✅ EXAMEN DESARROLLO WEB EN ENTORNO SERVIDOR - RESPUESTAS

Temas 0 y 1: Introducción y PHP Básico

📊 PARTE A: RESPUESTAS TEST

1. ¿Cuál es la principal diferencia entre API REST y API SOAP?

Respuesta correcta: c) REST usa HTTP con JSON (más ligero) y SOAP usa XML estructurado

Explicación: REST utiliza métodos HTTP estándar (GET, POST, PUT, DELETE) y formato JSON que es más ligero. SOAP usa protocolo XML más estructurado y complejo, ideal para aplicaciones empresariales que requieren mayor seguridad.

2. ¿Qué comando inicia un servidor web local con PHP?

Respuesta correcta: b) `php -S localhost:8000`

Explicación: El servidor de desarrollo integrado de PHP se inicia con `-S` seguido de la dirección y puerto. Es útil para desarrollo local sin necesidad de Apache.

3. En XAMPP, ¿qué puerto usa por defecto MySQL?

Respuesta correcta: c) 3306

Explicación: MySQL/MariaDB usa el puerto 3306 por defecto. Apache usa 80 (HTTP) y 443 (HTTPS).

4. ¿Cuál es la estructura correcta de etiquetas PHP?

Respuesta correcta: b) `<?php ... ?>`

Explicación: Las etiquetas PHP estándar son `<?php` para abrir y `?>` para cerrar. También existe la sintaxis corta `<?= ?>` para echo.

5. ¿Qué framework PHP es conocido por usar Eloquent ORM?

Respuesta correcta: c) Laravel

Explicación: Laravel incluye Eloquent ORM para manejo de bases de datos. Symfony usa Doctrine ORM.

6. ¿Cuál es la diferencia entre `include` y `require` en PHP?

Respuesta correcta: c) `require` detiene la ejecución si el archivo no existe, `include` continúa

Explicación: `require` genera error fatal si no encuentra el archivo. `include` solo genera warning y continúa.

7. ¿Qué significa el acrónimo XAMPP?

Respuesta correcta: b) Cross-platform Apache MariaDB PHP Perl

Explicación: X=Cross-platform, A=Apache, M=MariaDB/MySQL, P=PHP, P=Perl. Es un paquete de software multiplataforma.

8. ¿Cuál es la función principal de un archivo .htaccess?

Respuesta correcta: b) Configurar el servidor Apache

Explicación: .htaccess permite configurar Apache a nivel de directorio: reescritura URLs, seguridad, redirecciones, etc.

9. ¿Qué tipo de dato PHP almacena 42?

Respuesta correcta: c) int

Explicación: 42 es un número entero (integer). PHP tiene tipado dinámico y detecta automáticamente el tipo.

10. ¿Cuál es la sintaxis corta para hacer echo en PHP?

Respuesta correcta: b) `<?= $var ?>`

Explicación: `<?=` es equivalente a `<?php echo`. Es una sintaxis abreviada muy útil para mostrar variables en HTML.

11. En desarrollo servidor, ¿qué significa que PHP sea "interpretado"?

Respuesta correcta: b) Se ejecuta directamente línea por línea

Explicación: PHP es interpretado en tiempo de ejecución, no se compila previamente como lenguajes compilados (C, Java).

12. ¿Qué comando de terminal lista archivos en Linux/Ubuntu?

Respuesta correcta: c) `ls`

Explicación: `ls` lista archivos en sistemas Unix/Linux. `dir` es para Windows. `ls -la` muestra detalles completos.

13. ¿Cuál es la principal ventaja de Docker sobre XAMPP para desarrollo?

Respuesta correcta: b) Mayor consistencia entre entornos

Explicación: Docker garantiza que el entorno de desarrollo sea idéntico al de producción, evitando problemas de compatibilidad.

14. En un proyecto PHP, ¿dónde se deben colocar los archivos web?

Respuesta correcta: b) En `C:\xampp\htdocs\`

Explicación: htdocs es el directorio raíz del servidor web Apache en XAMPP. En Linux sería `/var/www/html/`.

15. ¿Qué tipo de framework es Symfony?

Respuesta correcta: c) PHP Framework

Explicación: Symfony es un framework de PHP para desarrollo web, conocido por su robustez y componentes reutilizables.

16. ¿Cuál es el propósito de `composer.json`?

Respuesta correcta: b) Gestionar dependencias PHP

Explicación: Composer es el gestor de paquetes de PHP. composer.json define las dependencias del proyecto.

17. ¿Qué puerto usa HTTP por defecto?

Respuesta correcta: b) 80

Explicación: HTTP usa puerto 80, HTTPS usa 443, FTP usa 21, MySQL usa 3306.

18. ¿Qué significa que una variable PHP sea tipado dinámico?

Respuesta correcta: b) El tipo se define en tiempo de ejecución

Explicación: PHP determina automáticamente el tipo de variable según el valor asignado, no necesita declaración previa del tipo.

19. En una API RESTful, ¿qué método HTTP se usa para crear recursos?

Respuesta correcta: b) POST

Explicación: POST crea recursos, GET obtiene, PUT actualiza/crea, DELETE elimina.

20. ¿Cuál es la función de `require_once` en PHP?

Respuesta correcta: a) Incluir un archivo solo si no se ha incluido antes

Explicación: `require_once` evita la inclusión múltiple del mismo archivo, útil para archivos de configuración.

21. ¿Qué es phpMyAdmin?

Respuesta correcta: b) Una herramienta para administrar MySQL

Explicación: phpMyAdmin es una interfaz web para gestionar bases de datos MySQL/MariaDB de forma visual.

22. ¿Cuál es la principal función de Apache en un entorno web?

Respuesta correcta: c) Servir archivos web vía HTTP

Explicación: Apache es un servidor web que recibe peticiones HTTP y devuelve respuestas (HTML, PHP procesado, archivos estáticos).

23. ¿Qué ventaja tienen los contenedores Docker para desarrollo?

Respuesta correcta: c) Garantizan entornos reproducibles

Explicación: Docker encapsula la aplicación y sus dependencias, asegurando que funcione igual en cualquier máquina.

24. En PHP, ¿cómo se declara una constante?

Respuesta correcta: d) Las opciones a y c son correctas

Explicación: Tanto `const NOMBRE = valor;` como `define('NOMBRE', valor);` son válidas. `const` es más moderna.

25. ¿Qué característica hace que PHP sea especialmente útil para desarrollo web?

Respuesta correcta: b) Se ejecuta en el servidor y genera HTML dinámico

Explicación: PHP se procesa en el servidor antes de enviar el resultado al navegador, permitiendo contenido dinámico y acceso a bases de datos.

💻 PARTE B: RESPUESTAS DE DESARROLLO

Pregunta 1 (5 puntos): Configuración de Entorno de Desarrollo

Proceso de instalación XAMPP:

1. **Descarga:** Ir a https://www.apachefriends.org/download.html
2. **Instalación:** Ejecutar el instalador como administrador
3. **Directorio:** Instalar preferiblemente en `C:\xampp\`
4. **Componentes:** Seleccionar Apache, MySQL, PHP, phpMyAdmin

Servicios que deben iniciarse:

Desde XAMPP Control Panel:

- Apache (puerto 80, 443)

- MySQL (puerto 3306)

Verificación por línea de comandos:

netstat -an | findstr :80   # Windows

netstat -an | grep :80   # Linux

Verificación de funcionamiento:

Crear archivo test.php en C:\xampp\htdocs\

<?php

phpinfo();

?>

- Acceder a `http://localhost/`
- Acceder a `http://localhost/test.php`
- Acceder a `http://localhost/phpmyadmin/`

Estructura de carpetas recomendada:

C:\xampp\htdocs\mi_proyecto\

├── index.php   # Página principal

├── config/

```
│   ├── config.php        # Configuración general
│   └── database.php      # Configuración BD
├── includes/
│   ├── header.php        # Cabecera común
│   ├── footer.php        # Pie común
│   └── functions.php     # Funciones reutilizables
├── assets/
│   ├── css/
│   ├── js/
│   └── images/
├── pages/
│   ├── about.php
│   └── contact.php
└── README.md
```

Pregunta 2 (5 puntos): APIs y Comunicación Servidor

API REST (Representational State Transfer):

Características principales:

- Arquitectura basada en recursos identificados por URLs
- Usa métodos HTTP estándar: GET, POST, PUT, DELETE
- Sin estado (stateless): cada petición es independiente
- Formato de datos: principalmente JSON, también XML
- Ligero y escalable

Métodos HTTP en REST:

```
GET    /api/users        # Obtener lista de usuarios
GET    /api/users/123    # Obtener usuario específico
```

```
POST   /api/users        # Crear nuevo usuario

PUT    /api/users/123    # Actualizar usuario completo

PATCH  /api/users/123    # Actualizar usuario parcialmente

DELETE /api/users/123    # Eliminar usuario
```

Ejemplo REST:

```
// Petición GET

fetch('/api/users/123')

  .then(response => response.json())

  .then(data => console.log(data));

// Respuesta JSON

{

  "id": 123,

  "name": "Juan Pérez",

  "email": "juan@example.com"

}
```

API SOAP (Simple Object Access Protocol):

Características principales:

- Protocolo basado en XML
- Mayor estructura y formalidad
- Incluye definición de servicios (WSDL)
- Soporte para seguridad avanzada (WS-Security)
- Transacciones complejas
- Principalmente sobre HTTP/HTTPS

Ejemplo SOAP:

```
<!-- Petición SOAP -->

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
```

```
  <soap:Body>
    <GetUser xmlns="http://example.com/">
      <UserId>123</UserId>
    </GetUser>
  </soap:Body>
</soap:Envelope>
<!-- Respuesta SOAP -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetUserResponse xmlns="http://example.com/">
      <User>
        <Id>123</Id>
        <Name>Juan Pérez</Name>
        <Email>juan@example.com</Email>
      </User>
    </GetUserResponse>
  </soap:Body>
</soap:Envelope>
```

Cuándo usar cada tipo:

- **REST:** Aplicaciones web/móviles, APIs públicas, servicios simples
- **SOAP:** Aplicaciones empresariales, transacciones complejas, alta seguridad

---

Pregunta 3 (4 puntos): Estructura de Proyecto PHP

```
mi_proyecto_web/
├── index.php                # Página principal
```

```
├── config/
│   ├── config.php            # Configuración general
│   ├── database.php          # Conexión BD
│   └── routes.php            # Definición de rutas
├── includes/
│   ├── header.php            # Cabecera común
│   ├── footer.php            # Pie de página común
│   ├── navigation.php        # Menú de navegación
│   └── functions.php         # Funciones comunes
├── pages/
│   ├── home.php              # Página inicio
│   ├── about.php             # Acerca de
│   ├── contact.php           # Contacto
│   ├── services.php          # Servicios
│   └── portfolio.php         # Portafolio
├── assets/
│   ├── css/
│   │   ├── style.css         # Estilos principales
│   │   ├── responsive.css    # Estilos responsive
│   │   └── admin.css         # Estilos administración
│   ├── js/
│   │   ├── main.js           # JavaScript principal
│   │   ├── form-validation.js  # Validación formularios
│   │   └── admin.js          # JS administración
```

```
│   ├── images/
│   │   ├── logo.png
│   │   ├── backgrounds/
│   │   └── icons/
│   └── fonts/
├── admin/
│   ├── index.php            # Panel administración
│   ├── login.php            # Login admin
│   └── dashboard.php        # Dashboard
├── api/
│   ├── users.php            # API usuarios
│   └── products.php         # API productos
├── vendor/                  # Dependencias Composer
├── .htaccess                # Configuración Apache
├── composer.json            # Dependencias PHP
├── README.md                # Documentación
└── .gitignore               # Archivos ignorados Git
```

Justificación de la organización:

- **Separación por tipo:** CSS, JS e imágenes en `assets/`
- **Reutilización:** Archivos comunes en `includes/`
- **Escalabilidad:** Estructura modular fácil de expandir
- **Mantenimiento:** Configuración centralizada en `config/`
- **Seguridad:** Separación de admin y API
- **Estándares:** Sigue convenciones PHP modernas

Pregunta 4 (4 puntos): Variables y Tipos de Datos en PHP

<?php

```php
/**
 * Demostración de variables y tipos de datos en PHP
 * Archivo: tipos_datos.php
 */

// Configuración para mostrar todos los errores
error_reporting(E_ALL);
ini_set('display_errors', 1);

echo "<h1>Tipos de Datos en PHP</h1>";

// 1. TIPOS ESCALARES
echo "<h2>1. Tipos Escalares</h2>";

// String (cadena de caracteres)
$nombre = "Juan Pérez";
$apellidos = 'García López';

echo "<h3>String:</h3>";
echo "Nombre completo: $nombre $apellidos<br>";
var_dump($nombre);
echo "<br><br>";

// Integer (número entero)
$edad = 25;
$año_nacimiento = 1998;

echo "<h3>Integer:</h3>";
echo "Edad: $edad años<br>";
var_dump($edad);
echo "<br><br>";
```

```php
// Float (número decimal)

$altura = 1.75;

$precio = 99.99;

echo "<h3>Float:</h3>";

echo "Altura: $altura metros<br>";

echo "Precio: $$precio<br>";

var_dump($altura);

echo "<br><br>";

// Boolean (verdadero/falso)

$es_estudiante = true;

$tiene_trabajo = false;

echo "<h3>Boolean:</h3>";

echo "Es estudiante: " . ($es_estudiante ? 'Sí' : 'No') . "<br>";

echo "Tiene trabajo: " . ($tiene_trabajo ? 'Sí' : 'No') . "<br>";

var_dump($es_estudiante);

echo "<br><br>";

// 2. TIPOS COMPUESTOS

echo "<h2>2. Tipos Compuestos</h2>";

// Array (matriz)

$colores = ["rojo", "verde", "azul"];

$persona = [

    "nombre" => "Ana",

    "edad" => 30,

    "ciudad" => "Madrid"
```

```
];

echo "<h3>Array:</h3>";

echo "Colores: " . implode(", ", $colores) . "<br>";

echo "Persona: {$persona['nombre']}, {$persona['edad']} años, {$persona['ciudad']}<br>";

var_dump($colores);

echo "<br>";

var_dump($persona);

echo "<br><br>";

// 3. TIPOS ESPECIALES

echo "<h2>3. Tipos Especiales</h2>";

// NULL

$valor_nulo = null;

echo "<h3>NULL:</h3>";

echo "Valor nulo: ";

var_dump($valor_nulo);

echo "<br><br>";

// 4. CONVERSIÓN ENTRE TIPOS

echo "<h2>4. Conversión entre Tipos</h2>";

$numero_string = "123";

$numero_int = (int)$numero_string;

$numero_float = (float)$numero_string;

echo "String '123' convertido a:<br>";

echo "Integer: $numero_int<br>";

echo "Float: $numero_float<br>";
```

```
var_dump($numero_string, $numero_int, $numero_float);

echo "<br><br>";

// Conversión automática (type juggling)

$resultado = "10" + 5; // PHP convierte automáticamente

echo "Conversión automática '10' + 5 = $resultado<br>";

var_dump($resultado);

echo "<br><br>";

// 5. CONSTANTES

echo "<h2>5. Constantes</h2>";

// Definir constantes

define('SITIO_WEB', 'www.miempresa.com');

define('VERSION', 1.2);

define('ES_DESARROLLO', false);

// Constantes de clase (PHP 5.6+)

const MAX_USUARIOS = 100;

const TIPO_USUARIO = [

    'admin' => 1,

    'editor' => 2,

    'usuario' => 3

];

echo "Sitio web: " . SITIO_WEB . "<br>";

echo "Versión: " . VERSION . "<br>";

echo "Es desarrollo: " . (ES_DESARROLLO ? 'Sí' : 'No') . "<br>";

echo "Máximo usuarios: " . MAX_USUARIOS . "<br>";
```

```
echo "Tipo admin: " . TIPO_USUARIO['admin'] . "<br>";

// 6. INFORMACIÓN DE VARIABLES

echo "<h2>6. Información de Variables</h2>";

$test_var = "Hola Mundo";

echo "Variable: $test_var<br>";

echo "Tipo: " . gettype($test_var) . "<br>";

echo "¿Es string?: " . (is_string($test_var) ? 'Sí' : 'No') . "<br>";

echo "¿Está definida?: " . (isset($test_var) ? 'Sí' : 'No') . "<br>";

echo "¿Está vacía?: " . (empty($test_var) ? 'Sí' : 'No') . "<br>";

// Mostrar todas las variables definidas

echo "<h3>Variables actuales:</h3>";

echo "<pre>";

print_r(get_defined_vars());

echo "</pre>";

?>
```

Pregunta 5 (3 puntos): Integración HTML-PHP

Archivo: index.php

```
<?php

// Configuración y variables para la página

$titulo_pagina = "Mi Sitio Web Dinámico";

$nombre_usuario = "Juan Pérez";

$productos = [

    ["nombre" => "Laptop", "precio" => 899.99],
```

```
    ["nombre" => "Mouse", "precio" => 25.50],
    ["nombre" => "Teclado", "precio" => 75.00]
];
$año_actual = date('Y');
$es_usuario_logueado = true;
?>
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title><?= $titulo_pagina ?></title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
    <?php include 'includes/header.php'; ?>

    <main class="container mt-4">
        <div class="row">
            <div class="col-md-8">
                <h1>Bienvenido<?= $es_usuario_logueado ? ", " . $nombre_usuario : "" ?>!</h1>

                <div class="alert alert-info">
```

```
            <h4>Información del sistema:</h4>
            <ul>
                <li>Fecha actual: <?= date('d/m/Y H:i:s') ?></li>
                <li>Servidor: <?= $_SERVER['SERVER_NAME'] ?></li>
                <li>IP del cliente: <?= $_SERVER['REMOTE_ADDR'] ?></li>
            </ul>
        </div>
        <h2>Productos Destacados</h2>
        <div class="row">
            <?php foreach ($productos as $producto): ?>
            <div class="col-md-4 mb-3">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title"><?= $producto['nombre'] ?></h5>
                        <p class="card-text">
                            Precio: <strong>$<?= number_format($producto['precio'], 2)
?></strong>
                        </p>
                        <a href="#" class="btn btn-primary">Comprar</a>
                    </div>
                </div>
            </div>
            <?php endforeach; ?>
        </div>
    </div>
```

```
            <div class="col-md-4">
                <div class="card">
                    <div class="card-header">
                        <h5>Panel de Usuario</h5>
                    </div>
                    <div class="card-body">
                        <?php if ($es_usuario_logueado): ?>
                            <p>Conectado como: <strong><?= $nombre_usuario ?></strong></p>
                            <a href="logout.php" class="btn btn-danger">Cerrar Sesión</a>
                        <?php else: ?>
                            <p>No has iniciado sesión</p>
                            <a href="login.php" class="btn btn-success">Iniciar Sesión</a>
                        <?php endif; ?>
                    </div>
                </div>
            </div>
        </div>
    </main>
    <?php include 'includes/footer.php'; ?>
</body>
</html>
```

Archivo: includes/header.php

```
<header class="bg-primary text-white">
```

```
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container">
            <a class="navbar-brand" href="index.php">
                <?= $titulo_pagina ?? 'Mi Sitio Web' ?>
            </a>

            <div class="navbar-nav ms-auto">
                <a class="nav-link" href="index.php">Inicio</a>
                <a class="nav-link" href="productos.php">Productos</a>
                <a class="nav-link" href="contacto.php">Contacto</a>
                <?php if ($es_usuario_logueado ?? false): ?>
                    <a class="nav-link" href="perfil.php">Mi Perfil</a>
                <?php else: ?>
                    <a class="nav-link" href="login.php">Login</a>
                <?php endif; ?>
            </div>
        </div>
    </nav>
</header>
```

Archivo: includes/footer.php

```
<footer class="bg-dark text-white mt-5 py-4">
    <div class="container">
        <div class="row">
            <div class="col-md-6">
```

```
            <h5><?= $titulo_pagina ?? 'Mi Sitio Web' ?></h5>

            <p>Desarrollado con PHP y Bootstrap</p>

        </div>

        <div class="col-md-6 text-end">

            <p>&copy; <?= $año_actual ?> Todos los derechos reservados</p>

            <p>Generado el: <?= date('d/m/Y H:i:s') ?></p>

        </div>

    </div>

  </div>

</footer>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
```

Pregunta 6 (4 puntos): Configuración Apache y .htaccess

¿Qué es .htaccess?

El archivo `.htaccess` (hypertext access) es un archivo de configuración que permite gestionar la configuración del servidor web Apache a nivel de directorio, sin necesidad de acceder al archivo de configuración principal del servidor.

Funciones principales:

- Reescritura de URLs (URL Rewriting)
- Control de acceso y autenticación
- Redirecciones HTTP
- Configuración de páginas de error personalizadas
- Compresión y caché
- Configuración de tipos MIME

Ejemplo de configuración básica .htaccess:

```
# Habilitar el motor de reescritura

RewriteEngine On
```

```
# ===== REESCRITURA DE URLs =====

# Eliminar extensión .php de las URLs

RewriteCond %{REQUEST_FILENAME} !-d

RewriteCond %{REQUEST_FILENAME} !-f

RewriteRule ^([^\.]+)$ $1.php [NC,L]

# URLs amigables para productos

RewriteRule ^producto/([0-9]+)/?$ producto.php?id=$1 [NC,L]

RewriteRule ^categoria/([a-zA-Z0-9-]+)/?$ categoria.php?slug=$1 [NC,L]

# ===== REDIRECCIONES =====

# Forzar HTTPS

RewriteCond %{HTTPS} off

RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]

# Redireccionar www a no-www

RewriteCond %{HTTP_HOST} ^www\.(.*)$ [NC]

RewriteRule ^(.*)$ https://%1/$1 [R=301,L]

# ===== SEGURIDAD BÁSICA =====

# Proteger archivos sensibles

<Files ~ "^\.ht">

    Order allow,deny

    Deny from all

</Files>

# Bloquear acceso a archivos de configuración

<FilesMatch "\.(env|json|config|ini)$">

    Order allow,deny
```

```
    Deny from all
</FilesMatch>
# Prevenir listado de directorios
Options -Indexes
# Proteger contra inyección de scripts
<IfModule mod_headers.c>
    Header always set X-Content-Type-Options nosniff
    Header always set X-Frame-Options DENY
    Header always set X-XSS-Protection "1; mode=block"
</IfModule>
# ===== COMPRESIÓN Y CACHÉ =====
# Habilitar compresión GZIP
<IfModule mod_deflate.c>
    AddOutputFilterByType DEFLATE text/plain
    AddOutputFilterByType DEFLATE text/html
    AddOutputFilterByType DEFLATE text/xml
    AddOutputFilterByType DEFLATE text/css
    AddOutputFilterByType DEFLATE application/xml
    AddOutputFilterByType DEFLATE application/xhtml+xml
    AddOutputFilterByType DEFLATE application/rss+xml
    AddOutputFilterByType DEFLATE application/javascript
    AddOutputFilterByType DEFLATE application/x-javascript
</IfModule>
# Configurar caché para archivos estáticos
```

```
<IfModule mod_expires.c>
    ExpiresActive On
    ExpiresByType image/jpg "access plus 1 month"
    ExpiresByType image/jpeg "access plus 1 month"
    ExpiresByType image/gif "access plus 1 month"
    ExpiresByType image/png "access plus 1 month"
    ExpiresByType text/css "access plus 1 month"
    ExpiresByType application/pdf "access plus 1 month"
    ExpiresByType application/javascript "access plus 1 month"
    ExpiresByType application/x-javascript "access plus 1 month"
    ExpiresByType application/x-shockwave-flash "access plus 1 month"
    ExpiresByType image/x-icon "access plus 1 year"
    ExpiresDefault "access plus 2 days"
</IfModule>
# ===== PÁGINAS DE ERROR PERSONALIZADAS =====
ErrorDocument 404 /error/404.php
ErrorDocument 500 /error/500.php
ErrorDocument 403 /error/403.php
# ===== CONFIGURACIÓN PHP =====
# Configuraciones PHP básicas
php_flag display_errors Off
php_value upload_max_filesize 64M
php_value post_max_size 64M
php_value max_execution_time 300
```

php_value max_input_vars 3000

Diferencias Apache vs Nginx para desarrollo PHP:

Apache:

Ventajas:

- Configuración descentralizada (.htaccess)

- Mayor flexibilidad por directorio

- Soporte nativo mod_php

- Más fácil para principiantes

- Gran cantidad de módulos

Desventajas:

- Consume más memoria

- Menor rendimiento con alto tráfico

- Configuración puede ser compleja

Nginx:

Ventajas:

- Mayor rendimiento y eficiencia

- Menor consumo de memoria

- Mejor para sitios con alto tráfico

- Configuración centralizada

- Excelente como proxy reverso

Desventajas:

- No soporta .htaccess (configuración centralizada)

- Curva de aprendizaje más pronunciada

- Necesita PHP-FPM para procesar PHP

- Menos módulos disponibles

```
# Ejemplo configuración Nginx para PHP:
server {
    listen 80;
    server_name localhost;
    root /var/www/html;
    index index.php index.html;
    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }
    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        include fastcgi_params;
    }
}
```

Comandos de terminal para gestionar servicios en Linux:

```
# ===== SYSTEMCTL (sistemas modernos) =====
# Iniciar servicios
sudo systemctl start apache2
sudo systemctl start nginx
sudo systemctl start mysql
sudo systemctl start php8.1-fpm
```

```
# Detener servicios

sudo systemctl stop apache2

sudo systemctl stop nginx

# Reiniciar servicios

sudo systemctl restart apache2

sudo systemctl reload nginx  # Recarga configuración sin parar

# Estado de servicios

sudo systemctl status apache2

sudo systemctl status mysql

# Habilitar inicio automático

sudo systemctl enable apache2

sudo systemctl enable mysql

# Deshabilitar inicio automático

sudo systemctl disable apache2

# ===== SERVICE (sistemas legacy) =====

sudo service apache2 start

sudo service apache2 stop

sudo service apache2 restart

sudo service apache2 status

# ===== GESTIÓN DE LOGS =====

# Ver logs de Apache

sudo tail -f /var/log/apache2/access.log

sudo tail -f /var/log/apache2/error.log

# Ver logs de Nginx
```

```
sudo tail -f /var/log/nginx/access.log

sudo tail -f /var/log/nginx/error.log

# Ver logs de MySQL

sudo tail -f /var/log/mysql/error.log

# ===== COMANDOS ESPECÍFICOS APACHE =====

# Verificar configuración

sudo apache2ctl configtest

sudo nginx -t  # Para Nginx

# Recargar configuración

sudo apache2ctl graceful

sudo nginx -s reload

# Ver módulos habilitados

apache2ctl -M

sudo a2enmod rewrite  # Habilitar módulo

sudo a2dismod rewrite # Deshabilitar módulo

# Gestión de sitios

sudo a2ensite mi-sitio.conf

sudo a2dissite mi-sitio.conf
```

⚡ PARTE C: RESPUESTAS PREGUNTAS CORTAS

26. ¿Qué significa PHP?

Respuesta: PHP: Hypertext Preprocessor (originalmente Personal Home Page)

27. ¿Cuál es la diferencia entre `echo` y `print` en PHP?

Respuesta: `echo` puede mostrar múltiples valores y es más rápido; `print` solo muestra un valor y retorna 1

28. ¿Qué función se usa para conectar a MySQL desde PHP?

Respuesta: `mysqli_connect()` o PDO con `new PDO()`

29. ¿Cómo se define una variable superglobal en PHP? Nombra 3 ejemplos.

Respuesta: No se definen, son predefinidas por PHP. Ejemplos: `$_GET`, `$_POST`, `$_SESSION`

30. ¿Qué hace el operador `.` en PHP?

Respuesta: Concatena (une) cadenas de texto

31. ¿Cuál es la diferencia entre `==` y `===` en PHP?

Respuesta: `==` compara valores, `===` compara valores y tipos de datos

32. ¿Qué directorio contiene los archivos web en XAMPP?

Respuesta: `C:\xampp\htdocs\`

33. ¿Cómo se inicia sesión en PHP?

Respuesta: `session_start();`

34. ¿Qué hace `phpinfo()`?

Respuesta: Muestra toda la información de configuración de PHP y el servidor

35. ¿Cuál es la sintaxis para comentarios de una línea en PHP?

Respuesta: `//` o `#`

📊 RESUMEN DE PUNTUACIÓN ACTUALIZADO

Parte A - Test: 25/25 puntos

Parte B - Desarrollo: 25/25 puntos

Parte C - Preguntas Cortas: 10/10 puntos

TOTAL: 60/60 puntos

Las respuestas incluyen ejemplos prácticos, código funcional y explicaciones detalladas enfocadas en desarrollo servidor-side con PHP y tecnologías web fundamentales.