# Despliegue de Aplicaciones Web

## Análisis Profundo del Script: setup.php

Clase de Desarrollo Web (UD7)

8 de enero de 2026

# Índice de Contenidos

# 1.1. La Importancia de no "Quemarlo" todo

Antes de conectar, definimos las credenciales en variables.

## ¿Qué es "Hardcoding" (Mala práctica)?

Es escribir los datos directamente en el código (ej: poner la contraseña dentro del comando de conexión).

## ¿Por qué usamos variables (Buena práctica)?

- **Mantenibilidad:** Si cambiamos de servidor, solo editamos 5 líneas al principio.
- **Seguridad:** Facilita mover estas variables a un archivo externo oculto (`config.php`) en el futuro.

# 1.2. El Código: La Agenda de Contactos

Este bloque define **a quién** vamos a llamar y **con qué credenciales**.

```php
// 1. CONFIGURACION (setup.php)
$host    = 'localhost';       // ¿Dónde está la BD?
$user    = 'root';            // ¿Quién llama?
$pass    = '';                // ¿Cuál es la llave?
$dbName  = 'tienda_ud7';      // ¿A qué habitación vamos?
$csvFile = 'datos.csv';       // ¿Dónde están los datos?
```

*Nota: Estas líneas no hacen nada por sí solas, solo preparan la información para usarla después.*

# 1.3. Concepto Clave: Localhost

**¿Qué es `$host`?**

- **Localhost** significa .Estе mismo ordenador".
- Traducción IP: `127.0.0.1`.
- Le dice a PHP: "No salgas a Internet, la base de datos está aquí instalada".

**Usuario Root (`$user`)**

- Es el "Super-Administrador"de MySQL.
- Tiene poder total (Crear, Borrar, Ver).
- **OJO:** En XAMPP la contraseña (`$pass`) viene vacía por defecto para facilitar el aprendizaje.

# 1.4. ¡CRÍTICO! El Problema del Puerto

MySQL suele vivir en el puerto **3306**. Pero a veces, XAMPP lo mueve al **3307** si el primero está ocupado.

## Solución si falla la conexión

Si en tu panel de XAMPP ves **Port: 3307**, el código **DEBE** cambiarse así:

```
// Añadimos dos puntos y el puerto
$host = 'localhost:3307';
```

Si no especificamos el puerto, PHP llamará a la puerta equivocada y nadie abrirá (Error: *Connection Refused*).

# 2.1. ¿Qué es PDO?

Para que PHP hable con la base de datos, necesitamos un traductor.

## PDO (PHP Data Objects)

Es una librería de PHP que actúa como un ".enchufe universal".

- Nos permite conectarnos a MySQL, pero también a PostgreSQL, SQLite, etc.
- Es más seguro que las funciones antiguas (`mysql_connect`) porque evita hackers (Inyección SQL).

**La Analogía:** Imagina que `$host` es el número de teléfono y **PDO** es el aparato telefónico que realiza la llamada.

# 2.2. El Código: Estableciendo la Línea

Aquí creamos el objeto (el ”cable”) que mantendrá la conexión abierta.

```php
// 2. CONEXIÓN (setup.php)
try {
    // A. El DSN (Origen de datos)
    $dsn = "mysql:host=$host;charset=utf8mb4";

    // B. Crear la instancia (Llamar al servidor)
    $pdo = new PDO($dsn, $user, $pass);

    // C. Activar modo de pánico (Errores)
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

} catch (PDOException $e) {
    die("Error de conexión: " . $e->getMessage());
}
```

# 2.3. Análisis: El DSN y la Base de Datos Fantasma

Fíjate en esta línea con lupa:

```
$dsn = "mysql:host=$host;charset=utf8mb4";
```

**¡Falta algo importante!**

**¿Dónde está `dbname=tienda_ud7`?**
NO lo ponemos todavía. Es intencionado.

**¿Por qué?** Porque si es la primera vez que ejecutamos el script, la base de datos **no existe**. Si intentamos conectar a algo que no existe, el script explota ("Fatal Error") antes de poder crearla.
*Estrategia:* Conectamos al "Hall del Hotel"(Servidor), y luego construiremos la habitación.

# 2.4. El Çhivato"de Errores (setAttribute)

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

**Sin esta línea (Mudo)**

- Si MySQL falla, PHP se queda callado.
- La página sale en blanco y te vuelves loco buscando el error.

**Con esta línea (Gritón)**

- Si hay un fallo, PHP lanza una **Excepción**.
- El script se detiene y nos muestra el mensaje exacto del error en pantalla.

[Image of sequence diagram showing PHP throwing a PDOException when MySQL query fails]

# 3.1. Infraestructura como Código

Ya estamos dentro del servidor. Ahora toca construir el edificio.

## Objetivo

No queremos abrir phpMyAdmin y crear la base de datos a mano. Queremos que el script lo haga por nosotros.

- **Automatización:** Si le pasas este script a un compañero, se le crea todo solo.
- **Idempotencia:** Propiedad clave que evita errores si ejecutamos el script múltiples veces.

# 3.2. Creación de la Base de Datos

Usamos el método `exec()` porque son órdenes de estructura (DDL) que no devuelven datos, solo "éxito o fracaso".

```php
// 3. CREAR LA BD (Infraestructura)

// A. Crear el contenedor (Si no existe)
$sqlDB = "CREATE DATABASE IF NOT EXISTS $dbName
          CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci";
$pdo->exec($sqlDB);

// B. Entrar en él
$pdo->exec("USE $dbName");
```

## Detalle: USE $dbName

Es CRÍTICO ejecutar `USE`. Al principio conectamos al "pasillo". Con esto, entramos en la "habitación". Si olvidas esto, la tabla se intentará crear en el pasillo y dará error.

# 3.3. Concepto Clave: Idempotencia

Fíjate en la cláusula: `IF NOT EXISTS`.

**Sin Idempotencia**

- Ejecutas el script: **Bien**.
- Recargas la página (F5): **ERROR FATAL**. "Database already exists".
- El script se rompe.

**Con Idempotencia**

- Ejecutas el script: **Bien**.
- Recargas la página: **Bien**. MySQL ve que existe y no hace nada.
- El script es robusto.

# 3.4. Creación de la Tabla

Definimos las columnas y, muy importante, el motor de almacenamiento.

```
$sqlTabla = "CREATE TABLE IF NOT EXISTS productos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    precio DECIMAL(10,2) NOT NULL,
    stock INT DEFAULT 0
) ENGINE=InnoDB";

$pdo->exec($sqlTabla);
```

## ¿Por qué ENGINE=InnoDB?

Es **obligatorio** para poder usar **Transacciones** en la fase final. Otros motores antiguos (como MyISAM) no soportan el "deshacer cambios" (Rollback).

# 3.5. Limpieza del Terreno

Antes de meter los datos nuevos, nos aseguramos de que la tabla esté vacía y limpia.

```
// Vaciar la tabla por completo
$pdo->exec("TRUNCATE TABLE productos");
```

## TRUNCATE vs DELETE

- **DELETE:** Borra fila a fila. Lento. No reinicia el contador de IDs (el siguiente sería el 51).
- **TRUNCATE:** Tira la tabla y la reconstruye. Instantáneo. **Reinicia el ID a 1**.

# 4.1. El Concepto de Transacción

Vamos a insertar miles de productos. ¿Cómo lo hacemos rápido y seguro?

## Transacción (ACID)

Es un conjunto de operaciones que se tratan como **una sola**.

- **Todo o Nada:** O se guardan los 1.000 productos, o no se guarda ninguno.
- **Velocidad:** En lugar de escribir en el disco duro 1.000 veces, escribimos en la RAM y volcamos al disco una sola vez al final.

**Analogía:** Es como el carrito de la compra. No pagas producto a producto. Llenas el carro y pagas todo junto al salir.

# 4.2. Inicio de la Transacción

Abrimos el archivo y congelamos.[el] guardado automático.

```php
// 4. CARGA MASIVA (setup.php)

// A. Abrir el archivo CSV en modo Lectura ('r')
$gestor = fopen($csvFile, "r");

// B. INICIAR TRANSACCI N (Velocidad extrema)
// Le decimos a MySQL: "No guardes nada en disco todavía"
$pdo->beginTransaction();
```

## Impacto en Rendimiento

Insertar 1.000 filas una a una: **5 segundos**.
Insertar 1.000 filas con transacción: **0.1 segundos**.

# 4.3. Sentencias Preparadas (Seguridad)

Compilamos la consulta SQL **fuera** del bucle.

```
// C. La Plantilla SQL (Prepared Statement)
$stmt = $pdo->prepare(
    "INSERT INTO productos (nombre, precio, stock) VALUES (?, ?, ?)"
);
```

**Eficiencia**

- MySQL compila la frase una sola vez.
- Luego solo rellenamos los huecos (?).

**Seguridad (Anti-Hackers)**

- Los marcadores ? evitan la **Inyección SQL**.
- Separan el código de los datos.

# 4.4. El Bucle de Lectura

Leemos el CSV línea a línea e inyectamos los datos.

```php
// Saltamos la primera línea (cabeceras del CSV)
fgetcsv($gestor);

// Leemos hasta que se acabe el archivo
while (($datos = fgetcsv($gestor, 1000, ",")) !== FALSE) {

    // $datos es un array: [0]=>Nombre, [1]=>Precio...

    // Ejecutamos la plantilla con los datos reales
    $stmt->execute([ $datos[0], $datos[1], $datos[2] ]);
}
```

En este punto, los datos están en la memoria RAM de MySQL, **pero NO en el disco duro todavía**. Si se va la luz ahora, se pierde todo.

# 4.5. El Momento de la Verdad

Confirmamos o deshacemos los cambios.

```
    // D. GUARDAR TODO (Volcar RAM a Disco)
    $pdo->commit();
    echo "Éxito: Datos cargados correctamente.";

} catch (Exception $e) {
    // SI ALGO FALLA...
    $pdo->rollBack(); // DESHACER (Z)
    echo "Error: " . $e->getMessage();
} finally {
    fclose($gestor); // Cerrar archivo siempre
}
```

## Rollback

Es el botón de "Deshacer". Si el CSV falla en la línea 999, `rollBack()` borra las 998 anteriores para que la base de datos no quede corrupta a medias.