Repaso Examen Parcial

Desarrollo Web en Entorno Servidor

Unidades 0-4

Fundamentos Web

Cliente vs. Servidor

Arquitectura Web: Lado Cliente

Lado Cliente (Navegador)

Es todo el código que se ejecuta en el navegador del usuario (ej. Chrome, Firefox). El usuario puede ver y interactuar con este código.

Responsabilidades:

- Estructura (Qué se ve)
- Estilo (Cómo se ve)
- Interactividad (Qué pasa si hago clic)

Tecnologías de Cliente

Estos lenguajes son "entendidos" por tu navegador:

- **HTML:** Define la estructura y el contenido.
- **CSS:** Define los estilos, colores, fuentes y diseño.
- **JavaScript:** Define el comportamiento y la interacción.

Arquitectura Web: Lado Servidor

Lado Servidor (Servidor)

Es todo el código que se ejecuta en un ordenador remoto (el servidor) antes de que la página se envíe al usuario.

Responsabilidades:

- Procesar lógica de negocio.
- Acceder a bases de datos (leer/escribir).
- "Construir" la página HTML final.
- Gestionar la seguridad y los usuarios.

Tecnologías de Servidor

El navegador **nunca** ve este código, solo el resultado (HTML):

- **PHP:** Lenguaje de scripting que se ejecuta en el servidor.
- **Apache:** El software de servidor que "escucha" peticiones.
- **MySQL / MariaDB:** El software de base de datos.

Conceptos Clave de la Web

HTTP

HyperText Transfer Protocol. Es el "idioma" o protocolo que usa el navegador (cliente) para pedir páginas y recursos al servidor.

W3C

World Wide Web Consortium. Es la organización internacional que se encarga de crear y mantener los estándares de la web, como HTML y CSS.

Interpretado

PHP es un lenguaje **interpretado**. Un intérprete lee y ejecuta el código "línea a línea" en tiempo real, a diferencia de un **compilado** (que se traduce a código máquina *antes* de usarse).

Nuestro Entorno de Trabajo

Herramientas Locales y Control de Versiones

Configuración del Servidor Local

XAMPP

Es un **paquete de software** gratuito que instala todo lo necesario para un servidor local: **A**pache, **M**ariaDB (MySQL), **P**HP y **P**erl.

Apache

Es el software de **servidor web**. Su trabajo es "escuchar" peticiones (ej. del navegador) y ejecutar el script de PHP para devolver una respuesta (HTML).

`htdocs`

Es la **carpeta raíz** de tu servidor Apache dentro de XAMPP. Aquí es donde debes guardar todos tus proyectos PHP (`C:\xampp\htdocs`).

`http://localhost`

Es la **URL** que escribes en tu navegador para acceder a los proyectos guardados en tu carpeta `htdocs`. Si Apache está apagado, el navegador mostrará un error de "Conexión Rechazada".

Herramientas de Desarrollo

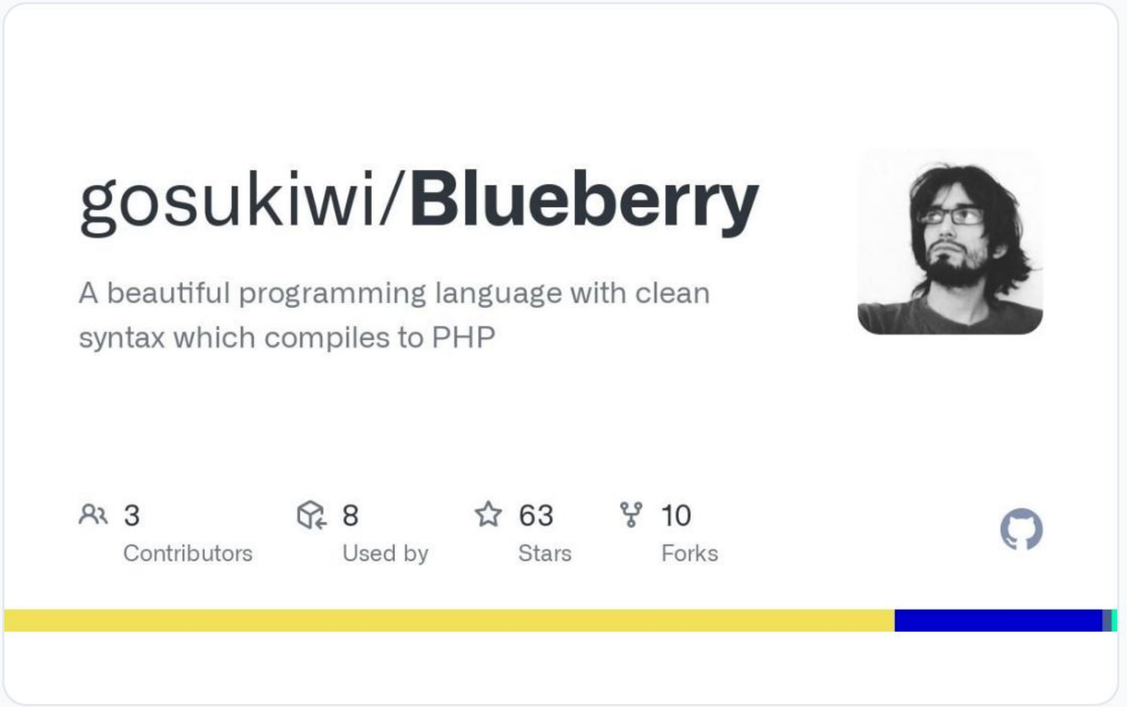
IDE (Entorno de Desarrollo)

Un **I**ntegrated **D**evelopment **E**nvironment es un editor de código avanzado (como Visual Studio Code, PhpStorm, etc.) que nos ayuda a programar más rápido.

GIT (Control de Versiones)

Es un **sistema de control de versiones**. Su propósito principal es "tomar fotos" (commits) de tu código a medida que trabajas. Esto te permite:

- Rastrear cambios.
- Volver a versiones anteriores.
- Colaborar con otros desarrolladores.



Flujo de Trabajo Básico de GIT

GIT funciona en un proceso de 3 pasos (local) + 1 (remoto):

1. `git add`
Añade tus cambios al "área de preparación" (Staging). Le dices a GIT: "Quiero incluir estos archivos en la próxima foto".

2. `git commit -m "mensaje"`
Toma la "foto" (commit) de todo lo que esté en el área de preparación. Guarda el cambio de forma permanente **en tu repositorio local**.

3. `git push`
Sube tus commits locales (tus "fotos") al **repositorio remoto** (como GitHub). Sin este paso, ¡tus cambios solo existen en tu ordenador!

Modelos Cloud y Frameworks

Modelos Cloud

Formas de usar recursos en la nube:

- **SaaS (Software):** Usas una app (ej. Gmail).
- **PaaS (Plataforma):** Te dan la plataforma (ej. PHP, Apache) y tú subes tu código (ej. Heroku, Cloud9).
- **IaaS (Infraestructura):** Alquilas las máquinas virtuales (ej. AWS EC2).

Frameworks

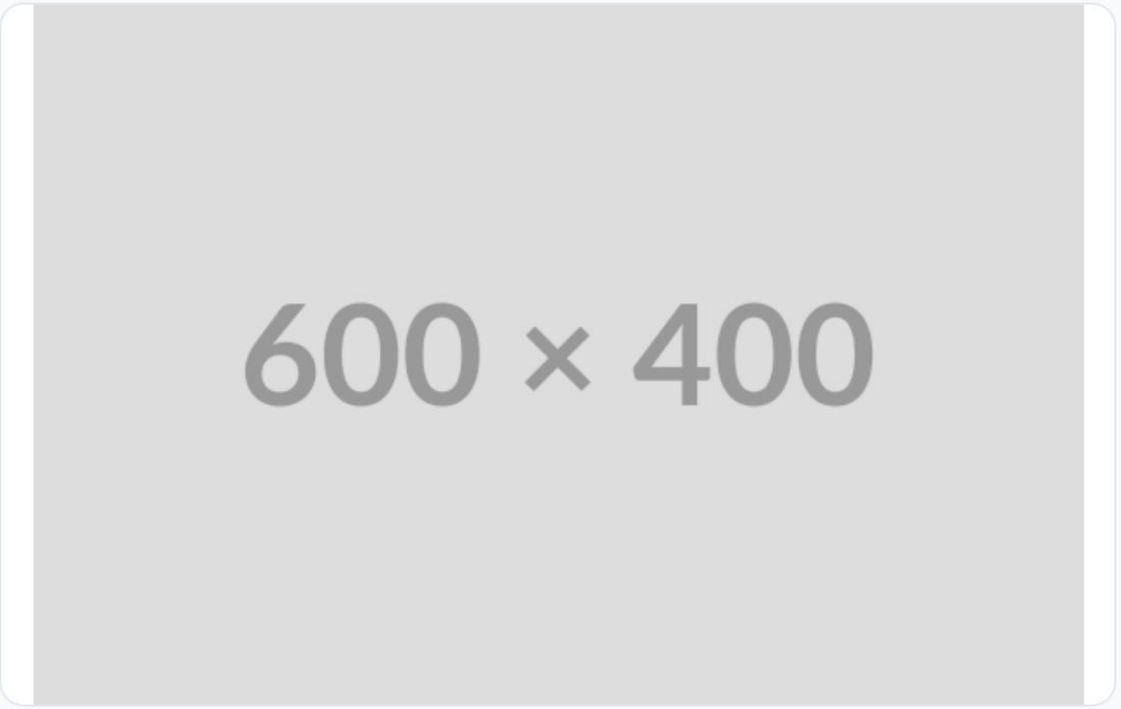
Un Framework (como **Symfony** o Laravel) es una base de código y un conjunto de herramientas de Programación Orientada a Objetos (POO) que te permite construir aplicaciones complejas más rápido y de forma más ordenada.

Fundamentos de PHP

Unidad 3: Sintaxis, Variables y Operadores

Sintaxis Básica de PHP

- **Etiquetas:** El código PHP siempre debe ir dentro de las etiquetas para que el servidor sepa qué ejecutar.
- **Variables:** Siempre empiezan con el símbolo dólar (`$`). Son sensibles a mayúsculas.
  `$nombre_alumno = "Juan";`
- **Comentarios:** Se usa `//` para una línea o `/* ... */` para bloques.
- **Punto y Coma:** Cada instrucción debe terminar con `;`.
- **Imprimir:** `echo` se usa para enviar salida al HTML.



Concepto: Cadenas (Strings)

Comillas Simples (' ')

Tratan el texto de forma **literal**. No interpretan variables ni caracteres especiales (excepto `\'`).

```
$edad = 20;
echo 'Tengo $edad años';
```

Salida: `Tengo $edad años`

Comillas Dobles (" ")

Interpretan (o "interpolan") las variables. Buscan símbolos `$` dentro de la cadena para reemplazarlos.

```
$edad = 20;
echo "Tengo $edad años";
```

Salida: `Tengo 20 años`

Concepto: Conversión de Tipos

Conversión Automática (Juggling)

PHP es un lenguaje "débilmente tipado". Intenta convertir tipos automáticamente según el contexto.

Contexto Numérico (+, -, *, /):

```
$valor = 1 + "3 coches";
```

PHP convierte `"3 coches"` al número `3` (ignora el texto) y luego suma `1 + 3`.

Salida (`echo $valor`): `4`

Concatenación de Cadenas (.)

El operador para unir strings es el punto (`.`).

Contexto de String (.):

```
$texto = "Nota media: " . 7.5;
```

PHP convierte el número `7.5` al string `"7.5"` y lo une.

Salida (`echo $texto`): `Nota media: 7.5`

Operadores de Comparación

Dadas las variables: `$a = 5;` (int) y `$b = "5";` (string)

Operador	Nombre	Descripción	Ejemplo	Resultado
`==`	Igualdad	Compara solo el **valor**. (Convierte tipos).	`$a == $b`	true
`===`	Identidad	Compara **valor Y tipo**. (No convierte).	`$a === $b`	false
`!=`	Distinto	Compara solo el valor.	`$a != $b`	false
`!==`	No Idéntico	Compara valor O tipo.	`$a !== $b`	true

Estructuras de Control

Unidad 3: Tomando Decisiones y Repitiendo Tareas

Condicionales: `if`, `elseif`, `else`

Permiten ejecutar diferentes bloques de código basándose en una condición.

- `if ($condicion)`
  Se ejecuta si la condición es `true`. Es el primer bloque que se comprueba.

- `elseif ($otra_condicion)`
  Se comprueba **solo si el** `if` **anterior fue** `false`. Puedes tener múltiples `elseif`.

- `else`
  Se ejecuta si **todas** las condiciones `if` y `elseif` anteriores fueron `false`.

Ejemplo de Notas (Examen Práctico):

```
if ($nota >= 9) { echo "Sobresaliente"; }
elseif ($nota >= 7) { echo "Notable"; }
elseif ($nota >= 5) { echo "Aprobado"; }
else { echo "Suspenso"; }
```

Bucles Repetitivos

Bucle `while`

El bucle `while` (mientras) repite un bloque de código **mientras** una condición sea verdadera.

1. Se comprueba la condición.
2. Si es `true`, se ejecuta el código del bloque.
3. Se vuelve al paso 1.

¡Cuidado! Debes asegurarte de que la condición cambie (ej. `$i++`), o crearás un **bucle infinito** que bloqueará el script.

Bucle `for`

El bucle `for` es ideal para cuando sabes cuántas veces quieres repetir algo. Agrupa todo en una línea:

```
for (inicio; condición; incremento)
```

Ejemplo (Contar del 1 al 5):

```
$i = 1; // 1. Inicio
while ($i <= 5) { // 2. Condición
echo $i;
$i++; // 3. Incremento
}
```

Equivalente en `for`:

```
for ($i = 1; $i <= 5; $i++) {
echo $i;
}
```

Estructuras de Datos

Unidad 4: Arrays y Cómo Recorrerlos

¿Qué es un Array?

Array Indexado (Numérico)

Una lista donde cada valor tiene una clave numérica (índice), que generalmente empieza en 0.

```
$menu = ["Home", "Nosotros", "Contacto"];
```

Equivale a:

```
[0 => "Home", 1 => "Nosotros", 2 => "Contacto"]
```

Acceso: `echo $menu[0]; // "Home"`

Array Asociativo (Texto)

Una colección de pares **clave => valor**, donde la clave es un string que tú defines.

```
$usuario = [
"nombre" => "Paco",
"edad" => 42,
"esProfesor" => true
];
```

Acceso: `echo $usuario["nombre"]; // "Paco"`

Recorrer Arrays: `for` vs `foreach`

Usando `for`

Ideal para arrays **indexados** numéricos. Necesitas saber el tamaño con `count()`.

```
for ($i = 0; $i < count($menu); $i++) {
echo $menu[$i];
}
```

Usando `foreach` (Recomendado)

Es la forma más fácil y segura. Funciona con **cualquier** array (indexado o asociativo).

Sintaxis de solo valor:

```
foreach ($menu as $item) {
echo $item;
}
```

Sintaxis de clave y valor:

```
foreach ($usuario as $clave => $valor) {
echo "$clave: $valor";
}
```

Funciones Útiles para Depurar

¡Importante! Estas funciones son solo para depuración. No deben usarse en el código de producción final.

Función	Descripción	Salida de Ejemplo
`print_r($array)`	Muestra la estructura del array de forma **legible para humanos**. No muestra tipos de datos.	`Array ( [0] => Home [1] => Nosotros )`
`var_dump($array)`	Muestra información **detallada** sobre la variable, incluyendo la estructura, los **tipos de datos** y la longitud.	`array(2) { [0]=> string(4) "Home" [1]=> string(8) "Nosotros" }`

¿Preguntas?

¡Mucha suerte en el examen!

Image Sources

https://external-preview.redd.it/3dEzQ_ODmdGplWw4J1KS7_T1iyPKgROk0ce3RLOdzdE.jpg?auto=webp&s=8f3265319b28f560ce51b597805968a4514aabe3

Source: www.reddit.com