

Desarrollo Web en entorno servidor

Unidad 4:

Aplicación completa

Tabla de contenido

1.	Introducción.....	3
1.1.	Metodología de desarrollo en espiral.	3
2.	Código PHP, tablas y base de datos a utilizar.	4
3.	Fase de planificación.....	4
3.1.	Análisis de requisitos.....	5
a)	Limitaciones de la aplicación.....	6
b)	Esquema entidad-relación	6
3.2.	Diseño de la aplicación	6
a)	Diseño lógico de la base de datos	6
b)	Diseño físico de la base de datos	7
c)	Diagrama de flujo de pantallas.....	8
d)	El carrito de la compra	10
e)	Control de acceso	11
f)	Ficheros de la aplicación	11
4.	Implementación.....	12
4.1.	Login	12
4.2.	Control de acceso	12
4.3.	La cabecera.....	12

4.4.	Lista de categorías	13
4.5.	Tabla de productos.....	13
4.6.	Añadir productos.....	13
4.7.	El carrito de la compra	13
4.8.	Eliminar productos	13
4.9.	Procesamiento del pedido	13
4.10.	La base de datos	14
a)	Configuración	14
b)	Descripción de las funciones	14
4.11.	Envío de correos	15

1. Introducción.

En este tema desarrollamos una aplicación web para realizar pedidos, similar a una tienda web. Con esto, pretendemos unir los elementos de los temas anteriores para plantear una aplicación web completa y mostrar cómo plantear un proyecto de aplicación web con base de datos desde cero siguiendo una metodología en espiral.

1.1. Metodología de desarrollo en espiral.

En esta metodología los proyectos pasan por distintas etapas, desde la de conceptualización, siguiendo el desarrollo, luego una etapa de mejoras, para finalizar con el mantenimiento.

Cada una de estas etapas se subdivide en seis fases:

1. Comunicación con el cliente.
2. Planificación.
3. Análisis de Riesgos.
4. Ingeniería.
5. Evaluación del cliente.
6. Construcción y entrega.

Podemos considerar lo expuesto en este tema como una primera iteración, el desarrollo de los conceptos, dentro de esta metodología, desarrollaremos las fases de planificación e ingeniería. Se puede considerar como la primera iteración dentro de un proyecto más completo.

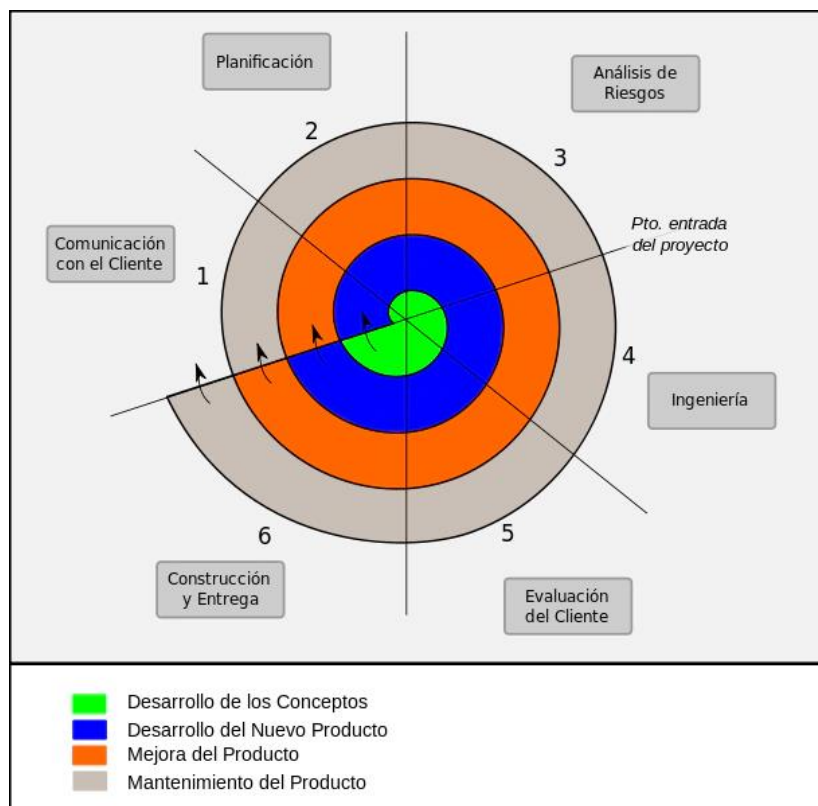


Ilustración 1.- Modelo en Espiral.

Las ventajas del modelo en Espiral, en líneas generales, son las siguientes:

- La funcionalidad adicional o los cambios se pueden hacer en una etapa posterior.
- La estimación del coste se hace fácil, ya que la construcción del prototipo se hace en pequeños fragmentos.
- El desarrollo continuo o repetido ayuda en la gestión de riesgos.
- El desarrollo es rápido y las características se añaden de forma sistemática.
- Siempre hay espacio para atender los comentarios de los clientes.

En la parte negativa, podemos decir que el modelo en Espiral nos presenta los siguientes retos a los que tendremos que hacer frente si nos decidimos a emplear esta aproximación en nuestros proyectos:

- Riesgo de no cumplir con la planificación o el presupuesto.
- Funciona mejor para proyectos grandes, aunque en estos también requiera de una estricta evaluación de riesgos.
- Para su buen funcionamiento, el protocolo del modelo en espiral debe ser seguido estrictamente.
- Se genera más documentación al tener fases intermedias.
- No es aconsejable para proyectos pequeños, la ratio coste beneficio no es rentable.

2. Código PHP, tablas y base de datos a utilizar.

Para poder manejar los archivos de la aplicación disponibles, vamos a crear en phpMyAdmin una nueva base de datos llamada **pedidos**.

Una vez creada, seleccionamos la pestaña SQL y pegamos el contenido del archivo **pedidos.sql**.

Para poder acceder a la aplicación, abrimos el archivo: **login.php**, los datos de acceso, según podemos ver en la tabla **restaurante** creada en la base de datos:

Usuario: madrid1@empresa.com, clave: 1234

Usuario: cadiz1@empresa.com, clave: 1234

3. Fase de planificación

En esta fase se define la funcionalidad de la aplicación y sus limitaciones, según la comunicación mantenida con el cliente. Se detallan los datos que se quieren almacenar y se realiza un esquema entidad relación para la base de datos, planteando un diseño para la aplicación en su conjunto.

En particular, describiremos la lógica de la base de datos, su diseño físico y el diagrama de flujo de pantallas. En un proyecto real se estimarían, a su vez, los costes, el calendario y los recursos para cada iteración.

Posteriormente pasaríamos a la fase de análisis de riesgos, qué, en esta primera iteración, se reduce en la mayoría de los proyectos en haber plasmado adecuadamente lo expresado por el cliente.

En esta primera sección definimos:

- 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.

- 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.

La situación es la siguiente, existe una empresa que es dueña de varios restaurantes y posee un almacén central. La empresa necesita una aplicación web para que el departamento de pedidos pueda gestionar los pedidos que realizan los encargados de los restaurantes.

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. Ya que la empresa sabe dónde está cada restaurante.

3.1. Análisis de requisitos

Según el departamento de pedidos, 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 autenticarse. 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 (por ejemplo, bebidas con alcohol) se quiere almacenar su código, su nombre y su descripción. De los productos (por ejemplo, cerveza o vino), 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:

- El restaurante que lo realizó.
- Los productos que se pidieron, incluyendo la cantidad de unidades de cada producto.
- Si ha sido enviado ya o no.
- 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:

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

a) 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 stock negativo, el pedido se tramita igualmente.

b) 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.

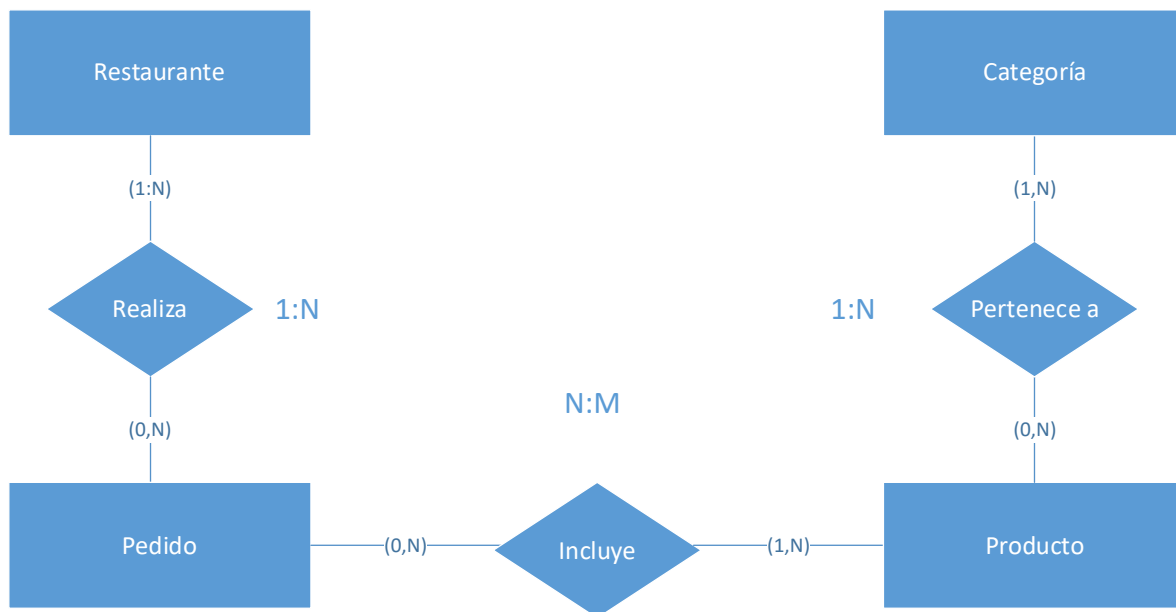


Figura 1.-Esquema E-R

3.2. Diseño de la aplicación

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

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

a) 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, *Categoría*)
- PedidosProductos(CodPedProd, *Pedido*, *Producto*, Unidades)
- Categorías(CodCat, Nombre, Descripción)

“Cod” son las claves primarias, las ajenas en cursiva.

b) 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.

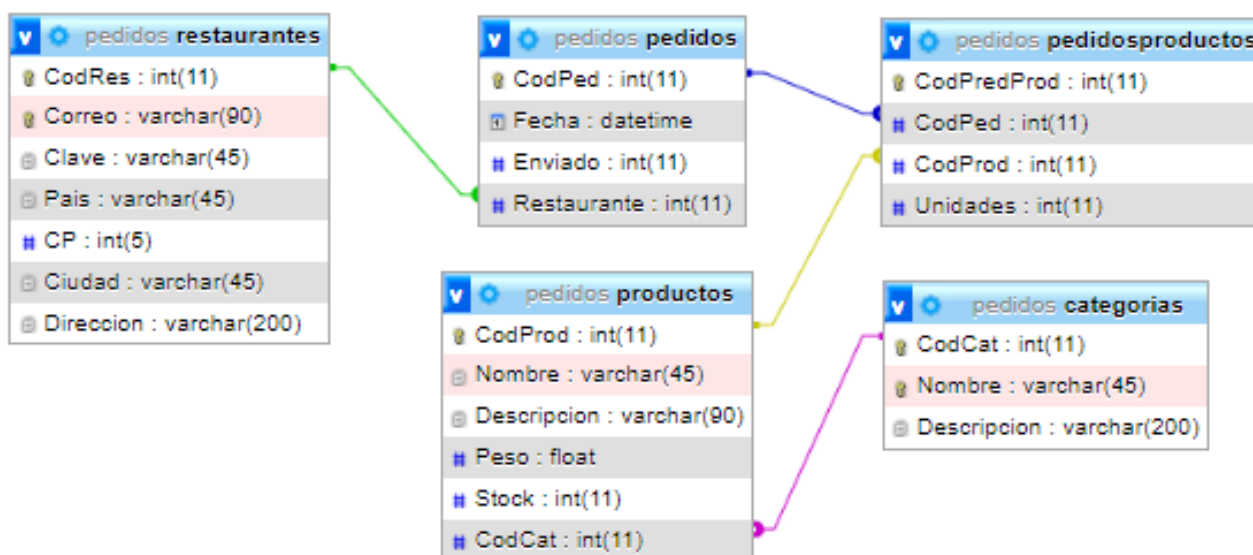


Ilustración 2.- Tablas de la base de datos

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.

Estas relaciones se han establecido en las sentencias del archivo pedidos.sql, se pueden ver a través de la pestaña “Diseñador” en phpMyAdmin.

c) 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:

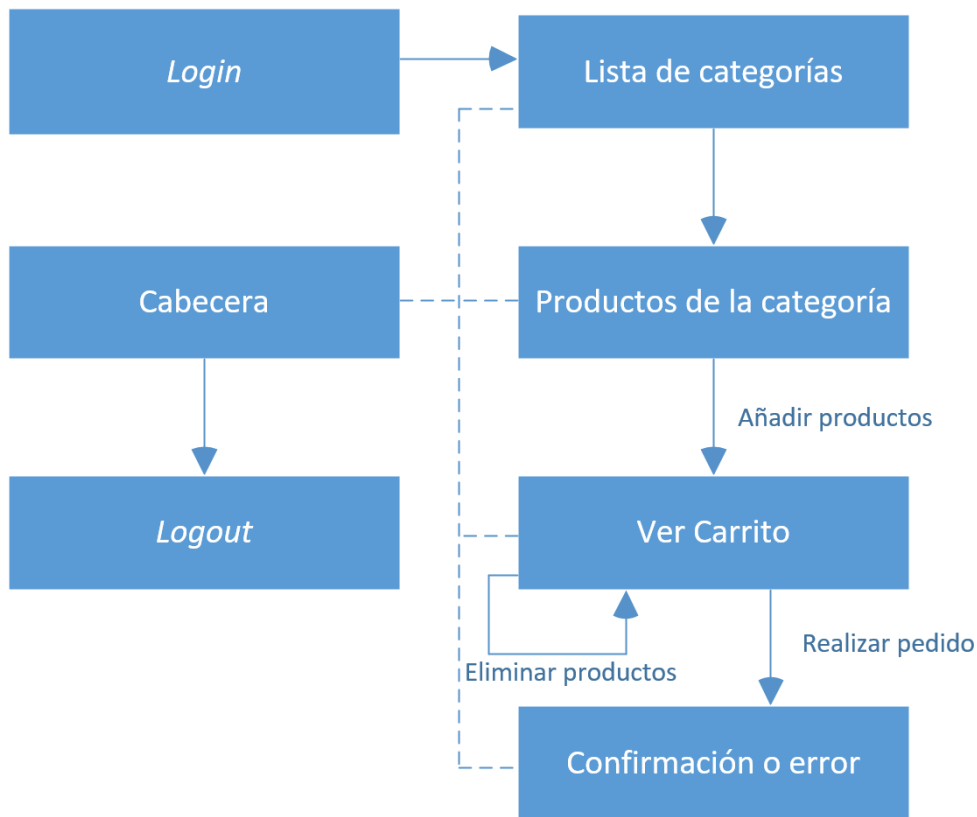


Ilustración 3.- Diagrama de flujo de pantallas

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 cogerlas de la base 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, 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.

Diagrama de flujo de pantallas.

Las pantallas vistas por el usuario son:



Usuario Clave

Ilustración 4.- Pantalla de login



Usuario: madrid1@empresa.com [Home](#) [Ver carrito](#) [Cerrar sesión](#)

Lista de categorías

- [Bebidas con](#)
- [Bebidas sin](#)
- [Comida](#)

Ilustración 5.- Lista de categorías



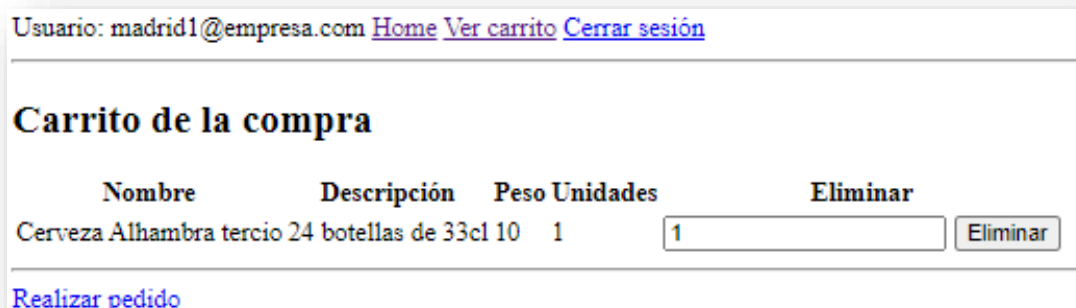
Usuario: madrid1@empresa.com [Home](#) [Ver carrito](#) [Cerrar sesión](#)

Bebidas con

Bebidas con alcohol

Nombre	Descripción	Peso	Stock		Comprar
Cerveza Alhambra tercio 24	botellas de 33cl	10	0	<input type="text" value="1"/>	<input type="button" value="Comprar"/>
Vino tinto Rioja 0.75	6 botellas de 0.75	5.5	10	<input type="text" value="1"/>	<input type="button" value="Comprar"/>

Ilustración 6.- Tabla de productos



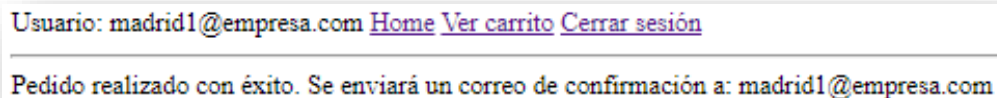
Usuario: madrid1@empresa.com [Home](#) [Ver carrito](#) [Cerrar sesión](#)

Carrito de la compra

Nombre	Descripción	Peso	Unidades		Eliminar
Cerveza Alhambra tercio 24	botellas de 33cl	10	1	<input type="text" value="1"/>	<input type="button" value="Eliminar"/>

[Realizar pedido](#)

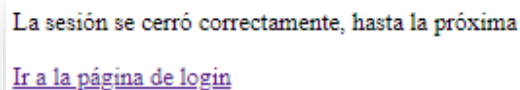
Ilustración 7.- Carrito de la compra



Usuario: madrid1@empresa.com [Home](#) [Ver carrito](#) [Cerrar sesión](#)

Pedido realizado con éxito. Se enviará un correo de confirmación a: madrid1@empresa.com

Ilustración 8.- Confirmación de pedido



La sesión se cerró correctamente, hasta la próxima

[Ir a la página de login](#)

Ilustración 9.- Pantalla de logout

d) 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.

```
$_SESSION['carrito'] = array();
```

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.

```
$_SESSION['carrito'] = array(4=>2);
```

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

```
$_SESSION['carrito'] = array(4=>2,5=>3);
```

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:

```
$_SESSION['carrito'] = array(4=>7,5=>3);
```

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:

```
$_SESSION['carrito'] = array(4=>4,5=>3);
```

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:

```
$_SESSION['carrito'] = array(5=>3);
```

e) Control de acceso

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

- Un array con dos campos. 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* por tanto no puede acceder. En ese caso se le redirige a la página de *login*.

f) 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.

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=código		
cabecera.php	Cabecera con vinculos 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 correos		

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.

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.

En la línea 8 se 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'] = $usu;` 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.

4.2. 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**.

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.

4.3. La cabecera

El archivo **cabecera.php** 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.

4.4. Lista de categorías

El fichero **categorias.php** 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=".$cat['codCat']`.

4.5. Tabla de productos

El fichero **productos.php** muestra una tabla con todos los elementos de una categoría y permite añadir los al carrito. Al mostrar la tabla de productos, se añade a cada fila un formulario para añadir una 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**.

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.

4.7. El carrito de la compra

El fichero **carrito.php** 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**.

4.8. Eliminar productos

El fichero **eliminar.php** 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 se da.

4.9. Procesamiento del pedido

En **procesar_pedido.php** nos encontramos con el procedimiento para tramitar pedidos. 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.

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 acceso 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.

Función	Descripción
<code>leer_config(\$nombre, \$esquema)</code>	Devuelve la configuración de la base de datos
<code>cargar_categorias()</code>	Devuelve un cursor ¹ con las categorías
<code>cargar_productos_categoria(\$codCat)</code>	Devuelve un cursor con los productos de la categoría
<code>cargar_categoria(\$codCat)</code>	Devuelve los datos de la categoría
<code>comprobar_usuario(\$nombre, \$clave)</code>	Comprueba usuario y clave en la base de datos
<code>cargar_productos(\$codigosProductos)</code>	Devuelve un cursor de productos a partir de sus códigos
<code>insertar_pedido(\$carrito, \$codRes)</code>	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**.

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

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

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.

function `cargar_categorias()`

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

¹ Con cursor hacemos referencia al objeto generado tras ejecutar una consulta a la base de datos. `$resul = $bd->query($ins)` es un cursor.

```
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 categorías no existe, devuelve FALSE.

```
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

```
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.

```
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 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.

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**

```
function crear_correo($carrito, $pedido, $correo)
```

y

```
function 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.

```
function 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 la función `crear_correo()` para crear el cuerpo, y luego llama `enviar_correo_multiples()`

Ejercicio 1

Al añadir un producto, la aplicación redirige al carrito. Modifícala para que redirija a la tabla de productos, con la misma categoría.

Ejercicio 2

Haz los cambios necesarios para encriptar las contraseñas de la tabla de usuarios (utiliza la función `bcrypt()`)

Ejercicio 3

Modifica la tabla de productos de `productos.php` para que no muestre los productos sin stock.

Ejercicio 4

Añade el peso total del pedido en el contenido del correo de confirmación.

Ejercicio 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.

Ejercicio 6

Crea un esquema XSD para la configuración de correo y valida con él el fichero de configuración antes de usarlo.