Documentación trabajo fin de curso

Iván Calderón Culebras

Indice

Introducción

Esta página es mi proyecto final, trata sobre la venta de consolas catalogadas como retro y accesorios nuevos .

Parte front

DESCRIPCIÓN FUNCIONAL

Introducimos nuestra pagina con un captcha que cambia con cada recarga, tiene dos botones refrescar y validar. En caso de que sea el numero y letras erroneo solo te da un mensaje, por lo contrario si es introdducido bien te llevará al cátalogo.
En el cátalogo (trabajo 1) gracias a la cabecera podemos acceder a distitas partes, pero con la que se puede acceder de otra maneraa es con un formulario. Este formulario es validado y corregido, en función al sexo escogidod te llevara a páginas hijos con distintas mercancias que se venderán en la página . En el formulario se valida un correcto dni, un formato correcto del gmail y que no exitan inputs vacios (campos sin respuesta). Por parte de las páginas hijos , se comprueba si es su cumpleaños y se le fecita, pero en caso de que sea menor de edad te devuelve al catálogo. En las páginas hijos tienes la opción de dirigirte a un contenido auidio

visual con un menú que te permite descargar, reproducir/parar, subir/bajar volumen o silenciar.
Es importante resaltar que varios datos de la parte back son enviados o validados con Ajax

Solución técnica

Página padre

Posee una cabecera, cuerpo y pie (en contacto y carrito). En la parte head:
<link rel="icon" href="icono/pink.ico" type="imagen/ico" />
<link rel="stylesheet" href="CSS/primero.css" type="text/css">

La primera linea conecta con la carpeta icono y selecciona el icono usado, la segunda linea conecta con la carpeta de los estilos; de ahí obtenemos el color, ordenación y las dimensiones de los contenedores.
En la parte del formulario existen tres contenedores , megacu (que contiene a los otros dos), rectángulo(el de la izquierda vacío) y cuadrado. El contenedor cuadrado posee imágenes que son a su vez enlaces , están ordenados en columnas con flex-vox
<img class="img" alt="" src="imagenes/descarga.jfif" title="Helado pantera rosa">
El formulario se conecta con las páginas hijos , cada input del formulario es nombrado y se le da una id en nuevo script con el que se recopila para ser enviado con lo siguiente:

```
for(var i = 0; i < sexo.length; i++) {// para recoja el sexo de manera corecta
        if(sexo[i].checked) {
            window.open("paginas2/"+ sexo[i].value +".html?" + nombre1 + "&" +
apells1  + "&" +  apellidos1 + "&" + fechas);
        }
```

.
El pie va por encima del script pero solamente es un contenedor , sin contenido destacable.

Páginas hijos y página audiovisual

La parte de html:
El diseño es el mismo que el padre así como colores o contenedor, la diferencia es que en la parte de navegación hay dos iframes , de otras páginas las cuales poseen enlaces para ver noticias o datos sobre el personaje de la página:

<iframe src="iframes/iframe1.html" frameborder="0" width="100%" height="50%" allowfullscreen=""></iframe>
En esta parte existe una referencia al script que saca la fecha, día y mes actual .
El script :

```
<script src="h.js">
</script>
```

Hace referencia a otra pagina con solo JavaScript , varia en las 3 paginas (como o.js, m.js) .
Por finalizar tiene enlace para volver a la pagina padre:
Volver

La parte de JavaScript:
Recopilamos los datos del usuario, mediante :
var params = window.location.search.substring(1);
Pasamos el string a un array con :
.var params2 = params.split(/&/);
Y para obtener la fecha
var cumpleUsu= params2[3].split(/-/);
var anoUsuario = params2[3].slice(0, 4);// año usu slice es para cortar
La fecha del primero se separa por “-” y se obtiene mes , día y año ,
La segunda la registra diferentemente toda la fecha.
var diferencia = parseInt(anoActual) - parseInt(anoUsuario);// diferencia de edad
var diferencia2 = parseInt(diaCumple) - parseInt(diaActual);
var diferencia3 = parseInt(mesCumple) - parseInt(mesActual);

Primero se calcula la diferencia de edad y las otras dos solo sirven para encontrar la diferencia entre día/mes del usuario y el día/mes de hoy asi con un if podemos determinar su cumpleaños si da 0.

```
var hoy = new Date();
var cumpleanos = new Date(fecha);
var edad = hoy.getFullYear() - cumpleanos.getFullYear();
var m = hoy.getMonth() - cumpleanos.getMonth();

if (m < 0 || (m === 0 && hoy.getDate() < cumpleanos.getDate())) {
    edad--;
}

return edad;
```

Mediante este último código obtenemos la edad real del usuario, diferenciando entre mes, para prever errores menores por algún mes o día.
Por último con una comparación de un if (>=18) da paso al usuario con el mensaje bienvenido y si no es mayor de 18 se le expulsa mediante:
window.close()

Captcha

Se crea un formulario con 4 inputs

```
<input type="text" id="CaptchaEnter" placeholder="Introduzca el captcha"><br/>
<input type="button" value="Refrescar" id="refreshbtn" onclick="genCapNuevo()">
<input type="button" value="Validar" id="checkbtn" onclick="comprobarCap()">
```

.Estos tres recopilan información y llaman a 2 funciones, "genCapNuevo()" , que genera los captcha y al pulsarlo genera nuevos con un math.ramdom

```
function genCapNuevo(){
  chars =
"1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
  captcha = chars[Math.floor(Math.random() * chars.length )];
  for(var i=0; i<6; i++){
    captcha = captcha + chars[Math.floor(Math.random() * chars.length )];
  }
  form1.text.value = captcha;
}
```

El siguiente botón válida creando una función que compara los datos introducidos con los de el math.random:

```
function comprobarCap(){
  var check = document.getElementById("CaptchaEnter").value;
  if(captcha == check){
    alert("CORRECTO!! Adelante");
    document.getElementById("CaptchaEnter").value = "";
    reedirecion();
  }else{
    alert(" UPSS, Intentalo de nuevo");
    document.getElementById("CaptchaEnter").value = "";
  }

  genCapNuevo();
```

Si es correcto te enviá a trabajo1.html y si no genera un nuevo capcha.

```
<input type="text" id="captchaTxtArea" name="text" value="" readonly><br/>
```

Este último input muestra por pantalla los datos al azar y tiene un background de una imagen que hace que los robots tengan fallos junto las letras rojas

Cookies

En caso de validar datos creo un patrón y distintas funciones , y si no coinciden los datos del usuario (del form) saltará un alert .

En el caso de las cookies primero las recopilo

```
document.getElementById("nombre").value = getCookie("nombre");
```

y luego las establezco `setCookies(doni,nombre1,apells1,apellidos1,fechas);`

en el caso que el input no este seleccionado hace localStorage.setItem("dni", doni);
recopilando de otra forma.
Mediante function getCookie(nombre) {
corto la cookies por el = y mediante un splice las separo por “;”.
function verificarCookies() {
Con esta función genero las cookies si existen en los formularios :
document.getElementById("nombre").value = getCookie("nombre");
var user = getCookie("nombre");
La ultima linea de código establezco el usuario para saludarle mediante recargas y se mantengan las cookies.
Para limpiar el usuario o cambiar de usuario : hago que caduquen las cookies y que el local se limpie
 localStorage.clear();
        document.cookie = "dni=doni;expires=1 Mar 1990 00:00:00 GMT";
El video/audio simplemente se le añade la direción de la carpeta y sus respectivas etiquetas de video/audio

```
        <video src="../videoAudio/segaUnboxing.mp4" autoplay controls loop></video>
        <audio controls loop>
           <source src="../videoAudio/kirbySinCopy.mp3" type="audio/mp3">
        </audio>
```

PARTE BACK

DESCRIPCIÓN FUNCIONAL

Los productos son mostrados de forma dinámica y traídos de una base de datos (menos sus imágenes), en esta parte podemos realizar varias acciones , como ver los detalles de cada producto pulsando el botón details (aquí existe un botón para agregar al carrito), o agregar al carrito y la última función es ver los productos ya agregados en el botón carrito. En este botón hay dos opciones añadir o restar la cantidad de productos o de finalizar compra y nos pide un que rellenemos un corto formulario con nuestros datos personales , al finalizarlo nos devuelve al catálogo guardando los datos de nuestra compra en varias base de datos.

Solución técnica

Lo primero es hacer referencia y necesario a las partes de config.php y database.php (esto se repite en prácticamente todas mis paginas .php).

```
require '../config/config.php';
```

```
require '../config/database.php';
```

La parte de config es básicamente donde se guardan las sesiones de nuestro carrito y también tengo una parte de la encriptación.

```
define("KEY_TOKEN", "APR.wqc-345*");
define("MONEDA", "€");// para que sea sencillo cambiarlo a otra moneda
y hacerse mas dinamico
session_start();
$num_cart = 0;
if(isset( $_SESSION['carrito']['productos'])){
    $num_cart = count( $_SESSION['carrito']['productos']);
}
```

En la parte de database se guarda la función para conectarse a la base de datos (con PDO) y luego lo ejecutamos con :

```
$db = new Database();
$con = $db->conectar();
$sql = $con->prepare("SELECT id, nombre, precio FROM productos WHERE
activo=1");// LO DE ACTIVO SIRVE PARA QUITAR PRODUCTOS PONIEND EN LA BD
SI CAMBIAMOS DE VALOR NO LOS MOSTRARÁ
$sql->execute();
```

Luego asociamos el resultado con el array asociativo.
En la parte de productos, los mostramos con un array asociativo

```
  <?php foreach($resulatdo as $row){?>
```

Para mostrar las imágenes traemos el id de productos y lo añadimos a la ruta de la imagen, porque ese id coincide con el nombre de la carpeta (ejemplo si saca el id 1 seleccionará la carpeta 1:

```
  $id = $row['id'];//Obtenemos la variable id de la BD
            $imagen ="../img/productos/". $id ."/principal.jpg";
```

En el caso de que sea encontrada o no exista mostrará por defecto otra imagen

```
            if(!file_exists($imagen)){
                $imagen = "../img/no-photo.jpg";// cuando no hay foto
existente o esta mal se añade una predefinida
            }
            ?>
```

En el cuerpo de cada producto se nos muestran dos botones, agregar al carrito o details (para ver los detalles).

```
                    <a href="details.php?id=<?php echo $row['id'];
?>&token=<?php echo hash_hmac('sha1', $row['id'], KEY_TOKEN);?>"
class="btn btn-primary">Detalles</a>
                </div><!--hash_hmac('sha1', $row['id'], KEY_TOKEN)
ciframos la ID CON NUESTRA CONTRASEÑA-->
                <button class="btn btn-outline-success"
                type="button" onclick="addProducto(<?php echo
$row['id'];?>, '<?php echo hash_hmac('sha1', $row['id'],
KEY_TOKEN);?>')">Agregar al carrito</button>
            </div><!--Se agrega de forma dinamica la id y el token -->
```

Estos datos son enviados o al carrito o a detalles para que nos muestre sus datos con la opción de añadirlos a nuestra compra.
Luego dentro de detalles codificamos y recopilamos nuestros datos cifrado,

```
$id = isset($_GET['id']) ? $_GET['id'] : ''; // en el caso de que este
definido la id lo recopila y si no le da ""
$token = isset($_GET['token']) ? $_GET['token'] : '';
if ($id == '' || $token == '') {
    echo 'Error al procesar al peticion 1';
    exit;
} else {
    $token_tmp = hash_hmac('sha1', $id, KEY_TOKEN);
```

Luego establecemos los datos y mediante un while traemos todas las imágenes de nuestro producto (mediante un carrusel se muestran todas).

Mediante una función script enviaremos nuestras selecciones de agregar carrito :

```
       function addProducto(id, token){
            let url = 'carrito.php'
            let formData = new FormData()//facilita recopilar datos
mediante post
            formData.append('id', id)
            formData.append('token', token)//hasta aquí es la conf de
la peticion por ajax
            fetch(url, {// aqui ya lo enviamos por post
                method: 'POST',
                body: formData,
                mode: 'cors'
            }).then(response => response.json())
            .then(data => {
                if(data.ok){//con data. se accede a los elemtos
enviados
                    let elemento = document.getElementById("num_cart")
                    elemento.innerHTML = data.numero
                }
```

```
            })
```

Lo enviamos a carrito.php con el método POST (es una petición ajax) el response no generará un resultado. En carrito.php se verifican los productos enviados que se van sumando y es donde se recopilan las sesiones . En el .then(data => … se accede a los datos de respuesta que se nos agregan . Toda la parte de script se añadirán donde están los botones agregar al carrito . También las .php se puede apreciar el número de productos agregados (no la cantidad por ahora pero si la variedad).

Después en checkout.php recogeremos los datos de la sesión los guardamos e un array .

```
$productos = isset($_SESSION['carrito']['productos']) ?
$_SESSION['carrito']['productos'] : null;// en caso de que exista la
recibiremos si no se le da el valor null
```

En caso de que no existan productos seleccionados saldrá un mensaje de que no hay productos. Luego muestro los productos en una tabla (de bootstrap) con un foreach muestro los productos, cantidad , subtotal y nombre. En el caso de la cantidad y el subtotal son dinámicos

```
<input type="number" min="1" max="10" value="<?php echo $cantidad?>"
size="7"
                        id="cantidad_<?php echo $_id;?>"
onchange="actualizaCantidad(this.value, <?php echo $_id; ?>)"
onclick="recargar()">
```

Llama a una función y se recarga al añadir o quitar cantidad, esto afecta a subtotal, cantidad y total, que nos altera la sesión de forma instantánea. En actualizar_carrito.php recibimos y comprobamos que el id existe del producto que cambiamos la cantidad, entonces volvemos a traer los datos de a base de datos (pero solo los necesarios), volvemos a calcularlo y retornamos su respuesta:

```
 $db = new Database();
        $con = $db->conectar();
        $sql = $con->prepare("SELECT precio, descuento FROM productos
WHERE id=? AND activo=1 LIMIT 1"); // LO DE ACTIVO SIRVE PARA QUITAR
PRODUCTOS PONIENDO EN LA BD SI CAMBIAMOS DE VALOR NO LOS MOSTRARÁ
        $sql->execute([$id]);
        $row = $sql->fetch(PDO::FETCH_ASSOC);// se asigna en un
asociativo

        $precio = $row['precio'];
        $descuento = $row['descuento'];
        $precio_desc = $precio - (($precio * $descuento) / 100);// si
queremos meter en la bd descuentos lo actualizará
        $res = $cantidad * $precio_desc;
        return $res;
```

Con `echo json_encode($datos);// esto es lo que regresa la petición` regresamos la petición , después es cambiado directamente en nuestra tabla. Esta función también recopila los datos de subtotal

```
let list = document.getElementsByName('subtotal[]')
                    for(let i = 0; i < list.length; i++){
                        total +=
parseFloat(list[i].innerHTML.replace(/[€,]/g, ''))// se quitan los
simbolos y comas, porque generan problemas

                    total = new Intl.NumberFormat('en-US', {// LE PONGO
EL FORMATO amricano para que me coincidan los . y ,
                        minimumFractionDigits: 2
                    }).format(total)
                    document.getElementById('total').innerHTML =  total
+'<?php echo MONEDA; ?>'
```

También se obtiene el total modificado, y se le quita el formato cabe resaltar que le puse el de US porque en el alemán daba errores(o europeo).

En el caso del botón eliminar llama a un modal para verificar si el usuario desea eliminar un producto, en el caso de que siga con el proceso llama a función eliminar(); , esta modifica los datos de la sesión

```
function eliminar($id){
    if($id > 0){
        if(isset($_SESSION['carrito']['productos'][$id])){
            unset($_SESSION['carrito']['productos'][$id]);
            return true ;
        }
    }else{
        return false;
```

Básicamente recopila el id y quita los datos que lleven ese id en actualizar_carrito.
Para finalizar esta parte en esta página existe el botón finalizar compra (checkout.php) que recopila el total y lo envía finalizar.php.
En finalizar.php recopiló los datos del usuario y su subtotal en dos bases de datos. Mediante commit confirmó que son correctos y si no realiza un rollback

```
$con->beginTransaction();
        $sql = $con->prepare("INSERT INTO compra (nombre, email,
direccion, total) VALUES(?,?,?,?)");
        $sql -> execute([$nombre, $email, $direccion, $total]);
        $id = $con->lastInsertId();// trae el id que se inserta
        if( $id > 0){
            $productos = isset($_SESSION['carrito']['productos']) ?
$_SESSION['carrito']['productos'] : null;//tramos la variable de sesion
```

```
            if($productos != null){// verificamos que no sea nula
                foreach($productos as $clave => $cantidad){// clave va
a ser la id del producto y cantidad la cantidad

            $sql = $con->prepare("SELECT id, nombre, precio, descuento
FROM productos WHERE
            id=? AND activo=1");// cantidad as cantidad solo se esta
pasando la cantidad a la consulta y no a la BD
            //PARA QUE LUEGO APAREZCA EN EL RESULTADO
            $sql->execute([$clave]);// EJECUTAMOS LA CONSULTA
            $row_prod = $sql->fetch(PDO::FETCH_ASSOC);//Solo fetch
porque vamos a sacar de 1 en uno cada producto

            $precio = $row_prod['precio'];
            $descuento = $row_prod['descuento'];
            $precio_desc = $precio-(($precio * $descuento)/100);

            $sql_insert = $con->prepare("INSERT INTO detalle_compra
(id_compra, id_producto, nombre, precio, cantidad)
            VALUES(?,?,?,?,?)");
            $sql_insert->execute([$id, $clave, $row_prod['nombre'],
$precio_desc, $cantidad]);
                }
                echo('<h3>Compra finalizada </h3> <br> <h2><a
href="trabajo1.php">Volver al inicio</a></h2>');
                unset($_SESSION['carrito']);// despue se elimina lo que
esatba en el carrito
                header('location: trabajo1.php');
            }


        }
        $con->commit();
```

Es importante señalar que esta parte tiene 2 botones finalizar (en caso de que todo esté bien realiza lo anterior y al finalizar destruye la sesión), devolviendote atrás al acabar (en el virtual host no deja modificar el head)

```
header('location: trabajo1.php');
```

El otro botón es seguir comprando que te envía atrás `onClick="history.go(-2);.`

La última parte son las tablas, mediante un select y varios options, verificamos la opción con un if y dependiendo de su selección hace una consulta u otra.

```
if(isset($_POST["mostrar"])) {
            $selecionTabla = $_POST['elegirTabla'];
            $sql = $con->prepare("SELECT * FROM $selecionTabla");
            $sql->execute();
            if($selecionTabla == 'compra'){
        echo('<table class="table table-dark table-striped">');

echo('<tr><th>ID</th><th>Nombre</th><th>Email</th><th>Direccion</th><th
>Total</th></tr>');
        foreach($sql as $elementos){

echo('<tr><td>'.$elementos["id"].'</td><td>'.$elementos["nombre"].

'</td><td>'.$elementos["email"].'</td><td>'.$elementos["direccion"].
            '</td><td>'.$elementos["total"].'</td></tr>');
        }
        echo('</table>');
        }
```

En caso de que seleccione detalle_compra se mostrarán las claves foráneas y si pulsa en mostrar las PK de las fk , no envía a ver con las claves que está relacionada .

```
<td><a href="tablasFK.php?produc=<?php echo
$elementos['id_producto'];?>&com=<?php echo $elementos['id_compra'];?>"
class="btn btn-primary">Mostrar Pk de las fk</a></td></tr>
```

Luego recopilamos los datos enviados y se muestran 2 tablas para cada pk (primary key)

Guia de estilos

Header: height: automático width: 100%
El contenido del div contenedor está dividido en 3 row row-cols-1 row-cols-sm-2 row-cols-md-3 g-3(el resto de páginas posee el mismo header y container)
Dimensiones imágenes: auto
Dimensione imágen w3c : height: auto width: auto
Dimensiones iframes: height: 70% width: 80%
En el captcha difieren los tamaños

padding-right: 5px;
background: white;
width: 98%;
border: 5px solid black;

border-radius: 12px;

Las dimensiones de sus botones son las siguientes :

width: 303px;
height: 40px;
border: 3px solid black;

En el caso de las letras que lo componen:

background: url(../imagenes/cap.jpg);
color: #FF4500;
font-size: 30px;
font-weight: bold;
cursor: pointer;
border: none;
outline: none;
pointer-events: none;
margin-top: 15px;
width: 154px;

Sus h1:

font-family: sans-serif;
text-transform: uppercase;
font-size: 40px;

El input para recoger las claves:

width: 303px;
height: 40px;
border: 3px solid black;

Colores:
Header: black
text header :white
button carrito: blue
main: white
Font main :black
button agregar carrito: green
Pie:grey :2px solid black
NOTA: las paginas hijo heredan estas dimensiones y colores

Manual de despliegue

1.Requerimientos hardware y software

En la parte de hardware, mínimo debe poseer un 1GB de RAM, para funcionamiento más óptimo deberá poseer 4 GB de RAM. El instalador de VirtualBox pesa 100MB, pero es aconsejable guardar al menos 5GB libres para el SO que le instalemos (en nuestro caso lo enfocamos con Linux mint).
En el caso de software, VirtualBox es compatible con versiones de 32 bits y 64 bits de Vista, Windows 7 y Windows 8, así como versiones de 32 bits de Windows XP. También funciona en plataformas de servidor Windows, incluida la edición de 32 bits de Windows Server 2003, tanto Windows Server 2008 como Windows Server 2012 de 32 y 64 bits. también puede ejecutarse en múltiples sistemas operativos. Oracle ofrece una versión OS X de VirtualBox que se ejecuta en las versiones 10.6, 10.7 y 10.8. Puede usar VirtualBox con cuatro tipos de Linux: Oracle Enterprise Linux, SUSE Linux, Ubuntu y Redhat Enterprise Linux. VirtualBox también es compatible con Solaris 10 y Solaris 11.
https://linuxmint.com/download.php
Para el funcionamiento de las bases de datos y parte del php se requiere tener instalado XAMPP (activando Apache y MySQL), por último es necesario tener nuestra base de datos con sus tablas respectivas, también es importante hacer que en la config de mysql (de xampp) comparta puerto (por defecto es el 3006).

2.Instrucciones de despliegue y configuración inicial

Después de instalar VirtualBox se debe descargar el SO (él enlace es para descargar Linux mint).
https://linuxmint.com/download.php
Vamos a la pestaña de máquina y seleccionamos nueva, a la hora de seleccionar su RAM se debería seleccionar 2GB. En las selecciones debemos clicar en VDI, su reserva es mejor hacerla dinámica (por si requerimos más espacio del esperado), en el caso de almacenamiento deberá ser mínimo de 5GB y por último seleccionamos crear el disco virtual ahora. El último paso arrancar la máquina virtual y seleccionar el SO.
Después de configurar nuestro usuario del respectivo SO abrimos la terminal y metemos el siguiente comando: “sudo apt-get install apache2”
Lo iniciamos con “sudo service apache2 start”

Luego añadimos en “etc/host”
#IP nombre-dominio
127.0.0.1 proyecto.com www.proyecto.com

En /etc/apache2/sites-available donde se definen los virtualhosts, cada virtualhost es un fichero de texto de configuración distinto, podemos copiar el 000-default y modificarlo.
Creando así un proyecto.conf, en este documento seleccionaremos el tipo de puerto 443 o 80 (o que te redirija al 443 que es el https).
Para activarlo realizamos un a2ensite proyecto (es aconsejable abrir desde la terminal donde se encuentra nuestro conf)
Para nuestros certificados primero instalamos el paquete openssl:
apt-get install openssl
Luego generamos un certificado autofirmado y la clave privada de tu autoridad de certificación (AC)
mkdir /etc/apache2/mis-cert/
cd /etc/apache2/mis-cert/
Estando en mis-cert:
openssl req -new -x509 -nodes -days 365 -keyout archivoClavePrivada.key -out archivoCertificado.crt

Después de comprobar que tenemos nuestros archivos crt y key editamos la configuración SSL por defecto en el
archivo /etc/apache2/sites-available/default-ssl para indicar el
certificado del servidor y su respectiva clave privada
asignando los siguientes valores a los parámetros :
SSLEngine on
SSLCertificateFile /etc/apache2/mis-ssl/proyecto.crt
SSLCertificateKeyFile /etc/apache2/mis-ssl/proyecto.key
Por último habilitamos el módulo ssl.
a2enmod ssl
a2ensite default-ssl
/etc/init.d/apache2 restart

Ahora instalaremos un soporte para php,
apt-get install libapache2-mod-php php-mysql
Después se accede a
gedit /etc/apache2/mods-enabled/dir.conf
Lo modificamos para que abra los archivos index.php antes que los html
<IfModule mod_dir.c>
DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm
</IfModule>
Por último se reinicia Apache.
En caso de que necesitemos mysql
Instalamos el servidor MYSQL.
apt-get install mysql-server mysql-client
mysql -u root -p
Se configura una contraseña

MYSQL_NATIVE_PASSWORD.
msql>ALTER USER root@localhost IDENTIFIED WITH mysql_native_password;
msql>ALTER USER root@localhost IDENTIFIED BY ‘root';
Se guarda y se sale.
msql>FLUSH PRIVILEGES;
msql>exit;
Es importante resaltar que se debe tener un servidor remoto para las BD.
El último paso es introducir nuestro proyecto en www/var/html/proyecto, respetando los niveles y se hace referencia en el Documentroot, finalizando con apache restart
.