PAGINAS CMS

La URL de una página depende del nombre del controlador, por esta razón el nombre del controlador debe ser óptimo para ser usado en una URI, las páginas no están almacenadas en la base de datos solo su contenido.

<http://misitio.com/mipagina/>

### 5.3.4. ARTÍCULOS CMS

Las URL's a los artículos depende de su ubicación en la tabla, se toma como base el id del mismo junto con el nombre de una función, este método recibe el Id del articulo y lo solicita al modelo.

<http://misitio.com/mipagina/articulo/43>

Lo que hace esta URL es llamar al método “**articulo**” de la clase “**mipagina**” y le pasa el número “**43**” como parámetro, el parámetro es el id del artículo.

### 5.3.5. IMÁGENES CMS

EL directorio de imágenes, las imágenes en si se guardan en un directorio del servidor pero las rutas se la almacena en una columna de la tabla que contienen los artículos.

`http://misitio/img/sitio/miimagen.extensión`

Al ser un sistema con un ambiente doble tenemos un subdirectorio para almacenar las imágenes de los artículos, esto es porque se tiene otro subdirectorio para las imágenes que forman el diseño del sitio y otras más para las páginas de administrador.

Una vez tomada la decisión para el manejo de los elementos del sitio veamos cómo se va a organizar todo, desde la ubicación de los ficheros hasta los enlaces a dichas imágenes.

```

├── appweb
│   ├── controllers
│   │   └── inicio.php
├── css
│   ├── bootstrap.css
│   └── slider.css
├── ico
│   └── favicon.ico
├── img
│   ├── article
│   │   ├── art1.png
│   │   └── art2.png
│   ├── portada
│   │   ├── img1.png
│   │   └── img2.png
│   ├── sitio
│   │   ├── slide1.jpg
│   │   └── slide2.jpg
├── js
│   ├── bootstrap.js
│   ├── jquery.js
│   └── vendor
│       ├── jquery-1.8.2.min.js
│       └── modernizr-2.6.2.min.js

```



### 5.3.6. GESTIÓN DE BASE DE DATOS CMS

La gestión de bases de datos está dada por el modelo dentro de la estructura de la aplicación, es el modelo quien interactúa directamente con la base de datos, este elemento está programado en el lenguaje PHP usando funcionalidades del Framework CodeIgniter.

El modelo implementa las siguientes funcionalidades.

- Obtener registros: Obtiene los registros de una tabla de preferencia este método debe aceptar condiciones.
- Obtener registro: Obtiene un único registro de la tabla, es útil en el caso de obtener un artículo.
- Eliminar registro: Elimina un registro en la tabla.
- Actualizar registro: Actualiza un registro.
- Contar registros: Cuenta el número de filas que tiene una tabla.
- Contar columnas: Cuenta el número de columnas que tiene una tabla.
- Listar columnas: Lista los nombres de las columnas de una tabla.
- Último Error: Retorna el contenido del último error devuelto por el motor de bases de datos.
- Última consulta SQL: Retorna el texto de la última consulta al servidor.
- Último ID ingresado a la base de datos: Retorna el valor del último identificador ingresado en la base de datos, sin importar la tabla.
- Ejecutar consulta SQL: Función para ejecutar una sentencia específica enviada por un controlador.

### 5.3.7. GESTIÓN DE ARTICULOS POR PÁGINAS (PAGINACIÓN)

Cuando los artículos de un sitio son tantos que tomaría demasiado tiempo mostrarlos a todos o simplemente solo se desea mostrar los últimos, existe una herramienta llamada paginación, esta divide los contenidos de una base de datos

en fragmentos, para optimizar tiempos de respuesta.

Para hacer paginación se usa una consulta SQL y un archivo PHP que las maneja, para hacer la paginación se usará la clase “**pagination**” de CodeIgniter

### 5.3.8. ADMINISTRACIÓN DE USUARIOS

Esta aplicación es capaz de interactuar con dos tipos diferentes de usuarios.

El usuario normal que hace peticiones HTTP al servidor para obtener contenidos y el Administrador usuario que tiene acceso a funcionalidades restringidas, identificarlo se conoce como administrar usuarios, esta técnica permite el acceso a determinada sección del sistema a determinados entes.

Administrar es un término un poco grande lo que en realidad se hace es identificar los tipos de usuario y permitirles el acceso a determinados elementos del sistema, para identificar al usuario administrador se dispone un nombre de usuario y una contraseña y el usuario normal no necesita de ninguna credencial para acceder al sistema, para validar las credenciales del usuario administrador se va a implementar un formulario web que envía los datos a un controlador y este los verifica iniciando una sesión que se inactiva cuando el usuario abandona el sitio.

### 5.3.9. GESTIÓN DE MENÚS

Los menús son muy importantes ya que son la puerta a los contenidos, es importante administrarlos bien, sabemos cómo están estructuradas las URL's, lo que se gestiona en los menús:

#### 5.3.9.1. El foco de los menús

Esto es muy útil para avisar al usuario en dónde se encuentra, se usa una clase en la etiqueta de la opción del menú que resalta el texto del menú diferenciándolo de los demás.

#### 5.3.9.2. Barra de menú lateral

Dependiendo de la página en la que se encuentra, la barra lateral muestra los artículos leídos de la página, para obtener este listado se dispone de una tabla tipo

vista que tiene un listado de los artículos ordenados por el número de visitas, se toma la lista y se crea un menú.

## **5.4. CREACIÓN DEL MODELO DE BASE DE DATOS**

El modelo de base de datos responde a los requerimientos del sistema al manejo de sus componentes y la descripción de los mismos.

### **5.4.1. CREANDO EL PRIMER MODELO**

Para empezar a definir la estructura del primer modelo, hay que revisar los componentes encontrados hasta el momento a estos componentes se los conoce como entidades:

- Una entidad para registrar las páginas
- Una entidad para registrar los artículos
- Una entidad para registrar los formularios
- Una entidad para registrar usuarios
- Una entidad para registrar los artículos más leídos

Se entiende que se necesitan 5 entidades para que el sitio funcione, para nombrar las entidades se usarán nombres en inglés, la razón de esto es que es uno de los requerimientos de la programación, a continuación un análisis completo de los atributos de las entidades listadas arriba.

### **5.4.2. DICCIONARIO DE DATOS**

#### **5.4.2.1. Entidad page**

Esta entidad contiene los registros de las páginas del sitio, controla que no se repitan las páginas.

### Atributos entidad page:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id_page	smallint(5)	No	Nada	Identificador de la página
title	varchar(500)	No	Nada	Título de la página
controller	varchar(50)	No	Nada	Se acorta el título de la página para controlar que no se repitan las páginas
keywords	varchar(500)	No	Nada	Keywords de la página
create_date	datetime	No	Nada	Fecha de creación
last_update	timestamp	No	CURRENT_TIMESTAMP	Fecha última edición

### Índices entidad page:

Tipo clave	Único	Columna	Nulo	Comentario
Primaria	Sí	id_page	No	Este índice ayuda a recuperar una página por su ID
Única	Sí	controller	No	Este es un índice que registra el nombre de la página conforme el nombre del controlador, ayuda que una página no se duplique.
Única	Sí	title	No	Este es un índice que registra el título de la página conforme ayuda a darle un título a las páginas y controla que este no se repita.

Con los índices aseguramos que los registros de la tabla “**page**” sean únicos.

#### 5.4.2.2. Entidad article

Almacena los artículos del sitio, recordamos los componentes del artículo, esta entidad debe controlar que los artículos sean únicos.

##### Atributos entidad article:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id_article	smallint(5)	No	Nada	Identificador del artículo
id_page	smallint(5)	No	Nada	Identificador de la tabla a la que pertenece, esta es la forma de enlazarlos a la página.
title	varchar(500)	No	Nada	Título del artículo
content	mediumtext	No	Nada	Contenido del artículo
counter	smallint(5)	No	Nada	Almacena el número de veces que un artículo ha sido visto, cada vez que un artículo es solicitado a la base de datos se edita este campo aumentando en uno su valor
visible	smallint(5)	No	Nada	booleano que indica si un artículo es visible o no
create_date	datetime	No	Nada	Fecha de creación

last_update	timestamp	No	CURRENT_TIMESTAMP	Fecha de la última modificación
publish_down	datetime	No	0000-00-00 00:00:00	Fecha en el que se le da de baja, se entiende por dar de baja el hacer invisible un artículo

#### Índices entidad article:

Tipo de clave	Único	Columna	Nulo	Comentario
Primaria	Sí	id_article	No	Esta controla el identificador del artículo
Única	Sí	title	No	Se comprueba que el título no se repita, esta es una forma de evitar artículos duplicados.
Foránea	No	id_page	No	Esta columna sirve de enlace con la columna página, se usa para saber a qué página pertenece un artículo, la razón por la que no es único está dada porque una página puede tener varios artículos.

Los índices de la entidad **“article”** sirven para controlar la integridad de la información contenida en la tabla y para relacionar los artículos con las páginas que el sitio posea, estas páginas están registradas en la entidad **“page”**.

#### 5.4.2.3. Entidad form

Esta entidad almacena el contenido enviado por los usuarios al administrador de ventas del sitio.

**Atributos entidad form:**

Columna	Tipo	Nulo	Predeterminado	Comentarios
id_form	smallint(5)	No	NULL	Identificador del registro
names	varchar(100)	Sí	NULL	Nombres del usuario
email	varchar(100)	Sí	NULL	Email del usuario
company	varchar(50)	Sí	NULL	Entidad a la pertenece
phone	varchar(25)	Sí	NULL	Número de teléfono
asunto	varchar(100)	Sí	NULL	Asunto del mensaje
description	varchar(1000)	Sí	NULL	Cuerpo del mensaje
browser	varchar(250)	Sí	NULL	Navegador usado por el usuario
ip	varchar(20)	Sí	NULL	Ip del usuario
country	varchar(50)	Sí	NULL	País desde el que escribe el usuario
create_date	datetime	Sí	NULL	Fecha de envío
last_update	timestamp	No	CURRENT_TIMESTAMP	Ultima modificación del registro normalmente este campo debe coincidir con el de creación si hay alguna diferencia entonces el registro fue modificado por terceros, este campo funciona como un ente de control.

### Índices entidad form:

Tipo de clave	Único	Columna	Nulo	Comentario
Primaria	Sí	id_form	No	Esta controla el identificador del formulario

#### 5.4.2.4. Entidad user

Entidad que registra a los usuarios que tienen privilegios en el sistema.

#### Atributos entidad user:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id_usuario	smallint(5)	No	Nada	Identificador del usuario
usuario	varchar(40)	No	Nada	Nombre del usuario
pass	varchar(200)	No	Nada	Contraseña del usuario
last_failure	datetime	Sí	NULL	Marca de tiempo que registra la última vez que el usuario no pudo ingresar al sistema
failure_count	smallint(5)	Sí	NULL	Cuenta el número de veces que el usuario falla al entrar al sistema
last_login	datetime	Sí	NULL	Última vez que ingresó
create	datetime	Sí	NULL	Fecha de creación
last_update	timestamp	No	CURRENT_TIMESTAMP	Última actualización



**Índices entidad User:**

Tipo de clave	Único	Columna	Nulo	Comentario
Primaria	Sí	id_usuario	No	Esta controla el identificador del usuario
Única	Sí	usuario	No	Con este índice se controla que no existan usuarios con los mismos nombres.

**5.5. DEFINICIÓN DE FUNCIONES Y VISTAS**

Entidad de artículos más leídos (ratings):

Basándonos en la entidad creada para la gestión de artículos creamos un listado de los artículos más leídos y con ellos elaboramos una vista, para determinar los artículos más leídos nos basamos en la columna “**counter**” ordenando los registros de forma descendente, para recuperar luego los registros se establece una condición para que retorne solo los artículos de una página.

**Entidad tipo vista para página:**

Columna	Tipo	Nulo	Predeterminado	Comentarios
id_article	smallint(5)	No	0	Identificador del artículo
id_page	smallint(5)	No	Nada	Identificador de página
controller	varchar(50)	Sí	NULL	Nombre del controlador de la página
title	varchar(500)	No	Nada	Título de la página
counter	smallint(5)	No	Nada	Contador de lecturas del artículo

Esta entidad es obtenida de un recorte a la entidad artículo por esta razón las

columnas y los índices son los mismos, los datos mostrados en la tabla son suficientes para crear el menú, y direccionarlo al artículo completo.

### 5.5.1. VISTA PARA PRESENTAR UN LISTADO DE ARTÍCULOS

Se necesita de una entidad que ayude a crear listados de los artículos de forma fácil y rápida, este listado se lo suele conocer como tablón, este tablón debe poseer un listado de los artículos, un resumen del mismo y el enlace que lleva al artículo.

#### Entidad tipo vista para tablón:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id_article	smallint(5)	No	0	Identificador del artículo
id_page	smallint(5)	No	Nada	Identificador de la página
controller	varchar(50)	Sí	NULL	Nombre del controlador de la página
title	varchar(500)	No		Título del artículo
content	varchar(243)	No		Contenido del articulo
counter	smallint(5)	No		Contador del artículo
create_date	datetime	No		Fecha de creación del artículo

Esta vista es un poco más complicada que la anterior ya que está basada en dos tablas, se toma el campo “controller” de la tabla “page” para poder construir la URL al artículo, recordamos que las URL's de CodeIgniter tienen el nombre del controlador, el nombre del método y los parámetros de este, para construir un enlace a un articulo desde esta vista es necesario conocer el nombre del controlador y este dato se encuentra en la tabla padre, esta acción es conocida

como producto cartesiano entre dos tablas, para hacerla se sirve de la sentencia JOIN de MySQL.

### **5.5.2. VISTA TABLÓN**

Con el afán de simplificar las consultas para elaborar el listado de artículos de una página se crea esta vista la cual contiene los campos principales para mostrar una introducción del sistema y poder construir un enlace que nos lleve a él, por defecto se organizan los registros descendente-mente por la columna fecha.

### **5.4.3. DIAGRAMA DE BASES DE DATOS**

El resultado de las anteriores entidades genera el siguiente diagrama.

A esto se lo conoce como diagrama de la base de datos, dos diagramas de datos ayudan a comprender mejor el funcionamiento y estructura de una base de datos.

### **5.4.4. SOMETIENDO A PRUEBAS EL MODELO**

Una vez creado nuestro modelo es necesario someterlo a pruebas para saber si lo que definimos es lo que se necesita, cuando se está desarrollando el proyecto, este se debe ajustar al modelo veamos cómo se comprueba que un modelo sirva.

El modelo es muy sencillo, la única complejidad es la relación que existe entre páginas y artículos, se dice que pueden existir muchas páginas y que a esta le debe corresponder muchos artículos, no puede existir un artículo sin una relación directa con una página, este modelo cumple con ese requerimiento.

En cuanto a las entidades sueltas tenemos la de formularios y la de sesiones, lo único que hacen estas entidades es ser un punto de control en el sistema, en el caso de la entidad de formularios almacena una copia del Email enviado por un cliente a la empresa o entidad dueña de sitio, posee un punto de control que es la fecha de actualización esta fecha y la de creación deben ser iguales, caso contrario alguien cambio el registro.

## **CAPÍTULO 6 IMPLEMENTACIÓN DE CMS.**

### **6.1. ANÁLISIS DE INTERFAZ DE USUARIO**

Las interfaces de usuario y usuario administrador son quizás los elementos más importantes de una aplicación de cualquier ámbito, las interfaces de usuario deben ser sencillas de usar, fáciles de entender y deben tener el contenido estructurado de forma que sea fácil acceder al él, las interfaces en nuestras aplicaciones están contruidos con tecnologías web HTML y JavaScript ambas estudiadas en los primeros capítulos del presente documento.

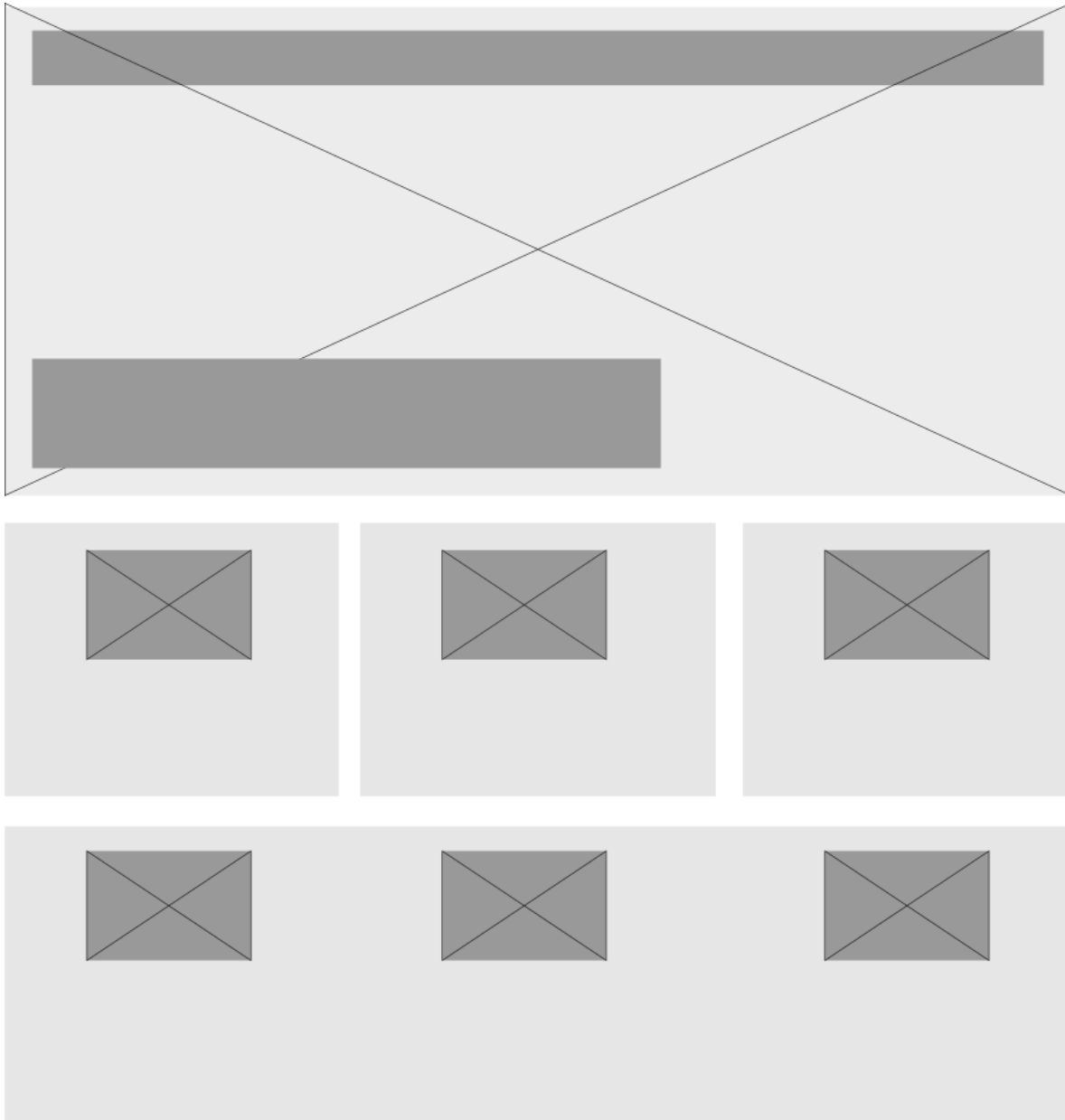
#### **6.1.1. INTERFAZ DE USUARIO**

La interfaz de usuario debe ser lo más sencilla posible, y de preferencia debe funcionar de igual manera para todos los navegadores y sistemas operativos, la interfaz de usuario debe ser amigable y cargarse en el menor tiempo posible.

Antes de crear la interfaz de usuario se procede a crear un diseño sobre papel esto permite al diseñador aprovechar los espacios y la distribución de los contenidos.

### 6.1.2. DISEÑO DE INTERFAZ DE INICIO

La pantalla de inicio es muy importante y debe contener información importante y resumida, para el inicio se puede establecer una plantilla lo más gráfica posible, este es el esquema de la plantilla:



### 6.1.3. DISEÑO DE INTERFAZ DE CONTENIDOS

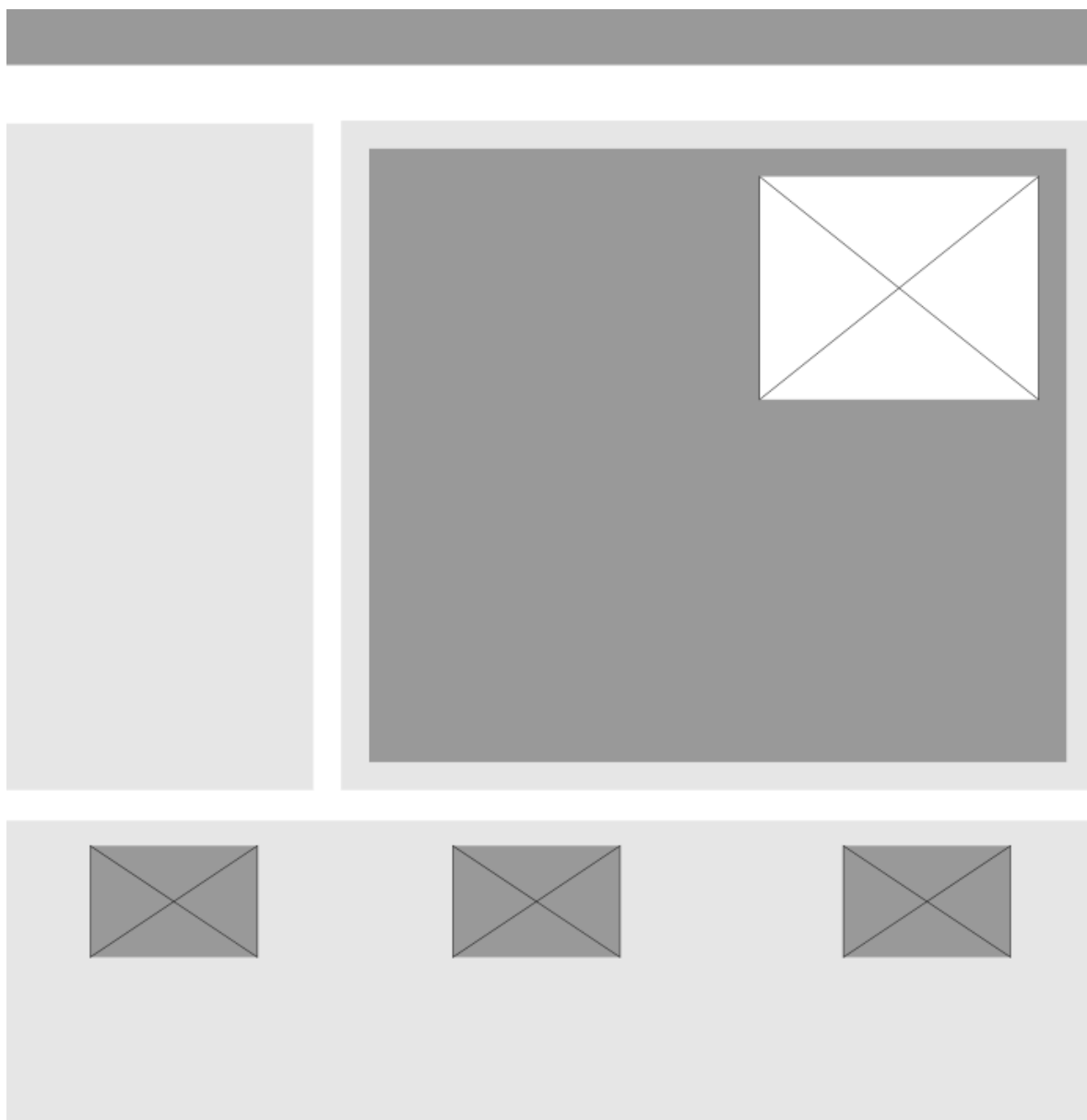
La interfaz de contenidos tiene que ser lo más sencilla posible, para presentar los artículos se lo hace con la siguiente interfaz:



La página de contenidos, presenta los artículos de forma paginada para acelerar la carga de las páginas, se presenta cinco artículos por página.

#### 6.1.4. DISEÑO DE INTERFAZ DE CONTACTOS

La interfaz de contactos posee un formulario web que permite al usuario ingresar información para luego enviarla al administrador del sitio:



Esta es la última plantilla de sitio, la estructura es similar a la de presentación de contenidos, esto para mantener una uniformidad en el diseño.

### 6.1.5. INTERFAZ DE ADMINISTRADOR

Al objetivo de las interfaces del administrador es permitir al usuario administrador acceder a todas las características de administración del sitio, se crea una sola plantilla para todas las páginas del administrador, ya que las pantallas tienen la misma tarea asignada (editar artículos de la página a la que representa), modelo





### **6.1.6. DISEÑO INTERFAZ DE LOGIN**

Esta interfaz es muy simple y lo único que posee es un formulario en el que el usuario administrador ingresa sus credenciales y se inicializa una sesión.



### **6.1.7. DEFINICIÓN ESTRUCTURA DE VISTAS**

Luego de definir las pantallas que se van a implementar en el sistema, se identifican patrones en las plantillas, por ejemplo en los pies de página, el listado de los contenidos, los menús, la estructura de las vistas del sitio se distribuye de la siguiente forma:

- Alertas
- Artículos del home
- Carrusel
- Pie de página
- Formulario
- Cabecera

- Menú lateral
- Menú
- Modal
- Presentación
- Recibido

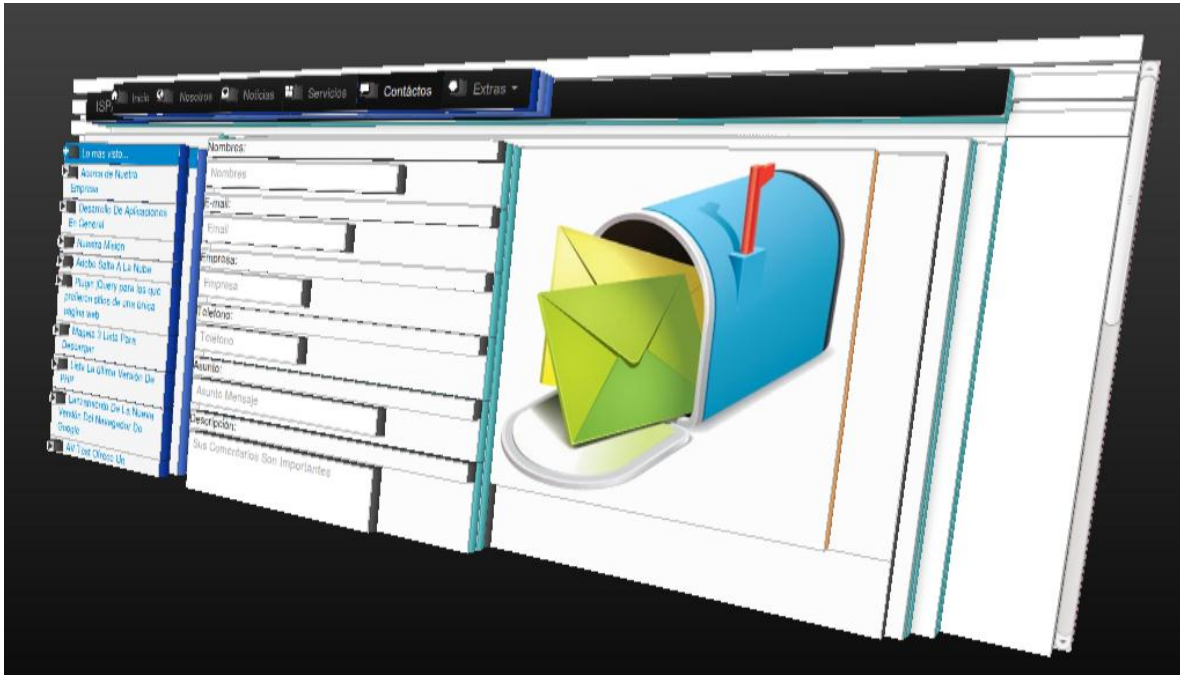
Cuando las vistas se dividen en partes alojadas en archivos se puede armar una página como si de un rompecabezas se tratara, una de las ventajas es la facilidad de cambiar el diseño de forma muy sencilla.

## 6.2. VISTAS TERMINADAS

### 6.2.1. PRESENTACIÓN DE CONTENIDOS



### 6.2.2. PÁGINA DE INICIO



### 6.2.3. ESTRUCTURA VISTAS ADMINISTRADOR

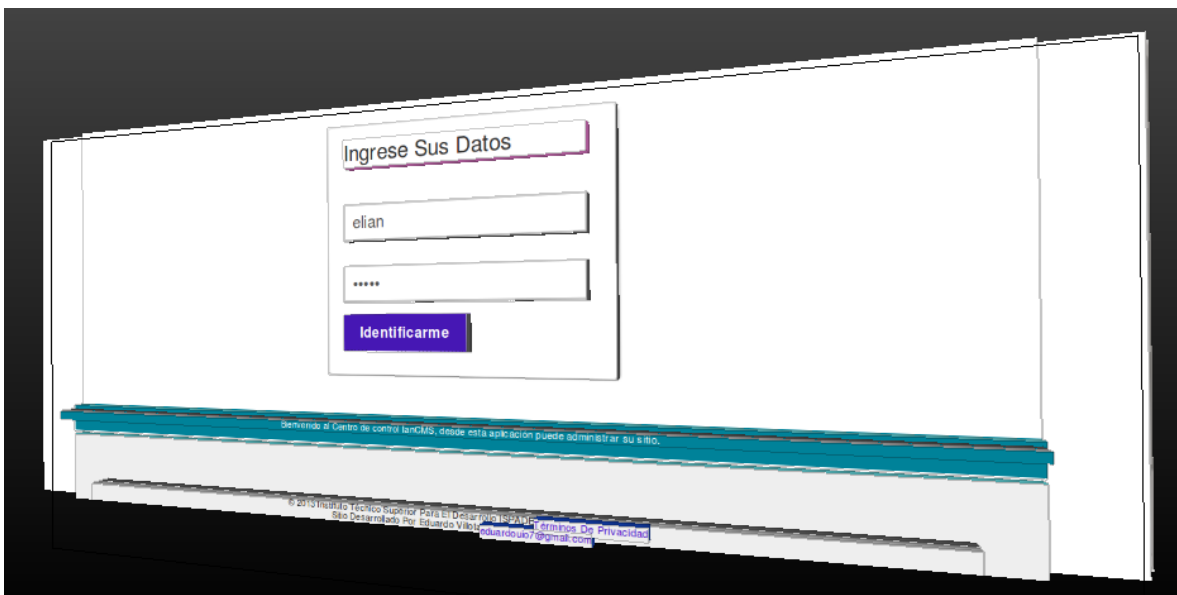
- Alerta
- Confirmar
- Pie de página
- Formularios
- Cabecera
- Información
- Login
- Menú
- Presentación
- Redirección
- Resultado
- Búsqueda
- Acción Completa

- Tabla

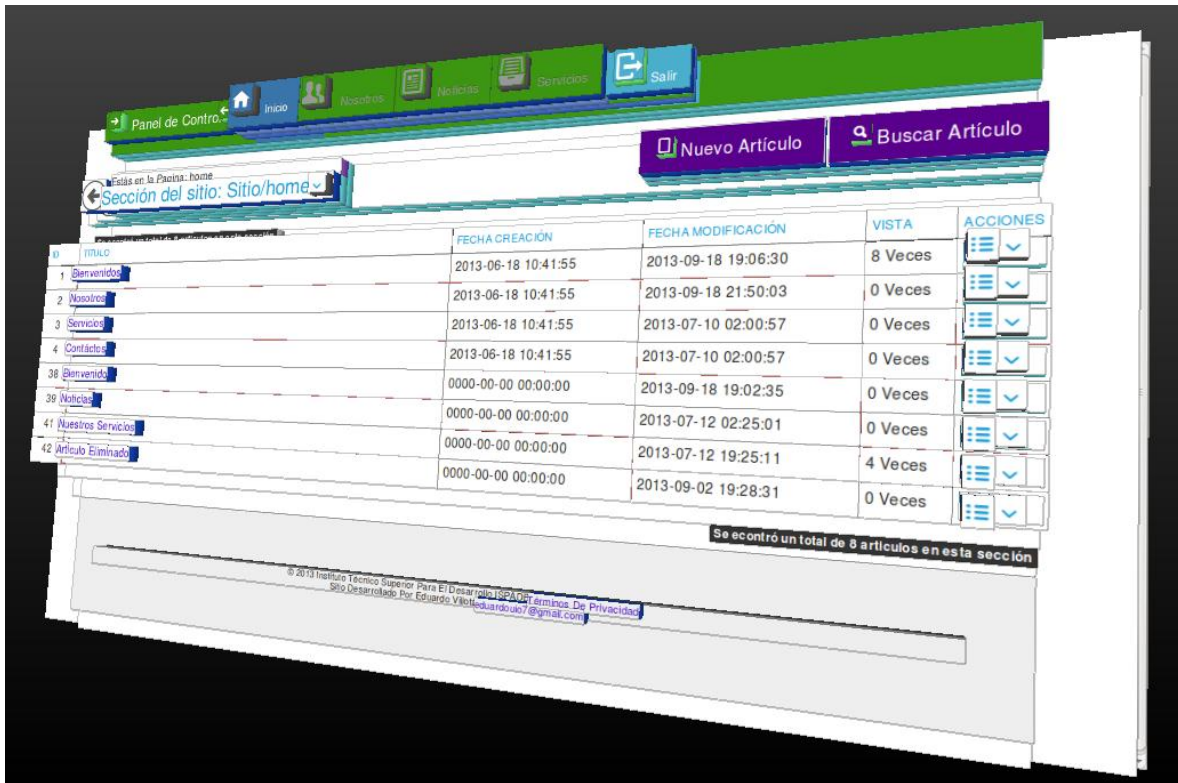
Se aplica el mismo principio que se aplicó al sitio, las vistas son fragmentadas, para facilitar su uso y mantenimiento, la vista de redirección es un artificio JavaScript para provocar una redirección en el navegador del cliente.

## 6.2.4. VISTA ADMINISTRADOR

### 6.2.4.1. Vista Login



### 6.2.4.2. Presentación de contenidos administrador



Cada una de las imágenes mostradas, dejan visualizar la forma en la que está estructurada los contenidos, cada una de las secciones forman parte de una vista, la unión de las vistas dan como resultado una plantilla.

## 6.3. ANÁLISIS DISEÑO LÓGICO DEL SISTEMA

El análisis lógico del sistema comprende, entender el funcionamiento en el papel, en esta fase se realizan diagramas para especificar y comprender mejor el funcionamiento, este análisis permite definir los componentes internos del sistema la metodología con la que se desarrolla este modelo es la de tres capas modelo-vista-controlador.

### 6.3.1. DIAGRAMAS DE MODELO UML

Esta es una técnica usada en la actualidad para crear aplicaciones web, la rama de la informática que se encarga de su estudio es la ingeniería del software, los

diagramas están diseñados para que sean comprendidos por cualquier persona que los estudie, se usan para describir el funcionamiento del sistema al usuario final, y para el desarrollo del mismo.

UML es un lenguaje de modelo, es decir es principalmente gráfico a estos gráficos se los llama diagramas, no es necesario conocerlos y aplicarlos todos para el desarrollo de una aplicación.

Existen herramientas específicas para trabajar con UML, el objetivo de estas herramientas es evitar que el desarrollador escriba código esto aún no se ha logrado por completo pero son herramientas indispensables para el correcto desarrollo de una aplicación.

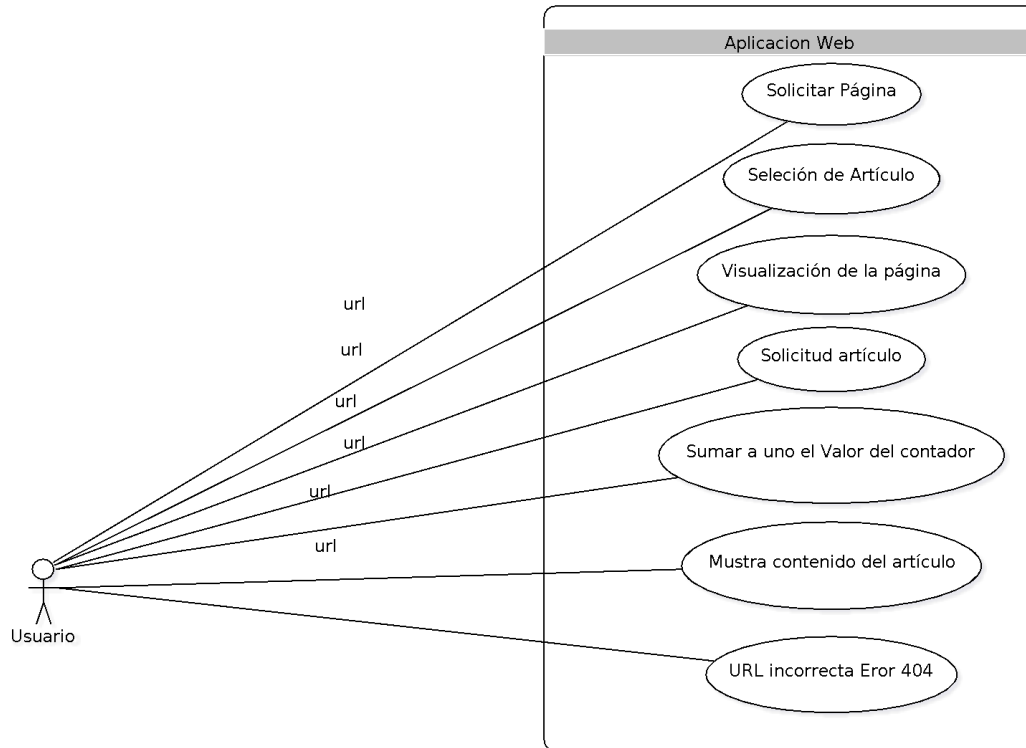
#### **6.3.1.1. Casos de uso aplicación web**

En la última década se ha masificado el uso de diagramas para la construcción de sistemas no solo digitales sino físicos y conceptuales, los diagramas de caso de uso no son la excepción, su objetivo principal es describir una interacción del sistema de manera muy sencilla, para que los lectores lo puedan interpretar de la misma forma, en ocasiones las palabras son ambiguas para describir la solución a un problema, esto debido a que un texto se puede interpretar de varias maneras un ejemplo claro son las leyes en el ámbito judicial.

Los casos de uso describen el comportamiento del sistema frente a otras entidades o usuarios, en el caso de lanCMS es un sistema gestor de contenidos creado en dos partes la parte del cliente y la parte del administrador, ambas partes funcionan de forma separada e independiente.

### 6.3.1.2. Caso de uso interacción del usuario con el sitio web

#### Descripción escenario interacciones sitio usuario:

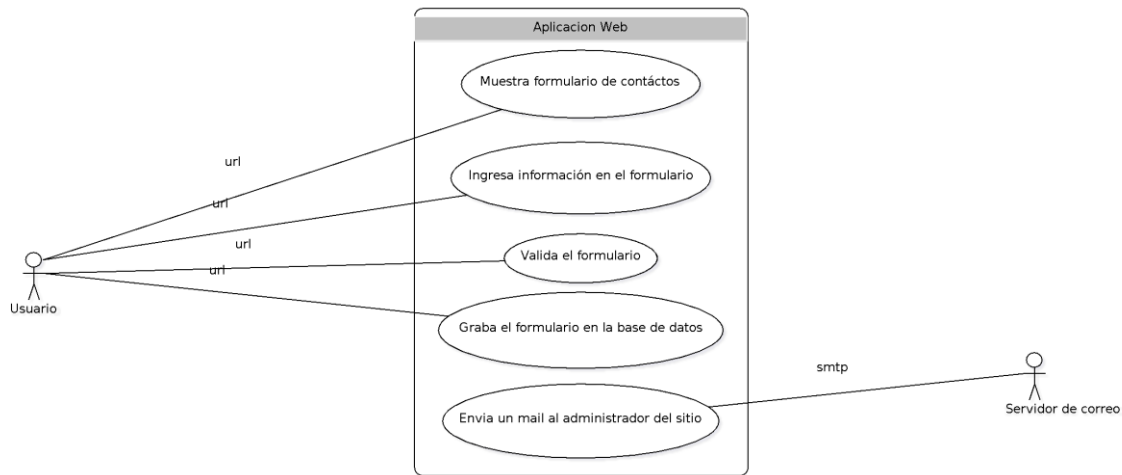


Descripción del escenario	
Quién lo comienza:	Usuario
Quien lo Finaliza:	Usuario
Excepciones:	Si el usuario solicita un recurso que no existe en el sitio, éste le responde con un mensaje 404
Descripción:	<p>Las interacciones del usuario con la página web son únicamente a través de url.</p> <ul style="list-style-type: none"> <li>• El usuario puede solicitar al servidor una página</li> <li>• Dentro de una página puede solicitar al servidor un artículo</li> <li>• Puede solicitar al servidor un recurso que éste no disponga, el servidor procesa esta petición, evita que el navegador dé el mensaje de error 404 en su lugar se</li> </ul>

	<p>envía una plantilla formateada que indica que el recurso no existe.</p> <p>Los datos de los contenidos del sitio están en una base de datos y las plantillas HTML se encuentran en un árbol de directorios en el sitio.</p>
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 6.3.1.3. Caso de uso interacción del usuario con el formulario de contactos

#### Descripción escenario formulario de contactos:



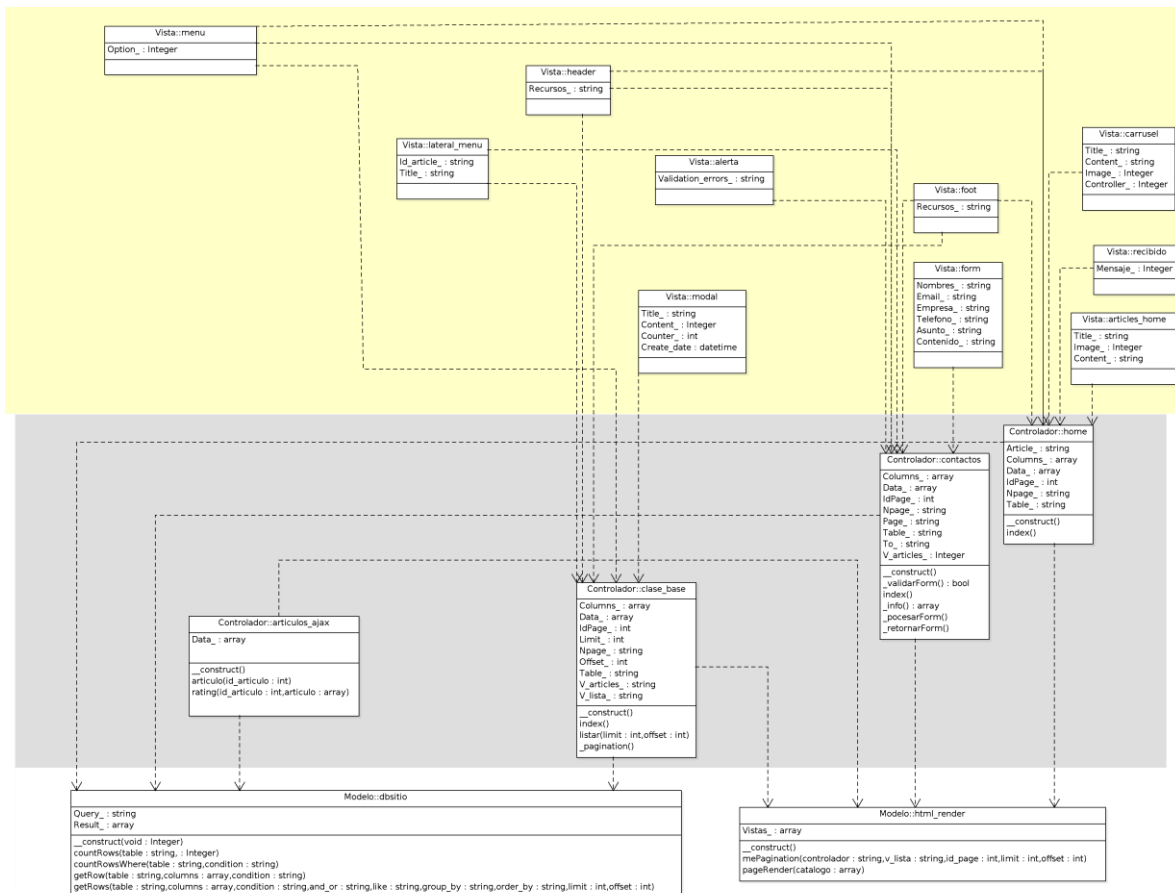
Descripción del escenario	
Quién lo comienza:	Usuario
Quien lo Finaliza:	Servidor de correo
Excepciones:	Si el usuario ingresa datos incorrectos, o no cumplen con los requerimientos del formulario, la información no se envía, si el usuario logra enviar la formación, esta se analiza en el servidor, si no cumple con las condiciones se retorna el formulario con la información.
Descripción:	Quando el usuario desea enviar una solicitud al administrador de sitio lo puede hacer únicamente por medio de un formulario web los pasos a seguir son:



- Solicitar el formulario
- Llenar el formulario
- Si el formulario no está correctamente completado se lo devuelve al usuario con los datos ingresados
- El formulario completo es empacado en un email y se lo envía al administrador

#### 6.3.1.4. Diagrama de clases sitio

Este es un diagrama en el que incluyen de forma gráfica todos los componentes del sistema, al ser esta una aplicación orientada a objetos los componentes de la misma son objetos, luego se establece la relación entre ellos, la forma en la que un objeto forma un paquete y este a su vez sirve a otro paquete en la aplicación.



Se puede apreciar en el diagrama como está estructurada la aplicación y las dependencias que existen.

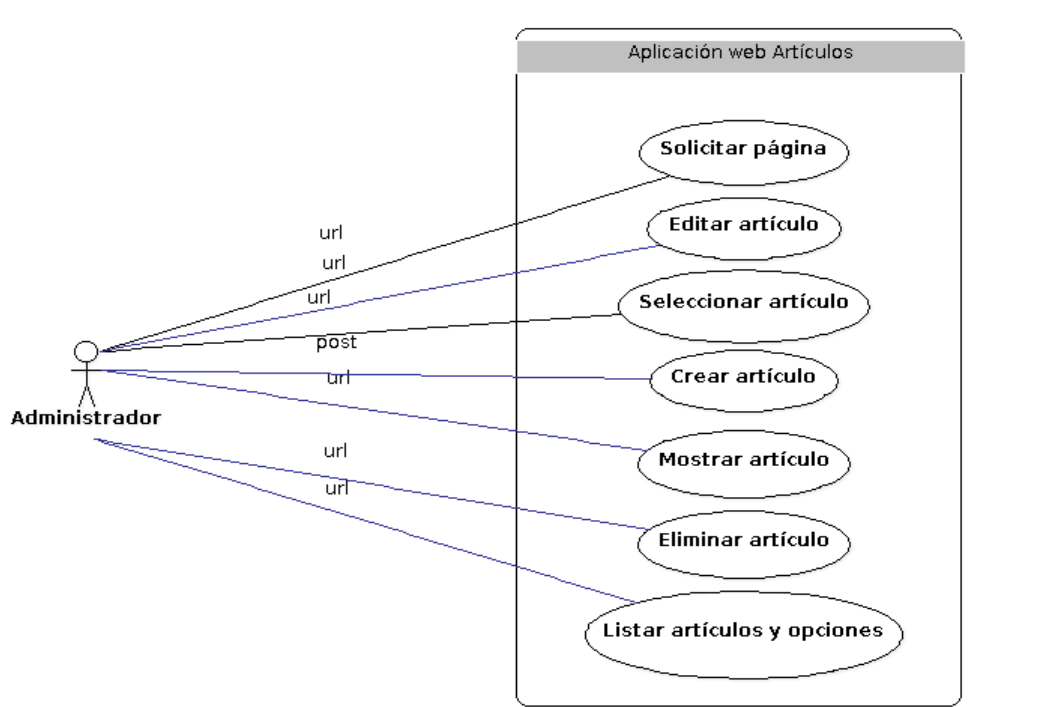
Esta aplicación está construida con el patrón de diseño MVC (modelo vista controlador) cada una de las capas está representada con un color diferente, siendo la blanca en modelo la gris el controlador y la amarilla las vistas, cada capa se representa como un paquete, y tienen dependencia hacia abajo, si una capa inferior falla todo lo que está para arriba fallará también.

#### 6.3.1.5. Casos de uso interacción del usuario con el sitio web administrador

El administrador posee casos de uso diferentes a los definidos para el sitio web, a continuación una revisión de los casos de uso encontrados.

Uso de aplicación:

#### Descripción escenario administrador web:

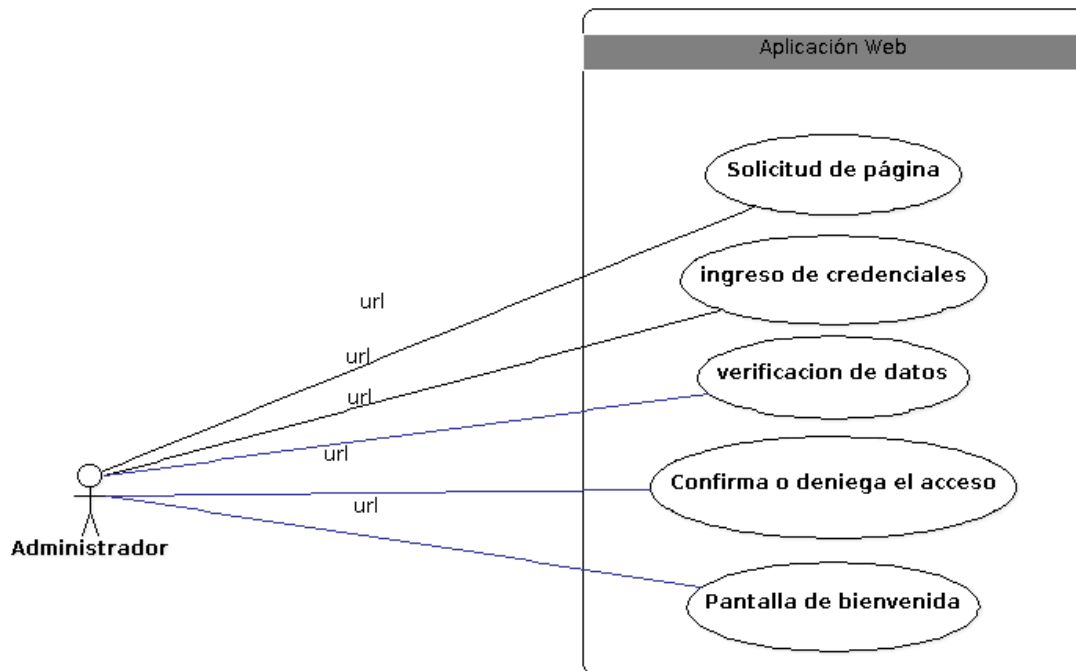


Descripción del escenario	
Quién lo comienza:	Administrador
Quien lo Finaliza:	Administrador
Excepciones:	La sesión caduca cuando el usuario cierra la ventana, si el

	recurso no existe se presenta una página con el error 404
Descripción:	<p>En este escenario se especifican las interacciones posibles entre el administrador del sistema y la aplicación, el usuario administrador está en capacidad de gestionar los artículos del sitio web, el administrador es un único ente en el sistema.</p> <p>Cada una de las páginas de la aplicación administrador corresponden a una página de la aplicación web, estas paginas están en capacidad de crear, eliminar y editar contenidos a los que se les llama artículos.</p>

#### 6.3.1.6. Caso de uso identificación de usuario

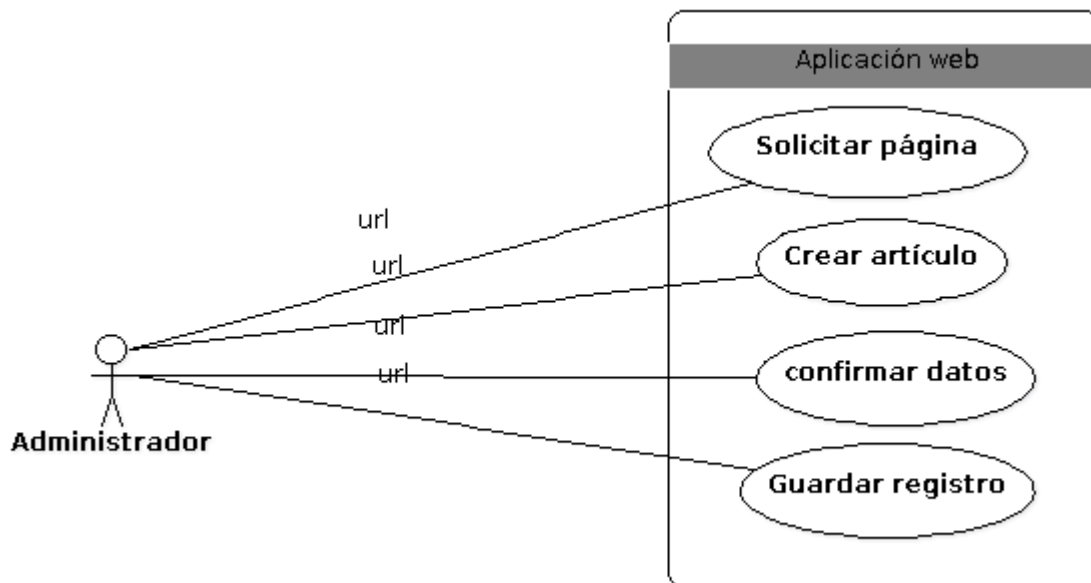
##### Descripción escenario identificación de usuario:



Descripción del escenario	
Quién lo comienza:	Administrador
Quien lo Finaliza:	Aplicación
Excepciones:	Cuando el usuario ingrese las credenciales incorrectas, el

	sistema retornada una vista con un mensaje de error, vuelve a mostrar un formulario HTML vacío.
Descripción:	En este escenario el usuario administrador ingresa sus credenciales al sistema, si las credenciales son las correctas el sistema inicializa una sesión creando una cookie con los datos del usuario y registrándolos en la base de datos.

#### 6.3.1.7. Caso de uso crear artículo



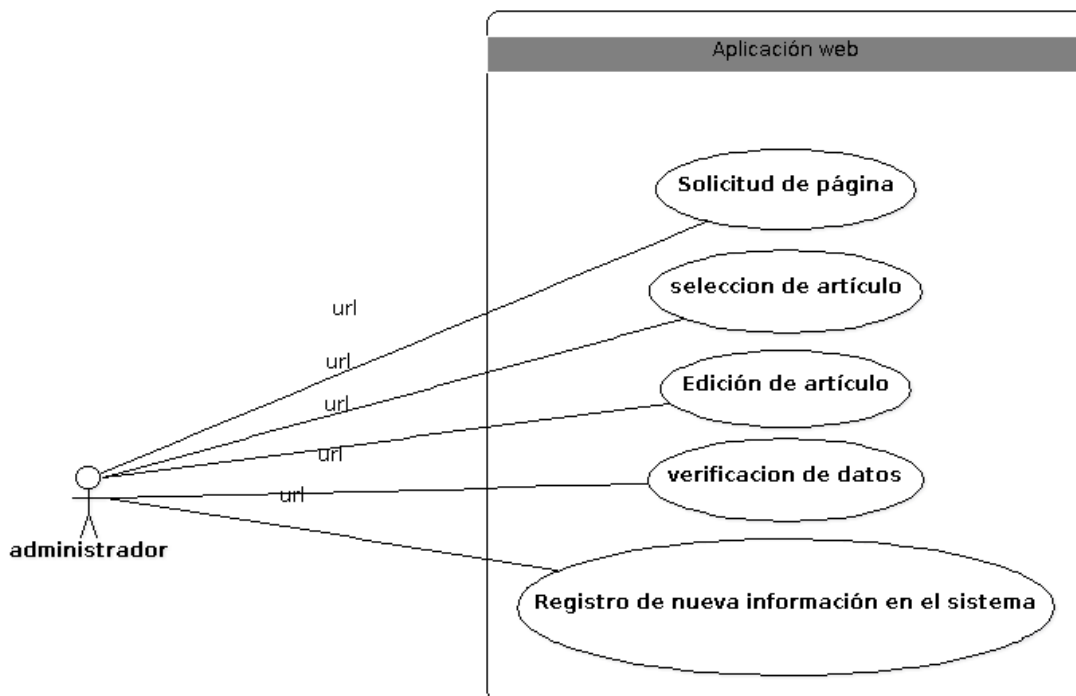
#### Descripción escenario creación de artículo:

Descripción del escenario	
Quién lo comienza:	Administrador
Quien lo Finaliza:	Aplicación
Excepciones:	La información ingresada incorrectamente es retornada al usuario con un mensaje de error para que éste los corrija, el proceso se repite si el usuario administrador no completa correctamente el formulario.

Descripción:	El usuario administrador puede crear nuevos contenidos en el sitio web, a estos contenidos se les da el nombre de artículos, cada artículo creado pertenece a la página en la que fue creado.
--------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### 6.3.1.8. Caso de uso editar artículo

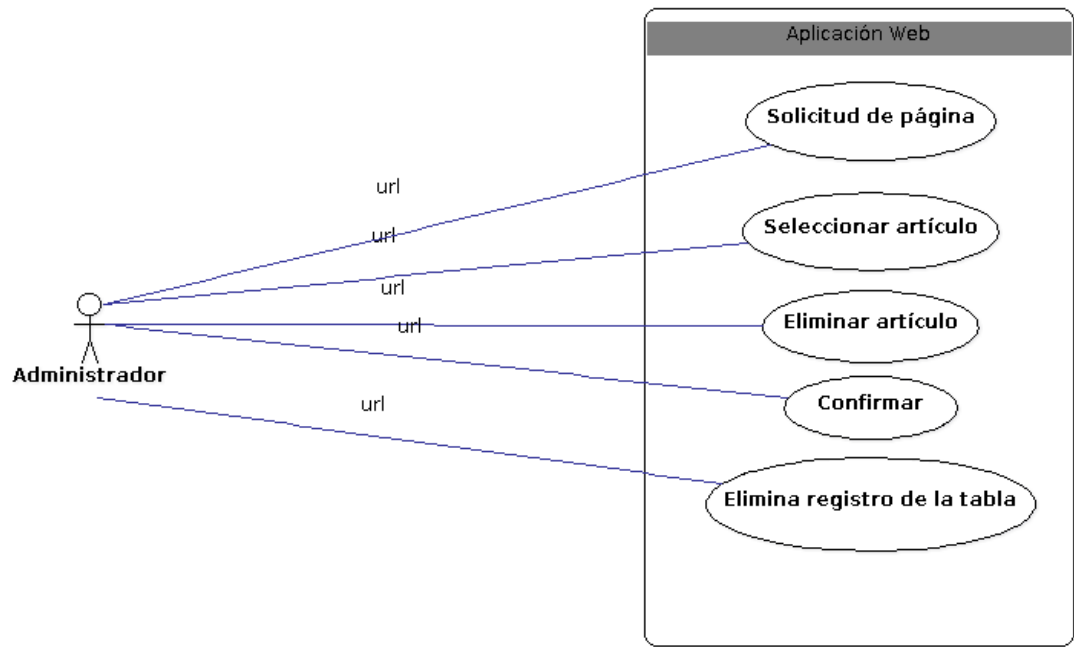
##### Descripción de escenario edición de artículo:



Descripción del escenario	
Quién lo comienza:	Administrador
Quien lo Finaliza:	Aplicación
Excepciones:	La información ingresada incorrectamente es retornada al usuario con un mensaje de error para que éste los corrija, el proceso se repite si el usuario administrador no completa correctamente el formulario.
Descripción:	El usuario administrador puede editar contenidos existentes en el sitio web, a estos contenidos se les da el nombre de

	artículos.
--	------------

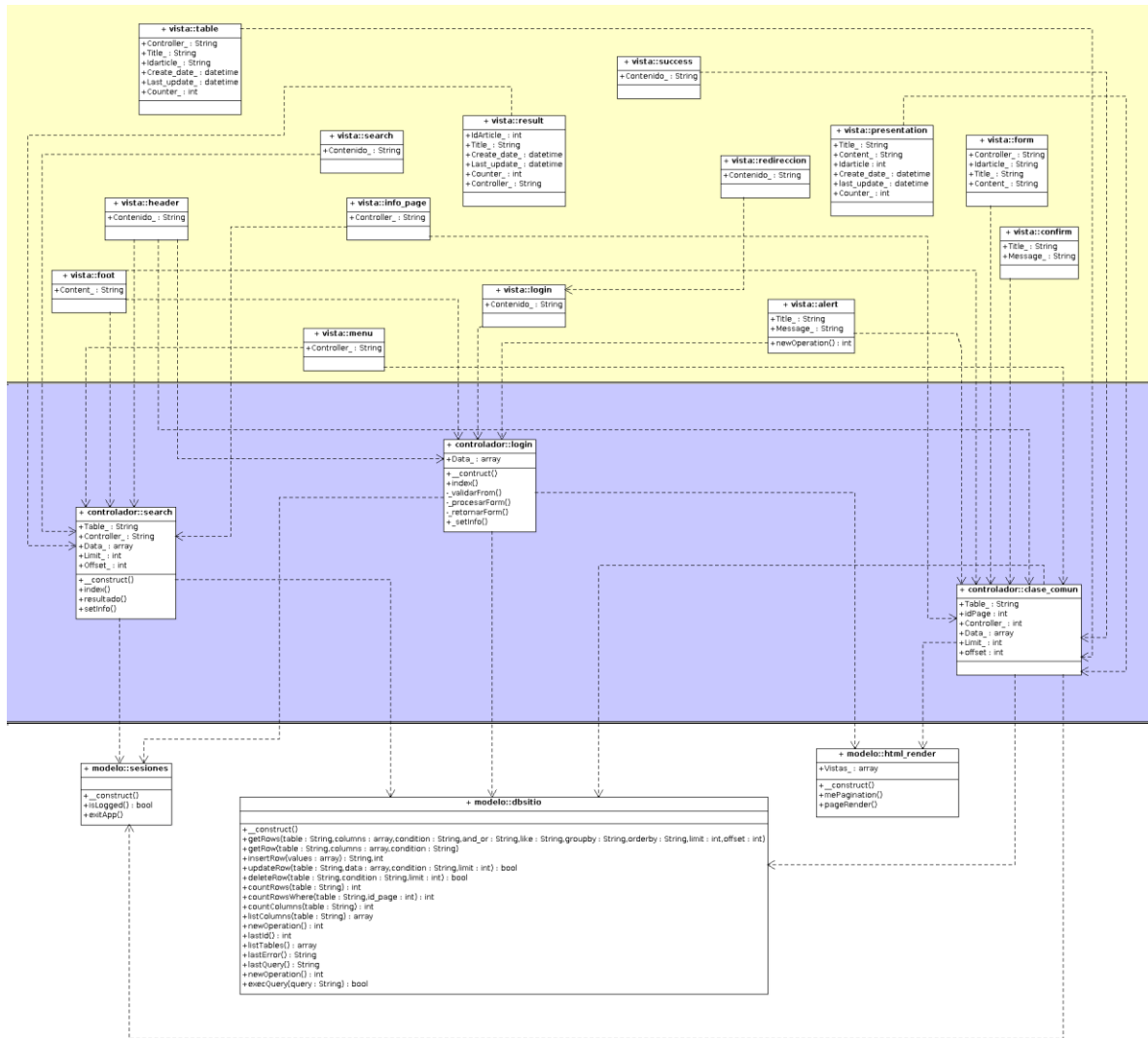
6.3.1.9. Caso de uso eliminar artículo



Descripción escenario eliminar artículo:

Descripción del escenario	
Quién lo comienza:	Administrador
Quien lo Finaliza:	Aplicación
Excepciones:	Si se trata de eliminar un contenido que no existe en el sitio, la aplicación re direcciona al usuario administrador al listado de los contenidos de la sección del sitio en el que se encuentre.
Descripción:	El usuario administrador puede eliminar contenidos existentes en el sitio web, a estos contenidos se les da el nombre de artículos.

### 6.3.1.10 diagrama de clases administrador



La aplicación de administrador esta creada con el patrón de diseño MCV(modelo-vista-controlador), en este diagrama se establece los objetos necesarios para el correcto funcionamiento del sistema, la sección de color blanco representa la capa modelo, la sección azul del gráfico representa la capa controlador del sistema y la sección amarilla del gráfico que representa las vistas del sitio.

### **6.3.2 PRUEBAS DEL SISTEMA**

Una vez terminada la aplicación se somete a pruebas antes de entrar en producción, gracias al diseño que esta presenta los cambios son muy pocos:

- Se elimina la gestión de imágenes, se usa una característica de Bootstrap para adjuntar imágenes en los artículos
- Se crea un modelo únicamente para manejar las sesiones, este modelo es el encargado de mantener la sesión del usuario administrador.
- Se cambia la url que adjunta un fichero PDF al sitio, haciendo que este recurso se abra en una pestaña nueva.



# ANEXOS

## 7.1. PUBLICACIÓN DEL SISTEMA

En este punto del proyecto estamos listos para poner el sistema en marcha, para publicar el sistema a producción se usará un servidor que administro, es servidor es propiedad de la empresa Liposerve Ecuador S.A. Cuyos dirigentes conocen de mi proyecto y cuento con su autorización, Listado con información necesaria para publicar el sistema.

- IP Servidor: 108.170.56.3
- Dominio: <http://liposerve.com>
- Subdominio: <http://ispade.liposerve.com>
- Motor DB: Mysql
- Usuario DB: liposerv\_ispade
- Contraseña DB: [.A(^=bqFUtn

### 7.1.2. CONFIGURACIONES INICIALES

Conocemos donde va a funcionar el sistema el nombre de todos los componentes antes de entrar al servidor vamos a preparar los ficheros.

#### **Archivos de Configuración Codelgniter**

Para que el sitio funcione de manera correcta hay que configurar algunos ficheros de Codelgniter en las dos aplicaciones con los datos citados anteriormente.

#### **Configuraciones Sitio Web**

Para que el sitio web funcione debemos decirle dónde está la base de datos, con qué usuario conectarse, también se le debe indicar las rutas a los ficheros de estilo y otros componentes, para esto es necesario modificar solo dos ficheros:

#### **ispade/sitio/appweb/config/config.php**

En este fichero se especifica la url del sitio.

```

/*
-----
Base Site URL
-----

URL to your CodeIgniter root. Typically this will be your base URL,
WITH a trailing slash:

    http://example.com/

If this is not set then CodeIgniter will guess the protocol, domain and
path to your installation.
*/
$config['base_url'] = 'http://ispade.liposerve.com';
/*

```

### ispade/sitio/appweb/config/database.php

```

47
48 $active_group = 'default';
49 $active_record = TRUE;
50
51 $db['default']['hostname'] = 'localhost';
52 $db['default']['username'] = 'liposerv_ispade';
53 $db['default']['password'] = '[.A(^=bqFUtn';
54 $db['default']['database'] = 'liposerv_ispade';

```

En este fichero se establecen las configuraciones de la conexión a la base de datos.

### Configuraciones Sitio Administrador

La configuraciones de la base de datos son las mismas para el sitio web la única diferencia está en el fichero

### ispade/sitio/appweb/config/config.php

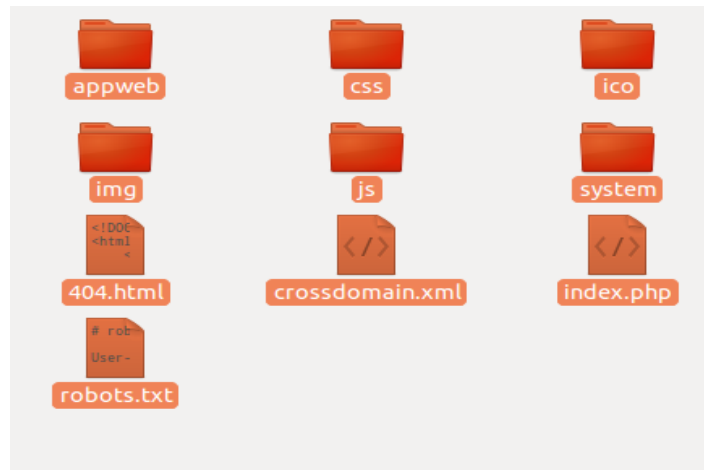
```

4 | -----
5 | Base Site URL
6 | -----
7 |
8 | URL to your CodeIgniter root. Typically this will be your base URL,
9 | WITH a trailing slash:
10 |
11 |     http://example.com/
12 |
13 | If this is not set then CodeIgniter will guess the protocol, domain and
14 | path to your installation.
15 |
16 | */
17 $config['base_url'] = 'http://ispade.liposerve.com/admin/';

```

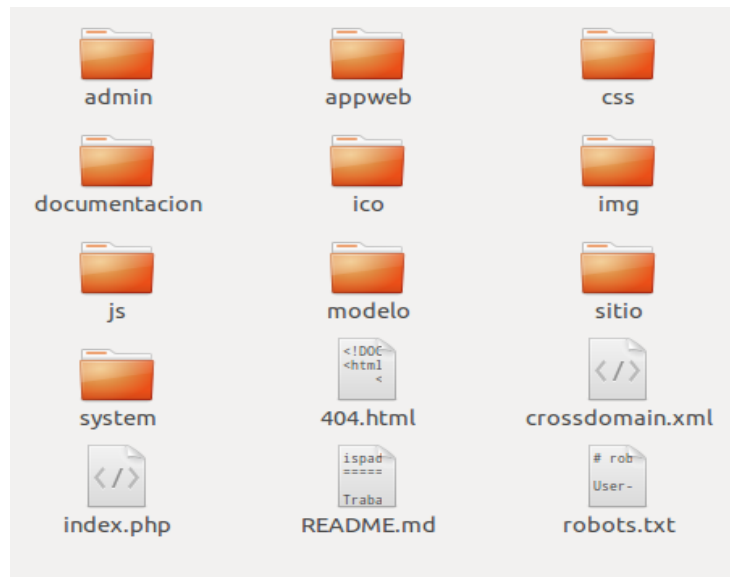
## Organizar Directorios y Comprimir

Lo único que se debe hacer es cortar el contenido de la carpeta **sitio** y pegarlo en la raíz de la carpeta **ispade**



Quedando la carpeta **ispade** de esta forma

Seleccionamos todos los archivos y carpetas damos clic derecho sobre un elemento y damos a la opción **añadir al archivo.zip** con esto estamos listos para el próximo paso.



### 7.1.3. CREAR SUBDOMINIO

Un subdominio es un dominio dependiente de otro, comparte los directorios con el dominio principal, los subdominios se reconocen por tener un nombre el signo punto seguido del nombre del dominio.

Ejemplo de subdominio:

<http://sudominio.dominio.com>

Para crear un subdominio hay que ingresar al panel de control con las credenciales proporcionadas por el proveedor de hosting en la ruta <http://liposerve.com/cpanel>.

Una vez dentro el sistema muestra una lista de opciones en forma de acordeón, para el caso nos interesa ingresar a sección **Dominios** la opción **Subdominios**.



Esta sección abrirá un formulario en el que se introduce el nombre del subdominio, en nuestro caso será ispade.

#### 7.1.4. SUBIENDO SITIO WEB

Crear un Subdominio

Subdominio :  .  ✓

Documento Principal :  ✓

Con esto el subdominio estará creado, además en el formulario se especifica la

ruta en la que se deben almacenar los archivos del sitio.

Para acceder a la lista de ficheros volvemos al home del Cpanel y seleccionamos archivos en la opción **Administrador de Archivos**.

A continuación se abre una nueva ventana con el listado de directorios del servidor, en este directorio vamos a encontrar una carpeta con el nombre del subdominio en nuestro caso sería ispade damos doble clic sobre ella y accedemos al directorio de nuestra aplicación.



public\_html

Colapsar Todo

(/home/poserv)

cpaddons

cpanel

fantasticdata

fontconfig

htpasswd

softaculous

sgmailattach

sgmaildata

trash

access-logs

cpmove.psql

cpmove.psql.1379756154

etc

mail

publicftp

public\_html

ssl

tmp

www

Home

Subir un nivel

Regresar

Adelantar

Recargar

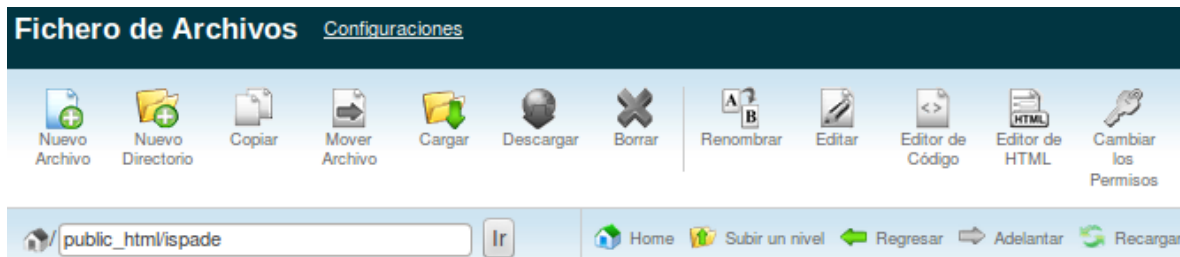
Seleccionar todo

Unseleccionar todo

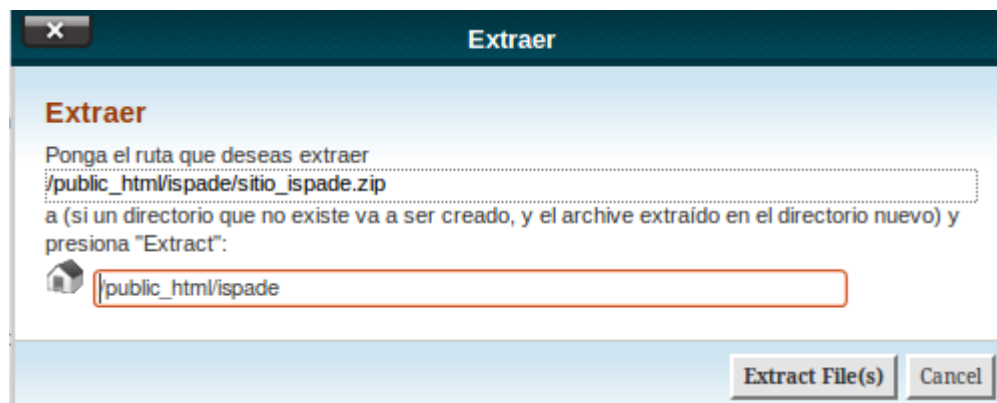
Name	Size	Last Modified (ECT)	Type	Perms
.smileys	4 KB	09/12/2010 19:23	httpdunix-directory	0755
cgi-bin	4 KB	16/11/2011 11:00	httpdunix-directory	0755
demo	4 KB	29/04/2013 11:59	httpdunix-directory	0750
esp	4 KB	16/10/2012 15:20	httpdunix-directory	0750
gmail	4 KB	22/10/2012 11:32	httpdunix-directory	0750
ispade	4 KB	18/09/2013 22:32	httpdunix-directory	0750
mail	4 KB	08/03/2012 12:35	httpdunix-directory	0755
mar	4 KB	16/10/2012 15:20	httpdunix-directory	0755
scgi-bin	4 KB	11/01/2012 11:37	httpdunix-directory	0755
vane	4 KB	16/10/2012 15:20	httpdunix-directory	0750
.htaccess	0 bytes	22/10/2012 11:04	text/x-generic	0644
index.html	951 bytes	22/10/2012 11:05	text/html	0644
index.html.zip	1.21 KB	17/05/2012 10:34	package/x-generic	0644
style.css	1.03 KB	27/02/2012 18:47	text/css	0644

Esto abrirá un directorio vacío, es posible que existan un par de archivos creados por el panel de control, no debemos eliminarlos, estos ficheros son parte de la configuración del sitio, en caso de que se elimine uno de ellos el sistema los

vuelve a crear, para subir ficheros al sitio vamos a la opción cargar en el menú superior.



Esta acción nos lleva a una nueva ventana con un formulario para subir ficheros, buscamos el comprimido de la aplicación y lo subimos, una vez terminada la carga cerramos la pestaña y volvemos a la página anterior la refrescamos y podremos ver el archivo comprimido damos clic derecho sobre este y lo extraemos.



Actualizamos el navegador y podremos ver los archivos del proyecto, la subida de archivos está completa, ahora hay que configurar la base de datos.

#### 7.1.5. SUBIR Y CONFIGURAR BASE DE DATOS

Volvemos al home del panel de control esta vez vamos a la sección **Bases de Datos** seleccionamos la opción **MySQL Bases de Datos**.

En la pantalla que se abre a continuación se puede crear el usuario de la base de datos y la base de datos es si.



Confirmamos y el sistema regresa automáticamente a la pantalla anterior ahora creamos el usuario, en la parte baja de la pantalla se encuentra otro formulario:

Este usuario es creado con el nombre y la contraseña especificada al inicio, luego de tener los dos elementos creados debemos asignar el usuario a la base de datos:

**añadir Nuevo Usuario** T DHR

Nombre Usuario:  ✓

Contraseña:  ✓

Contraseña (Otra vez):  ✓

Fuerza (por qué?): Muy Fuerte (100/100) Generador de contraseñas

Crear Usuario

Al añadirlo nos muestra una lista de privilegios para asignarle al usuario en la base de datos vamos a seleccionar las siguientes:

Enlazados y creados la base de datos y el usuario, estamos listos para subir la información a la base de datos, para subir los datos no usamos el usuario creado, ese usuario es de uso exclusivo de la aplicación, para subir los datos vamos al home del Cpanel y seleccionamos la opción **phpMyAdmin**, esto redirige a una



nueva pantalla, en ella podemos ver el listado de bases de datos, seleccionamos la nuestra liposerv\_ispade.

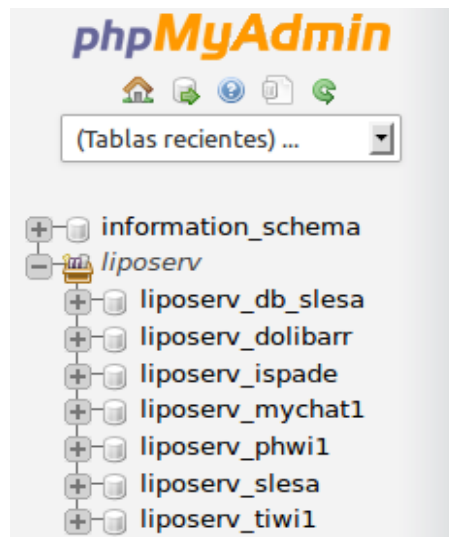
#### Manejar los Privilegios del Usuario

Usuario: liposerv\_ispad

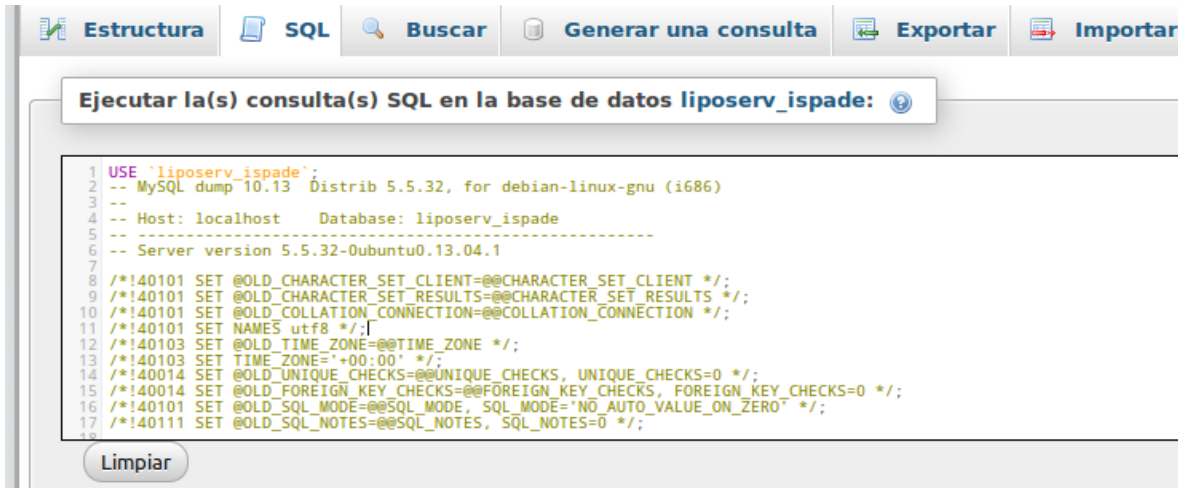
Base de Datos: liposerv\_ispade

☐ TODOS LOS PRIVILEGIOS	
<input type="checkbox"/> ALTER	<input checked="" type="checkbox"/> CREATE
<input type="checkbox"/> CREATE ROUTINE	<input checked="" type="checkbox"/> CREATE TEMPORARY TABLES
<input checked="" type="checkbox"/> CREATE VIEW	<input checked="" type="checkbox"/> DELETE
<input checked="" type="checkbox"/> DROP	<input type="checkbox"/> EXECUTE
<input checked="" type="checkbox"/> INDEX	<input checked="" type="checkbox"/> INSERT
<input type="checkbox"/> LOCK TABLES	<input checked="" type="checkbox"/> REFERENCES
<input checked="" type="checkbox"/> SELECT	<input checked="" type="checkbox"/> SHOW VIEW
<input type="checkbox"/> TRIGGER	<input checked="" type="checkbox"/> UPDATE

Hacer Cambios



En la parte superior encontraremos un menú damos clic en la opción **SQL**, esto abrirá un formulario en el que ingresaremos el script de nuestra base de datos, este script no debe contener la sentencia **create database** porque la base de datos ya fue creada solo debe tener las sentencias para crear las tablas y los datos de las mismas, en este formulario se debe pegar el contenido del fichero **ispade/modelo/liposerv\_ispade\_completa.sql**, con este fichero no habrá problemas a la hora de crear la estructura y datos de la base de datos.



Echo estos pasos el sitio está listo para funcionar.

### 7.2.1. ANEXO COMO USAR BOOTSTRAP

Bootstrap se usa directamente en las etiquetas CSS, en el capítulo de CSS se estudiaron los selectores de CSS, Bootstrap trabaja en su mayoría con selectores de clase, lo que significa que es necesario solo colocar el atributo de clase a las etiquetas o grupo de etiquetas, además en los ejemplos complejos o muy complejos Bootstrap ofrece un conjunto de etiquetas que obedecen a la estructura del elemento que deseamos construir por lo que en determinadas ocasiones es tan sencillo como copiar y pegar.

Un requisito importante es adjuntar las hojas de estilo, este código viene por defecto en las páginas de ejemplo de Bootstrap

```
<link href="ruta_al_fichero/bootstrap.css" rel="stylesheet">
```

#### Ejemplo de una página sin Bootstrap

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Sign in & middot; Twitter Bootstrap</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Pagina de Login">
  <meta name="author" content="Pagina de Login">
```

```

<!-- Le styles -->
<link href="../../assets/css/bootstrap.css" rel="stylesheet">
</head>
<body>
  <div>
    <form>
      <h2>Please sign in</h2>
      <input type="text" placeholder="Email address">
      <input type="password" placeholder="Password">
      <label >
        <input type="checkbox" value="remember-me"> Remember me
      </label>
      <button type="submit">Sign in</button>
    </form>
  </div> <!-- /container -->
</body>
</html>

```

Aspecto:



Ejemplo de una página con Bootstrap

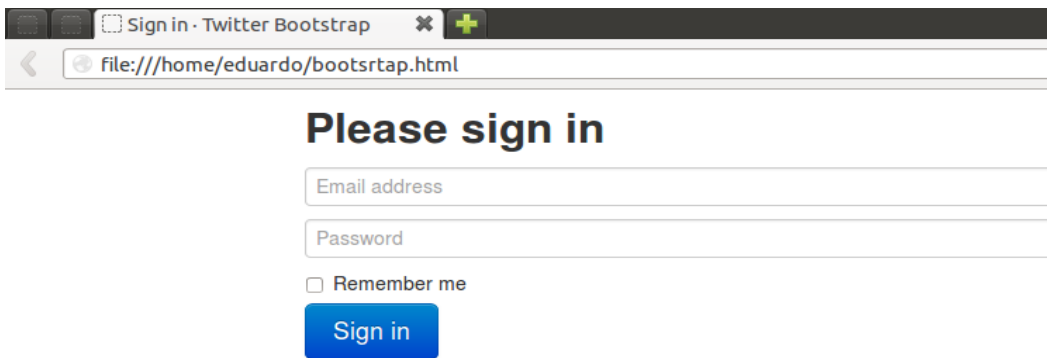
```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Sign in &middot; Twitter Bootstrap</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="">
  <meta name="author" content="">
  <link href="../../assets/css/bootstrap.css" rel="stylesheet">
</head>
<body>
  <div class="container">
    <form class="form-signin">
      <h2 class="form-signin-heading">Please sign in</h2>

```

```
<input type="text" class="input-block-level" placeholder="Email address">
<input type="password" class="input-block-level" placeholder="Password">
<label class="checkbox">
  <input type="checkbox" value="remember-me"> Remember me
</label>
<button class="btn btn-large btn-primary" type="submit">Sign in</button>
</form>
</div>
</body>
</html>
```

Aspecto:



Tomando el mismo ejemplo y añadiéndole el atributo “**class**” y un valor produce un código similar pero con un resultado completamente diferente.

### 7.2.2. SINTAXIS DE BOOTSTRAP

Bootstrap en si no tiene una sintaxis porque no es un lenguaje de programación es una herramienta que ayuda en el diseño de páginas web, no se escribe código se lo copia textualmente y se lo usa, Bootstrap lo reconoce y aplica los estilos de forma directa.

### 7.2.3. PRESTACIONES DEL FRAMEWORK BOOTSTRAP

En los siguientes subcapítulos se dará un recorrido por las características de Bootstrap, cabe recalcar que la mayoría de la documentación oficial está en ingles, y es de ahí de donde se han tomado algunos ejemplos y conceptos, son

importantes para el desarrollo y comprensión del sistema, pero al no ser una tecnología creada por mí, la documentación es una referencia hacia las fuentes, en este caso <http://bootstrap.com>

### 7.2.3.1. Scaffolding

Ayudan a crear la estructura del sitio, para que se adapte a múltiples pantallas, posee un conjunto de grillas que ajustan su contenido de manera automática al tamaño de la pantalla del dispositivo, es indispensable que se use HTML5 para que funcione.

### 7.2.3.2. Fluid grid system

Estas estructuras son idénticas a las anteriores, su única variante en código es la clase de las filas “fluid-row” por lo demás es exactamente similar, una de las características de estos elementos es la capacidad de adaptarse al tamaño de dispositivo, las estructuras se dimensionan y adaptan su contenido.

Ejemplo:

```
<html>
<head></head>
<div class="bs-docs-grid">
  <div class="row-fluid show-grid">
    <div class="span1">1</div>
    <div class="span1">1</div>
    <div class="span1">1</div>
    <div class="span1">1</div>
    <div class="span1">1</div>
    <div class="span1">1</div>
    <div class="span1">1</div>
    <div class="span1">1</div>
    <div class="span1">1</div>
    <div class="span1">1</div>
    <div class="span1">1</div>
  </div>
  <div class="row-fluid show-grid">
    <div class="span4">4</div>
```

```

    <div class="span4">4</div>
    <div class="span4">4</div>
  </div>
  <div class="row-fluid show-grid">
    <div class="span4">4</div>
    <div class="span8">8</div>
  </div>
  <div class="row-fluid show-grid">
    <div class="span6">6</div>
    <div class="span6">6</div>
  </div>
  <div class="row-fluid show-grid">
    <div class="span12">12</div>
  </div>
</div>
</html>

```

Apariencia:



A este concepto se lo conoce como “**responsive web design**”, tecnología que hace que un sitio se vea de forma similar en distintos tamaños de pantalla o ventanas.

### 7.2.3.3. Layouts

Los layouts proveen de la estructura del cuerpo de la página, posee dos elementos el cuerpo de la página y una barra lateral.

Ejemplo:

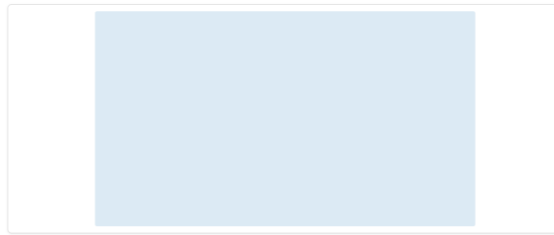
```

<html>
<head></head>

```

```
<body>
  <div class="container">
    <!--Contenido del Sitio-->
  </div>
</body>
</html>
```

Apariencia:



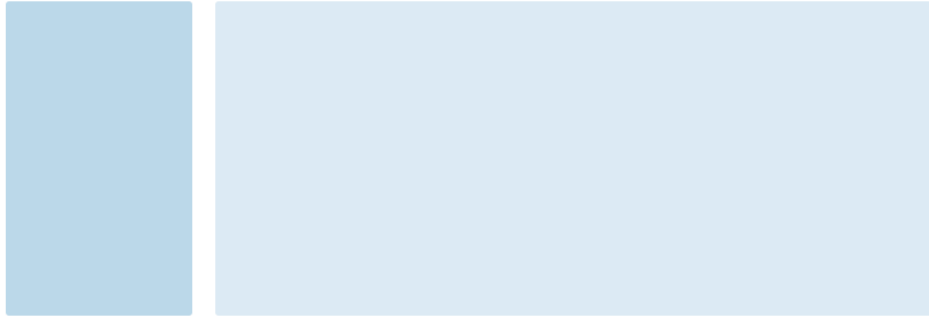
Hace que el contenido de la página aparezca en la parte central de la pantalla a un ancho de 950px.

Existe la forma de añadirle una barra al lado derecho esto demanda que el contenido sea fluido.

Ejemplo:

```
<html>
<head></head>
<div class="container-fluid">
  <div class="row-fluid">
    <div class="span2">
      <!--Contenido de la barra-->
    </div>
    <div class="span10">
      <!--Cuerpo de la página-->
    </div>
  </div>
</div>
</html>
```

Apariencia:



En este ejemplo tenemos un contenedor fluido adapta su contenido al tamaño de la pantalla del dispositivo.

#### 7.2.3.4. Base CSS

En esta sección encontramos elementos visuales de una apariencia agradable, cada uno de los elementos tienen que ver con la apariencia visual de la página con la ayuda de código CSS.

#### 7.2.3.5. Formularios con Bootstrap

Los formularios al igual que las tablas usan el código HTML normal, los estilos se aplican al formulario o a cualquiera de sus elementos, es importante revisar la estructura que deben llevar los formularios para un correcto despliegue.

Para especificar el nombre de un campo de manera visual, es decir indicar al usuario el nombre del campo para que sepa el tipo de datos que acepta, se usa una etiqueta “<label>” y luego el nombre del campo, si se desea incluir texto dentro del formulario como una especie de indicaciones se lo hace dentro de etiquetas “<span>”.

Los formularios son afectados por los estilos son la necesidad de especificar ninguna clase en su código, los campos de entrada de texto se somborean de color azul cuando tienen el foco.

Ejemplo de un formulario normal:

```
<form>
  <fieldset>
    <legend>Legend</legend>
    <label>Label name</label>
    <input type="text" placeholder="Type something...">
```



```
<span class="help-block">Example block-level help text here.</span>
<label class="checkbox">
  <input type="checkbox"> Check me out
</label>
<button type="submit" class="btn">Submit</button>
</fieldset>
</form>
```

## Encabezado del formulario

Nombre de etiqueta

Ejemplo de un elemetno de texto en el formulario

☒ Recuérdame...

### 7.2.3.5.1. buttons de Bootstrap

Cualquier elemento HTML puede ser un botón, por lo general se hace de los enlaces un botón, aunque se los puede hacer desde etiquetas “SPAN” o cualquiera de ellas, los botones son similares en su estructura, una de las diferencias es el color, mismos que se pueden usar en diferentes ocasiones:

### Estilos de botones en Bootstrap:

class=""	Descripción
btn	Un botón estándar de color gris.
btn btn-primary	Un botón estándar de color azul, generalmente usado para botones de acción .
btn btn-info	Un botón de color azul claro usado generalmente para mostrar información.

btn btn-success	Un botón de color verde que se usa para indicar una acción positiva.
btn btn-warning	Un botón de color verde que se usa para indicar una acción con posible peligro.
btn btn-danger	Un botón de color verde que se usa para indicar una acción con consecuencias irreversibles.
btn btn-inverse	Un botón estándar de color negro.
btn btn-link	Reemplaza un botón normal por un texto en forma de enlace, su uso es únicamente aplicable a los enlaces.



#### 4.6.3.5.2. Iconos en Bootstrap

Son elementos visuales que se pueden incluir en botones, menús, etc., la forma de incluirlas es como un ítem de lista y haciendo referencia al icono deseado.

Ejemplo:

```
<html>
<head></head>
<div class="btn-toolbar">
  <div class="btn-group">
    <a class="btn" href="#"><i class="icon-align-left"></i></a>
    <a class="btn" href="#"><i class="icon-align-center"></i></a>
    <a class="btn" href="#"><i class="icon-align-right"></i></a>
    <a class="btn" href="#"><i class="icon-align-justify"></i></a>
  </div>
</div>
</html>
```

Resultado:



### 7.2.3.7. Componentes de Bootstrap

Elementos que contienen funcionalidad programada en JavaScript, la mayoría de estos elementos se redimensionan de forma automática en caso de necesitarse.

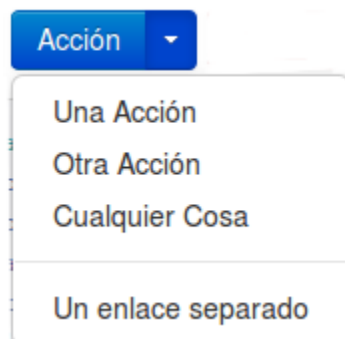
#### 7.2.3.7.1. Button Dropdowns

Estos son botones con un menú contextual, en HTML todos son enlaces, están disponibles en los mismos colores que los botones normales un ejemplo:

Ejemplo:

```
<html>
<head></head>
<div class="btn-group">
  <button class="btn">Action</button>
  <button class="btn dropdown-toggle" data-toggle="dropdown">
    <span class="caret"></span>
  </button>
  <ul class="dropdown-menu">
    <li><a href="#">Una Acción</a></li>
    <li><a href="#">Otra Acción</a></li>
    <li><a href="#">Cualquier Cosa</a></li>
    <li
class="divider"></li>
    <li><a href="#">Un enlace separado</a></li>
  </ul>
</div>
</html>
```

Resultado:



### 7.2.3.7.2. Navbar Bootstrap

Ayuda a crear menús de forma rápida, estos menús están diseñados para adaptarse entre dispositivos, una particularidad de este elemento es que cambia de forma de acuerdo al tamaño de la pantalla.

Ejemplo:

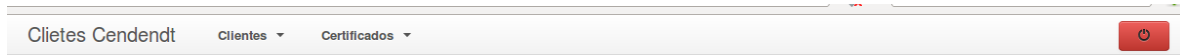
```
<div class="navbar-inner">
  <div class="container">
    <a class="btn btn-navbar collapsed" data-toggle="collapse" data-
target=".nav-collapse">
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
    </a>
    <a class="brand">Clietes Cendendt</a>
    <div style="height: 0px;" class="nav-collapse collapse">
      <ul class="nav">
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-
toggle="dropdown">&nbsp;&nbsp;&nbsp;<b>Clientes</b>&nbsp;&nbsp;&nbsp;<b class="caret">
            </b></a>
          <ul class="dropdown-menu">
            <li><a href="#"><i class="icon-th icon-black"></i>&nbsp;&nbsp;Explorar
Clientes</a></li>
            <li><a href="#"><i class="icon-file icon-black"></i>&nbsp;&nbsp;Nuevo(a)
Cliente</a></li>
          </ul>
        </li>
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-
toggle="dropdown">&nbsp;&nbsp;&nbsp;<b>Certificados</b>&nbsp;&nbsp;&nbsp;<b
class="caret"></b></a>
          <ul class="dropdown-menu">
            <li><a href="#"><i class="icon-th icon-black"></i>&nbsp;&nbsp;Explorar
Certificados</a></li>
            <li><a href="#"><i class="icon-file icon-black"></i>&nbsp;&nbsp;Nuevo
Certificado</a></li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</div>
```

```

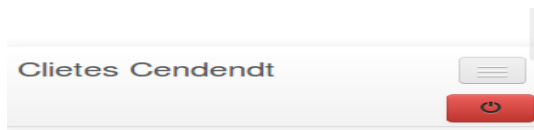
    </ul>
  </div><!--/.nav-collapse -->
  <a href="#" class="btn btn-danger navbar-form pull-right">&nbsp;<i
class="icon-off icon-black"></i>&nbsp;</a>
</div>
</div>

```

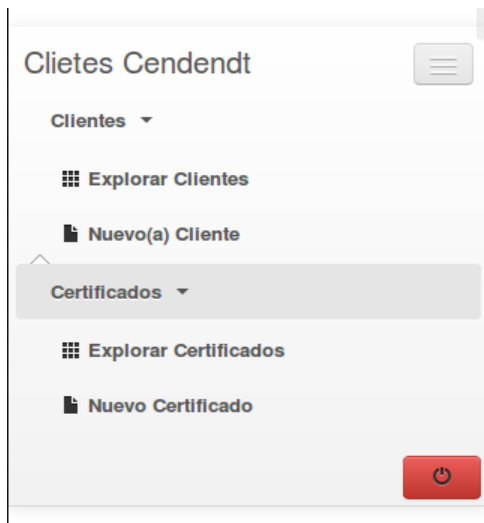
Resultado:



Resultado con la ventana 320X480 la pantalla estándar de un teléfono inteligente



Este menú se contrae y tiene una nueva disposición:



#### 7.2.3.7.3. *Pagination Bootstrap*

Son elementos que se puede usar en la paginación, son números de páginas, botones de siguiente, anterior que pueden ser usados como primeros y últimos.

Ejemplo:

```

<div class="pagination">
  <ul>
    <li><a href="#">Prev</a></li>
    <li><a href="#">1</a></li>
    <li><a href="#">2</a></li>
    <li><a href="#">3</a></li>
  </ul>

```

```
<li><a href="#">4</a></li>
<li><a href="#">Next</a></li>
</ul>
</div>
```

Resultado:



#### 7.2.3.7.4. Labels Bootstrap

La traducción literal significa etiquetas, las etiquetas se usan para resaltar un contenido, al igual que los botones tienen 6 colores diferentes.

#### Estilos labels en Bootstrap:

class=""	Descripción
label	Una etiqueta estándar de color gris.
label label-primary	Una etiqueta estándar de color azul, generalmente usado para indicar acción.
label label-info	Una etiqueta de color azul claro usado generalmente para mostrar información.
label label-success	Una etiqueta de color verde que se usa para indicar una acción positiva.
label label-warning	Una etiqueta de color verde que se usa para indicar una acción con posible peligro.
label label-danger	Una etiqueta de color verde que se usa para indicar una acción con consecuencias irreversibles.
label label-inverse	Una etiqueta estándar de color negro.

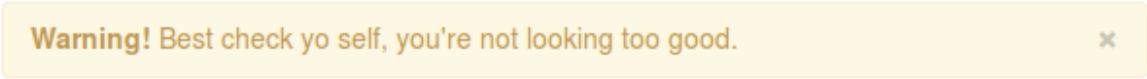
#### 4.6.3.7.5. Alertas Bootstrap

Similar a las etiquetas, se les agrega funcionalidad con JavaScript para que el usuario pueda cerrarlos.

Ejemplo:

```
<div class="alert">
  <button type="button" class="close" data-dismiss="alert">×</button>
  <strong>Warning!</strong> Best check yo self, you're not looking too good.
</div>
```

Resultado:



## 7.2.4. COMPONENTES JAVA SCRIPT EN BOOTSTRAP

Los elementos de esta sección contienen mucha programación JavaScript, son muy funcionales tienen utilidades como el aprovechamiento del espacio y hacer el contenido más accesible.

### 7.2.4.1. Modal Bootstrap

También son conocidos como cuadros de dialogo, la singularidad de estos cuadros es que ocultan el resto de contenidos de la página, haciendo que el usuario tome una decisión para continuar.

Ejemplo:

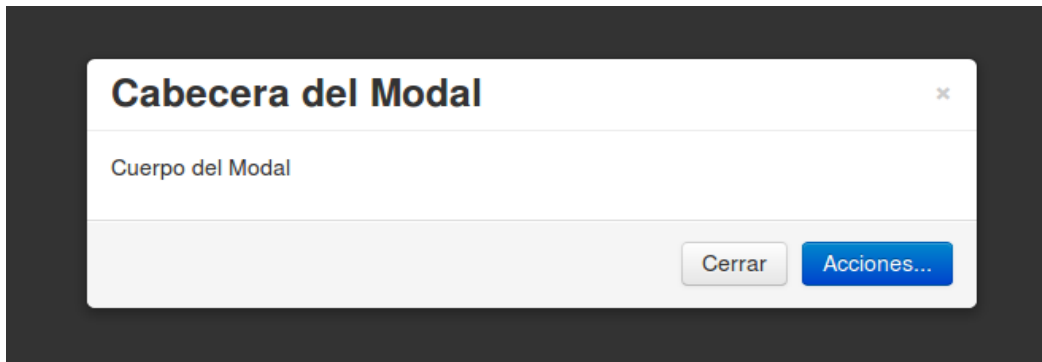
```
<!-- Button to trigger modal -->
<a href="#myModal" role="button" class="btn" data-toggle="modal">Launch demo
modal</a>
<!-- Modal -->
<div id="myModal" class="modal hide fade" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel" aria-hidden="true">
  <div class="modal-header">
    <button type="button" class="close" data-dismiss="modal" aria-
hidden="true">×</button>
    <h3 id="myModalLabel">Cabecera del Modal</h3>
```

```

    </div>
    <div class="modal-body">
        <p>Cuerpo del Modal</p>
    </div>
    <div class="modal-footer">
        <button class="btn" data-dismiss="modal" aria-
hidden="true">Cerrar</button>
        <button class="btn btn-primary">Acciones...</button>
    </div>
</div>

```

Resultado:



#### 7.2.4.2. Tabs Bootstrap

Coloca el contenido de una capa div en forma de pestañas, usa un mismo espacio para presentar varios contenidos, es usada para presentar información relacionada.

Ejemplo:

```

<ul class="nav nav-tabs" id="myTab">
    <li class="active"><a href="#home">Home</a></li>
    <li><a href="#profile">Profile</a></li>
    <li><a href="#messages">Messages</a></li>
    <li><a href="#settings">Settings</a></li>
</ul>

<div class="tab-content">
    <div class="tab-pane active" id="home">Contenido de la primera pestaña</div>
    <div class="tab-pane" id="profile">Contenido de la segunda pestaña</div>
    <div class="tab-pane" id="messages">Contenido de la tercera pestaña</div>
    <div class="tab-pane" id="settings">Contenido de la cuarta pestaña</div>
</div>

```



```
<script>
$(function () {
    $('#myTab a:last').tab('show');
})
</script>
```

Resultado:



### 7.2.4.3. Collapse Bootstrap

Similar a tab la diferencia radica en la forma en la que presenta el contenido, en una especie de acordeón.

Ejemplo:

```
<div class="accordion" id="accordion2">
  <div class="accordion-group">
    <div class="accordion-heading">
      <a class="accordion-toggle" data-toggle="collapse" data-
parent="#accordion2" href="#collapse0ne">
        Grupo Nro1
      </a>
    </div>
    <div id="collapse0ne" class="accordion-body collapse in">
      <div class="accordion-inner">
        El texto del primer artículo
      </div>
    </div>
  </div>
  <div class="accordion-group">
    <div class="accordion-heading">
      <a class="accordion-toggle" data-toggle="collapse" data-
parent="#accordion2" href="#collapseTwo">
        Grupo No2
      </a>
```

```
</div>
<div id="collapseTwo" class="accordion-body collapse">
  <div class="accordion-inner">
    El texto del segundo artículo
  </div>
</div>
</div>
</div>
```

Apariencia:

Grupo Nro1
El texto del primer artículo
Grupo No2

## CONCLUSIONES

Gracias por darle lectura a este documento de inicio a fin, espero haya sido de utilidad, una de las conclusiones más importantes está en que si una aplicación posee usuarios normales y usuarios administradores, es mejor implementar las aplicaciones de forma separada.

Es también importante separar la aplicación en partes que trabajen juntas, un problema es más fácil de resolver si está dividido en partes.

En el mundo de la tecnología existen muchas herramientas de libre uso, en este proyecto se usa más de una, es recomendable usarlas en lugar de trabajar desde cero.

## RECOMENDACIONES

Para obtener buenos resultados y un producto de calidad es recomendable seguir patrones de diseño y estándares para todos los niveles de desarrollo.

Llevar una buena documentación, tratando de documentar todos los componentes de la aplicación.

Usar un gestor de versiones, en el caso de este proyecto se usó git y github para tener un control completo del desarrollo de la aplicación

## BIBLIOGRAFÍA

En esta sección del documento presento una referencia a los documentos en los cuales me basé para escribir y desarrollar lo que en éste documento se encuentra.

<http://es.wikipedia.org/wiki/Internet>

<http://es.wikipedia.org/wiki/http>

<http://ellislab.com/codeigniter/user-guide/>

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

<http://eugeniabahit.blogspot.com/p/poo-y-mvc-en-php.html>

<http://www.desarrolloweb.com/articulos/que-es-html.html>

[http://es.wikipedia.org/wiki/HTML#Historia de HTML](http://es.wikipedia.org/wiki/HTML#Historia_de_HTML)

[http://www.w3schools.com/html/html\\_intro.asp](http://www.w3schools.com/html/html_intro.asp)

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

<http://www.w3schools.com>

<http://www.cristalab.com/tutoriales/introduccion-a-html5-c921711/>

[http://www.w3schools.com/cssref/css\\_selectors.asp](http://www.w3schools.com/cssref/css_selectors.asp)

<http://sublimetext.com>

<http://codeigniter.com>

<http://apache.org>

<http://php.net>

<http://git-scm.com>

<http://github.com>

<http://dev.mysql.com>

<http://argouml.tigris.com>