4

Ejemplo de aplicación completa en PHP

Objetivos

- ✓ Crear una aplicación completa de pedidos.
- ✓ Cargar dinámicamente las categorías y productos disponibles.
- ✓ Controlar el estado del pedido mediante una variable de sesión.
- ✓ Almacenar pedidos en la base de datos.
- ✓ Controlar el acceso con una tabla de usuarios.
- ✓ Enviar correos de confirmación.

Mapa conceptual

Glosario

Carrito de la compra. Abstracción habitual en las tiendas online. En el carrito se almacenan los productos que el cliente va escogiendo antes de realizar el pedido.

Diseño físico de la base de datos. Paso final para crear la base de datos. Se definen los tipos de datos e índices. Se pueden realizar cambios sobre el diseño lógico para optimizar el rendimiento.

Diseño lógico de la base de datos. En el modelo relacional consiste en obtener el conjunto de tablas que formarán la base de datos. Habitualmente se parte de un esquema ER o un diagrama de clases.

Especificación de requisitos de software. Descripción detallada de la aplicación que se va a realizar.

Esquema ER. Representación gráfica del esquema de datos de una aplicación. Es el primer paso para desarrollar una base de datos.

Mapa de flujo de pantallas. Diagrama que representa las vistas disponibles para el usuario y cómo se relacionan.

4.1. Definición del proyecto

En este capítulo se desarrolla una aplicación web para realizar pedidos, similar a una tienda web. Se pretende:

- Unir los elementos de los capítulos anteriores para plantear una aplicación web completa.
- Mostrar cómo plantear un proyecto de aplicación web con base de datos desde cero siguiendo una metodología apropiada.

Es un proyecto pequeño pero completo. Se desarrollan las fases de análisis, diseño e implementación. Se puede considerar como la primera iteración dentro de un proyecto más completo.

Actividad propuesta 4.1

Si no lo conoces, busca información sobre el modelo de desarrollo en espiral.

En el análisis, se define la funcionalidad de la aplicación y sus limitaciones. Se detallan los datos que se quieren almacenar y se realiza un esquema entidad relación para la base de datos.

El diseño es la parte más importante del capítulo, se definen:

- Las pantallas que verá el usuario.
- Los ficheros que formarán la aplicación y cómo se pasarán los parámetros entre ellos.
- La estructura de datos para el carrito de la compra y cómo manipularla, uno de los puntos más complicados de la aplicación.
- La base de datos.

Finalmente, en la implementación, se escriben los ficheros de la aplicación. La parte relacionada con la base de datos es la más larga.

Como la base de datos es un elemento fundamental para la aplicación, para poder entender bien el capítulo es necesario recordar los elementos básicos del modelo entidad-relación y, sobre todo, del modelo relacional y del SQL.

La aplicación es básicamente una tienda web sencilla. Se espera que tenga la funcionalidad habitual en una tienda online. La principal diferencia está en que, como los restaurantes son de la misma empresa, el pedido no requiere pago. Tampoco es necesario especificar la dirección de envío. La empresa sabe dónde está cada restaurante.

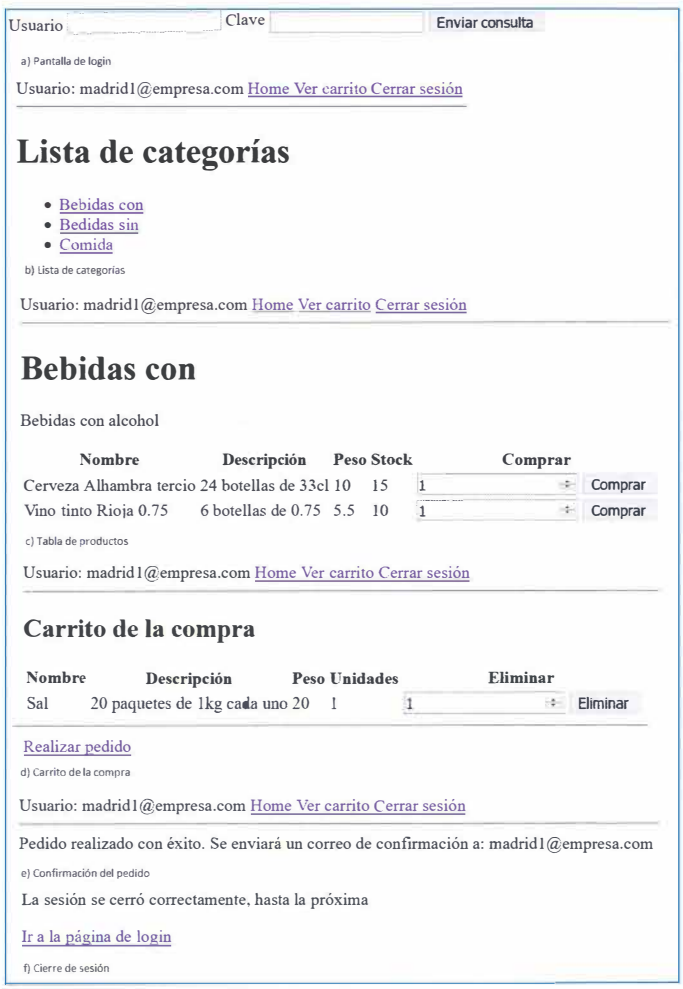

Figura 4.1
Resultado final de la aplicación.

RECUERDA

✓ Asegúrate de haber importado la base de datos pedidos, como se explica en el capítulo 1, y de arrancar la base de datos.

4.2. Análisis de requisitos

Se quiere realizar una aplicación para el Departamento de Pedidos para una cadena de restaurantes. Los restaurantes de la cadena utilizarán la aplicación web para realizar pedidos de comida, bebida y materiales.

La aplicación debe permitir:

- Consultar las categorías.
- Consultar los productos.
- Añadir una o más unidades de un producto al pedido.
- Consultar el pedido del carrito y eliminar productos de este.
- Realizar el pedido, introduciéndolo en la base de datos y enviando correos de confirmación al restaurante que hace el pedido y al Departamento de Pedidos de la empresa.

Para acceder a la aplicación será necesario autentificarse. Se supone que en cada restaurante habrá un responsable de pedidos que es quien tiene el usuario y la clave para acceder a la aplicación.

De cada categoría se quiere almacenar su código, su nombre y su descripción. De los productos, su código, nombre, descripción, peso, cantidad en *stock* y la categoría a la que pertenecen. Cada producto pertenece a una categoría.

De cada pedido interesa saber:

1. El restaurante que lo realizó.
2. Los productos que se pidieron, incluyendo la cantidad de unidades de cada producto.
3. Si ha sido enviado ya o no.
4. La fecha en la que se realizó el pedido.

Los pedidos se introducen en la base de datos como no enviados. Cuando se envíen el Departamento de Pedidos los marcará como enviados (directamente en la base de datos, la aplicación no se ocupa de esto).

De los restaurantes se guarda la siguiente información:

a) El código.
b) El correo electrónico. El correo es el nombre de usuario para acceder a la aplicación.
c) La clave.
d) País, dirección y código postal.

4.2.1. Esquema entidad-relación

De la descripción anterior se obtiene este esquema E-R. La relación *Incluye* es *muchos a muchos*, porque un pedido puede incluir varios productos y un producto puede aparecer en varios pedidos.

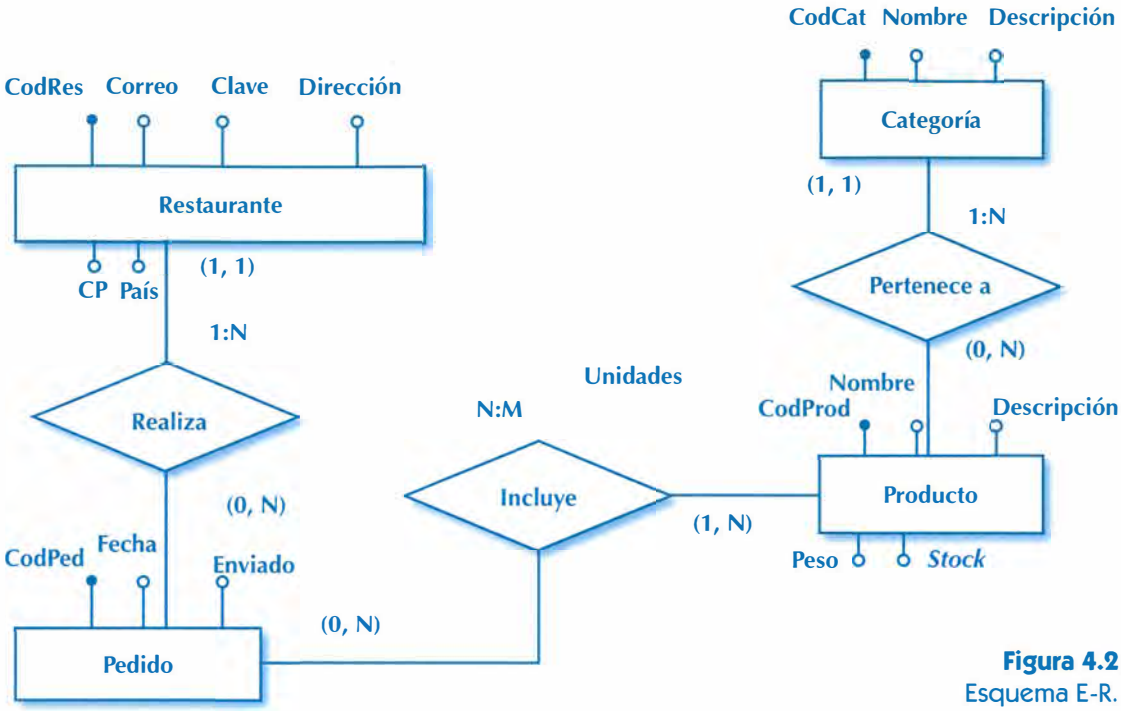

Figura 4.2
Esquema E-R.

4.2.2. Limitaciones de la aplicación

No hay panel de administración. Los usuarios, categorías y productos se tienen que introducir directamente en la base de datos.

No hay posibilidad de autorregistro.

No se controla el *stock*. Si al realizar un pedido algún producto queda con stcok negativo, el pedido se tramita igualmente.

4.3. Diseño de la aplicación

A partir del análisis anterior, se puede proceder al diseño de la aplicación. Los elementos más importantes son:

a) La base de datos.
b) El flujo de pantallas para realizar un pedido.
c) La estructura de datos para el carrito de la compra.
d) Los ficheros que forman la aplicación y cómo se pasan parámetros entre ellos.
e) El control de acceso.

4.3.1. Diseño lógico de la base de datos

Para obtener las tablas de la base de datos se parte del esquema E-R anterior. Para empezar, se crea una tabla por cada entidad. Las relaciones *Realiza* y *Pertenece a* implican un intercambio de

claves. La tabla producto recibirá la clave de la categoría, y la tabla pedido, la del restaurante que la realiza. Por otro lado, la relación *Incluye* es N:M y se resuelve mediante una tercera tabla que incluirá las claves de Pedido y Producto y los atributos de la relación.

Se obtienen las siguientes tablas:

Restaurantes(CodRes, Correo, Clave, Pais, CP, Ciudad, Dirección)

Pedidos(CodPed, Fecha, Enviado, *Restaurante*)

Productos(CodProd, Nombre, Descripción, Peso, Stock, *Categoria*)

PedidosProductos(CodPedProd, *Pedido*, *Producto*, Unidades)

Categorias(CodCat, Nombre, Descripción)

'Cod' son las claves primarias, las ajenas en cursiva.

4.3.2. Diseño físico de la base de datos

Para finalizar con la base de datos hay que decidir los tipos de datos de las columnas. El siguiente diagrama representa el resultado final.

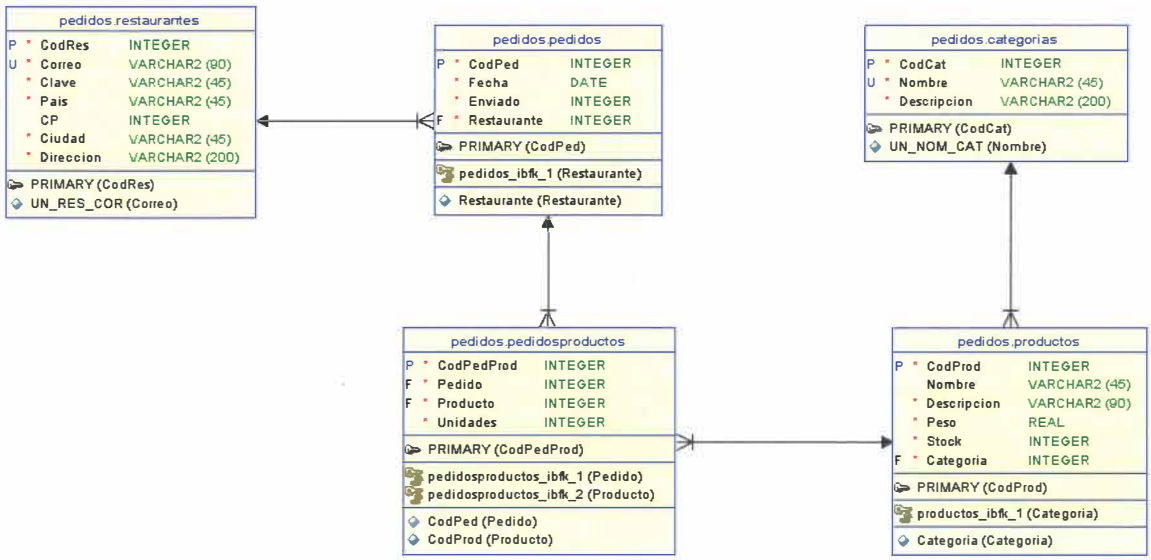

Figura 4.3
Tablas de la base de datos.

Conviene señalar que:

- Los códigos serán enteros con la opción de autoincremento.
- El correo de los restaurantes y el nombre de las categorías se marcan como *unique*.

Para insertar un pedido en la base de datos hay que insertar una fila en la tabla pedidos y una en PedidosProductos por cada producto diferente que incluya el pedido.

Ejemplo

Inserción de un pedido

El usuario con código 1 hace un pedido de 100 unidades del producto con código 16 y 150 del producto con código 20. El pedido recibe el código 1.000. En la tabla pedidos se insertará:

CodPed	Fecha	Enviado	Ejemplo
1.000	01/09/2019	0	1

Como el pedido incluye dos productos diferentes, hay que insertar dos filas en la tabla PedidosProductos. Estas filas reciben los códigos 20.000 y 20.001.

CodPedProd	CodPed	CodProd	Unidades
20.000	1.000	16	100
20.001	1.000	20	150

4.3.3. Diagrama de flujo de pantallas

El objetivo de este tipo de diagramas es representar las pantallas por las que pasa el usuario al realizar una operación. Los cuadrados representan las pantallas, y las flechas, acciones que llevan de unas a otras. El siguiente diagrama es una posible solución para este caso.

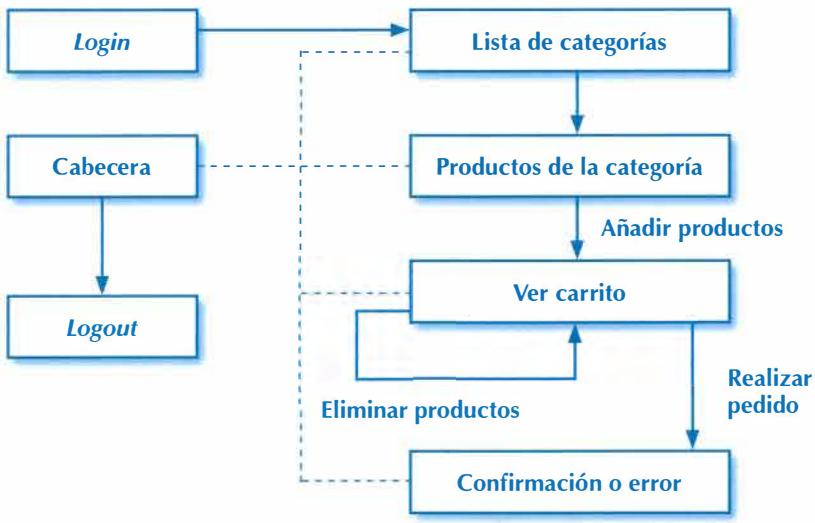

Figura 4.4
Diagrama de flujo de pantallas.

El punto de entrada a la aplicación es *login*, donde el usuario debe introducir un usuario y contraseña válidos para poder acceder a la aplicación.

Tras hacer *login* con éxito, se redirige al usuario a la página principal, que muestra las categorías existentes. Al seleccionar una categoría, se accede a sus productos. Como las categorías pueden ir cambiando a lo largo del tiempo, hay que cogerlos de las bases de datos.

En "Productos de la categoría" se muestran los datos de los productos de cierta categoría y se permite añadir un número variable de unidades al pedido. Si se añade algún producto, se redirige al usuario a "Ver carrito".

En "Ver carrito" se muestra en detalle el estado del pedido, se ofrece la posibilidad de eliminar productos y se puede confirmar el pedido. Al confirmar el pedido se muestra un mensaje de confirmación o error, según el caso.

En todas las páginas (salvo *login* y *logout*) habrá una cabecera con el nombre del restaurante y vínculos para:

- Ver carrito.
- Lista de categorías
- Cerrar la sesión.

4.3.4. El carrito de la compra

La estructura de datos utilizada para el carrito de la compra es uno de los puntos más importantes de la aplicación. Para almacenarlo se utilizará una variable de sesión.

El carrito será un *array* asociativo en el que las claves de los elementos representan el código de un producto, y el valor, el número de unidades pedidas de este producto.

El *array* comienza vacío. Cuando se añade un producto al pedido, hay que comprobar si ya hay en el *array* algún elemento que tenga como clave el código del producto. Si no lo hay, se añade un nuevo elemento tomando como clave el código y como valor el número de unidades.

Por ejemplo, si el carrito está vacío y se añaden 2 unidades del producto con código 4, se creará un elemento del *array* con clave 4 y valor 2.

Figura 4.5
Carrito de la compra con un elemento.

Si posteriormente se añaden 3 unidades del producto con código 5, el carrito será:

Figura 4.6
Se añade un nuevo producto.

Si ya hay un elemento con ese código, se suman las unidades al valor actual del elemento. Si se añaden otras cinco unidades del producto con código 4, el resultado será:

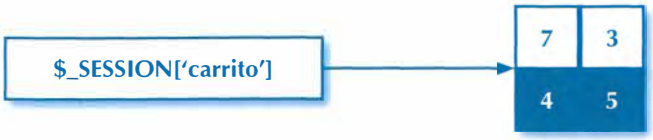

Figura 4.7
Se añade un nuevo producto.

De la misma manera, al eliminar productos del carrito habrá que buscar el elemento correspondiente en el *array* y restarle las unidades. Tras eliminar 3 unidades del producto con código 4, el carrito quedará así:

Figura 4.8
Se eliminan algunas unidades de un producto.

Cuando se eliminan todas las unidades de un producto, se suprime el elemento correspondiente en el *array*. Si para finalizar se eliminan 4 unidades del producto con código 4, se obtendrá:

Figura 4.9
Se eliminan todas las unidades de un producto.

Actividad propuesta 4.2

¿Crees que sería buena idea almacenar en el carrito todos los datos de los productos, y no solo el código? Analiza las ventajas e inconvenientes.

4.3.5. Control de acceso

Cuando se realiza *login* con éxito, se crea una nueva sesión y dos variables de sesión:

- Un *array* con dos campos. Uno para guardar el nombre del usuario (el correo del restaurante) y otro para su código, así no hay que buscarlo en la base de datos más adelante.
- La variable para el carrito de la compra.

El resto de los ficheros de la aplicación comienza uniéndose a la sesión y comprobando que la primera de estas variables existe. Si no se han creado, es que el usuario no ha hecho *login* y por tanto no puede acceder. En ese caso se le redirige a la página de *login*.

4.3.6. Ficheros de la aplicación

El siguiente paso es decidir qué ficheros formarán la aplicación y cómo se comunicarán entre ellos. Se resumen en la siguiente tabla.

CUADRO 4.1
Ficheros de la aplicación

Ruta	Descripción	Parámetros	Redirige a
login.php	Formulario de *login*	$_GET['redirigido'] $_POST['usuario'] $_POST['clave']	login.php?error=TRUE categorias.php
logout.php	Cierra la sesión		
sesiones.php	Comprueba que el usuario haya iniciado sesión correctamente		login.php (si no se ha iniciado sesión)
categorias.php	Muestra la lista de categorías con vínculos a productos.php?categoria=codigo		
cabecera.php	Cabecera con vínculos para ver el carrito, las categorías o cerrar sesión		
productos.php	Muestra los productos de la categoría, permite añadir al carro de la compra	$_GET['categoria'], el código de la categoría	
carrito.php	Muestra el carro de la compra, permite quitar productos y confirmar el pedido		
anadir.php	Añade productos al carrito	$_POST['cod'] $_POST['unidades']	carrito.php
eliminar.php	Elimina productos del carrito	$_POST['cod'] $_POST['unidades']	carrito.php
procesar_pedido.php	Inserta el pedido en la base de datos, envía correos de confirmación y muestra mensajes de error o éxito		
bd.php	Para agrupar las funciones de la base de datos		
correo.php	Funciones para enviar correo		

4.4. Implementación

Una vez completado el diseño de la aplicación, se puede comenzar con la implementación. Tomando como base la tabla anterior, en este apartado se van implementando todos los ficheros. Probablemente, el punto más complicado sean las funciones de acceso a la base de datos, que están agrupadas en bd.php.

4.4.1. *Login*

El fichero **login.php** incluye tanto el formulario HTML como el código para procesarlo. Al principio está el bloque PHP, que se ejecuta solo cuando el método es POST, es decir, cuando se envía el formulario.

```
1  <?php
2  require_once 'bd.php';
3  /*formulario de login habitual
4  si va bien abre sesión, guarda el nombre de usuario y redirige a princi-
   pal.php
5  si va mal, mensaje de error */
6  if ($_SERVER["REQUEST_METHOD"] == "POST") {
7
8      $usu = comprobar_usuario($_POST['usuario'], $_POST['clave']);
9      if($usu===FALSE){
10         $err = TRUE;
11         $usuario = $_POST['usuario'];
12     }else{
13         session_start();
14         //     $usu tiene campos correo y codRes, correo
15         $_SESSION['usuario'] = $usu;
16         $_SESSION['carrito'] = [];
17         header("Location: categorias.php");
18         return;
19     }
20 }
21 ?>
22 <!DOCTYPE html>
23 <html>
24     <head>
25         <title>Formulario de login</title>
26         <meta charset = "UTF-8">
27     </head>
28     <body>
29         <?php if(isset($_GET["redirigido"])){
30             echo "<p>Haga login para continuar</p>";
31         }?>
32         <?php if(isset($err) and $err == TRUE){
33             echo "<p> Revise usuario y contraseña</p>";
34         }?>
35         <form action = "<?php echo htmlspecialchars($_SERVER["PHP_
           SELF"]);?>" method = "POST">
36             <label for = "usuario">Usuario</label>
37             <input value = "<?php if(isset($usuario))echo $usuario;?>"
38             id = "usuario" name = "usuario" type = "text">
```

```
39            <label for = "clave">Clave</label>
40            <input id = "clave" name = "clave" type = "password">
41            <input type = "submit">
42        </form>
43    </body>
44 </html>
```

En la línea 8 comprueba usuario y contraseña. Si son correctos, crea una nueva sesión y las variables de sesión para el carrito y para el usuario. Para el usuario se crea una variable `$_SESSION['usuario']` con dos campos, 'correo' y 'codRes'. Para finalizar, reenvía al usuario a **categorias.php**, la página principal.

Si el *login* no es correcto o se accede al fichero por GET, se muestra el formulario HTML. Además de los elementos básicos, se pueden mostrar dos mensajes de error:

- Si el parámetro "redirigido" está presente en la URL, se muestra un mensaje indicando que se debe iniciar sesión para continuar (línea 29). El *script* que comprueba si se ha iniciado sesión, **sesiones.php**, redirige a login.php?redirigido=true cuando no encuentra la variable `$_SESSION['usuario']`.
- Si se ha enviado el formulario, pero los datos no son correctos, el primer bloque de PHP crea la variable `$err` con valor TRUE. Esto se comprueba en la línea 32 y se muestra el mensaje correspondiente.

4.4.2. Control de acceso

Para comprobar que solo puedan acceder a la aplicación los usuarios que hayan hecho *login*, se utiliza la función `comprobar_sesion()` del fichero **sesiones.php**.

```
<?php
function comprobar_sesion(){
    session_start();
    if(!isset($_SESSION['usuario'])){
    header("Location: login.php?redirigido=true");
    }
}
```

Simplemente se une a la sesión existente y comprueba que la variable `$_SESSION['usuario']` exista. Si no es el caso, quiere decir que el usuario no ha hecho *login* correctamente y, por tanto, se le redirige al formulario de *login*. El parámetro redirigido hace que se muestre un mensaje de error junto con el formulario.

El resto de las páginas de la aplicación (excepto **login.php** y **logout.php**) llaman a este para restringir el acceso de manera que solo puedan acceder los usuarios que hayan hecho *login* con éxito.

4.4.3. La cabecera

Esta es la cabecera común a todas las páginas una vez se hace *login*. Muestra el nombre del usuario utilizando la variable de sesión y vínculos para cerrar la sesión y para volver a la página principal.

```
<header>
    Usuario: <?php echo $_SESSION['usuario']['correo']?>
    <a href="categorias.php">Home</a>
    <a href="carrito.php">Ver carrito</a>
    <a href="logout.php">Cerrar sesión</a>
</header>
<hr>
```

4.4.4. Lista de categorías

Este fichero es la página principal de la aplicación. En el cuerpo de la página se muestra una lista con las categorías. Cada elemento de la lista es un vínculo con el nombre de la categoría y un vínculo a productos.php?categoria=<cod>.

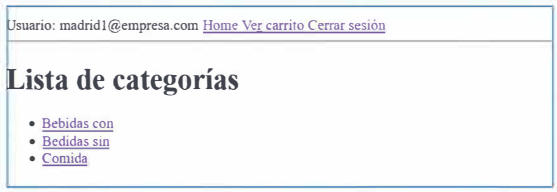

Figura 4.10
Lista de categorías.

```
<?php
/*comprueba que el usuario haya abierto sesión o redirige*/
require 'sesiones.php';
require_once 'bd.php';
comprobar_sesion();
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset = "UTF-8">
    <title>Lista de categorías</title>
</head>
    <body>
        <?php require 'cabecera.php';?>
        <h1>Lista de categorías</h1>
        <!--lista de vínculos con la forma
        productos.php?categoria=1-->
        <?php
        $categorias = cargar_categorias();
        if($categorias===FALSE){
            echo "<p class='error'>Error al conectar con la base datos</
            p>";
        }else{
            echo "<ul>"; //abrir la lista
            foreach($categorias as $cat){
                /*$cat['nombre] $cat['codCat']*/
                $url = "productos.php?categoria=".$cat['codCat'];
                echo "<li><a href='$url'>".$cat['nombre']."</a></li>";
            }
            echo "</ul>";
        }
        ?>
```

```
    </body>
</html>
```

4.4.5. Tabla de productos

El fichero **productos.php** muestra una tabla con todos los elementos de una categoría y permite añadirlos al carrito. Al mostrar la tabla de productos, se añade a cada fila un formulario para añadir 1 o más unidades de ese producto al carrito. El formulario contiene campos para el código del producto, las unidades y un botón de envío. El campo con el código está oculto, no se muestra al usuario. Se envía a **anadir.php**.

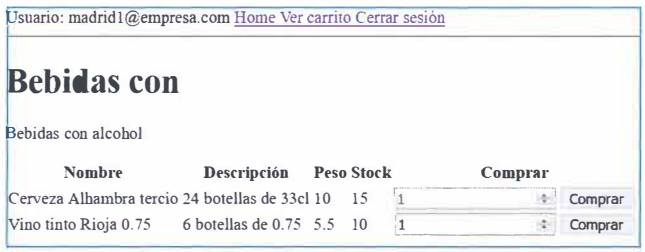

Figura 4.11
Tabla de productos.

```
<?php
    /*comprueba que el usuario haya abierto sesión o redirige*/
    require 'sesiones.php';
    require_once 'bd.php';
    comprobar_sesion();
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset = "UTF-8">
        <title>Tabla de productos por categoría</title>
    </head>
    <body>
    <?php
        require 'cabecera.php';
        $cat = cargar_categoria($_GET['categoria']);
        $productos = cargar_productos_categoria($_GET['categoria']);
        if($cat=== FALSE or $productos === FALSE){
            echo "<p class='error'>Error al conectar con la base datos</
            p>";
            exit;
        }
        echo "<h1>". $cat['nombre']. "</h1>";
        echo "<p>". $cat['descripcion']."</p>";
        echo "<table>"; //abrir la tabla
        echo "<tr><th>Nombre</th><th>Descripción</th>
              <th>Peso</th><th>Stock</th><th>Comprar</th></tr>";
        foreach($productos as $producto){
            $cod = $producto['CodProd'];
```

```
            $nom = $producto['Nombre'];
            $des = $producto['Descripcion'];
            $peso = $producto['Peso'];
            $stock = $producto['Stock'];
        echo "<tr><td>$nom</td><td>$des</td><td>$peso</td><td>$stock</td>
              <td>
                <form action = 'anadir.php' method = 'POST'>
                 <input name = 'unidades' type='number' min = '1'value = '1'>
                 <input type = 'submit' value='Comprar'>
                 <input name = 'cod' type='hidden' value = '$cod'>
                </form>
              </td>
        </tr>";
        }
    echo "</table>"
  ?>
  </body>
</html>
```

Actividad propuesta 4.3

¿Qué opciones se te ocurren para añadir productos en lugar de un formulario? Analiza sus ventajas e inconvenientes.

4.4.6. Añadir productos

El fichero **anadir.php** se encarga de añadir elementos al carrito. No tiene salida, modifica la variable de sesión y redirige a **carrito.php**. Recibe los datos del formulario de **productos.php** y los parámetros `cod` y `unidades` que recibe para modificar el carrito. Primero comprueba si el código ya existe en el *array*. Si existe, se suma `unidades` al valor existente. Si no existe, se crea con valor `unidades`.

```
<?php
    /*comprueba que el usuario haya abierto sesión o redirige*/
    require_once 'sesiones.php';
    comprobar_sesion();
    $cod = $_POST['cod'];
    $unidades = (int)$_POST['unidades'];
    /*si existe el código sumamos las unidades*/
    if(isset($_SESSION['carrito'][$cod])){
        $_SESSION['carrito'][$cod] += $unidades;
    }else{
        $_SESSION['carrito'][$cod] = $unidades;
    }
    header("Location: carrito.php");
```

Actividad propuesta 4.4

El fichero anterior permite añadir más unidades de las disponibles para un producto. ¿Qué alternativas se te ocurren para impedirlo?

4.4.7. El carrito de la compra

Muestra una tabla con una fila por cada producto (diferente) del carrito. La fila muestra los datos del producto y el número de unidades pedidas. Además, en cada fila hay un formulario para eliminar ese producto del carrito. El funcionamiento es análogo al del formulario para añadir de **productos.php**.

El formulario incluye el número de unidades que hay que eliminar y el código del producto, este último como campo oculto. Se envía a **eliminar.php**.

```
<?php
    /*comprueba que el usuario haya abierto sesión o redirige*/
    require_once 'sesiones.php';
    require_once 'bd.php';
    comprobar_sesion();
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset = "UTF-8">
        <title>Carrito de la compra</title>
    </head>
    <body>
        <?php
        require 'cabecera.php';
        echo "<h2>Carrito de la compra</h2>";
        $productos = cargar_productos(array_keys($_SESSION['carrito']));
        if($productos === FALSE){
            echo "<p>No hay productos en el pedido</p>";
            exit;
        }
        echo "<h2>Carrito de la compra</h2>";
        echo "<table>"; //abrir la tabla
        echo "<tr><th>Nombre</th><th>Descripción</th><th>Peso</th>
        <th>Unidades</th><th>Eliminar</th></tr>";
        foreach($productos as $producto){
            $cod = $producto['CodProd'];
            $nom = $producto['Nombre'];
            $des = $producto['Descripcion'];
            $peso = $producto['Peso'];
            $unidades = $_SESSION['carrito'][$cod];
```

```
            echo "<tr><td>$nom</td><td>$des</td><td>$peso</td><td>$unida-
            des</td>
            <td>
               <form action = 'eliminar.php' method = 'POST'>
                  <input name = 'unidades' type='number' min = '1'value =
                  '1'>
                  <input type = 'submit' value='Eliminar'></td>
                  <input name = 'cod' type='hidden' value ='$cod'>
               </form>
            </td>
         </tr>";
         }
            echo "</table>";
         ?>
         <hr>
         <a href = "procesar_pedido.php">Realizar pedido</a>
   </body>
</html>
```

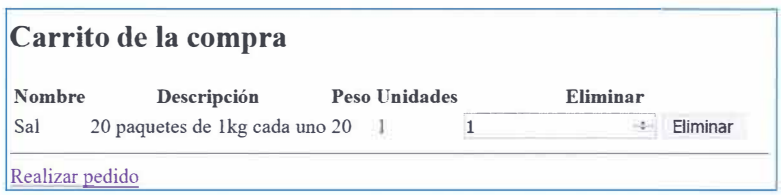

Figura 4.12
Carrito de la compra.

4.4.8. Eliminar productos

Este *script* se encarga de eliminar elementos del carrito. No tiene salida, modifica la variable de sesión y redirige a **carrito.php**. Lo primero que hace es comprobar que los parámetros `cod` y `unidades` estén presentes. Para modificar el carrito, primero comprueba si el código ya existe en el *array*. Si existe, se resta unidades al valor existente. Si el valor resultante es menor o igual que cero, se elimina ese elemento del *array*.

Si no existe, no hace nada, porque no hay nada que eliminar. Como a **eliminar.php** se accede desde los vínculos de **carrito.php**, que se generan a partir del carrito, en principio este caso no debería darse.

```
<?php
/*comprueba que el usuario haya abierto sesión o redirige*/
require_once 'sesiones.php';
comprobar_sesion();
$cod = $_POST['cod'];
$unidades = $_POST['unidades'];
/*si existe el código restamos las unidades, con mínimo de 0*/
if(isset($_SESSION['carrito'][$cod])){
   $_SESSION['carrito'][$cod] -= $unidades;
   if($_SESSION['carrito'][$cod] <= 0){
      unset($_SESSION['carrito'][$cod]);
```

```
        }
    }
header("Location: carrito.php");
```

4.4.9. Procesamiento del pedido

Este proceso consiste en:

- Insertar al pedido usando la función `insertar_pedido()`.
- Si el pedido se inserta correctamente:
  - Llamar a `enviar_correos()` para mandar los correos de confirmación.
  - Vaciar el carrito.
- Mostrar mensajes de confirmación o error.

```
<?php
    /*comprueba que el usuario haya abierto sesión o redirige*/
    require '..\correo\enviar_correo.php';
    require 'sesiones.php';
    require_once 'bd.php';
    comprobar_sesion();
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset = "UTF-8">
        <title>Pedidos</title>
    </head>
    <body>
        <?php
        require 'cabecera.php';
        $resul = insertar_pedido($_SESSION['carrito'],
            $_SESSION['usuario']['codRes']);
        if($resul === FALSE){
            echo "No se ha podido realizar el pedido<br>";
        }else{
            $correo = $_SESSION['usuario']['correo'];
            echo "Pedido realizado correctamente<br>";
            //vaciar carrito
            $compra = $_SESSION['carrito'];
            $_SESSION['carrito'] = [];
            echo "Pedido realizado con éxito.
            Se enviará un correo de confirmación a: $correo ";
                enviar_correos($compra, $pedido, $correo);
        }
    ?>
```

```
    </body>
</html>
```

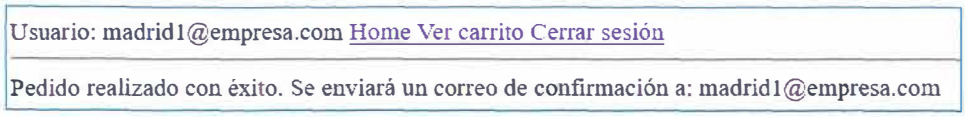

Figura 4.13
Confirmación del pedido.

Actividad propuesta 4.5

Las tiendas online suelen pedir la confirmación del usuario antes de procesar el envío. ¿Qué cambios habría que hacer en la aplicación para incluir ese paso?

4.4.10. La base de datos

Las funciones para manejar la base de datos se agrupan en el fichero **bd.php**. La base de datos se utiliza para controlar el accesos al sistema, al ver las categorías y productos y al realizar el pedido.

La siguiente tabla resume las funciones para que relacionadas con la base de datos.

CUADRO 4.2
Funciones de bd.php

Función	Descripción
leer_config($nombre, $esquema)	Devuelve la configuración de la base de datos
cargar_categorias()	Devuelve un cursor con las categorías
cargar_productos_categoria($codCat)	Devuelve un cursor con los productos de la categoría
cargar_categoria($codCat)	Devuelve los datos de la categoría
comprobar_usuario($nombre, $clave);	Comprueba usuario y clave en la base de datos
cargar_productos($codigosProductos)	Devuelve un cursor de productos a partir de sus códigos
insertar_pedido($carrito, $codRes)	Inserta el pedido en la base de datos

A) Configuración

Para almacenar los datos de configuración de la base de datos se usa el fichero **configuracion.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuracion>
```

```
    <ip>127.0.0.1</ip>
    <nombre>pedidos</nombre>
    <usuario>root</usuario>
    <clave></clave>
</configuracion>
```

Este fichero tiene que cumplir el esquema `configuracion.xsd`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDe-
fault="qualified">
    <xs:element name="configuracion">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ip" type="xs:string"/>
                <xs:element name="nombre" type="xs:string"/>
                <xs:element name="usuario" type="xs:string"/>
                <xs:element name="clave" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Todas las funciones que acceden a la base de datos utilizan los datos de este fichero (a través de la función `leer_config()`.).

B) Descripción de las funciones

1. function leer_config($nombre, $esquema);

- $nombre es la ruta del fichero con los datos de conexión a la base de datos.
- $esquema es la ruta del fichero XSD para validar el anterior.

Si el fichero de configuración existe y es válido, devuelve un *array* con 3 valores: la cadena de conexión, el nombre de usuario y la clave.

Si no encuentra el fichero o no es válido, lanza una excepción.

2. function cargar_categorias();

Devuelve un cursor con el código y nombre de las categorías de la base de datos.

```
function cargar_categorias(){
    $res = leer_config(dirname(__FILE__)."/configuracion.xml",
    dirname(__FILE__)."/configuracion.xsd");
    $bd = new PDO($res[0], $res[1], $res[2]);
```

```
    $ins = "select codCat, nombre from categorias";
    $resul = $bd->query($ins);
    if (!$resul) {
        return FALSE;
    }
    if ($resul->rowCount() === 0) {
        return FALSE;
    }
    //si hay 1 o más
    return $resul;
}
```

3. function cargar_categoria($codCat);

Recibe como argumento el código de una categoría y devuelve un *array* con su nombre y descripción. Si hay algún error con la base de datos o la categoría no existe, devuelve FALSE.

4. function comprobar_usuario($nombre, $clave);

Esta es la función que se usa para comprobar los datos del formulario de *login*. Si los datos son correctos, devuelve un *array* con dos campos: 'codRes', el código del restaurante y 'correo', su correo. Si hay algún error con la base de datos o los datos no son correctos, devuelve FALSE.

5. function cargar_productos_categoria($codCat);

Recibe como argumento el código de una categoría y devuelve un cursor con sus productos. Incluye todas las columnas de la base de datos. Si hay algún error con la base de datos, la categoría no existe o no tiene productos, devuelve FALSE.

6. function cargar_productos($codigosProductos);

Recibe como argumento un *array* de códigos de productos y devuelve un cursor con todas las columnas de estos. Si hay algún error con la base de datos, devuelve FALSE. Esta función se usa al mostrar el carrito de la compra.

```
function cargar_productos($codigosProductos){
    $res = leer_config(dirname(__FILE__)."/configuracion.xml",
    dirname(__FILE__)."/configuracion.xsd");
    $bd = new PDO($res[0], $res[1], $res[2]);
    $texto_in = implode(",", $codigosProductos);
    $ins = "select * from productos where codProd in($texto_in)";
    $resul = $bd->query($ins);
    if (!$resul) {
```

```
            return FALSE;
        }
        return $resul;
    }
```

7. function insertar_pedido($carrito, $codRes);

Esta función es la que se encarga de insertar el pedido en la base de datos. Recibe el carrito de la compra y el código del restaurante que realiza el pedido. Si todo va bien, devuelve el código del nuevo pedido. Si hay algún error devuelve FALSE.

Se encarga de transformar el carrito de la compra en un pedido en la base de datos, para lo cual:

- Crea una nueva fila en la tabla pedidos.
- Crea una fila en la tabla PedidosProductos por cada producto diferente que se pida. Hay que usar la clave del nuevo pedido.

Las inserciones deben ocurrir como una transacción. Si alguna falla, hay que deshacer las anteriores.

```
function insertar_pedido($carrito, $codRes){
    $res = leer_config(dirname(__FILE__)."/configuracion.xml", dirname(__
    FILE__)."/configuracion.xsd");
    $bd = new PDO($res[0], $res[1], $res[2]);
    $bd->beginTransaction();
    $hora = date("Y-m-d H:i:s", time());
    // insertar el pedido
    $sql = "insert into pedidos(fecha, enviado, restaurante)
        values('$hora',0, $codRes)";
    $resul = $bd->query($sql);
        if (!$resul) {
        return FALSE;
    }
    // coger el id del nuevo pedido para las filas detalle
    $pedido = $bd->lastInsertId();
    // insertar las filas en pedidoproductos
    foreach($carrito as $codProd=>$unidades){
        $sql = "insert into pedidosproductos(Pedido, Producto,
        Unidades)values($pedido, $codProd, $unidades)";
        echo $sql;
        $resul = $bd->query($sql);
        if (!$resul) {
            $bd->rollback();
            return FALSE;
        }
        $sql = "update productos set stock = stock - $unidades
where codProd = $codProd";
```

```
$resul = $bd->query($sql);
if (!$resul) {
    $bd->rollback();
    return FALSE;
}
    }
    $bd->commit();
    return $pedido;
}
```

4.4.11. Envío de correos

Si todo ha ido bien, envía un correo de confirmación al restaurante que lo ha realizado y al Departamento de Pedidos. Es el mismo correo para los dos. El correo incluirá el número del pedido, el restaurante que lo realiza y una tabla HTML con los productos del pedido, bastante parecida a la del carrito.

Las funciones para enviar los correos están en el fichero **correo.php**:

1. crear_correo($carrito, $pedido, $correo);

```
function crear_correo($carrito, $pedido, $correo){
    $texto = "<h1>Pedido nº $pedido </h1><h2>Restaurante:
    $correo </h2>";
    $texto.= "Detalle del pedido:";
    $productos = cargar_productos(array_keys($carrito));
    $texto.= "<table>"; //abrir la tabla
    $texto.= "<tr><th>Nombre</th><th>Descripción</th><th>Peso</th>
    <th>Unidades</th><th>Eliminar</th></tr>";
    foreach($productos as $producto){
        $cod = $producto['CodProd'];
        $nom = $producto['Nombre'];
        $des = $producto['Descripcion'];
        $peso = $producto['Peso'];
        $unidades = $_SESSION['carrito'][$cod];
        $texto.= "<tr><td>$nom</td><td>$des</td><td>$peso</td>
        <td>$unidades</td><td></tr>";
    }
    $texto.= "</table>";
    return $texto;
}
```

2. enviar_correo_multiples ($lista_correos, $cuerpo, $asunto = "");

Recibe un *array* de direcciones de correo, el cuerpo del correo y opcionalmente el asunto. Envía el correo a todas las direcciones.

3. enviar_correos($carrito, $pedido, $correo);

Recibe el carrito de la compra, el número de pedido y el correo del restaurante que lo hace. Primero llama a la función **crear_correo**() para crear el cuerpo, y luego llama **enviar_correo_multiples().**

Figura 4.14
Confirmación del pedido.

TOMA NOTA

Hay que tener instalado PHPMailer, también hay que introducir los datos de la cuenta de Gmail y permitir el acceso a aplicaciones menos seguras, como se explica en el capítulo 3.

Resumen

- La aplicación permite que los restaurantes de una cadena realicen pedidos al Departamento de Pedidos.
- Muestra los productos disponibles organizados por categorías.
- El carrito de la compra es una variable de sesión. Es un *array* en que la clave de cada elemento representa el código del producto, y su valor, las unidades pedidas.
- El carrito se manipula a través de formularios que envían el código del producto y las unidades que hay que añadir o eliminar.
- Al añadir productos al carrito, si no hay ya un elemento con ese código, se crea. Si lo hay, se suman las unidades.
- Al eliminar productos del carrito hay que comprobar si quedan 0 o menos unidades para eliminar el elemento correspondiente.
- Al pulsar el vínculo de realizar pedido, los datos del carrito se utilizan para modificar la base de datos.
- El pedido se realiza como una transacción; si hay algún error con la base de datos, se deshacen las operaciones anteriores.
- Si el pedido se realiza con éxito, se envían correos de confirmación al Departamento de Pedidos y al restaurante que lo realiza.
- Hay muchas ampliaciones sencillas por hacer: control de *stock*, panel de administración, contraseñas encriptadas...

Ejercicios propuestos

1. Al añadir un producto, la aplicación redirige al carrito. Modifícala para que redirija a la tabla de productos, con la misma categoría.
2. Haz los cambios necesarios para encriptar las contraseñas de la tabla de usuarios (utiliza la función bcrypt()).
3. Modifica la tabla de productos para que no muestre los productos sin *stock*.
4. Añade el peso total del pedido en el contenido del correo de confirmación.
5. Almacena la información del servidor de correo en un fichero de XML como se hace con el de la base de datos. Modifica las funciones de envío de correo para que utilicen ese fichero.
6. Crea un esquema XSD para la configuración de correo y valida con él el fichero de configuración antes de usarlo.