Desarrollo de una Aplicación Web Servidor completa en PHP

Marta López de los Bueis

ÍNDICE

4 Desarrollo Aplicación Web en PHP

4.0 Objetivos de aprendizaje

Se debe de crear una aplicación de comercio electrónico, en esta ocasión para un proveedor de material de oficina para empresas.

Cargar dinámicamente los tipos de productos y de estos, la variedad de productos disponibles.

Hacer pedidos y controlar el estado del pedido mediante una variable de sesión.

Almacenar pedidos en la base de datos

Controlar el acceso a la aplicación con una tabla de usuarios(admin y empresas-cliente).

Enviar correos de confirmación de pedido y datos de factura

4.1Definición del Proyecto

Se trata de desarrollar una aplicación web para realizar pedidos business-to-business. (podemos tener como referencia los comercios online que conocemos como clientes).

En esta tarea se pretende que:

─ Unamos los elementos que hemos visto en U2 y U3 para plantear una aplicación web completa.

─ Tengamos un modelo de cómo plantear, desde cero, un proyecto de aplicación servidora web con base de datos, siguiendo una metodología adecuada.

Es un proyecto pequeño pero completo; por lo que sus fases son: análisis, diseño, implementación y pruebas.

Seguiremos el modelo de desarrollo en espiral.

En el Análisis,se define la funcionalidad de la aplicación y sus limitaciones. También se describen los datos que se quieren almacenar.

El Diseño es la parte más importante. Tiene que incluir:

- Las pantallas que verá el usuario.
- Los ficheros que forman la aplicación y cómo se pasarán los parámetros entre ellos.
- La estructura de datos para el carrito de la compra y cómo manipularla cuando el usuario añada o elimine productos, o varíe su cantidad.
- La base de datos. Se realiza un esquema entidad-relación.

En la Implementación, se escriben los ficheros php de la aplicación. La parte relacionada con la base de datos fundamental para la aplicación.

La aplicación es un comercio web sencillo, con la funcionalidad habitual de compra online. La aplicación va a simplificar algunos aspectos, por ejemplo, no hay pagos; es decir, no se paga por el pedido. Tampoco gestiona entregas; es decir, no hay que especificar la dirección de envío.

Conceptos a tener en cuenta:

• Carrito de la compra: Es un concepto habitual de los comercios online que representa el lugar en el que se almacenan los productos que el cliente va escogiendo, antes de realizar el pedido.

• Diseño lógico de la base de datos: En el modelo relacional, consiste en obtener el conjunto de tablas que forman la base de datos. Generalmente se parte de un esquema ER o un diagrama de clases.

• Especificación de requisitos del software: Descripción detallada la aplicación que se va a realizar. Se puede simplificar sin los requisitos no funcionales.

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

4.2 Análisis de Requisitos

Se quiere realizar una aplicación de pedidos entre empresas.

Las empresas cliente utilizarán la aplicación web para realizar pedidos de productos, materiales, etc; según el tipo de empresa proveedora que hayas escogido. La aplicación debe permitir:

• Consultar los tipos de productos o categorías.

• Consultar los productos.

• Añadir una o más unidades de un producto al pedido.

• Consultar el pedido del carrito y eliminar productos del carrito.

• Realizar el pedido, que supone: incorporarlo en la base de datos, enviar correos de confirmación y datos de facturación al cliente que hace el pedido y al Departamento de pedidos de la empresa.

Para acceder a la aplicación hay que autenticarse. Se supone que, en cada empresa cliente, hay un responsable de pedidos que es quien tiene el usuario y la contraseña para acceder a la aplicación.

De cada tipo o categoría de producto, hay que almacenar su, nombre, descripción y opcionalmente una imagen.

De los productos hay que almacenar su código o id, nombre, peso(si aplicara), dimensiones(si aplicara), cantidad en stock, tipo o categoría a la que pertenecen. Cada producto pertenece a una categoría.

De cada pedido se necesita conocer:

1. La empresa cliente que realizó ese pedido, con sus datos de facturación.
2. Productos que se pidieron, incluyendo la cantidad de unidades de cada producto.
3. Si ya se ha enviado o está pendiente de enviar.
4. La fecha en que se realizó el pedido.

Los pedidos se incluyen en la base de datos como no enviados. Cuando se envíen, el Departamento de Pedidos los marcará como enviados en la base de datos; o sea, que la aplicación no se ocupa esto.

De las empresas clientes se guarda la información siguiente:

a) El id

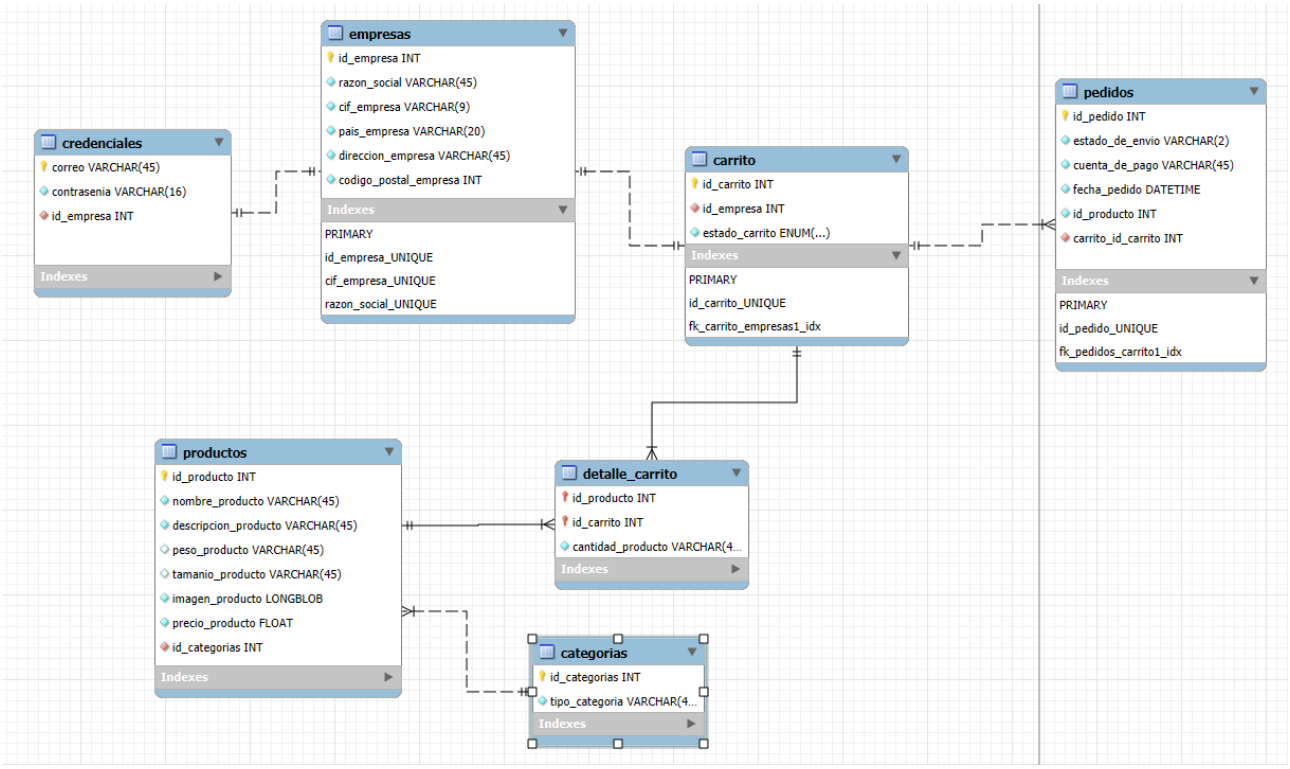
b) El correo electrónico que será el nombre de usuario para acceder a la aplicación.

c) La contraseña

d) Nombre de la razón social y CIF.

e) País, dirección, código postal.

4.2.1 Esquema entidad-relación



4.2.2 Limitaciones de la aplicación

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

No hay posibilidad de auto registro.

No se controla el stock. Si al realizar un pedido algún producto queda con stock negativo, el pedido se tramita de todas formas.

No hay pagos.

4.3 Diseño de la aplicación

A partir del análisis anterior se empieza el 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 y para la factura.

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 tener las tablas de la base de datos, se parte del esquema E-R. Primero se crea una tabla por cada entidad.

Las relaciones Realiza y Pertenece-a, implican un intercambio de claves. La tabla Producto recibirá la clave del tipo o categoría de producto y la tabla Pedido la de l a empresa cliente que lo realiza. Además, la relación Incluye N:M se resuelve mediante una tercera tabla que tenga las claves de Pedido y Producto y los atributos de la relación. Vamos a tener las tablas siguientes:

4.3.2 Diseño físico de la base de datos

Para terminar el diseño de la base de datos hay que decidir de qué tipo son los datos de las columnas. Se pide que se incluyen las siguientes características: • Los id serán números enteros con la opción de autoincremento.

• El correo de las empresas clientes y el nombre de los tipos o categorías de producto se marcarán como unique.

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

4.3.3 Diagrama de flujo de pantallas

Visualización de los productos ► index.php

Se debe estar logueado para poder visualizar la información

De primeras te ofrece redirigir a la pagina de logueo. –> login.php

Si esta logeado, permite cerrar la sesión –>logout.php

Si el usuario quiere agregar productos, redirige a carrito_insert.php

Si quiere, puede ir a ver el carrito con los productos que lleva, en el caso de no tener un carrito activo se (si el usuario aún no ha introducido ningún producto) le informa –> carrito.php

Hacer login login.php

Permite introducir las credenciales del usuario, si se introducen correctamente envía a

–>index.php

Esto utiliza las funciones de consultas.php

Hacer logout logout.php

Permite cerrar la sesión cuando el usuario lo pida. Desde index.php –> redirige a index.php

Añadir productos carrito_insert.php

Comprueba si hay un producto en el carrito, en el caso de que ya exista, actualiza la cantidad, si no existe lo añade al carrito, esto se gestiona desde –> consultas.php

Gestion de consultas consultas.php

Esto almacena la gran parte de consultas que se utilizan

Generar factura factura.php

Desde aquí se genera la factura en HTML, esta página no es vista por el usuario, redirige directamente a index.php.

Esto utiliza las funciones de consultas.php, que redirigen a –> enviar_email.php

Conexion con la base de datos conecion_bd.php

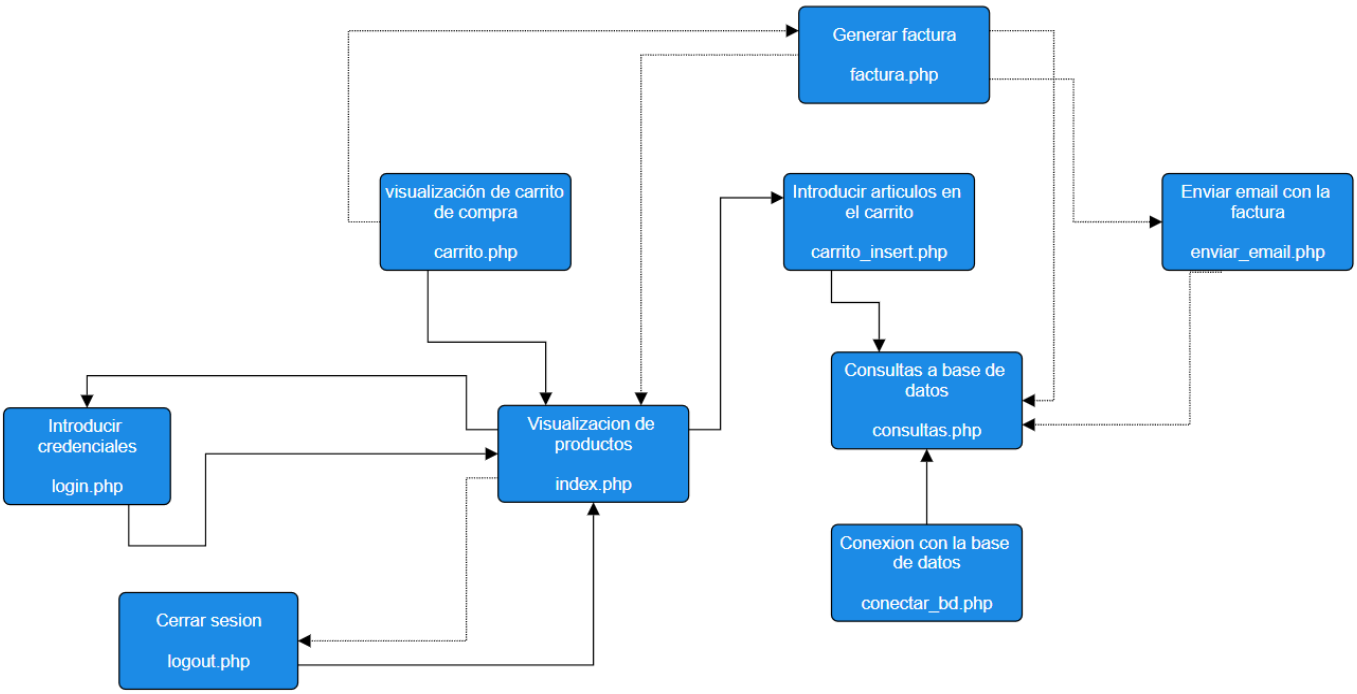
Desde aqui se crea una clase para poder conectar la base de datos, no redirige a ningun sitio

Gestionar el envio de correos enviar_email.php

Desde aqui se gestiona el envio de la factura por correo, esta pagina no refirige a ninguna otra.

Visualizacion del carrito carrito.php

Desde aqui podemos ver la cantidad de productos que tiene el usuario en su carrito, si desea finalizarlo redirige a factura.php



4.3.4 El carrito de la compra

Esta parte del proyecto, es bastante importante, ya que almacena la información de los productos que tiene el usuario (empresa) almacenado para proceder con el pedido.

Es importante que exista un estado del carrito, en este caso es “activo” o “completado”, ya que una misma empresa, puede tener varios carritos completados, pero uno solo activo, ya que es el que se está utilizando.

El estado del carrito se modifica una vez termina el proceso de compra y se envía la factura, esto se cambia después para evitar problemas en un futuro, los cuales actualmente no se contemplan, pero puede implementarse mas adelante, si se genera un error con la factura, no se perdería este estado de carrito, por lo que podría buscarse una solución.

Se crea un carrito en el momento en el que el usuario introduce un producto en el carrito, mientras tanto si el usuario quiere visualizar un carrito en el que no hay nada, se le indica que no tiene un carrito activo.

Si quiere ver el carrito y ya tiene productos en el, este le mostrará una tabla con los productos y la cantidad que lleva.

Si desea terminar el carrito, tiene la posibilidad de pulsar el boton finalizar carrito para que pueda gestionarse el pedido.

4.3.5 Control de acceso

Al entrar en la aplicación, el usuario no tiene permitido ver nada de los productos que hay en ella, ya que para ello, debe de estar logueado en la aplicación con sus credenciales.

Desde la pagina de inicio, tiene acceso a un botón que le redirige a la pagina de login.php, para que desde esta pueda loguearse y confirmar que es usuario de la aplicación.

Si se loguea correctamente, vuelve a la pagina de inicio index.php pero esta vez ya se le mostrarán todos los productos, de primera le aparecen todos los productos, pero tiene la opción de filtrar por categorías.

Igualmente si el usuario quiere cerrar la sesión, tiene un botón para poder cerrar, este redirige a logout.php pero el usuario no lo ve, y este le redirige de nuevo a la pagina de login.php.

4.3.6 Ficheros de la aplicación

Nombre	Descripcion	Parámetros	Redirige a
Index.php	Visualización de productos una vez logueado	$_POST["cambiar"] Almacena un valor enviado por form en hidden para el filtro $_POST['categoria'] almacena el valor de la categoría a mostrar $_SESSION['id_empresa'] almacena el id_empresa para comprobar que esté logueado	Login.php para iniciar sesión carrito.php para visualizar la cantidad de productos carrito_insert.php para introducir o modificar productos logout.php para cerrar la sesion
Login.php	Permite introducir las credenciales del usuario	$_POST['correo']Almacena el correo $_POST['contrasenia'] almacena la contraseña Estos son enviados por el formulario de registro	Redirige de nuevo a index,php
Logout.php	Cierra la sesión		Redirige a index.php pero no se puede visualizar nada de la pagina por que no hay sesión
Carrito.php	Muestra el carrito, en el caso de no existir uno, muestra un mensaje	$_SESSION['id_empresa']Se almacena para saber a que empresa le pertenece el carrito $_SESSION['id_carrito']Se almacena para saber que carrito mostrar	Redirige a factura.php en el caso de que quiera terminar con el carrito

Factura.php	Genera la factura del cliente, y usando funciones de consultas.php y enviar_email.php cambia el estado del carrito, almacena el pedido en la tabla pedidos y envia el email	Utuliza las variables globales que están en consultas.php	Redirige a index.php tras confirmar el pedido y enviar el correo
Enviar_email.php	Gestiona el envío del correo		No redirige, ya que se gestiona desde factura.php
Consultas.php	Almacena funciones que son utilizadas en otros ".php"	$_SESSION['id_empresa'] $_SESSION['id_carrito'] Se almacena ya que las consultas requieren de esta información	No redirige a ningún lado
carrito_insert.php	Crea o almacena los datos del carrito	$_POST['id_producto'] $_POST['cantidad'] Esto se almacena para poder conocer que producto y cuanta cantidad de este se introduce en el carrito	Redirige a index.php aun que esta pagina el usuario no la visualiza.
Conexión_bd.php	Clase que crea una conexión con la base de datos		No redirige a ningún lado

4.4 Implementación

4.4.1 Login

Este código PHP implementa un formulario de inicio de sesión básico. Se valida la identidad del usuario mediante una función externa "consultaLogin", dentro de consultas.php, incluido mediante `require_once`. Al enviar el formulario por el método POST, las credenciales del usuario se almacenan, se validan y se verifica la existencia del usuario. Si las credenciales son correctas, se redirige al usuario a index.php usando `header("Location: index.php")`. De lo contrario, se muestra un mensaje de error en rojo.

Dentro de "consultas.php", en la función "consultaLogin" se almacena en $_SESSION el id_empresa, ya se utilizará más adelante.

4.4.2 Control de acceso

Validación de sesión activa:

Antes de mostrar la información principal de la página, el código verifica si existe una variable de sesión llamada id_empresa, que se le asigna al usuario al iniciar sesión.

Si esta variable no está definida (lo que significa que el usuario no ha iniciado sesión), se muestra un mensaje indicando que debe iniciar sesión, acompañado de un botón que redirige al formulario de inicio de sesión (login.php).

Bloqueo de funcionalidad:

Si la sesión no está activa, no se ejecutan las funciones relacionadas con la consulta y visualización de productos.

Esto asegura que los usuarios no autenticados no puedan acceder a los datos ni interactuar con la página, mas que para iniciar sesión.

Control para usuarios autenticados:

Solo si $_SESSION['id_empresa'] está definido, mostrará productos según la categoría seleccionada, por defecto aparecen de primera todos los productos, y permitir añadir productos al carrito.

4.4.3 La cabecera

La cabecera del código tiene una barra de navegación básica con tres elementos principales: un botón para cerrar sesión que redirige al script logout.php, un encabezado central (<h1>) que muestra el nombre "Empresa de papelería" como título principal, y un botón que redirige a carrito.php para acceder al carrito de compras.

4.4.4 Lista de categorías

El filtro de categorías se implementa mediante un formulario HTML que permite al usuario seleccionar una categoría de productos para mostrar. El formulario incluye un <select> con opciones como "Todo", "Artículos de papelería" y "Mobiliario de oficina", y envía la selección a través del método POST. En el servidor, el valor de la categoría seleccionada almacena usando $_POST['categoria'], y según el valor, se llaman funciones específicas como “consultaObtenerTodosProductos” o “consultaObtenerProductosCategoria" para obtener los productos correspondientes de la base de datos. Si no se selecciona ninguna categoría o el formulario no se envía, por defecto se muestran todos los productos, esto lo hacemos creando $_POST["cambiar"], el cual se verifica con isset, este dato se envía con el formulario mediante un hidden, si el formulario no se pulsa, no envía nada a $_POST["cambiar"]

```php
if (!isset($_POST["cambiar"])) {
//  $consultaTodos->execute();
//  $productos = $consultaTodos->fetchAll(PDO::FETCH_ASSOC);
    echo "<h2>Todos los productos</h2>";
    $productos = consultaObtenerTodosProductos();
    mostrarProductos($productos);
}
```

Por lo que muestra todos los productos.

En el caso de que si exista algo en $_POST["cambiar"] mostraria la categoría correspondientes

```php
if($_SERVER["REQUEST_METHOD"] == "POST"){

    if(isset($_POST['categoria'])){
        $categoria=$_POST['categoria'];
    }else{
        $categoria=1;
    }



if($categoria==1 || $_SERVER["REQUEST_METHOD"] != "POST"){

    echo "<h2>Todos los productos</h2>";
    $productos = consultaObtenerTodosProductos();
    mostrarProductos($productos);

}elseif ($categoria==2 || $categoria==3) {

    echo $categoria == 2 ? "<h2>Artículos de papelería</h2>" : "<h2>Mobiliario de oficina</h2>";
    $productos=consultaObtenerProductosCategoria($categoria);
    mostrarProductos($productos);

    }
}
```

4.4.5 Tabla de productos

```php
function mostrarProductos($productos) {

    echo "<div class='contenedor'>";
    foreach ($productos as $producto) {
        echo "<div>";
        echo "<p>Nombre: " .($producto['nombre_producto']) . "</p>";
        echo "<p>Descripción: " . ($producto['descripcion_producto']) . "</p>";
        echo "<p>Peso: " . ($producto['peso_producto']) . "</p>";
        echo "<p>Tamaño: " . ($producto['tamanio_producto']) . "</p>";
        echo '<img src="' . base64_encode($producto['imagen_producto']) . '" alt="Imagen del producto" width="300px" height="300px"" />';
        echo '<form action="carrito_insert.php" method="post">';
        echo '<input type="hidden" name="id_producto" value="' . ($producto['id_producto']) . '">';
        echo '<label for="cantidad">Cantidad:</label>';
        echo '<input type="number" name="cantidad" id="cantidad" value="1" min="1">';
        echo '<input type="submit" value="Añadir al carrito">';
        echo '</form>';
        echo "</div>";
    }
    echo "</div>";
}
```

Función que muestra los productos con un formulario para añadir productos.

Con dos consultas distintas obtenemos la lista de productos que queremos mostrar en función a la categoría. De las consultas obtenemos un array asociativo el cual mostraremos por su clave.

4.4.6 Añadir productos

Este código gestiona productos en el carrito. Recibe la información del producto (id_producto y cantidad) mediante una solicitud POST. Comprueba si existe un carrito activo para la empresa

mediante “consultaCarritoActivo”, si no hay uno, lo crea con “insertarCarritoActivo” y guarda su ID en la sesión. Luego verifica si el producto ya está en el carrito usando “comprobarProductoCarrito”. Si está, actualiza la cantidad con “actualizarCantidad”, si no está, lo agrega como un nuevo producto con “insertarProductos”. Finalmente, redirige al usuario a la página principal (index.php) tras completar la operación.

Estas consultas se almacenan en funciones dentro de consultas.php.

```php
<?php

require_once "consultas.php";


if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Recibimos los datos del producto
    $id_producto = $_POST['id_producto'];
    $cantidad_producto = $_POST['cantidad'];

    // Comprobamos si ya existe un carrito activo para la empresa
    $carritoActivo = consultaCarritoActivo();

    if (!$carritoActivo) {
        // Si no existe un carrito activo, lo creamos
        insertarCarritoActivo();
        // Recuperamos el carrito recién creado desde la sesión
        $carritoActivo = consultaCarritoActivo();
    }

    // Obtenemos el ID del carrito activo
    $_SESSION['id_carrito'] = $carritoActivo['id_carrito'];

    // Comprobamos si el producto ya existe en el carrito
    if (comprobarProductoCarrito($id_producto, $id_carrito)) {
        // Si ya existe, actualizamos la cantidad del producto
        actualizarCantidad($id_producto, $cantidad_producto);
    } else {
        // Si no existe, lo insertamos como un nuevo producto en el carrito
        insertarProductos($id_producto, $cantidad_producto);
    }

    // Redirigimos al inicio después de la operación
    header('Location: ../php/index.php');
}

?>
```

```php
function consultaCarritoActivo(){

    global $base_de_datos;

    $id_empresa = $_SESSION['id_empresa'];

    $consultaCarritoActivo = $base_de_datos->prepare("SELECT id_carrito
                                                      FROM carrito
                                                      WHERE id_empresa = :id_empresa
                                                      AND estado_carrito = 'activo'");

    $consultaCarritoActivo->bindParam(":id_empresa",$id_empresa, PDO::PARAM_INT);
    $consultaCarritoActivo->execute();

    // Retorna un solo resultado si existe el carrito activo
    return $consultaCarritoActivo->fetch(PDO::FETCH_ASSOC);
}


function insertarCarritoActivo(){
    global $base_de_datos;

    $id_empresa = $_SESSION['id_empresa'];

    $insertarCarrito = $base_de_datos->prepare("INSERT INTO carrito (id_empresa, estado_carrito)
                                                VALUES (:id_empresa, 'activo')");
    $insertarCarrito->bindParam(":id_empresa", $id_empresa, PDO::PARAM_INT);
    $insertarCarrito->execute();


    /*lastInsertId() es un método de PDO que devuelve el último
    ID generado automáticamente por una consulta INSERT en una
    base de datos que utiliza claves primarias autoincrementales
    Por lo que guardamos la id_carrito en la variable sesion*/
    $_SESSION['id_carrito'] = $base_de_datos->lastInsertId();

}
```

```php
function comprobarProductoCarrito($id_producto,$id_carrito){
    global $base_de_datos;

    $consultaComprobarProductoCarrito = $base_de_datos->prepare("SELECT id_producto
                                                                 FROM detalle_carrito
                                                                 WHERE id_carrito = :id_carrito
                                                                 AND id_producto = :id_producto");
    $consultaComprobarProductoCarrito->bindParam(":id_carrito", $id_carrito, PDO::PARAM_INT);
    $consultaComprobarProductoCarrito->bindParam(":id_producto", $id_producto, PDO::PARAM_INT);
    $consultaComprobarProductoCarrito->execute();

    // Retorna true si el producto ya está en el carrito
    return $consultaComprobarProductoCarrito->fetch(PDO::FETCH_ASSOC) !== false;
}

function insertarProductos($id_producto,$cantidad_producto){
    global $base_de_datos;

    // Comprobamos si el producto ya existe en el carrito
    if (comprobarProductoCarrito($id_producto, $id_carrito)) {
        // Si ya está, actualizamos la cantidad
        actualizarCantidad($id_producto, $cantidad_producto);
    } else {
        // Si no está, insertamos el producto
        $insertarProductos = $base_de_datos->prepare("INSERT INTO detalle_carrito (id_producto, id_carrito, cantidad_producto)
                                                      VALUES (:id_producto, :id_carrito, :cantidad_producto)");
        $insertarProductos->bindParam(":id_producto", $id_producto, PDO::PARAM_INT);
        $insertarProductos->bindParam(":id_carrito", $_SESSION['id_carrito'], PDO::PARAM_INT);
        $insertarProductos->bindParam(":cantidad_producto", $cantidad_producto, PDO::PARAM_INT);
        $insertarProductos->execute();
    }
}


function actualizarCantidad($id_producto,$cantidad_producto){
    global $base_de_datos;

    $actualizarCantidad = $base_de_datos->prepare("UPDATE detalle_carrito
                                                   SET cantidad_producto = cantidad_producto + :cantidad_producto
                                                   WHERE id_carrito = :id_carrito
                                                   AND id_producto = :id_producto");
    $actualizarCantidad->bindParam(":cantidad_producto", $cantidad_producto, PDO::PARAM_INT);
    $actualizarCantidad->bindParam(":id_carrito",$_SESSION['id_carrito'], PDO::PARAM_INT);
    $actualizarCantidad->bindParam(":id_producto", $id_producto, PDO::PARAM_INT);
    $actualizarCantidad->execute();

}
```

4.4.7 El carrito de la compra

En el carrito se verifica si el usuario ha iniciado sesión correctamente mediante la existencia de la variable $_SESSION['id_empresa'], la cual ya se almacena una vez que el usuario inicia sesión. Luego, si el carrito está activo y asociado con la empresa (verificado mediante una consulta a la base de datos), esto solo ocurre cuando el usuario de la empresa introduce un articulo, mientras este no añada productos el carrito no se genera.

Se obtienen los productos almacenados en el carrito a través de una consulta SQL que une la tabla detalle_carrito con la de productos. Los productos se muestran en una tabla HTML, mostrando su

información. Al final de la página, se incluye un formulario para finalizar el carrito y proceder con la facturación.

4.4.8 Eliminar productos

4.4.9 Procesamiento del pedido

Este código PHP maneja la generación y procesamiento de una factura al finalizar un pedido realizado en el carrito de compras de una empresa. Cuando el usuario pulsa dentro del formulario en carrito.php el botón de finalizar carrito, mediante el método POST, el script obtiene los detalles de los productos del carrito mediante la función “obtenerDatosCarritoFactura()”. Luego, genera un documento HTML para la factura, mostrando información de cada producto en una tabla. Además, calcula el total de la factura sumando los subtotales de cada producto.

Después de crear la factura en HTML, el código obtiene los correos electrónicos tanto del departamento encargado como de la cuenta que realizó el pedido, mediante las funciones “obtenerCorreoDepartamento()” y “obtenerCorreoCuenta()”. El código está preparado para enviar la factura por correo electrónico a ambos destinatarios.

Una vez enviada la factura se actualiza la base de datos:

Se actualiza la tabla de pedidos con los detalles del pedido.

El estado del carrito se cambia a "completado", lo que indica que el proceso de compra ha finalizado.

El script redirige al usuario a la página principal (index.php).

4.4.10 La base de datos

Creamos una clase llamada “conectarBaseDeDatos” que permite la conexión a la base de datos utilizando datos de conexión almacenados en un archivo XML y su validación con XSD.

La clase realiza los siguientes pasos:

1. **Carga y valida los archivos XML y XSD**: En el constructor de la clase, se indican las rutas del archivo XML que contiene los datos de conexión (datos_conexion.xml) y el archivo XSD (schema.xsd). Si alguno de estos archivos no existe o si el archivo XML no se valida correctamente contra el XSD, se lanza una excepción.
2. **Crea la conexión PDO**: Si los archivos son válidos, se carga el contenido del XML usando la clase DOMDocument y se valida con schemaValidate(). Después, se llama a la función “conexionConBD()” para extraer los datos de conexión del XML y crear una instancia de PDO, que es utilizada para interactuar con la base de datos.

3. **Extracción de los datos de conexión**: La función “introducirDatos()” utiliza DOMXPath para leer los valores del XML, como el tipo de base de datos, el nombre, el host, el usuario y la contraseña. Estos datos se organizan en un array asociativo que luego se utiliza para crear la conexión PDO.
4. **Método público para obtener la conexión**: La clase tiene un método público “getConexion()”, que devuelve la instancia de PDO, permitiendo que otros scripts puedan acceder a la base de datos de manera sencilla.

4.4.11 Envío de correos

Los correos se envían una vez que el usuario confirma el pedido, esto se gestiona desde factura.php, desde donde se consiguen los correos necesarios para enviar los correos, en este caso se envían dos iguales, uno al usuario que maneja la cuenta de la empresa y otro al correo del departamento indicado.

Desde factura.php se genera lo que se envía por correo.

La función “enviarEmail()” recibe el mensaje y el destinatario, ya que los correos que hay introducidos no existen, he puesto el mio personal almacenado en la variable $address. Para enviar a las direcciones de correo de la base de datos, habría que cambiar esa variable, por la que recibe la función en $destinatario.