Programación Web. Práctica evaluable II

Sergio Vela Pelegrina

Índice

Resumen de mi práctica

Mi práctica ha sido realizada con el Modelo Vista Controlador. Por tanto, vamos a tener varios directorios:

- **Controllers** donde tendremos mis ficheros controladores.
- **Models** con los modelos.
- **Views** con las vistas.
- **Public** donde tendremos ficheros JavaScript, CSS e imágenes.
- **Libs** donde tenemos ficheros más "generales" y clases padres de las cuales heredarán otras clases.
- **Config** en el cual tenemos un fichero donde definiremos una serie de constantes que nos facilitarán el trabajo.

Fuera de los directorios, nos encontramos con dos ficheros, **.htaccess** en el cual podemos cambiar alguna configuración del servidor con una serie de reglas, como por ejemplo, que el fichero de partida de mi página sea el **index.php**, que es mi fichero principal.
Es un fichero en el que solo haremos el include del fichero en el cual tenemos nuestras constantes, de nuestra clase principal controladora, vista, modelo, de la base de datos y lo más importante, el fichero **app.php** que será el encargado de "centralizar" mi aplicación.
Tras los includes, lo único que hacemos es crear un objeto de mi clase **App**.

En el fichero **app.php**, tenemos, por tanto, la definición de la clase **App**, la cual tiene tres datos miembros:

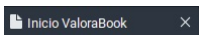
- $url. Tendremos almacenada la **URL** de nuestra página, en caso de que valga **NULL**, le asignaremos la URL de nuestra página principal, o sea, **main.php**.
- $archivoController. Tendrá la **URL** de nuestro **controlador**. En caso de que el fichero controlador exista, haremos un include de dicho fichero, la variable **controller** llamará a un método de la clase Controller llamado **loadModel**. En el cual crearemos el modelo asociado a cada controlador. También tenemos en cuenta el número de parámetros de nuestra URL.
  En caso de que no exixtiera el fichero controlador, llamamos a nuestra clase **CError** el cual nos informará de que dicha página no existe.

Mensaje de error al intentar acceder a una página que no existe.

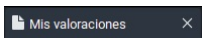
- $controller. Como hemos mencionado anteriormente, se encargará de cargar el model asociado a cada controller.

A modo de resumen de como funciona mi aplicación, tenemos una clase principal **controller.php** en la cual tendremos un objeto de tipo Vista y crearemos otro de tipo Modelo en caso de que la página requiera de un modelo (para acceso a la BD). En el fichero **model.php** tenemos un objeto de tipo dataBase que tendrá atributos tales como el nombre de la BD, del usuario, contraseña…

Por lo tanto, mis ficheros controladores heredan de la clase Controller. Tenemos el método **render** para llamar a la vista asociada de cada controlador y mandaremos un mensaje especificado en el controlador en el que estamos, con el cual podemos hacer cosas como cambiar el título de la pestaña en la que estemos como podemos ver a continuación:



Pestaña de inicio



Pestaña de mis valoraciones.

Esto lo gestiono con un Switch como podemos ver:

```
$titulo = "";

switch($this->getVista()){
    case 'main':
        $titulo = "Inicio ValoraBook";
    break;
```

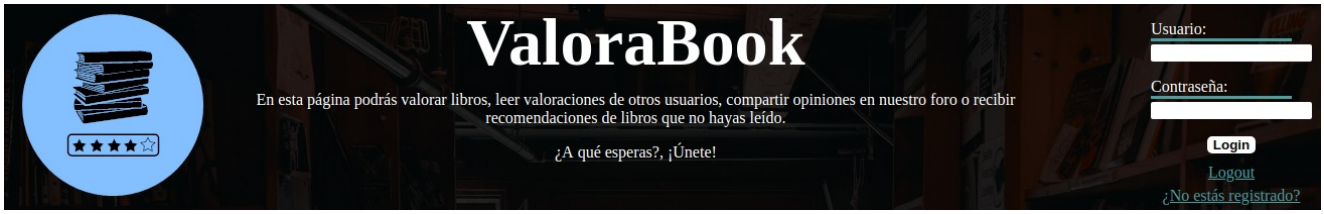
```
        case 'cerror':
            $titulo = "PÁGINA DE ERROR";
        break;

        case 'mislibros':
            $titulo = "Mis libros";
        break;
        .
        .
        .
       ...
```

Los controladores que dispongan de un modelo asociado, tendrán métodos con los cuales acceder a la base de datos para insertar, eliminar, modificar o eliminar registros.
Dicho métodos se explicarán más adelante.

De las vistas hay poco que mencionar, básicamente en ellas encontraremos HTML y alguna instrucción que otra en **php**.
Un detalle a destacar es que tengo un fichero para el menú de navegación que en función de la página en la que me encuentre, nos la marcará con un color distinto como podemos ver:

A partir de los formularios que tenemos de la práctica anterior y **tal y como se especifica en el guión de la práctica**, mi aplicación permite que los usuarios se puedan registrar y añadir valoraciones de libros.
Al iniciar sesión, esta, se mantiene activa hasta que el usuario pulsa sobre el botón de **logout**.
La ventaja de estar logeado será poder visualizar el menú de navegación y poder añadir valoraciones.
En la esquina superior derecha, aparece el nombre de usuario de la persona.

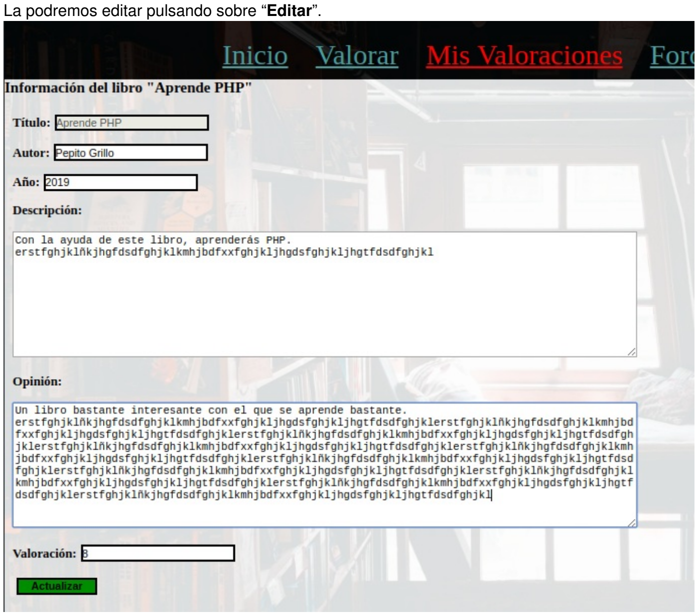
Si pulsamos sobre **logout**, nos desaparecerá el menú de navegación:



El usuario podrá crear la valoración del libro que quiera:

Dicha valoración se podrá visualizar en el menú **Mis valoraciones**:

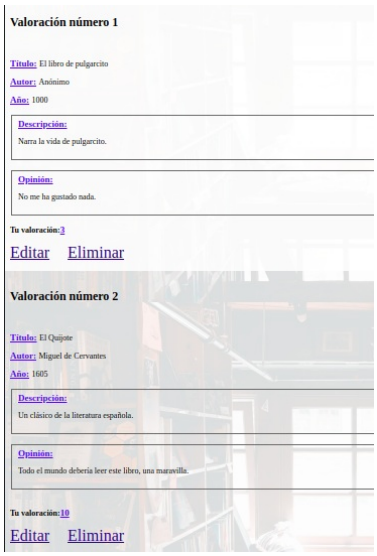
La podremos editar pulsando sobre "**Editar**".



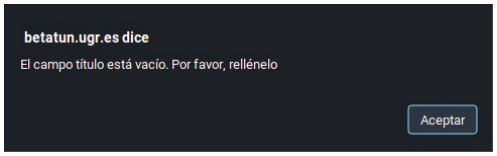
Y dicha edición la podremos ver, nuevamente en la página **Mis valoraciones**

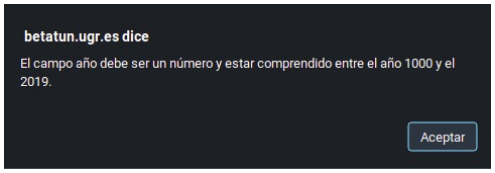
Si pulsamos sobre **Eliminar** no volveremos a ver dicha valoración.

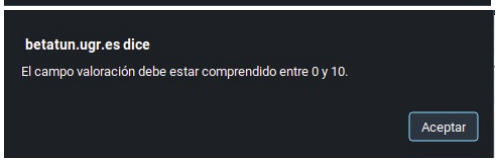
Podremos visuaizar todas las valoraciones que hay en el sistema:

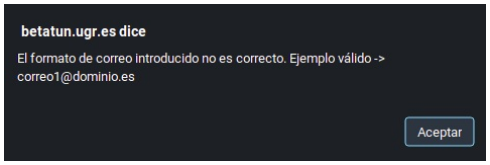


Todos los formularios los valoramos con JavaScript, mostrando alertas cuando algo no se haya introducido correctamente, como por ejemplo:









Innovación añadida

- A la hora de modificar las valoraciones, el campo **título** será el único **NO** modificable, puesto que es la clave primaria en mi BD, el problema, es que como sabemos, desde HTML lo podríamos modificar y es algo que no nos podemos permitir, esta es la razón por la cual, al modificar una valoración, el campo **título** lo captamos a partir de la sesión, de forma que aunque se modifique con HTML, dicho cambio no tendrá efecto.
  Aquí tenemos las funciones con las cuales gestiono lo mencionado:

```
public function verValoracion($param = null){
            $idVal = $param[0]; //Por muchos parámetras que tengamos, solo nos interesa el primero
            $valoration = $this->model->getByID($idVal);

            //Evitamos que puedan modificar el título del libro, uso de sesión
            session_start();
            $_SESSION['id_verValoracion'] = $valoration->titulo;

            $this->view->mensaje = "";
            $this->view->valoration = $valoration;

            $this->view->render('misvaloraciones/detalle');
        }

public function actualizarValoracion(){
            session_start();
            //Aquí nos metemos.
            $titulo = $_SESSION['id_verValoracion'];
            $autor = $_POST['autor'];
            $anio = $_POST['anio'];
            $descripcion = $_POST['descripcion'];
            $opinion = $_POST['opinion'];
            $valoracion = $_POST['valoracion'];

            unset($_SESSION['id_verValoracion']);
}
```

Como podemos ver, tras modificar la valoración, cerramos la sesión.

- **Uso del Modelo Vista Controlador**. Este seguramente sea el detalle más a destacar, ya que de primeras puede resultar difícil la gestión de la aplicación, pero después resulta una forma mucho más legible y organizada. El uso de clases y de orientación a objetos facilita la gestión de nuestra página.
- **Fichero .htaccess**. Con este fichero hemos modificado algunas características del servidor, como por ejemplo que el primer archivo que se cargue sea el **index.php**, con el uso de la regla `RewriteRule ^(.*)$ index.php?url=$1 [L,QSA]` .
- **Rutas relativas de todos mis ficheros y variables**. El uso de un fichero para definir constantes me ha permitido definir rutas relativas, de forma que la portabilidad de mi código a otro servidor se podría hacer de forma rápida únicamente modificando dichas constantes, a modo de ejemplo, veamos lo siguiente:

  `<link rel="stylesheet" href="<?php echo constant('URL');?>public/css/estilo.css">`

  Aquí especificamos el lugar de mi fichero CSS. La constante URL mientras trabaja en local, tenía el valor `http://localhost/` mientras que para subir la práctica al servidor de la asignatura cambié el valor de la variable por `http://betatun.ugr.es/~xMiDNI/bookrecsysII/` .
  De no haber hecho uso de dichas constantes, habría tenido que cambiar la URL de cada foto, fichero o variable a la que hagamos referencia.
- **Ocultación de la extensión del fichero**. Al usuario, no le interesa saber si la página en la que se encuentra es HTML, PHP o lo que sea, por eso en la URL no muestro dicha información:

  betatun.ugr.es/~x75933698/bookrecsysII/mislibros

  betatun.ugr.es/~x75933698/bookrecsysII/misvaloraciones
- **Intercambio el color de fondo de cada valoración** con el fin de que sea más fácil visualizar el fin de una valoración con el comienzo de otra. El código que he implementado es el siguiente:

```
<?php
  $color = true;
  $contador = 1;
  foreach($this->vals as $row){
    $v = new Valoraciones();
    $v = $row;
    if($color){
      $color=false;
?>
```

```
<main class="main-pedidos">
      <form class="formulario-busqueda" action="formulario.php" method="post">
        <br>
        <h2> Valoración número <?php echo $contador; $contador++; ?> </h2>
          <p><span class="preguntas-formulario"></span> <span></span></p>
          <p><span class="preguntas-formulario">Título:</span> <span><?php echo $v->titulo; ?></span></p>
          <p><span class="preguntas-formulario">Autor:</span> <span><?php echo $v->autor; ?></span></p>
          <p><span class="preguntas-formulario">Año:</span> <span><?php echo $v->anio; ?></span></p>
          <fieldset>
            <p><span class="preguntas-formulario">Descripción:</span></p>
            <p class="parrafo"><?php echo $v->descripcion; ?></p>
          </fieldset>
          <fieldset>
            <p><span class="preguntas-formulario">Opinión:</span></p>
            <p class="parrafo"><?php echo $v->opinion; ?></p>
          </fieldset>
            <br>
            <label>Tu valoración:</label><span class="preguntas-formulario"><?php echo $v->valoracion; ?></span>
              <br><br>
              <a class="edit" href="<?php echo constant('URL') . 'misvaloraciones/verValoracion/' . $v->titulo; ?>">Editar</a>
              <a class="edit" href="<?php echo constant('URL') . 'misvaloraciones/eliminarValoracion/' . $v->titulo; ?>">Eliminar</a>
            <br><br>
      </form>
    </main>

    <?php }else{
      $color=true;
    ?>

      <form class="formulario-busqueda" action="formulario.php" method="post">
      <br>
      <h2> Valoración número <?php echo $contador; $contador++; ?> </h2>
          <p><span class="preguntas-formulario"></span> <span></span></p>
          <p><span class="preguntas-formulario">Título:</span> <span><?php echo $v->titulo; ?></span></p>
          <p><span class="preguntas-formulario">Autor:</span> <span><?php echo $v->autor; ?></span></p>
          <p><span class="preguntas-formulario">Año:</span> <span><?php echo $v->anio; ?></span></p>
          <fieldset>
            <p><span class="preguntas-formulario">Descripción:</span></p>
            <p class="parrafo"><?php echo $v->descripcion; ?></p>
          </fieldset>
          <fieldset>
            <p><span class="preguntas-formulario">Opinión:</span></p>
            <p class="parrafo"><?php echo $v->opinion; ?></p>
          </fieldset>
          <br>
              <br>
              <label>Tu valoración:</label><span class="preguntas-formulario"><?php echo $v->valoracion; ?></span>
              <br>
              <br>
              <a class="edit" href="<?php echo constant('URL') . 'misvaloraciones/verValoracion/' . $v->titulo; ?>">Editar</a>
              <a class="edit" href="<?php echo constant('URL') . 'misvaloraciones/eliminarValoracion/' . $v->titulo; ?>">Eliminar</a>
              <br><br>
      </form>
      <?php }
              }?>
    </main>
```

- **Uso de herencia con php** con el fin de gestionar mi aplicación de forma más sencilla y “elegante”.
- **Uso de funciones de php** como **rtrim** para quitar caracteres de un string o **explode** para dividir un string.

Conclusión

Como conclusión de esta práctica, he aprendido dos lenguajes fundamentales en el ámbito de la programación web, PHP y JavaScript.
Quizás PHP de primeras sea un poco engorroso, pero dota de utilizad y funcionalidad los formularios de nuestra página web. Dicha página se genera de forma dinámica y nos ahorramos el “copy-paste” de HTML como en la anterior práctica.
JavaScript es un lenguaje un tanto particular pero nos permite validar formularios, y mejorar la interfaz de usuario y dinamismo de una forma muy sencilla.

Definitivamente, esta práctica me ha resultdo muy útil y he aprendido mucho.