



**UNIVERSIDAD  
DE GRANADA**

**PROGRAMACIÓN WEB.**

## **Práctica 2.**

**Daniel Terol Guerrero.**

# ÍNDICE

---

1.	Introducción.....	3
2.	Base de datos.....	3
3.	Elementos característicos .....	3
4.	Index.html .....	5
5.	Altausuario.html .....	5
6.	Altausuario.php.....	7
7.	Index.php .....	7
8.	Index2-Login.php .....	8
9.	LoguearUsuario.php.....	9
10.	Index2.php .....	9
11.	DatosPersonales.php .....	9
12.	Index2-modificarDatos.php .....	10
13.	MisLibros.php.....	10
14.	AltaLibro.php.....	11
15.	Altalibro-2.php .....	12
16.	nuevoLibro.php .....	12
17.	LibroLeido.php .....	12
18.	ModificarDatos.php .....	13
19.	actualizarOpinionLibro.php.....	13
20.	Libro.php.....	13
21.	valorarLibro.php.....	15
22.	darOpinionLibro.php.....	15

## 1. INTRODUCCIÓN

---

En la segunda práctica de Programación Web se ha dotado de funcionalidad a la página web desarrollada en la primera práctica. Entre las funcionalidades desarrolladas tenemos creación de un usuario, inicio de sesión de usuario a través de credenciales, creación de un libro, capacidad de opinar y valorar un libro además de consultar la opinión de los demás usuarios y la valoración media de un libro específico.

## 2. BASE DE DATOS.

---

La disposición de la base de datos que he considerado para implementar es la siguiente:

- **Usuarios:** En esta tabla se almacena la información de los usuarios.
- **Libros:** En esta tabla se almacena la información de los libros. Además, se almacena qué usuario ha dado de alta el libro.
- **Valoraciones:** En esta tabla se almacena la información de las valoraciones. Es decir, si no has dado de alta un libro pero lo has leído, tu opinión y valoración se almacenan en esta tabla. Por el contrario, si has dado de alta el libro, tu opinión y valoración se almacenan en la tabla *Libros*.

## 3. ELEMENTOS CARACTERÍSTICOS

---

Antes de comenzar a analizar cómo se ha realizado las diferentes funcionalidades, quiero destacar algunas características, presentes en casi todas las páginas.

```
/* Una vez se tiene la clase para loguear creada, creamos la instancia
para poder acceder a la base de datos */
$dsn = "mysql:host=localhost;dbname=db09076204_pw1819;charset=utf8";
$usuario= "x09076204";
$password= "09076204";

/*Se realiza la conexión a la base de datos */
try {
    $conexion = new PDO( $dsn, $usuario, $password );
    $conexion->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );
} catch ( PDOException $e ) { echo "Conexión fallida: " . $e->getMessage(); }
```

Vamos a ver cómo se realiza la conexión a la base de datos para poder explicar con claridad los siguientes pasos. La conexión con la base de datos se realiza a través de especificar *dsn*,

*usuario* y *password*. Lo más significativo, es el atributo *charset=utf8* ya que esto permite que se almacenen las tildes y ñ en la base de datos y a la hora de mostrar cualquier valor, si contiene una tilde o ñ, lo muestre de forma correcta. Una vez explicado esto, se procede a explicar las características más importantes.

```
//Se crea la sesión.
session_start();

//Tenemos el DNI y vamos a realizar una función para conseguir
//los datos del usuario que nos hace falta para
//mostrar, de forma correcta, la página web.
$_SESSION["DNI"] = $_POST['usuario'];
buscarNombreApellidos($conexion);
```

La primera característica a destacar es cómo se crea la sesión del usuario y se guarda los datos del usuario en la variable `$_SESSION`. Cuando el usuario inicia sesión su DNI, almacenado en la variable `usuario` de la

variable `$_POST`. Una vez tenemos el DNI, se ejecuta la función `buscarNombreApellidos` que busca, en la tabla `Usuarios`, la información de la tupla que tenga como DNI, el DNI introducido por el usuario.

Una vez se tiene la sesión creada, la foto de perfil y el nombre se recupera de forma “dinámica”.

```

<p class="conectado"> <?php echo $_SESSION['NOMBRE']; " " .$_SESSION['APELLIDOS']; ?> </p>
```

La segunda característica a destacar es el control de sesiones, es decir, restringir el acceso a ciertas páginas a no ser que hayas iniciado sesión.

Como se puede ver a la izquierda, lo primero que se hace es iniciar una nueva sesión o reanudar la existente y se comprueba si el campo `$_SESSION['DNI']` está definido. Si no lo está, te devuelve al `index.html` para que inicies sesión.

```
<?php
//Iniciar una nueva sesión o reanudar la existente.
session_start();

if (!(isset($_SESSION['DNI']))) {
    header('Location: index.html');
    die() ;
}
?>
```

```
<?php
session_start();
$_SESSION = array();
session_destroy();
?>
```

Por último, todas las páginas tienen un enlace para desconectar la sesión. Ese enlace te redirecciona a una página exactamente igual a `index.html` pero llamada `logout.php`. En esta página, se reanuda la sesión, se vacía la variable `$_SESSION` y se destruye la sesión.

## 4. INDEX.HTML

---

```
function encriptarPassword() {
    var password = document.getElementById("password1").value.trim();
    var passwordCodificada = btoa(password);
    document.getElementById("password1").value = passwordCodificada;
}

function comprobarDNI() {
    var dni = document.getElementById("usuario1");
    //Se crea una expresión regular para comprobar si está compuesto por
    //8 número y una letra.
    var expresionRegular = /^[d{8}[a-zA-Z]$/;
    if (!(expresionRegular.test(dni.value)))
        alert ("Formato no válido del DNI. Recuerda que debe ser 8 números y una letra")
}

function comprobarPassword(){
    var password = document.getElementById("password1");

    //Lo primero que se comprueba es que no sea vacío.
    if((password.value.length < 8) || (password.value.length > 40)){
        alert("El campo contraseña debe tener más de 8 caracteres o menos de 40");
        password.style.borderColor = "red";
    }
}
```

Es la primera página que nos encontramos al entrar en el sitio web. Aquí se puede iniciar sesión o registrar un usuario. Para iniciar sesión, hace falta rellenar los campos *DNI* y *Contraseña*. Estos dos campos están validados a través de Javascript, como se puede ver a la izquierda, mediante un evento *onblur* (este evento actúa cuando el usuario abandona el campo

de texto). El DNI se comprueba a través de una expresión regular que comprueba si hay 8 números y una letra entre la A-Z, incluyendo minúsculas o mayúsculas. Si el formato del DNI no es válido, se muestra un *alert* recordando cuál es el formato correcto.

Con la contraseña pasa exactamente igual, se comprueba si es menor de 8 caracteres o mayor de 40. Si lo es, muestra un mensaje informando de cómo debe ser. Además, pone en rojo el color del borde del campo.

Por último, se puede ver una función llamada *encriptarPassword* que coge el valor de la contraseña, eliminando todos los espacios en blanco por la izquierda y por la derecha, le aplica la función *btoa* (codifica en base64) y devuelve la contraseña codificada en base 64.

## 5. ALTAUSUARIO.HTML

---

Se puede ver el formulario para introducir un usuario al sistema. Entre los campos a rellenar, se pueden ver DNI, Nombre, Apellidos, etc.

Este formulario está validado a través de Javascript con el evento *onblur*, anteriormente utilizado.

Entre las funciones de validación, se encuentran la función para encriptar la contraseña y comprobar el DNI, mencionados anteriormente, las cuales no voy a hacer hincapié otra vez ya que fueron ya han sido explicadas.

Además, hay una función llamada *comprobarNombre*, que es igual que la función *comprobarApellidos*. Por tanto, solo voy a tratar una de ellas.

```
function comprobarApellidos() {
    var apellidos = document.getElementById("apellidos-usuario");
    //Se comprueba que no sea más largo que 60 caracteres.
    if ((apellidos.value.length <= 60) && (apellidos.value.length >=10)){
        apellidos.style.borderColor = "green";
    }

    else {
        alert ("Los apellidos no pueden exceder los 60 caracteres y deben ser mayor a 10 caracteres");
        apellidos.style.borderColor = "red";
    }
}
```

Esta función muestra el color del borde verde o rojo conforme el formato de los apellidos sea correcta.

Esta función se ejecuta cuando se abandona el campo *Comprobar Contraseña*. Comprueba si ambas contraseñas son iguales, si no lo son, muestra un mensaje de que las contraseñas no coinciden. Si son iguales, pero sobrepasan la longitud, también hay un mensaje de error.

```
function comprobarPassword() {
    var password1= document.getElementById("contrasenia-usuario");
    var password2= document.getElementById("contrasenia-usuario2");

    //Se va a comprobar que tenga, como mínimo 8 caracteres y que sean iguales.
    if (password1.value == password2.value){
        if ((password1.value.length < 8) || (password1.value.length > 40)) {
            alert("Las contraseñas tienen que ser mayor a 8 caracteres por su seguridad o menor a 40 caracteres");
            password1.style.borderColor = "red";
            password2.style.borderColor = "red";
        }
        if ((password1.value.length >= 8) && (password1.value.length <= 40)) {
            password1.style.borderColor = "green";
            password2.style.borderColor = "green";
        }
    }

    else {
        alert ("Las contraseñas no coinciden");
        password1.style.borderColor = "red";
        password2.style.borderColor = "red";
    }
}
```

```
function comprobarEmail() {
    var email = document.getElementById("email-usuario");
    //Se realiza una expresión regular que compruebe que el email es válido.
    var expresionRegular = /^[?:[^<>()[]\.,;:\s@"]+(\.[^<>()[]\.,;:\s@"]+)*"["'"]?@(?:[^<>()[]\.,;:\s@"]+\.)(?:(?!(2,63)$)/i;
    if ((expresionRegular.test(email.value) && email.value.length<=50)){
        email.style.borderColor = "green";
    }

    else {
        alert ("El email no tiene un formato correcto o sobrepasa los 50 caracteres.");
        email.style.borderColor = "red";
    }
}
```

La función para comprobar el email utiliza una expresión regular que comprueba si hay un @ y un .

antes del final del correo electrónico.

## IMPORTANTE.

Antes de continuar, explicar que, aunque se seleccione una foto, no se va a subir y se va a establecer una foto por defecto. Esto es debido a que siempre que intentaba utilizar la función *move\_uploaded\_files* me salía un error de que no tenía permisos para realizar la acción. Por tanto, la funcionalidad de poder subir una foto no está validada.

## 6. ALTAUSUARIO.PHP

Esta página solo es visitada en el caso en el que el DNI que sea introducido en el formulario ya exista en la base de datos. Entonces, se muestra un mensaje de que el DNI ya está presente en otro usuario.

DATOS

FOTO:	<input type="button" value="Seleccionar archivo"/>	Ningún archivo seleccionado
DNI:	<input type="text"/>	El DNI introducido ya pertenece a otro usuario.
NOMBRE:	<input type="text" value="Daniel"/>	
APELLIDOS:	<input type="text" value="Terol Guerrero"/>	
CONTRASEÑA:	<input type="password" value="....."/>	
CONFIRMAR CONTRASEÑA:	<input type="password" value="....."/>	
CORREO ELECTRÓNICO:	<input type="text" value="danielterol9298@outlook.es"/>	

## 7. INDEX.PHP

Si el DNI que se ha utilizado en el formulario no coincide con ningún otro en la base de datos, se mostrará esta página que es de la misma apariencia que *index.html* y que contiene toda la funcionalidad de añadir un usuario a la base de datos.

```
/* Se crea una variable con el directorio donde se va a guardar la imagen.*/
function generarRuta ($post){

    $directorio = "/home/x09076204/public_html/bookrecsysII/imagenes/";
    $nombreImagen = $post['imagen'];

    //move_uploaded_file($_FILES['foto']['tmp_name'],$directorio.$nombreImagen)

    return("imagenes/foto-usuario-2.jpeg");

}

/* Antes de llamar al constructor, paso todos los valores del $_POST a un array.
En imagen se guarda el nombre de la imagen.*/
$post = array ("imagen" => $_FILES['foto']['name'],
               "dni" => $_POST['DNI'],
               "nombre" => $_POST['nombre'],
               "apellidos" => $_POST['apellidos'],
               "password" => $_POST ['password'],
               "email" => $_POST ['email']);

/* La imagen la convertimos a ruta*/
$post['imagen'] = generarRuta($post);

//Se crear una instancia de la clase.
$nuevoUsuario = new anadirUsuario ($post);
```

De forma inicial, se recogen los valores de la variable `$_POST` y se copian a una variable llamada `$post`, esto se realiza para una mayor comodidad a la hora de tratar con los datos del usuario.

Posteriormente, se genera la ruta de la imagen que, como se puede ver en el código, la función `move_uploaded_files` está comentada por el motivo mencionado anteriormente. La función `generarRuta` devuelve, por defecto, la misma imagen para todo el mundo.

Por último, se crea una instancia de la clase `anadirUsuario` que crea una instancia con toda la información del

usuario que, más tarde, será añadida a la base de datos. La clase `anadirUsuario` solo tiene un constructor y métodos de consulta además de un método para comprobar si el DNI introducido ya está presente en la base de datos.

```

/*Se comprueba si existe un DNI como el introducido, si no existe, se añade a la
if ($nuevoUsuario->existeUsuario($conexion)){

    //Como el DNI introducido ya existe, vamos a devolverle a la página de altausu
    //mostrándole un mensaje de que el DNI ya existe. Para poder mandar los datos
    //para que el usuario no los vuelva a introducir, se pasan mediante cookies.
    setcookie("nombre",$nuevoUsuario->getNombre());
    setcookie("apellidos",$nuevoUsuario->getApellidos());
    setcookie("password", $_POST['password']);
    setcookie("email",$nuevoUsuario->getEmail());
    header("Location: altausuario.php?");
}

else {
    $consultaSQL = "INSERT INTO Usuarios VALUES (?, ?, ?, ?, ?)";
    $prepared = $conexion->prepare($consultaSQL);
    $prepared->execute([$nuevoUsuario->getrutaImagen(), $nuevoUsuario->getDNI(),
        $nuevoUsuario->getNombre(), $nuevoUsuario->getApellidos(),
        $nuevoUsuario->getPassword(), $nuevoUsuario->getEmail()]);
}

```

Se comprueba si existe el DNI introducido, si existe, se crean cookies para poder mostrar en el formulario de *altausuario.php* los valores en el campo de texto.

Si no lo está, se inserta una tupla con los valores del usuario en la tabla *Usuarios*.

```

public function existeUsuario ($conexion){

    $dni = $this->getDNI();
    $existe = true;
    $consultaSQL = "SELECT * from Usuarios where dni= '$dni'";

    $resultados = $conexion->query($consultaSQL);

    if (!$resultados->rowCount()>0)
        $existe = false;

    return ($existe);
}

```

Para comprobar si existe el usuario, se realiza una consulta sobre la tabla *Usuarios* y se comprueba si el número de tuplas resultantes es mayor a 0 o no.

## 8. INDEX2-LOGIN.PHP

A esta página se accede al intentar iniciar sesión en el sitio web. Se comprueba si el usuario y contraseña, codificada, coinciden con alguna tupla almacenada en la base de datos. Si no coincidiera, a través de *header location*, se devuelve el usuario para que pueda volver a insertar sus credenciales.

```

function buscarNombreApellidos ($conexion){

    $dni = $_SESSION['DNI'];
    $consultaSQL = "SELECT rutaImagen, nombre, apellidos, email, password from Usuarios where dni = '$dni'";
    $resultados = $conexion->query($consultaSQL);

    foreach ($resultados as $fila){
        $rutaImagen = $fila['rutaImagen'];
        $nombre = $fila['nombre'];
        $apellidos = $fila['apellidos'];
        $email = $fila['email'];
        $password= $fila['password'];
    }

    $_SESSION["RUTAIMAGEN"] = $rutaImagen;
    $_SESSION["NOMBRE"] = $nombre;
    $_SESSION["APELLIDOS"] = $apellidos;
    $_SESSION["EMAIL"] = $email;
    $_SESSION["PASSWORD"] = $password;
}

```

La particularidad de esta página es que se crea la sesión y se almacena la información del usuario en el vector *\$\_SESSION*.

A la izquierda se puede ver la consulta que devuelve toda la información del usuario y la almacena en *\$\_SESSION*.



```
//Se busca los tres libros mejor valorados
$consultaSQL = "SELECT * from Libros ORDER BY valoracion DESC";
$resultados = $conexion->query($consultaSQL);

if ($resultados->rowCount() == 0) {
    echo '<article id="nohaylibro">
        <p id="nolibro"> No has leído ningún libro todavía </p>
    </article>';
}

else {
    $i = 0;
    foreach ($resultados as $fila){
        $i++;
        $isbn = $fila['isbn'];
        echo '<article class="libro2-3">
            
            <p class="titulo-libro"> <a href="libro.php?i='.$isbn.'">'. $fila['titulo']. ' </a></p>
            <p class="autor-libro">'. $fila['autor']. '</p>
        </article>
        <br>';
        if ($i == 3)
            break;
    }
}
```

Además, hay un *section* que presenta los libros mejor valorados del sistema. Esto es posible ya que hace una consulta a la tabla *Libros* ordenando los resultados por el atributo *valoración* en orden descendente. Es decir, de mayor a menor. Por último, solo se muestran los últimos tres libros.

Los libros poseen la misma foto por la misma razón, que cuando se registraba el usuario. Es decir, no tengo permisos para subir una foto.

## 9. LOGUEARUSUARIO.PHP

Esta página se muestra si, a la hora de iniciar sesión, se introducen las credenciales incorrectamente. La apariencia es idéntica a *index.html* e *index.php*. Por último, te informa que has introducido las credenciales mal a partir de un mensaje *alert*.

## 10. INDEX2.PHP

Es la misma página que *index2-login* pero sin ninguna interacción con la base de datos referida al inicio de sesión de un usuario, como había en *index2-login*.

## 11. DATOSPERSONALES.PHP

Contiene un formulario para poder modificar la información almacenada del usuario excepto el DNI. Tiene validación realizada a través de Javascript pero, al ser los mismo campos que cuando se registra un usuario, no se va a profundizar en ello. Lo único relevante en esta página es que, en el formulario, aparece la imagen del usuario y si se pasa el ratón por encima de la foto, aparecen los libros introducidos por el usuario.

El evento que es capaz de realizar esto es *onmouseover*.

```

```

```
$dni = $_SESSION['DNI'];
$consultaSQL = "SELECT titulo FROM Libros where dni='$dni'";
$resultados = $conexion->query($consultaSQL);
$libros = array();
foreach ($resultados as $fila){
    array_push($libros,$fila['titulo']);
}
```

La lista de libros del usuario se obtiene con la consulta de la izquierda. Para que Javascript pueda acceder a los libros, se tiene que crear un vector llamado *\$libros* donde se meten todos los libros que ha

leído el usuario.

```
function mostrarLibros(){
    var libros=<?php echo json_encode($libros);?>;
    var text="";
    var i;
    for (i = 0; i < libros.length; i++) {
        text += libros[i] + "\n";
    }

    if(text== "")
        alert("No has leído ningún libro todavía");
    else
        alert(text);
}
```

La función de Javascript es la siguiente. Se crea una variable *libros* y se inserta código php para poder asignar a la variable *libros* de Javascript, la variable *libros* de php. Además, esto no se puede realizar de forma automática, por tanto hay que pasar la variable *libros* de php a JSON siendo así Javascript capaz de interpretar la variable de forma correcta.

Si no hay ningún libro, se muestra un mensaje de que no ha leído libros todavía. Y si los hay, los muestra.

## 12. INDEX2-MODIFICAR DATOS.PHP

Se accede desde *datospersonales.php* y tiene toda la funcionalidad relacionada con actualizar la

```
public function actualizarUsuario ($conexion){
    $nombre = $this->getNombre();
    $apellidos = $this->getApellidos();
    $password = $this->getPassword();
    $email = $this->getEmail();
    $dni = $_SESSION['DNI'];

    $consultaSQL = "UPDATE Usuarios SET nombre= '$nombre', apellidos='$apellidos',
        password='$password', email='$email' where dni='$dni'";

    $resultados = $conexion->query($consultaSQL);
    $_SESSION["NOMBRE"] = $nombre;
    $_SESSION["APELLIDOS"] = $apellidos;
    $_SESSION["EMAIL"] = $email;
    $_SESSION["PASSWORD"] = $password;
}
```

información del usuario. De apariencia es exactamente igual a *index2.php*.

La consulta sería del tipo *UPDATE tabla SET*. Además, al haber actualizado los datos del usuario, se sobrescriben los valores del vector *\$\_SESSION*.

## 13. MISLIBROS.PHP

Contiene dos *section* diferenciados; 'Libros leídos' y 'Últimos libros añadidos'. Por las decisiones de

```
//Se busca los libros introducidos por el usuario.
$dni = $_SESSION["DNI"];
$consultaSQL = "SELECT * from Libros where dni= '$dni'";
$resultados = $conexion->query($consultaSQL);

if ($resultados->rowCount() == 0) {
    echo '<article id="nohaylibro">
        <p id="nolibro"> No has leído ningun libro todav&iacute;a </p>
    </article>';
}

else {
    foreach ($resultados as $fila){
        $isbn = $fila['isbn'];
        echo '<article class="libro2-3">
            
            <p class="titulo-libro"> <a href="libroleido.php?i='.$isbn.'">'. $fila['titulo']. ' </a></p>
            <p class="autor-libro">'. $fila['autor']. ' </p>
        </article>
        <br>';
    }
}
```

diseño que he tomado a lo largo del desarrollo de la página web, solo los libros que has añadido tú te salen en 'Libros leídos'. Si has valorado un libro, no va a salir en 'Libros leídos' sino que tu opinión aparecerá en el libro que hayas opinado.

A la izquierda se ve los libros, de 'Libros Leídos' que han sido introducidos

por el usuario. Si nos fijamos en la etiqueta *<a>*, vemos que el link está formado por

*libroleido.php?i=ISBN*. Esto es así para que, en *libroleido.php*, se pueda mostrar la información del libro de forma correcta.

A la derecha se puede ver cómo se muestran los libros de 'Últimos libros añadidos'. Además, el link de la página está formado por *libro.php?i=ISBN*, siguiendo el mismo procedimiento anteriormente mencionado.

Además de estos dos *section*, hay un enlace, que redirige a *altalibro.php*, para poder dar de alta nuestro libro. Una vez se da de alta, aparece en 'Libros leídos'.

```
//Se busca los libros introducidos por algún usuario
$dni = $_SESSION["DNI"];
$consultaSQL = "SELECT * from Libros where dni != '$dni'";
$resultados = $conexion->query($consultaSQL);

if ($resultados->rowCount() == 0) {
    echo '<p id="nolibro"> Nadie ha añtilde;adido ning&uacute;n libro todav&iacute;a </p>';
}

else {

    echo '<nav>
        <ul>';
    foreach ($resultados as $fila){
        $isbn = $fila['isbn'];
        echo '<li> <a href="libro.php?i='.$isbn.'">'.$fila['titulo'].' </a></li>
        <br>';
    }

    echo ' </ul>
        </nav>';
}
```

## 14. ALTALIBRO.PHP

Consiste en un formulario sobre los datos acerca del libro que queramos dar de alta. Tiene validación realizada a través de Javascript.

```
function comprobarISBN() {
    var isbn = document.getElementById("ISBN-libro");

    //Se crea una expresión regular para comprobar si está compuesto por
    //trece números.
    var expresionRegular = /^[0-9]*$/;
    if ((expresionRegular.test(isbn.value)) && (isbn.value.length == 13)){
        isbn.style.borderColor = "green";
    }

    else {
        alert ("Formato no válido de ISBN. Recuerda que el ISBN de un libro son 13 números");
        isbn.style.borderColor = "red";
    }
}
```

Para el caso del ISBN, se comprueba si los caracteres introducidos son numéricos y, además, si la longitud de la cadena introducida son 13 caracteres.

La comprobación del título, autor, editorial y edición son exactamente igual menos la condición de la longitud

mínima o máxima que ha de tener. Por ello, solo se va a mostrar la comprobación de la edición.

```
function comprobarEdicion() {
    var edicion = document.getElementById("edicion-libro");

    //Lo primero que se comprueba es que no sea vacío.
    if(edicion.value.length == 0){
        alert("El campo edición es obligatorio");
        edicion.style.borderColor = "red";
    }

    else {
        //Se comprueba que no sea más largo que 30 caracteres.
        if ((edicion.value.length <= 30)){
            edicion.style.borderColor = "green";
        }

        else {
            alert ("La edición no puede ser mayor a 30 caracteres");
            edicion.style.borderColor = "red";
        }
    }
}
```

La validación de la edición se realiza comprobando si la cadena introducida no es vacía o si la edición es menor a 30 caracteres.

El año se comprueba verificando que sea mayor o igual que 0 y menor o igual que 2019.

```
//Se va a comprobar que tenga, como mínimo 8 d
if ((anyo.value >= 0) && (anyo.value <= 2019))
    anyo.style.borderColor = "green";
```

OPINIÓN

1234 HOLA QUE TAL

17 /350

A la hora de introducir la opinión, hay un contador que muestra el número de caracteres (contando los espacios) hasta un máximo. Esto es posible debido a la siguiente estructura;

```
<label class="anadir-libro" for="opinion-libro">
  <textarea name="opinion" id="opinion-libro" rows="4" cols="75" maxlength="350" onkeyup="contarOpinion()" required></textarea>
  <span id="mi-opinion">0</span>
  <span>/350</span>
</label>
```

Se puede observar que, cada vez que se levante una tecla, evento *onkeyup*, salta la función *contarOpinion()*. Para entender la función *contarOpinion()* hay que fijarse en la etiqueta *<span>* con un identificador y su contenido es '0'. La función recoge la longitud del valor de la etiqueta *textarea*

```
function contarOpinion(){
  var caracteres = document.getElementById("opinion-libro").value.length;
  document.getElementById("mi-opinion").innerHTML = caracteres;
}
```

y sustituye la etiqueta *<span>*, mencionada anteriormente, por la longitud correspondiente.

Al igual que está implementado en la opinión, también lo está en la descripción.

## 15. ALTALIBRO-2.PHP

Se accede a esta página cuando, a la hora de dar de alta un libro, el ISBN coincide con otro existente en la base de datos. Contiene las mismas funcionalidades que *altalibro.php*. Simplemente muestra un mensaje de aviso de que el ISBN introducido ya existe en la base de datos.

## 16. NUEVOLIBRO.PHP

```
$consultaSQL = "INSERT INTO Libros VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
$prepared = $conexion->prepare($consultaSQL);
$prepared->execute([$nuevoLibro->getrutaImagen(), $nuevoLibro->getISBN(),
  $nuevoLibro->getDNI(), $nuevoLibro->getTitulo(),
  $nuevoLibro->getAutor(), $nuevoLibro->getEditorial(),
  $nuevoLibro->getAnyo(), $nuevoLibro->getEdicion(),
  $nuevoLibro->getDescripcion(), $nuevoLibro->getOpinion(),
  $nuevoLibro->getValoracion()]);
```

Contiene todas las funcionalidades para agregar un nuevo libro al sistema. Si el ISBN introducido existe en el sistema, devuelve a la página *altalibro-2.php*.

## 17. LIBROLEIDO.PHP

MI DESCRIPCION

Hobbits, orcos y anillos.

MI OPINIÓN

Gran libro.

MI VALORACIÓN

★★★★★

[¿Modificar opinión?](#)

Una vez se introduce un libro en el sistema, en el section 'Libros leídos' aparece un libro automáticamente. Si se pincha en ese libro, redirige a esta página en la que se muestra la información correspondiente al libro y la posibilidad de modificar la opinión y valoración del libro. El link de la

página se genera 'dinámicamente' agregando el ISBN del libro para poder extraer la información en la siguiente página

## 18. MODIFICARDATOS.PHP

OPINION

Opinion modificada.

19 / 350

VALORACIÓN

★ ★ ★ ★ ★

Contiene la información general del libro y la posibilidad de rellenar una opinión y valoración para modificar la que ya hay introducida en el sistema.

Además, contiene el ISBN de forma *hidden* para poder consultarla en la siguiente página y

actualizar, de forma correcta, la información del libro.

```
<!-- Se manda de forma hidden el ISBN del libro. -->
<input type="hidden" name="isbn" value="<?php echo $isbn;?>" />
```

## 19. ACTUALIZAROPINIONLIBRO.PHP



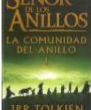
Contiene la funcionalidad para actualizar la información del libro que se recibe a través del vector `$_POST`. La información se actualiza con la siguiente consulta;

```
function actualizarOpinionLibro($isbn, $opinion, $valoracion, $conexion) {
    $consultaSQL = "UPDATE Libros SET opinion='$opinion', valoracion='$valoracion' WHERE isbn='$isbn'";
    $resultados = $conexion->query($consultaSQL);
}
```

## 20. LIBRO.PHP

Se accede a esta página clicando en algún libro de los últimos añadidos. Antes de comenzar a explicar código, esta página contiene un mecanismo de errores por si se diese una situación muy concreta.

### Libros mejor valorados

	<a href="#">EL SEÑOR DE LOS ANILLOS LA COMUNIDAD DEL ANILLO</a> J.K. ROWLING
	<a href="#">Crónicas Robóticas</a> Isaac Asimov
	<a href="#">EL SEÑOR DE LOS ANILLOS: LAS DOS TORRES</a> J.R.R. TOLKIEN

Recordemos que en páginas como *index2.php* y demás, hay un *section* que contiene los libros mejor valorados. Los enlaces de esos libros redirigen a esta página, por tanto, hace falta un mecanismo de errores por si un usuario, que ha sido el que ha dado de alta el libro, accede a esta página, no se muestre igual que se le muestra a los demás usuarios. Esto se consigue gracias a comprobar si existe alguna tupla en la tabla *Libros* que coincida el ISBN y el DNI.

```
if (esCreadordelLibro($_SESSION['DNI'], $isbn, $conexion))
    header('Location:libroleido.php?i='.$isbn);
```

Una vez explicado el mecanismo de errores, se procede a explicar la funcionalidad de la página;

DESCRIPCIÓN

Robots.

VALORACIÓN MEDIA:4

OPINIONES

Buen libro.

[¿Valorar libro?](#)

Si se observa la imagen de arriba, se puede ver que aparece la valoración media. Esto se consigue de la siguiente manera;

```
foreach ($resultados as $fila){  
  
    $opinion_creador = $fila['opinion'];  
    $valoracion_creador = $fila['valoracion'];  
    echo'  
    <h2>'. strtoupper($fila['titulo']). '</h2>  
    <article>  
          
        <p class="titulo-libro"> <b> TITULO:</b> ' . $fila['titulo']. ' </p>  
        <p class="autor-libro"> <b> AUTOR:</b> ' . $fila['autor']. ' </p>  
        <p class="autor-libro"> <b> EDITORIAL:</b> ' . $fila['editorial']. ' </p>  
        <p class="autor-libro"> <b> A&Ntilde;O:</b> ' . $fila['anyo']. ' </p>  
        <p class="autor-libro"> <b> EDICION:</b> ' . $fila['edicion']. ' </p>  
    </article>  
    <br><br>
```

A la hora de mostrar la información general, se accede a la tabla *Libros*, y se almacena la opinión del creador y la valoración que dio el creador cuando dio de alta el libro.

Entonces, se crea un vector de valoraciones y se introduce la valoración del creador.

Se realiza una consulta sobre la tabla *Valoraciones* para conseguir todas las valoraciones de aquellos usuarios que hayan valorado el libro en algún momento.

Una vez se tienen las tuplas, se recorre y se van añadiendo a nuestro vector de valoraciones. Se suman y se muestra dividiendo el resultado de la suma entre el número de valoraciones que hubiera.

```
$valoraciones = array();  
array_push($valoraciones, $valoracion_creador);  
$consultaSQL = "SELECT valoracion FROM Valoraciones WHERE isbn='$isbn'";  
$resultados = $conexion->query($consultaSQL);  
  
//Establezco esta igualdad aquí ya que si no hubiera valoraciones en Valora  
//no crearía la variable $valoracion_media. Por tanto, el for empieza en 1  
$valoracion_media = $valoracion_creador;  
  
foreach($resultados as $fila){  
    array_push($valoraciones, $fila['valoracion']);  
}  
  
//Se calcula la valoración media.  
for ($i=1; $i < count($valoraciones); ++$i){  
    $valoracion_media += $valoraciones[$i];  
}  
  
echo '  
    <article>  
        VALORACI&Oacute;N MEDIA: ' . ($valoracion_media)/(count($valoraciones)). '  
    </article>';
```

```
$consultaSQL = "SELECT opinion FROM Valoraciones WHERE isbn='$isbn'";
$opiniones = array();
array_push($opiniones, $opinion_creador);
$resultados = $conexion->query($consultaSQL);
foreach ($resultados as $fila){
    array_push($opiniones, $fila['opinion']);
}
```

Con las opiniones ocurre algo similar.

Se almacenan en un vector de opiniones para luego poder mostrarla como queremos.

```
<?php
$columnas = 1;
$pintar = true;
//Se pintan tantas filas como haga falta hasta que
//el número de opiniones sea igual al de columnas que hemos
//pintado.
for($i=0; $pintar;++$i){
    echo '<tr>';

    if(($columnas%4)==0){
        echo '<th>'. $opiniones[$columnas-1]. '</th>';
        ++$columnas;
    }

    for($columnas; (((($columnas)%4)!=0) && ($columnas <= count($opiniones)); ++$columnas)
        echo '<th>'. $opiniones[$columnas-1]. '</th>';

    if(($columnas-1) == count($opiniones))
        $pintar = false;

    echo '</tr>';
}
}>
```

Cuando se tienen todas las opiniones en un vector, se va creando las filas y columnas de una tabla de forma dinámica mostrando tres valoraciones por fila.

Cuando se llega a tres valoraciones por fila, se crea una nueva fila.

## 21. VALORARLIBRO.PHP

---

Se accede a esta página cuando se quiere valorar un libro que no hemos dado de alta nosotros. Contiene un formulario para introducir tu opinión y tu valoración respecto al libro.

## 22. DAROPINIONLIBRO.PHP

---

Cuando se manda el formulario de *valorarLibro*, dirige a esta página. De apariencia es idéntica a *index2.php* y similares. La única característica es que tiene que comprobar si la opinión del usuario ya estaba presente en el sistema o es la primera vez que da su opinión a ese libro. Esta comprobación es necesaria ya que como el usuario no ha dado de alta el libro, no ha tenido que introducir una opinión. Entonces puede darse el caso que sea la primera vez que opina o ya haya opinado.

```
if(buscarUsuarioOpinion($_POST['isbn'], $_SESSION['DNI'], $conexion))
    actualizarOpinionLibro($_POST['isbn'], $_SESSION['DNI'], $_POST['opinion'], $_POST['valoracion'],$conexion);
else
    darOpinionLibro($_POST['isbn'], $_SESSION['DNI'], $_POST['opinion'], $_POST['valoracion'],$conexion);
```