Tarea DWES03:

Aplicación de gestión de productos con BBDD

Christian Miguel Giménez Puerma - 2ºDAW

Sumario

1. Explicación previa

Para la resolución de esta tarea, en orden de cumplir todos los requerimientos del cliente, la aplicación se ha modularizado en los siguientes 5 ficheros PHP, **todo con tratamiento de errores con excepciones:**

- **listado.php**
- **editar.php**
- **actualizar.php**
- **db_conector.php**
- **db_utils.php**

En los primeros 3 ficheros se realizaron las labores requeridas junto a las funcionalidades CRUD de los ficheros db_conector.php y db_utils.php, todo ello se desglosará y explicará en los apartados siguientes.

En los ficheros de la capa de presentación y a su vez de la capa de la lógica de negocio (listado.php, editar.php y actualizar.php) se han incluido las siguientes líneas. Las cuales son la importación obligatoria de los ficheros necesarios para la interacción con la capa de modelo de datos, de forma que se realice la correcta consulta, modificación y persistencia de los datos:

```php
require_once "./db_conector.php";
require_once "./db_utils.php";

//Debugg
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
```

También se añadieron líneas para que PHP nos muestre los errores, ya que por defecto están deshabilitados.

2. Fichero ‘listado.php’

El fichero **listado.php** se ocupa primero de revisar si ya se ha mandado anteriormente una consulta seleccionando una familia, tratando los errores en el proceso:

```php
//Si no hay ningún id seleccionado (null) introducimos una string vacía para evitar errores
try {
    $idSeleccionado = $_GET["familias"] ?? "";
} catch (Exception $e) {
    //Lanzamos la excepción si el id es null
    echo "El id seleccionado no puede ser null. Mensaje error: " . $e->getMessage();
}
```

Luego muestra la vista al usuario y genera de forma dinámica un formulario con el método **mostrarFamilias()** de db_utils.php, que consultará a la base de datos para mostrarlos.

```php
<div id="encabezado">
  <h1>Tarea: Listado de productos de una familia </h1>
  <!-- Formulario select generado dinámicamente para mostrar todas las familias de la BBDD-->
  <form id="form_seleccion" action="<?php echo htmlspecialchars($_SERVER['PHP_SELF']); ?>" method="get">
    <label for="familias">Familia:</label>
    <select name="familias">
      <!-- Función que genera los <option> para cada familia-->
      <?php
      try {
        mostrarFamilias($db, $idSeleccionado);
      } catch (Exception $e) {
        echo "Hubo un error al mostrar las familias: " . $e->getMessage();
      }
      ?>
    </select>
    <input type="submit" name="mostrar_productos" value="Mostrar productos" />
  </form>
</div>
```

Por último se muestran los productos de la familia seleccionada en el formulario anterior, si no hubiera productos en esa familia, también se le mostrará al usuario la ausencia de ellos:

```php
<div id="contenido">
  <h2>Productos de la familia:</h2>
  <?php
  //Si hubo alguna familia seleccionada en el formulario anterior, mostramos productos de esa familia
  if ($idSeleccionado !== "") {
    mostrarProductosFamilia($db, $idSeleccionado);
  }
  ?>
</div>
```

3. Fichero ‘editar.php’

En el fichero **editar.php** se mostrará sólo si el usuario le dió al botón de Editar en un producto mostrado en listado.php.

Hecho esto, se decodifica el producto llegado por POST y se muestra dinámicamente el mismo con el método **editarProducto($producto)** del fichero db_utils.php, en un formulario para poder editar los datos del mismo.

```php
<div id="contenido">
  <h2>Producto:</h2>
  <?php
  //Si se le dió a "editar" en algún producto
  if (isset($_POST["editar"])) {
    //des-serializamos el objeto producto
    $producto = json_decode($_POST["producto"]);
    //Y mostramos el formulario con cada campo editable del mismo
    try {
      editarProducto($producto);
    } catch (Exception $e) {
      //Cazamos la excepción si no se pudo editar el producto
      echo "Hubo un error al editar el producto. Mensaje error: " . $e->getMessage();
    }
  }
  ?>
```

4. Fichero ‘actualizar.php’

El fichero **actualizar.php** mostrará la correcta actualización o cancelación de la modificación del producto realizado en editar.php.

Primero se comprueba si el usuario le dio al botón “Actualizar”, si es así se comprueban y almacenan todos los datos del producto en distintas variables para llamar al método **actualizarProducto()** de db_utils.php para realizar la consulta a la BBDD, si la consulta fue exitosa, se le mostrará desde ese método al usuario la correcta actualización.

Si en cambio el usuario le dio al botón “Cancelar”, no se mandará la consulta a la base de datos, sólo se le mostrará al usuario que se canceló la actualización del producto tal y como él lo ha querido.

```php
<?php
//Si le dió al botón de "Actualizar"
if (isset($_POST["actualizar"])) {
  //Almacenamos todos los datos del formulario limpiando los datos
  try {
    $cod = trim($_POST["cod"]) ?? "";
    $nombre = trim($_POST["nombre"]) ?? "";
    $nombreCorto = trim($_POST["nombre_corto"]) ?? "";
    $descripcion = trim($_POST["descripcion"]) ?? "";
    $pvp = trim($_POST["PVP"]) ?? "";
    //Y actualizamos la base de datos con un UPDATE
    actualizarProducto($db, $cod, $nombre, $nombreCorto, $descripcion, $pvp);
  } catch (Exception $e) {
    echo "Hubo un error a la hora de actualizar el producto. Mensaje error: " . $e->getMessage();
  }
} else if (isset($_POST["cancelar"])) {
  //Si le dió al botón de "Cancelar", mostramos que no se actualizó tal y como pidió el cliente
?>
  <div id="encabezado">
    <h1>Ejercicio: Cancelación del proceso de actualización</h1>
  </div>

  <div id="contenido">
    <h2>Se canceló la actualización del producto.</h2>
    <!-- Botón para regresar a listado.php-->
    <button><a href="<?= htmlspecialchars("./listado.php"); ?>">Continuar<a /></button>
  </div>
<?php
} else {
  throw new Exception("Hubo un error en el envío de los datos a actualizar.php");
}
?>
```

5. Fichero ‘db_conector.php’

El fichero **db_conector.php** se ocupá de almacenar las credenciales de la base de datos, así como de establecer la codificación UTF-8 de las consultas que se realizan a la misma, y de instanciar el objeto perteneciente a la clase PDO, el cual tiene la funcionalidad de poder hacer consultas a la base de datos indicada.

```php
<?php
//Constantes para almacenar las credenciales de la BBDD
const DATABASE = "dwes";
const USER = "dwes";
const PASSWORD = "abc123.";
const HOST = "localhost";
//Codificación UTF-8 para la instancia de PDO
$utf8 = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8");
try {
  //Instanciamos el objeto PDO indicando el tipo de SGBD (mysql), el nombre del host
  //el nombre de la base de datos, el usuario que la administra, su contraseña y la codificación de caracteres
$db = new PDO("mysql:host=" . HOST . ";dbname=" . DATABASE, USER, PASSWORD, $utf8);
  $db->getAttribute(PDO::ATTR_SERVER_VERSION);
} catch (Exception $e) {
  //Si hubo un error en la conexión con la base de datos, mostramos la excepción
  echo "Conexión fallida con la base de datos 'dwes'.";
}
```

6. Fichero ‘db_utils.php’

Como se ha comentado anteriormente, el fichero **db_utils.php** contiene una serie de métodos que realizarán las consultas a la base de datos.

mostrarFamilias()

Este método se ocupa de preguntar a la base de datos por el listado de las familias que haya en la base de datos, de forma que el usuario pueda seleccionar alguna y mandar esa información posteriormente al método mostrarProductosFamilia().

```php
//Función para mostrar las familias en los <option> del <select>
function mostrarFamilias($db, $optionSelec)
{
  //Preparamos la consulta obteneindo todos de la familia
  $query = $db->prepare("SELECT * FROM `familia`;");
  //Ejecutamos la consulta
  $exito = $query->execute();
  if ($exito) {
    //Transformamos cada fila del resultado en un objeto
    while ($fila = $query->fetch(PDO::FETCH_OBJ)) {
      //Si el código del producto = al seleccionado
      if ($fila->cod === $optionSelec) {
        //Hacemos que se quede marcado ese producto en el <select>
        echo "<option value='$fila->cod' selected>$fila->nombre</option>";
      } else {
        //Si no, simplemente creamos el <select>
        echo "<option value='$fila->cod'>$fila->nombre</option>";
      }
    }
  } else {
    //Si no se pudo ejecutar la consulta, lanzamos la excepción
    throw new Exception("No se pudo ejecutar la consulta 'SELECT * FROM `familia`;'. Opción seleccionada: $optionSelec");
  }
}
```

mostrarProductosFamilia()

Este método se ocupa de preguntar a la base de datos por el listado de productos de la familia que haya seleccionado el usuario en el formulario anterior, dándo la opción de poder editar alguno de ellos con el botón “Editar”, mandando así toda la información del producto seleccionado a través de un input oculto, para ser tratado posteriormente por el método editarProducto().

```php
function mostrarProductosFamilia($db, $seleccionado)
{
  //Preparamos la consulta, seleccionando todos los datos de los productos de la familia indicada
  //ejemplo: SELECT nombre_corto, PVP FROM producto WHERE familia = "NETBOK";
  $query = $db->prepare("SELECT * FROM producto WHERE familia = :idFamilia;");
  //Ejecutamos la consulta, pasando a PDO los argumentos evitando SQL Injection
  $exito = $query->execute([":idFamilia" => $seleccionado]);

  //Contamos el número de filas de la consulta para mostrar si hay o no productos
  $numFilas = $query->rowCount();

  //Si la consulta tuvo éxito sin errores
  if ($exito) {
    //Si la consulta devolvió productos
    if ($numFilas > 0) {
      //Transformamos cada fila de la consulta en un objeto
      while ($fila = $query->fetch(PDO::FETCH_OBJ)) {
?>
        <!-- Formulario para que el usuario pueda editar el producto deseado-->
        <form method='post' action='<?php echo htmlspecialchars("./editar.php"); ?>'>
          <label for="<?php echo "producto"; ?>">
            <?php echo "Producto: $fila->nombre_corto. Precio: $fila->PVP." ?>
          </label>
          <!-- Guardamos todo el objeto $producto para no necesitar hacer otra consulta a la BBDD a la hora de modificarlo-->
          <input type="hidden" name="<?php echo "producto"; ?>" value="<?php echo htmlspecialchars(json_encode($fila)); ?>" />
          <input type='submit' name='editar' value='Editar' />
        </form>
    <?php
      }
    } else {
      //Si la consulta no devolvió ningún producto, se lo informamos al usuario
      echo "<p>No hay productos dados de alta en esta famila</p>";
    }
  } else {
    //Lanzamos la excepción si hubiera un error en la consulta
    throw new Exception("No se pudo ejecutar la consulta: 'SELECT * FROM producto WHERE familia = $seleccionado'");
  }
}
```

editarProducto()

Método que tras recibir el objeto producto seleccionado anteriormente, se ocupa de imprimir en un formulario editable el nombre, descripción, precio y nombre largo del producto, de forma que el usuario puede modificar estos datos. Si el usuario decide hacer efectiva la modificación, podrá darle al botón “Actualizar”, si no es así, podrá darle al botón “Cancelar”, de forma que todo se gestionará correctamente mandando la información desglosada por POST al método actualizarProducto().

```php
//Función para mostrar los datos del producto modificables a través de un formulario
function editarProducto($producto)
{
  if (isset($producto)) {
    ?>
    <!-- Formulario rellenado con los datos ya existentes en la BBDD-->
    <form method='post' action='<?php echo htmlspecialchars("./actualizar.php"); ?>'>
      <label for='nombre_corto'>Nombre corto:</label>
      <input type='text' name='nombre_corto' id='nombre_corto' value='<?= trim($producto->nombre_corto); ?>' />
      <br /><br />
      <label for='nombre'>Nombre:</label>
      <br /><br />
      <textarea name='nombre'>
      <?php
      //Si el nombre no fuera null, lo mostramos
      if ($producto->nombre !== null) {
        echo trim($producto->nombre);
      }
      ?>
    </textarea>
      <br /><br />
      <label for='descripcion'>Descripción:</label>
      <br /><br />
      <textarea name='descripcion' id='descripcion'>
      <?php
      //Si la descripción del producto no fuera null, la mostramos
      if ($producto->descripcion !== null) {
        echo trim($producto->descripcion);
      }

      ?>
    </textarea>
      <br /><br />
      <label for='PVP'>PVP:</label>
      <input name='PVP' id='PVP' type="number" min="0" step="any" value='<?= $producto->PVP; ?>' />
      <br /><br />
      <!-- Enviamos el código del producto para poder posteriormente actualizarlo si el usuario así lo desea-->
      <input type="hidden" id="cod" name="cod" value="<?= htmlspecialchars($producto->cod); ?>" />
      <!-- Botón para guardar los cambios efectuados-->
      <input type="submit" id='actualizar' name='actualizar' value="Actualizar" />
      <!-- Botón para cancelar los cambios-->
      <input type="submit" id='cancelar' name='cancelar' value="Cancelar" />
    </form>
    <?php
  } else {
    //Si hubo un error al mostrar los datos del producto, lanzamos la excepción
    throw new Exception("Hubo un error al mostrar los datos del producto para poder editarlo. Producto: $producto
  }
}
```

actualizarProducto()

Este método, si el usuario pulsó anteriormente en “Actualizar”, se ocupará de hacer la consulta a la base de datos para actualizar el registro de ese producto, mostrando al usuario la actualización realizada con éxito y un botón “Continuar” para volver de nuevo a listado.php, si en cambio el usuario le dio a “Cancelar”, se le mostrará la cancelación de la actualización y el botón “Continuar” para redirigirle también a listado.php. Si hubiera un error, igual que en el resto de métodos, se lanzará una excepción, en este caso la excepción mostrará explícitamente todos los datos del producto que le llegó por argumento, de manera que el desarrollador pueda debugguear la excepción más cómodamente.

```php
//Función para actualizar el producto que fue modificado en editar.php en la BBDD
function actualizarProducto($db, $cod, $nombre, $nombreCorto, $descripcion, $pvp)
{
  //Si el producto tiene código, lo modificamos, si no no, así evitamos consultas inválidas
  if ($cod !== "") {
    //Preparamos la consulta para evitar SQL Injection, modificando cada campo donde el id coincida con el seleccionado
    $query = $db->prepare("UPDATE producto SET nombre = :nombre, nombre_corto = :nombre_corto,
    descripcion = :descripcion, PVP = :pvp WHERE cod = :cod;");
    //Ejecutamos la consulta, pasándole a PDO los valores a las variables del array asociativo
    $exito = $query->execute([
      ":nombre" => $nombre,
      ":nombre_corto" => $nombreCorto,
      ":descripcion" => $descripcion,
      ":pvp" => $pvp,
      ":cod" => $cod
    ]);

    //Si la consulta se realizó con éxito, informamos al usuario
    if ($exito) {
    ?>
      <div id="encabezado">
        <h1>Ejercicio: Actualización correcta del producto</h1>
      </div>

      <div id="contenido">
        <h2>Se han actualizado los datos.</h2>
        <!-- Botón para regresar a listado.php-->
        <button><a href="<?= htmlspecialchars("./listado.php"); ?>">Continuar<a /></button>
      </div>
<?php
    } else {
      //Lanzamos la excepción si la consulta falló
      throw new Exception(
        //Si la consulta falló, mostramos los datos para revisar si hubo algún fallo en la ejecución
        "<p class='error'>Ups... Ha habido un error al ejecutar el UPDATE. Datos:</p>
      cod: " . $cod . "<br/>
      nombre: " . $nombre . "<br/>
      nombre_corto: " . $nombreCorto . "<br/>
      descripcion: " . $descripcion . "<br/>
      pvp: " . $pvp . "<br/>
      "
      );
    }
  } else {
    //Lanzamos la excepción si el código del producto llegó vacío tras el formulario
    throw new Exception("El código del producto no puede estar vacío.");
  }
}
```