Práctica: “El Catálogo Conectado v3 – Backend Real con PHP y MySQL”

Objetivo general

Modificar el *Catálogo de Productos* para que funcione contra una **base de datos real MySQL**, gestionada mediante **PHP en AwardSpace**, utilizando **peticiones asíncronas (fetch)** desde JavaScript.
El alumno deberá comprender y aplicar la asincronía real derivada de la comunicación cliente–servidor.

Contexto: De la simulación a la realidad

Hasta ahora, vuestro catálogo:

- Funcionaba en memoria (arrays JS).
- La validación del ID y las operaciones CRUD eran **instantáneas y síncronas**.
- No existía latencia, ni errores de red, ni respuesta del servidor.

En un entorno profesional **esto no es realista**.

Ahora el cliente exige que:

- Los productos se almacenen en una **base de datos central**.
- El frontend JavaScript se comunique con el backend PHP **de forma asíncrona**.
- La aplicación siga siendo usable mientras el servidor responde.

---

1. Justificación de la Asincronía: PHP + Base de Datos

Cuando JavaScript envía datos a PHP:

- El navegador **no puede esperar bloqueando el hilo principal**.
- PHP necesita tiempo para:
    - Conectarse a MySQL
    - Ejecutar consultas
    - Validar restricciones (ID único)
    - Devolver una respuesta JSON

Por ello, el uso de **fetch + Promesas / async–await** es obligatorio.

Con esta práctica aprenderás a:

- Esperar respuestas del servidor con await fetch(...)
- Separar validaciones:
    - **Síncronas (frontend)**: formato, campos vacíos
    - **Asíncronas (backend)**: ID duplicado, errores SQL

- Mantener la interfaz activa mientras el servidor procesa la petición

---

2. Experiencia de Usuario (UX) en entornos reales

Cuando el backend tarda en responder:

- El usuario **debe recibir feedback**
- La interfaz **no debe permitir acciones duplicadas**

Se requiere implementar:

- Botón **"Guardando…"** deshabilitado mientras PHP procesa
- Tarjetas **opacas** durante el borrado
- Mensajes de éxito y error basados en la respuesta del servidor
- Restauración visual controlada mediante:
    - .then()
    - .catch()
    - .finally()
    - o try / catch / finally

---

3. Manejo avanzado de errores (Frontend + Backend)

Ahora existen **dos niveles de error claramente diferenciados**:

Errores síncronos (Frontend – JS)

- Campos vacíos
- Precio no numérico
- URL de imagen inválida (antes de enviar)

Errores asíncronos (Backend – PHP)

- ID duplicado en la base de datos
- Error de conexión MySQL
- Error al borrar un producto inexistente

El alumno deberá interpretar correctamente las respuestas JSON del servidor.

Práctica Guiada

"El Catálogo Conectado con PHP y MySQL"

Tarea 1: Preparar el Backend en AwardSpace (PHP + MySQL)

1.1 Base de datos

Crear una base de datos con una tabla productos:

1.2 API en PHP (backend)

Crear un archivo api.php que:

- Reciba peticiones mediante fetch
- Responda **siempre en JSON**
- Distinga acciones mediante $_GET['action'] o JSON recibido

Acciones mínimas:

1. **guardar**
    - Comprueba si el ID ya existe
    - Inserta el producto si no existe
    - Devuelve:
        - { success: true, message: "Producto guardado" }
        - { success: false, error: "ID duplicado" }
2. **borrar**
    - Elimina el producto por ID
    - Devuelve error si no existe

Tarea 2: Alta de productos (Asincronía real con fetch)

Requisitos:

- Al pulsar **“Guardar”**:
    - Deshabilitar botón
    - Mostrar “Guardando…”
- Enviar los datos a api.php usando fetch
- Usar async / await

Flujo obligatorio:

- Validación síncrona → JS
- Petición asíncrona → PHP
- Interpretación de la respuesta JSON
- Actualización del DOM **solo si el backend confirma**

Tarea 3: Borrado asíncrono con feedback visual

- Click derecho → “Eliminar”
- La tarjeta:
    - Se vuelve semitransparente
- Se envía petición fetch a PHP
- Solo si PHP responde OK:
    - Se elimina el nodo del DOM
- Si falla:
    - Se restaura la opacidad
    - Se muestra un alert con el error

Tarea 4: Validación de imágenes antes de enviar

Crear una función:

```
function validarImagen(url) {
  return new Promise((resolve, reject) => {
    const img = new Image();
    img.onload = () => resolve();
    img.onerror = () => reject("La imagen no se puede cargar");
    img.src = url;
  });
}
```

- Debe ejecutarse **antes del fetch**
- Si falla:
    - No se envía nada al servidor
    - Se muestra el error al usuario