	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA	NOVIEMBRE 2023

## OBJETO

[Indicaciones generales:](#)

[Contenido de la práctica](#)

[Mapa conceptual](#)

[Requisitos](#)

[Login](#)

[Categorías](#)

[Bebidas](#)

[Carrito de la compra](#)

[Cierre de sesión](#)

[Esquema de Entidad-Relación](#)

[Limitaciones de la aplicación](#)

[Diseño de la aplicación](#)

[Autenticación](#)

[Ficheros de la aplicación](#)


## OBJETO

Se trata de programar una aplicación web completa en PHP, una pequeña tienda online con categorías, productos, pedidos, carrito de la compra y envío de notificaciones vía email.

## Indicaciones generales:

Toma como base el proyecto base que hemos utilizado en clase con su estructura de carpetas.

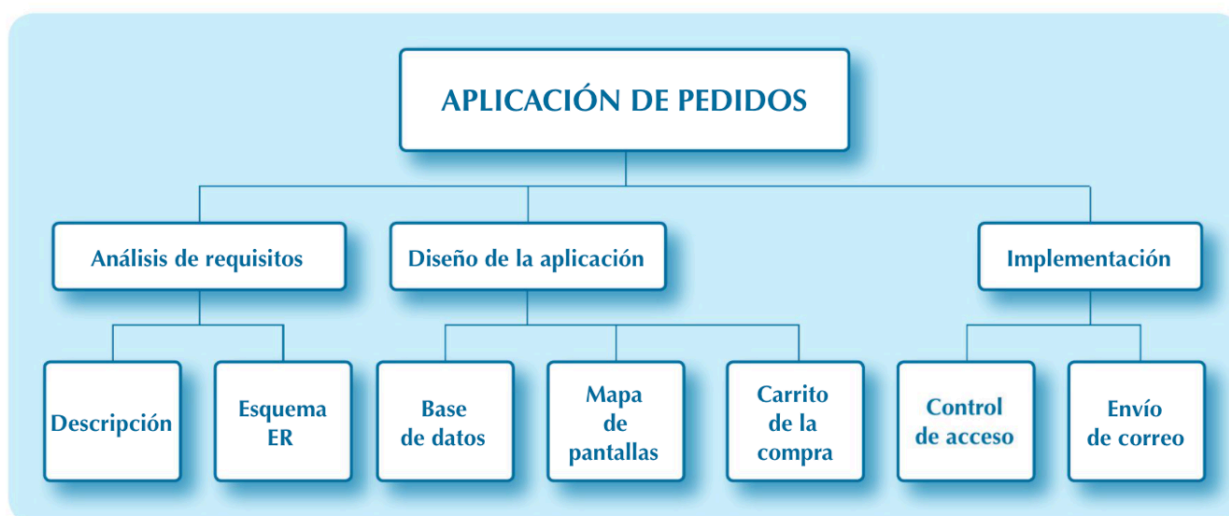
- 1) Lee detenidamente los requisitos.
- 2) Sube Práctica UD4\_AC1\_<nombre>\_<apellido>.pdf con la ejecución de la aplicación.
- 3) Sube también el código fuente a la plataforma.
- 4) Sube la bbdd (SQL).

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA	NOVIEMBRE 2023

## Contenido de la práctica

### Mapa conceptual

#### Mapa conceptual




### 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.

	<b>NOMBRE:</b> <b>APELLIDOS:</b>	<b>FECHA:</b>
	<b>MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA</b>	<b>NOVIEMBRE 2023</b>

Para acceder a la aplicación será necesario autenticarse. 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.

## Ejemplos de Pantallas


### Login

Usuario	<input type="text"/>	Clave	<input type="password"/>	<input type="button" value="Enviar consulta"/>
---------	----------------------	-------	--------------------------	--

a) Pantalla de login

### Categorías

<h2>Lista de categorías</h2> <ul style="list-style-type: none"> <li>• <a href="#">Bebidas con</a></li> <li>• <a href="#">Bebidas sin</a></li> <li>• <a href="#">Comida</a></li> </ul> <p>b) Lista de categorías</p> <p>Usuario: madrid1@empresa.com <a href="#">Home</a> <a href="#">Ver carrito</a> <a href="#">Cerrar sesión</a></p>
--

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA	NOVIEMBRE 2023

## Bebidas

### Bebidas con

Bebidas con alcohol

Nombre	Descripción	Peso	Stock		Comprar
Cerveza Alhambra tercio	24 botellas de 33cl	10	15	1	<input type="button" value="Comprar"/>
Vino tinto Rioja 0.75	6 botellas de 0.75	5.5	10	1	<input type="button" value="Comprar"/>

c) Tabla de productos

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

## Carrito de la compra


### Carrito de la compra

Nombre	Descripción	Peso	Unidades		Eliminar
Sal	20 paquetes de 1kg cada uno	20	1	1	<input type="button" value="Eliminar"/>

[Realizar pedido](#)

d) Carrito de la compra

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

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA	NOVIEMBRE 2023

## Cierre de sesión

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

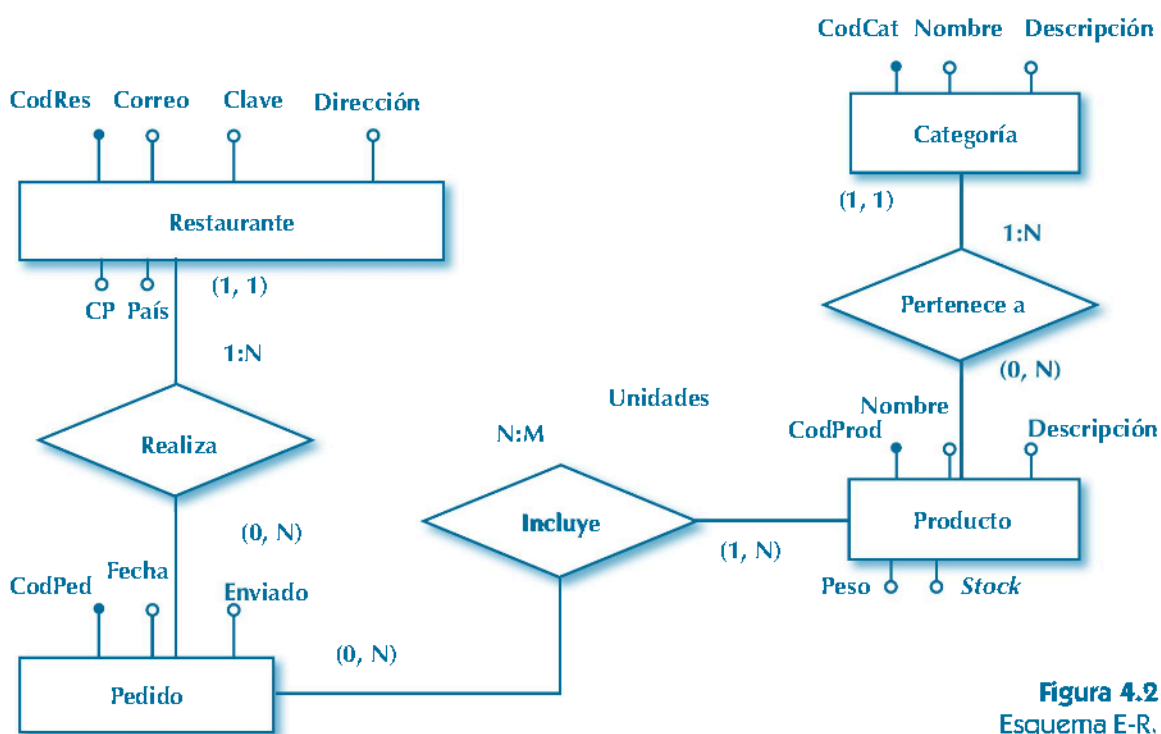
e) Confirmación del pedido

La sesión se cerró correctamente, hasta la próxima


[Ir a la página de login](#)

f) Cierre de sesión

## Esquema de Entidad-Relación



**Figura 4.2**  
Esquema E-R.

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA	NOVIEMBRE 2023

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

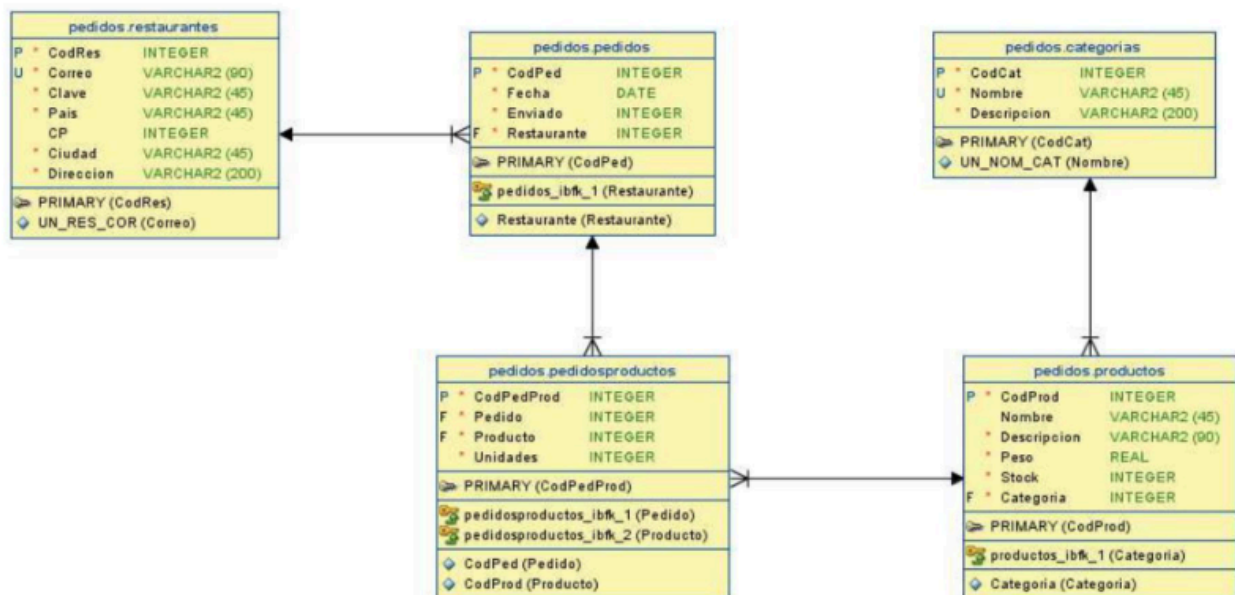
Pedidos(CodPed, Fecha, Enviado, *Restaurante*)

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.



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.

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA	NOVIEMBRE 2023

## 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.

## 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.

Las entidades del diagrama E-R son:


- Restaurante
- Categoría
- Producto
- Pedido
- (relación N:M) Pedido–Producto (incluye unidades)

En código las representaremos con las siguientes clases:

- Usuario → representa el restaurante que se autentica en la aplicación.
  - En BD está en la tabla Restaurante.
- Categoría
- Producto
- Pedido
- LineaPedido (corresponde a la tabla intermedia de la relación N:M).

A partir de estas clases implementaremos:

- La autenticación (login y logout) a través de la clase Usuario.
- La creación de pedidos a partir del carrito de la compra.
- Las operaciones básicas contra la base de datos (consultar categorías, productos, insertar pedidos, etc.).

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA	NOVIEMBRE 2023

## 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.




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



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:



	<b>NOMBRE:</b> <b>APELLIDOS:</b>	<b>FECHA:</b>
	<b>MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA</b>	<b>NOVIEMBRE 2023</b>

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:



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:




```
// clave: código de producto (int)

// valor: unidades de ese producto (int)

$_SESSION['carrito'][$codProd] = $unidades;
```

## ¿Qué ocurre si cerramos la sesión?

## Autenticación

	<b>NOMBRE:</b> <b>APELLIDOS:</b>	<b>FECHA:</b>
	<b>MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA</b>	<b>NOVIEMBRE 2023</b>

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*.

La clase **Usuario** se encargará de: (¿?)

1. **Representar** un restaurante (usuario de la aplicación).
2. **Realizar el login y el logout** usando la base de datos.

A nivel conceptual:

- En la BD tenemos la tabla Restaurante con los campos:  
CodRes, Correo, Clave, CP, Pais, Ciudad, Direccion.
- En el código tendremos la clase Usuario con atributos equivalentes.

Método estático de login, ¿se pasa el \$pdo?, ¿se le pasa usuario y contraseña desde el formulario?, ¿Qué devuelve el método?, ¿un booleano?, ¿un objeto?

De esta forma, los ficheros de la carpeta public/ solo tienen que llamar a estos métodos.


### 6.3. Clases orientadas a objetos y relaciones del modelo

Las clases que vamos a crear deben reflejar las **relaciones** del diagrama E-R, pero de forma sencilla para esta primera aproximación a la POO.

#### 6.3.1. Clase **Categoría**

Representa la entidad **Categoría**:

- Atributos:
  - \$codCat, \$nombre, \$descripcion.
- Métodos:
  - Getters y setters.

	<b>NOMBRE:</b> <b>APELLIDOS:</b>	<b>FECHA:</b>
	<b>MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA</b>	<b>NOVIEMBRE 2023</b>

- Métodos estáticos para consultar la BD, por ejemplo:
  - `public static function todas(): array` → devuelve un array de objetos `Categoria`.
  - `public static function buscarPorId(int $codCat): ?Categoria`.

### 6.3.2. Clase **Producto**

Representa la entidad **Producto** y su relación con `Categoria`:


- Atributos:
  - `$codProd`, `$nombre`, `$descripcion`, `$peso`, `$stock`, `$codCat`.
  - Para simplificar, podemos almacenar solo el código de la categoría (`$codCat`).
- Métodos:
  - Getters y setters.
  - Métodos estáticos de consulta:
    - `public static function productosDeCategoria(int $codCat): array`  
(devuelve un array de `Producto`).
    - `public static function buscarPorId(int $codProd): ?Producto`.

En futuras unidades se podría añadir un atributo `Categoria $categoria`, pero en esta fase nos basta con el código de categoría.

### 6.3.3. Clase **Pedido**

Representa la entidad **Pedido** y su relación con `Usuario` y con las líneas de pedido:

- Atributos:
  - `$codPed`, `$fecha`, `$enviado`, `$codRes`.
  - `$lineas`: array de objetos `LineaPedido`.

	<b>NOMBRE:</b> <b>APELLIDOS:</b>	<b>FECHA:</b>
	<b>MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA</b>	<b>NOVIEMBRE 2023</b>

- Métodos:
  - Getters y setters.
  - Método para añadir líneas al pedido:
    - `public function anadirLinea(LineaPedido $linea).`
  - Métodos relacionados con la BD:
    - `public function guardar(): void`
      - Inserta el pedido en la tabla `Pedido`.
      - Inserta sus líneas en la tabla intermedia (ver clase `LineaPedido`).

De nuevo, para simplificar, en lugar de guardar un objeto `Usuario` dentro de `Pedido`, almacenaremos solo el código del restaurante (`$codRes`).


#### 6.3.4. Clase `LineaPedido`

Corresponde a la relación N:M “**Incluye**” entre `Pedido` y `Producto`:

- Atributos:
  - `$codPedProd`, `$codPed`, `$codProd`, `$unidades`.
- Métodos:
  - Getters y setters.
  - Un método `guardar(PDO $pdo)` que inserte la línea en la tabla correspondiente.

#### 6.3.5. Resumen de las relaciones en POO

- **Restaurante–Pedido (1:N)**
  - Se modela guardando `codRes` dentro de `Pedido`.
- **Categoría–Producto (1:N)**
  - Se modela guardando `codCat` dentro de `Producto`.
- **Pedido–Producto (N:M)**

	NOMBRE: APELLIDOS:	FECHA:
	MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA	NOVIEMBRE 2023

- Se modela con la clase `LineaPedido`, que guarda `codPed` y `codProd`.
- El **carrito** no aparece en el diagrama E-R porque es una estructura temporal; en esta unidad seguirá siendo **un array en la sesión**, no un objeto ni una tabla.

## Ficheros de la aplicación

Trabaja con programación orientada a objetos.


## 7. FICHEROS DE LA APLICACIÓN (VERSIÓN POO SIN DAO)

En esta sección se indica **qué ficheros deben crear los alumnos** y qué responsabilidad tiene cada uno, utilizando la plantilla del proyecto (Composer y estructura de carpetas).

### 7.1. Estructura de directorios

Se utilizará la estructura que proporciona el profesor:

- `config/`
  - `config.ini` → configuración de la base de datos, idioma, etc.
- `public/`
  - Ficheros que se llaman desde el navegador (páginas y controladores sencillos).
- `src/`
  - Clases de la aplicación (modelo + lógica + acceso a BD).
- `tools/`
  - Clases de apoyo ya dadas:

	NOMBRE:	FECHA:
	APELLIDOS:	
	MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA	NOVIEMBRE 2023


- `Conexion.php`, `Config.php`, `HtmlHelpers.php`, `Mailer.php`, `Validador.php`, ...

- `vendor/`
  - Directorio generado por Composer.
- `composer.json`
  - Configuración de autoload y dependencias.


## 7.2. Ficheros PHP en `public/`

Los alumnos deben crear (o completar) los siguientes ficheros:

- `index.php`
  - Página de inicio.
  - Puede mostrar un enlace al login o redirigir directamente a `login.php`.
- `login.php`
  - Muestra el formulario de acceso (correo y clave).
  - Si el formulario se envía por POST, llama a `Usuario::login($correo, $clave)`:
    - Si devuelve un objeto `Usuario`, redirige a `categorias.php`.
    - Si devuelve `null`, muestra un mensaje de error.
- `logout.php`
  - Llama a `Usuario::logout()` y muestra un mensaje de salida.
- `categorias.php`
  - Comprueba que haya un usuario autenticado (revisando la sesión).

	<b>NOMBRE:</b> <b>APELLIDOS:</b>	<b>FECHA:</b>
	<b>MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA</b>	<b>NOVIEMBRE 2023</b>

- Llama a `Categoria::todas()` para obtener la lista de categorías.
- Muestra la lista con enlaces a `productos.php?codCat=...`
- `productos.php`
  - Recibe el parámetro `codCat`.
  - Comprueba que el usuario esté autenticado.
  - Utiliza `Producto::productosDeCategoria($codCat)` para obtener los productos.
  - Muestra los productos y un formulario para añadir unidades al carrito.
- `carrito.php`
  - Comprueba que el usuario esté autenticado.
  - Inicializa el carrito en la sesión si no existe.
  - Muestra el contenido de `$_SESSION['carrito']`.
  - Permite modificar unidades o eliminar productos.
- `anadir_carrito.php`
  - Recibe por POST `codProd` y `unidades`.
  - Inicializa el carrito si es necesario y actualiza `$_SESSION['carrito']`.
  - Redirige a `carrito.php` o `productos.php`.
- `eliminar_carrito.php`
  - Recibe `codProd`.
  - Elimina ese índice del array `$_SESSION['carrito']`.
  - Redirige a `carrito.php`.

	<b>NOMBRE:</b> <b>APELLIDOS:</b>	<b>FECHA:</b>
	<b>MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA</b>	<b>NOVIEMBRE 2023</b>

- `confirmar_pedido.php`
  - Comprueba que el usuario esté autenticado.
  - Crea un objeto `Pedido` con los datos del usuario actual.
  - Recorre `$_SESSION['carrito']` y va creando objetos `LineaPedido`.
  - Llama al método `$pedido->guardar()` para almacenar el pedido y sus líneas en la base de datos.
  - Utiliza `Mailer` para enviar los correos de confirmación.
  - Vacía el carrito (`$_SESSION['carrito'] = [];`).
  - Muestra un mensaje de pedido realizado.


### 7.3. Clases en `src/`

Cada clase irá en su propio fichero, con el mismo nombre que la clase:

- `src/Usuario.php`
- `src/Categoria.php`
- `src/Producto.php`
- `src/Pedido.php`
- `src/LineaPedido.php`

Cada una de estas clases:

1. **Representa** una tabla o relación del modelo de datos (atributos privados).
2. Tiene **métodos de acceso** (getters y setters).
3. Implementa métodos estáticos y/o de instancia para realizar las consultas SQL necesarias, utilizando la clase `Conexion`:
  - En `Usuario`:

	<b>NOMBRE:</b> <b>APELLIDOS:</b>	<b>FECHA:</b>
	<b>MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA</b>	<b>NOVIEMBRE 2023</b>


- `public static function login(string $correo, string $clave): ?Usuario`
- `public static function logout(): void`
- En `Categoria`:
  - `public static function todas(): array`
  - `public static function buscarPorId(int $codCat): ?Categoria`
- En `Producto`:
  - `public static function productosDeCategoria(int $codCat): array`
  - `public static function buscarPorId(int $codProd): ?Producto`
- En `Pedido`:
  - `public function guardar(): void`  
(inserta en la tabla `Pedido` y llama a `guardar()` de cada `LineaPedido`).
- En `LineaPedido`:
  - `public function guardar(PDO $pdo): void`

## Los métodos que manejan la persistencia ¿Deberían devolver `bool`, una excepción, o `void`?

Aquí hay tres estilos típicos:

### 1. Devolver `bool` (`true/false`)

- Ventaja: muy fácil de entender para el alumnado.

	NOMBRE: APELLIDOS:	FECHA:
	MÓDULO DWES MULTI3 - UD4 AC1 APLICACIÓN COMPLETA	NOVIEMBRE 2023

- Inconveniente: hay que acordarse siempre de comprobar el resultado.

## 2. Lanzar excepciones en caso de error y no devolver nada útil (`void`)

Se configura PDO para que lance `PDOException` si algo va mal:

```
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

- 
- Si no hay excepción, asumimos que ha ido bien.
- El `catch` suele hacerse en los scripts públicos.

## 3. Mezcla: devolver algo útil (por ejemplo, el id insertado) y además lanzar excepciones en errores graves.