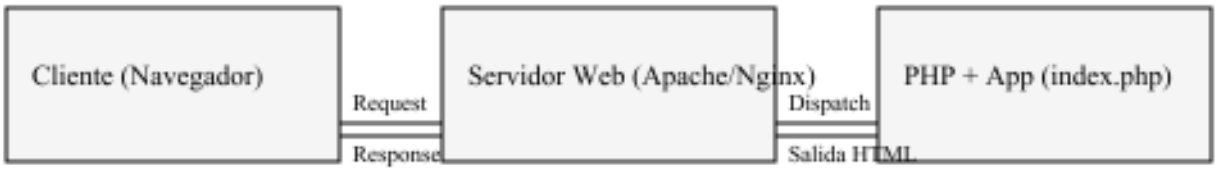
■ Plan Intensivo de PHP desde Cero (12 semanas)

Objetivo: llegar a una tienda online con carrito de compras, seguridad básica y buenas prácticas.

Carga horaria: ~10 horas/semana · Nivel inicial: principiante total en PHP



Estructura sugerida del proyecto (DocumentRoot en public/)

```
project/
  public/  (index.php, assets)
  src/
    Controllers/
    Models/
    Views/
    Services/
    Support/
  config/
  database/ (schema.sql, seeds)
  tests/
  vendor/
  .env
  composer.json
```

Resumen semanal

Semana	Tema principal
Semana 1	Introducción a PHP y entorno de trabajo
Semana 2	Sintaxis base, control de flujo y funciones
Semana 3	Arrays y strings para catálogo inicial
Semana 4	Formularios, GET/POST, validación y CSRF
Semana 5	MySQL con PDO y CRUD seguro
Semana 6	POO en PHP y servicios de dominio
Semana 7	Estructura MVC ligera, Composer y autoload PSR-4
Semana 8	Sesiones, cookies y autenticación robusta
Semana 9	Catálogo y carrito persistente
Semana 10	Checkout y órdenes transaccionales
Semana 11	Calidad, testing y seguridad
Semana 12	Deploy y cierre del proyecto final

Semana 1 — Introducción a PHP y entorno de trabajo

Concepto clave: PHP es un lenguaje del lado del servidor. Cada solicitud HTTP ejecuta un script que genera HTML. Trabajaremos con un servidor local (XAMPP/Laragon/Docker) y ubicaremos el DocumentRoot en public/.

Diferencia con JavaScript/Python: A diferencia de JavaScript (que suele correr en el navegador) y Python (scripts persistentes), PHP corre por solicitud. Usa etiquetas para incrustar en HTML. Variables siempre con $, arrays son flexibles (índices numéricos o claves string).

Buena práctica: Habilita errores en desarrollo y desactívalos en producción. Estructura desde el inicio carpetas claras.

Ejercicios prácticos:

- 1. Instala PHP y crea un archivo index.php que imprima 'Hola, Mundo' y la fecha actual.
- 2. Usa phpinfo() para verificar configuración; crea una página /about con tu nombre.
- 3. Declara variables ($nombre, $edad) y concaténalas en un párrafo HTML.
- 4. Muestra superglobales mínimas (REQUEST_METHOD, REQUEST_URI) de forma segura.
- 5. Crea encabezados HTTP con header() para redirigir de / a /about.

Semana 2 — Sintaxis base, control de flujo y funciones

Concepto clave: Condicionales (if/elseif/else), bucles (for/while/foreach) y funciones con parámetros tipados opcionales. declare(strict_types=1) al inicio para forzar tipado estricto en firmas.

Diferencia con JavaScript/Python: En PHP la comparación estricta (===) es crucial para evitar coerciones. Las funciones se definen con function y no requieren clase.

Buena práctica: Prefiere return en funciones y deja echo para la capa de presentación. Evita comparaciones flojas (==).

Ejercicios prácticos:

- 1. Función esPar(int $n): bool y tests simples (eco del resultado).
- 2. Tabla de multiplicar con for y versión con foreach sobre range(1,10).
- 3. Función promedio(array $nums): float (valida array no vacío).
- 4. Simula un login básico comparando usuario/clave en variables (sin BD).
- 5. Escribe una mini-librería de helpers numéricos (sumar, max, min) y úsalos.

Semana 3 — Arrays y strings para catálogo inicial

Concepto clave: Arrays indexados y asociativos con foreach; funciones útiles (array_map, array_filter, array_reduce). Manejo de strings (strlen, strtolower, preg_replace).

Diferencia con JavaScript/Python: PHP tiene un único tipo array para listas y diccionarios. DateTimeImmutable es preferible para manipular fechas.

Buena práctica: Escapa siempre el output de usuario con htmlspecialchars(). No mezcles lógica y presentación.

Ejercicios prácticos:

- 1. Arma un 'catálogo' en un array asociativo con name, price, category.
- 2. Filtra por categoría con array_filter y renderiza en una tabla HTML.
- 3. Crea slugs para productos (minúsculas + reemplazo de espacios por guiones).
- 4. Calcula el total del carrito simulado con array_reduce.
- 5. Ordena por precio asc/desc usando usort con función de comparación.

Semana 4 — Formularios, GET/POST, validación y CSRF

Concepto clave: Recepción de datos con $_GET/$_POST. Validación (requerido, tipos, límites). Token CSRF en sesión para prevenir ataques.

Diferencia con JavaScript/Python: PHP no trae middleware por defecto; implementás tu token CSRF y validación manual. La seguridad del lado servidor es obligatoria.

Buena práctica: Valida todo en servidor; normaliza strings (trim) y escapa al renderizar. Generá y verificá token CSRF en POST.

Ejercicios prácticos:

- 1. Formulario de contacto con validación de requerido y email válido.
- 2. Helper old('campo') para repoblar formularios tras errores.
- 3. Implementa token CSRF: genera en sesión, inclúyelo oculto, verifica.
- 4. Muestra mensajes de error arriba del formulario en una lista.
- 5. Agrega un formulario 'Nuevo producto' que valide price numérico y stock entero.

Semana 5 — MySQL con PDO y CRUD seguro

Concepto clave: Conectar a MySQL vía PDO, ejecutar SELECT/INSERT/UPDATE/DELETE con prepared statements. Manejo de excepciones try/catch.

Diferencia con JavaScript/Python: Evita mysqli_* o concatenar SQL. PDO permite transacciones y múltiples drivers con la misma API.

Buena práctica: Usa consultas preparadas siempre; registra errores a logs, no al usuario. Define índices en columnas de búsqueda.

Ejercicios prácticos:

- 1. Crea BD tienda y tabla products(id, sku, name, price, stock, category_id).
- 2. Clase Database que retorne PDO configurado (ERRMODE_EXCEPTION).
- 3. Inserta 10 productos desde un script seed.
- 4. Página products.php que lista con paginación LIMIT/OFFSET parametrizados.
- 5. Implementa create/update/delete con formularios validados y CSRF.

Semana 6 — POO en PHP y servicios de dominio

Concepto clave: Clases, propiedades tipadas (PHP 7.4+), constructores, visibilidad; Value Objects para precios; servicios con reglas de negocio.

Diferencia con JavaScript/Python: PHP tiene traits, propiedades readonly (8.2) y tipos en firmas. Inyección de dependencias simple vía constructor.

Buena práctica: Clases final por defecto, interfaces para servicios externos, SRP (Single Responsibility Principle).

Ejercicios prácticos:

- 1. Clase Product con validaciones (precio > 0, stock ≥ 0).
- 2. Cart (puro) con add/remove/total; no tocar I/O ni sesión.
- 3. Interface CartStorageInterface (session/db) + implementación SessionCartStorage.
- 4. AuthService con login/logout usando repositorio de usuarios.
- 5. Escribe tests rápidos (sin PHPUnit) invocando métodos y verificando resultados.

Semana 7 — Estructura MVC ligera, Composer y autoload PSR-4

Concepto clave: Composer para gestionar dependencias, autoload PSR-4 para App*. Router simple que resuelva Controlador@acción. Vistas como plantillas.

Diferencia con JavaScript/Python: Composer ~ npm/pip, pero además configura autoload. PHP requiere front controller (public/index.php) y reescrituras.

Buena práctica: Separá public/ del core. Cargá variables de entorno (.env) con phpdotenv. Evitá require spaghetti.

Ejercicios prácticos:

- 1. Inicializa composer.json y define autoload PSR-4 para App\.
- 2. Implementa un router mínimo (GET/POST) y HomeController.
- 3. Crea vistas con función render($view, $data).
- 4. Controlador ProductsController con list/show/create/update/delete.
- 5. Middleware CSRF y Auth aplicado a rutas protegidas.

Semana 8 — Sesiones, cookies y autenticación robusta

Concepto clave: Gestión de sesión (session_start, regenerar id), cookies seguras (HttpOnly, SameSite), contraseñas con password_hash/verify.

Diferencia con JavaScript/Python: Sin framework, construirás rate limiting básico y verificación de correo (token) de forma manual o con librería PHPMailer.

Buena práctica: Regenera ID al iniciar sesión; guarda solo el id de usuario en sesión; limita intentos de login; registra eventos.

Ejercicios prácticos:

- 1. Tabla users con email único y password_hash.
- 2. Registro y login con validación y mensajes de error específicos.
- 3. Middleware role('admin') para panel de administración.
- 4. Flujo 'olvidé mi contraseña' con token de un solo uso.
- 5. Registro de eventos de login (ip, user agent, resultado).

Semana 9 — Catálogo y carrito persistente

Concepto clave: Listar catálogo con filtros (categoría, precio), carrito en sesión con fusión a DB al iniciar sesión, cálculo de totales en servidor.

Diferencia con JavaScript/Python: El servidor es la fuente de verdad para totales; no aceptes precios desde el cliente. Revalida stock en cada operación.

Buena práctica: Idempotencia en agregar/eliminar. Valida qty ∈ [1..stock]. Recalcula totales del lado servidor siempre.

Ejercicios prácticos:

- 1. GET /products?category=x&min;=..&max;=.. con paginación.
- 2. POST /cart/add con validaciones y mensajes flash.
- 3. Vista /cart con subtotal, impuestos y total.
- 4. Persistencia del carrito en DB y fusión al loguearse.
- 5. Tests manuales: simula add concurrentes y verifica stock.

Semana 10 — Checkout y órdenes transaccionales

Concepto clave: Flujo de checkout por pasos: dirección y envío → revisión → confirmar. Inserción transaccional de order + items y decremento de stock.

Diferencia con JavaScript/Python: Implementarás una 'pasarela' simulada para centrarse en la consistencia. Usa transacciones y, si es posible, SELECT ... FOR UPDATE.

Buena práctica: Si cualquier paso falla, rollback. Envía email de confirmación (PHPMailer). No vacíes el carrito hasta commit exitoso.

Ejercicios prácticos:

- 1. Modelo Address y formulario validado.
- 2. Resumen de pedido recálculo (no aceptar valores del cliente).
- 3. OrderService::placeOrder() con transacción completa.
- 4. Página de 'Mis pedidos' y detalle por id.
- 5. Admin: cambiar estado (pendiente, pagado, enviado, cancelado).

Semana 11 — Calidad, testing y seguridad

Concepto clave: PHPUnit para tests, PHPStan para análisis estático, PHPCS para estilo PSR-12, cabeceras de seguridad y checklist de vulnerabilidades.

Diferencia con JavaScript/Python: Herramientas distintas pero propósito igual que en JS/Python: feedback temprano, contratos de tipo, estilo consistente.

Buena práctica: Antes de commitear: phpunit, phpstan analyse, phpcs. Configura Content-Security-Policy y otras cabeceras.

Ejercicios prácticos:

- 1. Unit tests de Cart (add/remove/total).
- 2. Integration tests de repos con BD de prueba.
- 3. Configura PHPStan (nivel 6–8) y corrige hallazgos.
- 4. Añade rate limiting en login y add-to-cart.
- 5. Checklist de XSS, CSRF, SQLi, IDOR y revisión ruta por ruta.

Semana 12 — Deploy y cierre del proyecto final

Concepto clave: Despliegue en Apache/Nginx con PHP-FPM, DocumentRoot en public/, variables de entorno separadas, OPcache y backups.

Diferencia con JavaScript/Python: A diferencia de dev, en prod se deshabilitan errores en pantalla; logs a archivo/servicio; no exponer .env ni vendor/.

Buena práctica: Automatiza pasos de build (composer install --no-dev, dump-autoload -o). Prepara README y .sql de la BD.

Ejercicios prácticos:

- 1. Configura virtual host y reescrituras a public/index.php.
- 2. Prepara .env.prod con credenciales reales seguras.
- 3. Activa OPcache y verifica tiempos de respuesta.
- 4. Script de backup y restauración de la BD.
- 5. Entrega final: código, README, capturas y checklist de QA.

Buenas prácticas transversales

Tema	Recomendación
Nomenclatura	PSR-12; Clases en PascalCase; métodos/variables en camelCase; constantes en UPPER_
Estructura	Separá public/ del core; MVC ligero; servicios y repositorios. Vistas sin lógica.
Seguridad	Escapado de salida (htmlspecialchars), prepared statements (PDO), CSRF, regenerar id de
Configuración	Credenciales y claves en .env; nunca en el código.
Errores/Logs	En dev mostrar errores; en prod solo logs. Mensajes de usuario genéricos.
Rendimiento	Paginación, índices de BD, OPcache, seleccionar solo columnas necesarias.
Testing	Tests unitarios del dominio (Cart), integración de repositorios y flujos críticos (login, placeOr

Proyecto Final — Tienda online con carrito

Desarrollá una tienda completa con catálogo, autenticación, carrito persistente, checkout y órdenes. Usa PDO para la BD y aplica CSRF, escaped output y roles para seguridad básica.

- Registro/login, recuperación de contraseña, roles (cliente/admin).
- Catálogo con categorías, búsqueda y paginación.
- Carrito en sesión con persistencia al loguearse y fusión de carritos.
- Checkout por pasos y creación de órdenes transaccional.
- Página de 'Mis pedidos' y panel de administración básico.
- Criterios de aceptación: flujo completo, stock decrementado una sola vez, tests mínimos y estilos PSR-12.