📚 RESUMEN EXAMEN DWES - PHP

1. ARQUITECTURAS WEB

Páginas Estáticas vs Dinámicas

- **Estática**: contenido fijo (HTML puro). No cambia sin editar el archivo.
- **Dinámica**: contenido generado en tiempo de ejecución (PHP, JSP...).

¿Cuándo usar cada una?

- **Presentación/info fija** → Estática (o dinámica si quieres reutilizar plantillas)
- **Formularios/datos de BBDD** → Dinámica SIEMPRE

Validación cliente vs servidor

Validación	Tecnología	Uso
Cliente	JavaScript	Comprobar formato (ej: tiene @)
Servidor	PHP	Comprobar si existe en BBDD

Arquitectura LAMP

- **L**inux (SO)
- **A**pache (servidor web)
- **M**ySQL/MariaDB (BBDD)
- **P**HP (lenguaje)

Tipos de lenguajes

- **Guiones/Script**: PHP, JavaScript (interpretados)
- **Compilado a código nativo**: C, C++
- **Compilado a código intermedio**: Java (bytecode → JVM)

---

2. VARIABLES SUPERGLOBALES

```
$_GET['param']      // Parámetros de URL (?param=valor)
$_POST['param']     // Datos de formulario POST
$_REQUEST['param']  // GET + POST + COOKIE
$_SESSION['var']    // Datos de sesión (persisten entre páginas)
$_COOKIE['nombre']  // Cookies del navegador
$_FILES['campo']    // Archivos subidos
$_SERVER['REQUEST_METHOD']  // GET o POST
```

---

3. FORMULARIOS

Estructura básica

```
<form action="destino.php" method="POST">
    <input type="text" name="nombre">
    <input type="submit" value="Enviar">
</form>
```

Recibir datos

```
// Siempre validar que existen
if (isset($_POST['nombre'])) {
    $nombre = $_POST['nombre'];
    // Sanitizar
    $nombre = htmlspecialchars($nombre);
    $nombre = trim($nombre);
}
```

Comprobar método

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Procesar formulario
}
```

Subir archivos

```
// Formulario
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="archivo">
</form>

// PHP
$archivo = $_FILES['archivo'];
$nombre = $archivo['name'];
$tmp = $archivo['tmp_name'];
$size = $archivo['size'];
$tipo = $archivo['type'];
$error = $archivo['error'];

// Validar tamaño (2MB = 2*1024*1024)
if ($size > 2097152) {
    echo "Archivo muy grande";
}
```

```php
// Validar tipo
$extension = pathinfo($nombre, PATHINFO_EXTENSION);
if ($extension !== 'pdf' || $tipo !== 'application/pdf') {
    echo "Solo PDF";
}

// Guardar
$destino = __DIR__ . '/uploads/' . $nombre;
if (!file_exists($destino)) {
    move_uploaded_file($tmp, $destino);
}
```

---

4. COOKIES

Crear cookie

```php
// ANTES de cualquier HTML/echo
setcookie("nombre", "valor", time() + 3600);  // 1 hora
setcookie("nombre", "valor", time() + 86400); // 1 día
```

Leer cookie

```php
if (isset($_COOKIE['nombre'])) {
    $valor = $_COOKIE['nombre'];
}
```

Eliminar cookie

```php
setcookie("nombre", "", time() - 3600);
```

Ejemplo contador de visitas

```php
<?php
if (!isset($_COOKIE['visitas'])) {
    setcookie('visitas', 1, time() + 86400);
    echo "BIENVENIDO";
} else {
    $visitas = $_COOKIE['visitas'] + 1;
    if ($visitas >= 10) {
        setcookie('visitas', '', time() - 3600);
        echo "Cookie eliminada. Contador reseteado.";
    } else {
        setcookie('visitas', $visitas, time() + 86400);
```

```
        echo "VISITA $visitas";
    }
}
?>
```

---

5. SESIONES

Iniciar sesión

```
session_start();  // SIEMPRE al principio del script
```

Guardar datos

```
$_SESSION['usuario'] = 'pepe';
$_SESSION['autenticado'] = true;
$_SESSION['contador'] = 0;
```

Leer datos

```
if (isset($_SESSION['usuario'])) {
    echo $_SESSION['usuario'];
}
```

Eliminar variable

```
unset($_SESSION['contador']);
```

Cerrar sesión completa

```
session_start();
session_unset();      // Elimina todas las variables
session_destroy();    // Destruye la sesión
```

Ejemplo login con sesión

```
// validar.php
session_start();
$user_ok = 'admin';
```

```php
$pass_ok = '1234';

if ($_GET['user'] === $user_ok && $_GET['pass'] === $pass_ok) {
    $_SESSION['autenticado'] = true;
    header('Location: calculos.php');
    exit;
} else {
    header('Location: error.php');
    exit;
}

// calculos.php
session_start();
if (!isset($_SESSION['autenticado']) || !$_SESSION['autenticado']) {
    header('Location: index.php');
    exit;
}
// Código protegido aquí
```

---

6. REDIRECCIONES

```php
header('Location: pagina.php');
exit;  // IMPORTANTE: siempre después de header

// Con parámetros GET
header('Location: pagina.php?mensaje=ok&valor=5');
exit;
```

---

7. CONEXIÓN A BASE DE DATOS (PDO)

Clase de conexión

```php
class Conexion {
    private static $conexion = null;

    public static function getConexion(): PDO {
        if (self::$conexion === null) {
            $host = 'localhost';
            $dbname = 'mi_base';
            $user = 'root';
            $pass = '';

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

            self::$conexion = new PDO($dsn, $user, $pass, [
                PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
            ]);
```

```
        }
        return self::$conexion;
    }
}
```

CRUD básico

```
$pdo = Conexion::getConexion();

// INSERT
$sql = "INSERT INTO usuarios (nombre, email) VALUES (:nombre, :email)";
$stmt = $pdo->prepare($sql);
$stmt->execute([':nombre' => $nombre, ':email' => $email]);

// SELECT todos
$sql = "SELECT * FROM usuarios";
$stmt = $pdo->query($sql);
$usuarios = $stmt->fetchAll(PDO::FETCH_OBJ);

// SELECT uno
$sql = "SELECT * FROM usuarios WHERE id = :id";
$stmt = $pdo->prepare($sql);
$stmt->execute([':id' => $id]);
$usuario = $stmt->fetch(PDO::FETCH_OBJ);

// UPDATE
$sql = "UPDATE usuarios SET nombre = :nombre WHERE id = :id";
$stmt = $pdo->prepare($sql);
$stmt->execute([':nombre' => $nombre, ':id' => $id]);

// DELETE
$sql = "DELETE FROM usuarios WHERE id = :id";
$stmt = $pdo->prepare($sql);
$stmt->execute([':id' => $id]);
```

---

8. TRANSACCIONES

¿Para qué sirven?

- Ejecutar varias operaciones como UNA SOLA
- Si una falla, se deshacen TODAS (rollback)
- Garantiza integridad de datos

Sintaxis

```
$pdo = Conexion::getConexion();
```

```php
try {
    $pdo->beginTransaction();

    // Operación 1
    $stmt1 = $pdo->prepare("INSERT INTO tabla1 ...");
    $stmt1->execute([...]);

    // Operación 2
    $stmt2 = $pdo->prepare("INSERT INTO tabla2 ...");
    $stmt2->execute([...]);

    // Si todo OK
    $pdo->commit();
    echo "Transacción completada";

} catch (PDOException $e) {
    // Si algo falla
    $pdo->rollBack();
    echo "Error: " . $e->getMessage();
}
```

Ejemplo: alta simultánea

```php
public function altaSimultanea(array $registros): bool {
    $pdo = Conexion::getConexion();

    try {
        $pdo->beginTransaction();

        $sql = "INSERT INTO libros (titulo, autor, paginas)
                VALUES (:titulo, :autor, :paginas)";
        $stmt = $pdo->prepare($sql);

        foreach ($registros as $libro) {
            $stmt->execute([
                ':titulo' => $libro['titulo'],
                ':autor' => $libro['autor'],
                ':paginas' => $libro['paginas']
            ]);
        }

        $pdo->commit();
        return true;

    } catch (PDOException $e) {
        $pdo->rollBack();
        return false;
    }
}
```

9. ENVÍO DE CORREOS (PHPMailer)

Instalación

```
composer require phpmailer/phpmailer
```

Uso básico

```php
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\SMTP;
use PHPMailer\PHPMailer\Exception;

require 'vendor/autoload.php';

$mail = new PHPMailer(true);

try {
    // Configuración servidor
    $mail->isSMTP();
    $mail->Host = 'smtp.gmail.com';
    $mail->SMTPAuth = true;
    $mail->Username = 'tu@gmail.com';
    $mail->Password = 'contraseña_app';  // No la normal!
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
    $mail->Port = 587;

    // Remitente y destinatario
    $mail->setFrom('tu@gmail.com', 'Nombre');
    $mail->addAddress('destino@email.com');
    $mail->addCC('copia@email.com');        // Opcional

    // Adjuntos
    $mail->addAttachment('/path/archivo.pdf');

    // Contenido
    $mail->isHTML(true);
    $mail->Subject = 'Asunto';
    $mail->Body = '<h1>Mensaje HTML</h1>';

    $mail->send();
    echo 'Enviado!';

} catch (Exception $e) {
    echo "Error: {$mail->ErrorInfo}";
}
```

---

10. CLASES Y POO

Clase básica

```php
class Hobby {
    private int $id;
    private string $nombre;
    private string $descripcion;
    private ?string $fotografia = null;  // Puede ser null

    public function __construct(string $nombre, string $descripcion) {
        $this->nombre = $nombre;
        $this->descripcion = $descripcion;
    }

    // Getters
    public function getNombre(): string {
        return $this->nombre;
    }

    // Setters
    public function setFotografia(string $path): void {
        $this->fotografia = $path;
    }
}
```

Serialización

```php
// Guardar objeto en sesión/cookie
$hobby = new Hobby("Lectura", "Me gusta leer");
$serializado = serialize($hobby);

// Recuperar objeto
$hobby = unserialize($serializado);
echo $hobby->getNombre();
```

---

11. INCLUDES Y NAMESPACES

Incluir archivos

```php
require 'archivo.php';      // Error fatal si no existe
require_once 'archivo.php'; // Solo incluye 1 vez
include 'archivo.php';      // Warning si no existe
include_once 'archivo.php'; // Solo incluye 1 vez
```

Namespaces

```php
// En src/Clases/Hobby.php
namespace App\Clases;

class Hobby {
    // ...
}

// En otro archivo
require_once 'src/Clases/Hobby.php';
use App\Clases\Hobby;

$h = new Hobby();
```

Autoload con Composer

```php
// composer.json
{
    "autoload": {
        "psr-4": {
            "App\\": "src/"
        }
    }
}
```

```
composer dump-autoload
```

```php
require 'vendor/autoload.php';
use App\Clases\Hobby;
```

---

12. FUNCIONES ÚTILES

```php
// Strings
htmlspecialchars($str)    // Evita XSS
trim($str)                // Quita espacios
strlen($str)              // Longitud
strtolower($str)          // Minúsculas
strtoupper($str)          // Mayúsculas

// Arrays
count($arr)               // Número elementos
in_array($val, $arr)      // ¿Existe valor?
array_push($arr, $val)    // Añadir al final
```

```
// Archivos
file_exists($path)        // ¿Existe archivo?
dirname(__FILE__)         // Directorio actual
pathinfo($file, PATHINFO_EXTENSION)  // Extensión

// Conversión tipos
(int) $var                // A entero
(float) $var              // A decimal
(bool) $var               // A booleano
(string) $var             // A string

// Validación
isset($var)               // ¿Existe y no es null?
empty($var)               // ¿Vacío/null/0/false?
is_numeric($var)          // ¿Es número?
filter_var($email, FILTER_VALIDATE_EMAIL)  // Validar email
```

⚡ TIPS RÁPIDOS EXAMEN

1. **session_start()** y **setcookie()** → SIEMPRE antes de cualquier echo/HTML
2. **header()** → SIEMPRE seguido de **exit**
3. **PDO** → Siempre usar **prepare()** con parámetros, NUNCA concatenar SQL
4. **Transacciones** → beginTransaction, commit, rollBack en try-catch
5. **Archivos** → Validar extensión Y tipo MIME
6. **Validación cliente** (JS) → Formato, campos vacíos
7. **Validación servidor** (PHP) → Existencia en BBDD, seguridad