Cuaderno Técnico: Desarrollo Web

Arquitectura Servidor • PHP • Programación Orientada a Objetos

📝 Datos del Alumno

Nombre Completo:

Antonio Cordero

Fecha de Entrega:

07/12/2025

1. Arquitectura y Entorno de Trabajo

Comprender dónde se ejecuta nuestro código y cómo configurarlo es el primer paso para ser desarrollador.

🚀 Reto 1: El viaje de la Petición INVESTIGACIÓN

Un usuario escribe `www.mitienda.com` en su navegador. Describe técnicamente los 4 pasos que ocurren hasta que ve la web. Debes usar términos como *DNS, Petición HTTP, Servidor Web (Apache) y Renderizado*.

El usuario escribe www.mitienda.com en su navegador. Los 4 pasos técnicos que ocurren hasta que se ve la web son:
- Resolución DNS: el navegador consulta un servidor DNS para obtener la dirección IP del dominio. Unavez resuelto, sabe a qué servidor tiene que conectarse.
- Petición HTTP: el navegador establece una conexión TCP con la IP del servidor y envía una petición HTTP al servidor web Apache
- Servidor Web Apache: el servidor Apache recibe la pertición, interpreta la URL, busca el rescurso solicitado y ejecuta el código PHP generando una respuesta con html
- Renderizado: el servidor envía la respuesta HTTP al navegador, qu ela recibe, interpreta y muestra la página al usuario.

🛠️ Reto 2: Despliegue con Git PRÁCTICA

Imagina que te incorporas a un proyecto ya empezado. Escribe la secuencia exacta de comandos de consola para:

1. Descargar el código a tu ordenador.
2. Crear una rama de trabajo llamada `mi-funcionalidad`.
3. Subir tus cambios al repositorio remoto.

1. Descargar el archivo al ordenador: git clone ruta-del-repositorio.git
2. Crear una rama de trabajo: git checkout -b mi-funcionalidad
3. Subir cambios al repositorio:
   git add .
   git commit -m "mis cambios"
   git push -u -origin mi-funcionalidad

2. Lógica de Programación y Algoritmos

El servidor toma decisiones. Aquí practicaremos cómo controlar el flujo de la información.

🕵️ Reto 3: Debugging Mental ANÁLISIS

Analiza el siguiente código sin ejecutarlo. ¿Qué imprimirá por pantalla y **por qué**?

```
$x = "50";
$y = 50;
if ($x === $y) {
    echo "Son idénticos";
} elseif ($x == $y) {
    echo "Son equivalentes";
} else {
    echo "Son distintos";
}
```

Imprimirá: Son equivalentes, porque con === compara valor y tipo, string y entero no entran en el primer if, pero con == solo compara el valor y "50" y 50 los considera iguales.

Reto 4: La Taquilla Inteligente CÓDIGO

Crea un script PHP que calcule el precio de una entrada de cine (Precio base: 8€).

- Si es miércoles: -2€.
- Si tiene "Carnet Joven": -10%.
- Ambos descuentos son acumulables.

```php
<?php
   $precioBase = 8;
   $esMiercoles = true;
   $tieneCarnetJoven = false;

   $precio = $precioBase;

   if ($esMiercoles) {
      $precio *= 0.8;
   }

   if ($tieneCarnetJoven) {
      $precio *= 0.9;
   }

   echo "Precio final: $precio €";
?>
```

🔢 Reto 5: Tabla de Multiplicar Dinámica LÓGICA ANIDADA

No queremos una lista simple. Genera una **Tabla HTML (`<table>`)** completa que muestre las tablas de multiplicar del 1 al 10.

- Usa bucles `for` anidados (uno para filas, otro para celdas).
- El cruce de fila/columna debe mostrar el resultado.
- Las celdas donde el resultado sea par deben tener fondo gris (CSS inline).

```
<?php
   echo "<table border='1' cellpadding='5'>";

   for ($fila = 1; $fila <= 10; $fila++) {
      echo "<tr>";
      for ($col = 1; $col <= 10; $col++) {
         $resultado = $fila * $col;
         $style = ($resultado % 2 == 0) ? " style='background-color: #ddd;'" : "";
         echo "<td$style>$resultado</td>";
      }
      echo "</tr>";
   }

   echo "</table>";
?>
```

3. Estructuras de Datos (Arrays)

La información real viene en listas y colecciones. Aprende a manipularlas.

📋 Reto 6: Gestión de Inventario ARRAYS SIMPLES

Tienes un array: `$stock = [10, 5, 20, 0, 15];`

Escribe un script que:

1. Elimine los productos con stock 0 (usa unset).
2. Ordene la lista de mayor a menor cantidad (usa funciones de ordenación).
3. Añada un nuevo producto con stock 50 al final.

```php
<?php
   $stock = [10, 5, 20, 0, 15];

   foreach ($stock as $indice => $cantidad) {
      if ($cantidad == 0) {
         unset($stock[$indice]);
      }
   }

   sort($stock);
   $stock = array_reverse($stock);
   $stock[] = 50;

   print_r($stock);
?>
```

🎓 Reto 7: Boletín de Notas Digital MULTIDIMENSIONAL

Gestiona las notas de 3 alumnos usando un **Array Multidimensional Asociativo**.

Estructura: `Nombre Alumno => [Asignatura => Nota]`.

Debes recorrerlo con foreach e imprimir una ficha HTML por alumno que muestre su **Nota Media**. Si la media es suspenso (< 5), el nombre debe salir en rojo.

```php
<?php
$alumnos = [
    "Aitor Menta" => [
        "Matemáticas" => 7,
        "Lengua"      => 4,
        "Inglés"      => 6
    ],
    "Elsa Pato" => [
        "Matemáticas" => 3,
        "Lengua"      => 5,
        "Inglés"      => 4
    ],
    "Esteban Quito" => [
        "Matemáticas" => 9,
        "Lengua"      => 8,
        "Inglés"      => 7
    ]
];

foreach ($alumnos as $nombre => $notas) {
    $suma = 0;
    $numAsignaturas = 0;

    foreach ($notas as $asignatura => $nota) {
        $suma += $nota;
        $numAsignaturas++;
    }

    $media = $suma / $numAsignaturas;

    $colorNombre = ($media < 5) ? "red" : "black";

    echo "<div style='border:1px solid #ccc; margin:10px; padding:10px;'>";
    echo "<h3 style='color:$colorNombre;'>$nombre</h3>";
    echo "<p>Nota media: $media</p>";
    echo "</div>";
}
?>
```

4. Programación Orientada a Objetos (POO)

El estándar de la industria. Modelamos la realidad con Clases y Objetos seguros.

🔒 Reto 8: Seguridad de Datos CONCEPTO

¿Por qué definimos las propiedades de una clase como `private` en lugar de `public`? Explica qué problema evita esto y cómo accedemos a los datos entonces (Getters/Setters).

Definimos las propiedades como private para que no puedan modificarse libremente desde fuera de la clase.
Se evita la inconsistencia al cambiar un dato sin validar y se evita saltarse las reglas de la aplicación como poner datos que no corresponden.

En lugar de acceder directo ($obj->propiedad), se usan los métodos públicos getter y setter
- Getter: devuelve el valor con getPropiedad(), es solo para lectura.
- Setter: puede cambiar el valor con setPropiedad($valor) aplicando validaciones si es necesario.

🏦 Reto 9: Simulador Bancario POO COMPLETA

Crea una clase `CuentaBancaria` en un archivo aparte.

- **Propiedades:** titular (public), saldo (private).
- **Constructor:** Inicia titular y saldo.
- **Método retirar($cantidad):** Resta saldo solo si hay suficiente dinero (if). Si no, muestra error.
- **Script de prueba:** Crea una cuenta con 100€, intenta sacar 200€ (error) y luego saca 50€ (éxito).

```php
<?php
   class CuentaBancaria {
      public $titular;
      private $saldo;

      public function __construct($titular, $saldoInicial) {
         $this->titular = $titular;
         $this->saldo   = $saldoInicial;
      }

      public function retirar($cantidad) {
         if ($cantidad <= $this->saldo) {
            $this->saldo -= $cantidad;
            echo "Retirados $cantidad €. Saldo actual: {$this->saldo} €<br>";
         } else {
            echo "Error: saldo insuficiente. Saldo actual: {$this->saldo} €<br>";
         }
      }
   }
?>

<?php

   require_once 'CuentaBancaria.php';

   $cuenta = new CuentaBancaria("Pepe", 100);

   echo "Titular: {$cuenta->titular}<br>";
   echo "(imprime error)<br>";
   $cuenta->retirar(200);

   echo "(imprime éxito)<br>";
   $cuenta->retirar(50);
?>
```

Reto 10: Juego de Cartas RETO FINAL

Combina Arrays y POO. Crea una clase `Carta` (palo, número).

Genera una baraja de 40 cartas usando bucles. Barájala aleatoriamente (busca función `shuffle`) y reparte las 5 primeras cartas en pantalla.

```php
<?php
    class Carta {
        public $palo;
        public $numero;

        public function __construct($palo, $numero) {
            $this->palo = $palo;
            $this->numero = $numero;
        }

        public function mostrar() {
            return "$this->numero de $this->palo";
        }
    }

    $palos = ["oros", "copas", "espadas", "bastos"];
    $baraja = [];

    foreach ($palos as $palo) {
        for ($num = 1; $num <= 10; $num++) {
            $baraja[] = new Carta($palo, $num);
        }
    }

    shuffle($baraja);

    for ($i = 0; $i < 5; $i++) {
        echo $baraja[$i]->mostrar() . "<br>";
    }
?>
```

🏆 Rúbrica de Evaluación

Así es como se evaluará tu trabajo. Revisa esta tabla antes de entregar.

Criterio	Nivel Básico (Insuficiente)	Nivel Medio (Aceptable)	Nivel Pro (Excelente)
Sintaxis PHP	Errores frecuentes (falta `;`, `$`). Código desordenado.	Código funcional. Indentación básica correcta.	Código limpio, comentado y siguiendo estándares.
Lógica	Uso excesivo de variables innecesarias. Copia/pega evidente.	Uso correcto de bucles y condicionales básicos.	Lógica eficiente. Uso de bucles anidados y validaciones.
Datos	Solo usa variables sueltas. No entiende arrays.	Maneja arrays simples y los recorre con foreach.	Domina arrays multidimensionales y asociativos complejos.
Objetos (POO)	Usa clases como simples contenedores. Todo público.	Entiende Clases vs Objetos. Usa constructor básico.	Aplica encapsulamiento (private/getters) y lógica de métodos.