📚 GUÍA COMPLETA DEL CURSO

Programación Full Stack con JavaScript, HTML, PHP y MySQL

Sistema de Gestión para Gimnasios

📖 TABLA DE CONTENIDOS

#	Módulo	Duración	Páginas
15	Dashboard: Métricas + Gráficos Chart.js	25 min	12
16	CRUD de Socios con Subida de Fotos	42 min	16
17	Control de Asistencia (Registro Diario Único)	22 min	10
18	Membresías + Inscripciones (Relaciones 1 a Muchos)	43 min	17
19	Módulo de Pagos + Filtros + Exportación CSV	22 min	11
20	CRUD de Entrenadores (Módulo Final)	31 min	12
PARTE V: DESPLIEGUE Y PROFESIONALISMO			
21	Pruebas de API con Postman (Testing Profesional)	26 min	13
22	Despliegue con FileZilla (FTP/SFTP + Permisos)	7 min	6
23	Control de Versiones con GitHub y Git	9 min	7
CONCLUSIÓN	Resumen Final + Próximos Pasos	-	5
APÉNDICES	Libros Recomendados + Recursos Adicionales	-	4

📄 PORTADA

```
┌──────────────────────────────────────────────────────────┐
│                                                          │
│              PROGRAMACIÓN FULL STACK                     │
│    Sistema de Gestión para Gimnasios con PHP y MySQL     │
│                                                          │
│              Guía Completa del Curso                     │
│          23 Sesiones • Proyecto Profesional • Portafolio │
│                                                          │
│                    Por Gustavo Arias                     │
│           Senior Data Architect • Instructor Full Stack  │
│                                                          │
└──────────────────────────────────────────────────────────┘
```

📌 INTRODUCCIÓN AL CURSO

¿Por qué este curso es diferente?

Este no es un curso teórico ni un tutorial fragmentado. Es un **proyecto real desde cero hasta producción** que construirás paso a paso, explicando cada línea de código para que entiendas no solo el "qué", sino el "porqué".

Resultado final:

Un sistema de gestión profesional para gimnasios con:

- ✅ Autenticación segura (login + recuperación con tokens)
- ✅ CRUD completo de socios con subida de fotos
- ✅ Control de asistencias diarias únicas
- ✅ Gestión de planes y membresías
- ✅ Registro de pagos con filtros y exportación CSV
- ✅ Dashboard con métricas en tiempo real y gráficos
- ✅ Despliegue en dominio real con Hostinger
- ✅ Código en GitHub con control de versiones

Stack tecnológico:

Frontend	Backend	Base de Datos	Herramientas
HTML5	PHP 8	MySQL	Visual Studio Code
CSS3	API REST	PHPMyAdmin	XAMPP
JavaScript	Autenticación	Consultas SQL	Postman
Bootstrap 5	Password Hashing	Normalización	Git + GitHub
jQuery	Sesiones PHP	Claves Foráneas	FileZilla
Ajax	Variables de Entorno	Sentencias Preparadas	Hostinger
DataTables	PHPMailer	Sanitización	Easy Panel
Chart.js	.env	XSS Prevention	SSL/HTTPS
SweetAlert	MySQLi	SQL Injection Prevention	FTP/SFTP

🏵️ PARTE I: FUNDAMENTOS Y ENTORNO

Sesión 1: Introducción al Curso Full Stack (8 min)

Objetivo: Entender la visión global del curso y el sistema que construiremos.

Contenido clave:

- Duración: 4 sesiones de 2 horas (8 horas totales)
- Nivel: Principiante a intermedio (sin experiencia previa requerida)
- Resultado: Sistema 100% funcional listo para producción y portafolio
- Arquitectura modular sin frameworks pesados
- Enfoque en seguridad desde el primer día:
    - Sentencias preparadas (100% de consultas)

- Password hashing con `password_hash()`
- Variables de entorno (.env) para credenciales
- Validación y sanitización de datos

Módulos del sistema:

```
auth/          → Login, logout, recuperación contraseña
dashboard/     → Métricas, gráficos Chart.js, tabla pagos
members/       → CRUD socios + subida fotos
attendance/    → Registro diario único + validación membresía
memberships/   → Planes (mensual, trimestral, anual, VIP)
enrollments/   → Inscripciones socio + plan + fechas
payments/      → Registro pagos + filtros + exportación CSV
trainers/      → CRUD entrenadores + especialidades
PHP/           → Backend modular (auth.php, members.php, etc.)
config/        → database.php + .env (credenciales seguras)
images/        → Fotos socios (image_members/) y entrenadores
vendor/        → PHPMailer para envío de correos
```

Sesión 2: Presentación del Sistema de Gimnasio (6 min)

Demostración en vivo del sistema completo:

1. **Página de inicio (index.html)**
   - Hero section con logo y título H1 único
   - Tres tarjetas con iconos Font Awesome:
     - Gestión de socios (CRUD + fotos)
     - Control de asistencias (registro diario)
     - Pagos y membresías (planes + facturación)
   - Diseño 100% responsive (3 columnas escritorio → 1 móvil)
2. **Autenticación**
   - Formulario login con validación frontend/backend
   - Recuperación contraseña con token expirable (1 hora)
   - Protección contra fuerza bruta y SQL injection
3. **Dashboard**
   - 4 tarjetas métricas: socios activos, asistencias hoy, membresías vencer, pagos recientes
   - Tabla últimos 5 pagos con DataTables (búsqueda + paginación)
   - Gráfico barras asistencia últimos 7 días con Chart.js
4. **Módulos funcionales**
   - Socios: CRUD completo + subida fotos + búsqueda instantánea
   - Asistencias: selector socios activos + registro único diario
   - Membresías: planes con duración, precio, beneficios

- Inscripciones: vincular socio + plan + fechas automáticas
- Pagos: monto, método (efectivo/tarjeta/transferencia), notas
- Entrenadores: CRUD con especialidades y contacto

Sesión 3: Instalación de Herramientas (12 min)

Herramientas esenciales (instalación paso a paso):

Herramienta	Propósito	Versión Recomendada
Visual Studio Code	Editor de código principal	Última estable
XAMPP	Servidor local (Apache + PHP + MySQL)	8.2.x
Postman	Pruebas API REST sin frontend	Última estable
FileZilla	Transferencia FTP/SFTP a servidor	Última estable
Git	Control de versiones	Última estable

Flujo de trabajo recomendado:

```
1. Desarrollo local (XAMPP + VS Code)
   ↓
2. Pruebas API (Postman)
   ↓
3. Debugging (consola navegador + error_log)
   ↓
4. Control de versiones (Git + GitHub)
   ↓
5. Despliegue producción (FileZilla + Hostinger)
```

Notas importantes:

- MySQL Workbench no es necesario (usamos phpMyAdmin incluido en XAMPP)
- Librerías vía CDN: Bootstrap, jQuery, Font Awesome, Chart.js, DataTables, SweetAlert
- Figma no se utiliza (diseño directo en código)

Sesión 4: Arquitectura Técnica y Stack Tecnológico (6 min)

Stack técnico ligero, seguro y enfocado en resultados prácticos:

Desarrollo local:

- Visual Studio Code como editor principal
- XAMPP para simular servidor web real con Apache, PHP 8 y MySQL
- Postman para validar cada endpoint de nuestra API
- Git para gestionar cada cambio, manteniendo historial limpio y colaborativo

Producción en Hostinger:

- Sistema se ejecuta en dos contenedores independientes:
  - Servicio PHP: aloja frontend (HTML, CSS, JavaScript) y backend (PHP)
  - Servicio MySQL: base de datos aislada
- Conexión segura por nombre de servicio, nunca por localhost
- Credenciales gestionadas mediante variables de entorno (.env)

Sistema resultante incluye:

- Autenticación segura con login y recuperación de contraseñas
- Gestión completa de socios, entrenadores, membresías y asistencia
- Dashboard dinámico con gráficos Chart.js y tablas DataTables
- API REST lista para futuras integraciones con CRM o herramientas BI

Sesión 5: Diagrama de Contexto del Sistema (5 min)

Visión más alta y esencial del sistema:

Actores principales:

1. **Usuarios finales/clientes/socios:** Acceden al portal público para ver servicios y planes
2. **Personal administrativo y entrenadores:** Inician sesión en panel privado con credenciales seguras
3. **Sistema de correo electrónico:** Integra PHPMailer para notificaciones automáticas

Importante: El sistema es autónomo en su MVP, no depende de servicios externos críticos. Todo desde base de datos hasta lógica de negocio se ejecuta en entorno controlado (local o Hostinger + Easy Panel).

Este diagrama responde tres preguntas fundamentales:

1. ¿Qué hace el sistema?
2. ¿Quién lo usa?
3. ¿Con qué se comunica?

Define qué NO está en el alcance: Procesadores de pagos automáticos, manteniendo enfoque en MVP sólido, usable y entregable en poco tiempo.

Sesión 6: Estructura Final del Proyecto (7 min)

Organización por funcionalidades:

```
Gym System/
├── auth/
│   ├── index.html
│   └── auth.js
├── dashboard/
│   ├── index.html
│   └── dashboard.js
├── members/
│   ├── index.html
│   └── members.js
```

```
├── attendance/
│   ├── index.html
│   └── attendance.js
├── memberships/
│   ├── index.html
│   └── memberships.js
├── enrollments/
│   ├── index.html
│   └── enrollments.js
├── payments/
│   ├── index.html
│   └── payments.js
├── trainers/
│   ├── index.html
│   └── trainers.js
├── PHP/
│   ├── auth.php
│   ├── dashboard.php
│   ├── members.php
│   ├── attendance.php
│   ├── memberships.php
│   ├── enrollments.php
│   ├── payments.php
│   └── trainers.php
├── config/
│   ├── database.php
│   └── .env
├── images/
│   ├── image_members/
│   └── image_trainers/
├── vendor/
│   └── PHPMailer/
├── .gitignore
├── index.html
└── README.md
```

Beneficios de esta estructura:

- Modularidad pura: un desarrollador trabaja en pagos mientras otro mejora dashboard
- Backend modular: cada archivo PHP actúa como mini controlador con Ajax
- Seguridad centralizada: database.php nunca expone credenciales
- .env dentro de .gitignore: secretos nunca subidos a GitHub
- Fotos organizadas en carpetas específicas
- Autenticación aislada para auditorías
- PHPMailer listo para enviar correos

PARTE II: BASE DE DATOS Y FRONTEND

Sesión 7: Base de Datos - Esquema SQL Completo (10 min)

Archivo esquema.sql: columna vertebral del sistema

Tablas principales:

- `users`: personal interno (admin, recepcionista, entrenadores) con roles y contraseñas hasheadas
- `members`: socios con nombre, DNI, contacto y foto (ruta relativa)
- `memberships`: planes con duración, precio y beneficios
- `member_memberships`: vincula socios con planes, fechas inicio/vencimiento, estados
- `attendance`: registro diario evitando duplicados con restricción UNIQUE
- `payments`: pagos manuales asociados a membresías (no socios)
- `trainers`: entrenadores con especialidades y contacto
- `password_resets`: tokens expirables para recuperación segura
- `logs`: auditoría completa de quién hizo qué, desde qué IP y cuándo

Características clave:

- Base de datos `gym_management` con soporte UTF8MB4
- Claves foráneas y reglas de integridad
- Datos de prueba incluidos (admin, recepcionista, socios, planes)
- Funciona igual en localhost y producción

Sesión 8: Automatización con PowerShell (6 min)

Script crear_gym_system.ps1:

Crea automáticamente toda la estructura del proyecto:

- Carpetas de módulos (auth, dashboard, members, etc.)
- Archivos HTML y JS vacíos con comentario inicial
- Estructura de imágenes con subcarpetas
- Carpetas PHP, config, vendor
- Archivos raíz (.gitignore, README.md, index.html)

Comando para ejecutar:

```
.\crear_gym_system.ps1
```

Beneficio: Ahorra tiempo y evita errores manuales al crear cada carpeta y archivo.

Sesión 9: Página de Inicio - HTML5 + Bootstrap Responsive (22 min)

Estructura del index.html:

HEAD (la cocina invisible):

- Caracteres UTF-8 para acentos y símbolos
- Viewport para compatibilidad móvil
- Título y descripción para SEO
- Favicon desde carpeta images

- Estilos Bootstrap y librerías Font Awesome
- CSS personalizado para identidad única

BODY (el comedor visible):

- Hero section: logo, título H1 único, subtítulo, botón "Iniciar Sesión"
- Features section: 3 tarjetas con iconos Font Awesome
  - Gestión de socios (CRUD + fotos)
  - Control de asistencias (registro diario)
  - Pagos y membresías (planes + facturación)
- Footer: logotipo pequeño, derechos de autor, enlace a marca personal

Diseño 100% responsive:

- Columnas `col-md-4` = 3 columnas escritorio, 1 móvil
- Comportamiento adaptativo gracias a Bootstrap
- Sin Bootstrap: página sin estilos (demostración visual)

Sesión 10: Configuración de Dominio en Hostinger (15 min)

Pasos para crear dominio .shop:

1. **Comprar dominio:**
   - gymystem.shop por $0.99 primer año (97% descuento)
   - Configuración DNS y Name Server
2. **Panel de control ESTIA:**
   - Crear usuario "entrenador" con datos de contacto
   - Configurar dominio web con soporte DNS
   - Habilitar redireccionamiento www
   - Activar certificado SSL gratuito
   - Crear cuenta FTP adicional
3. **FileZilla para subir archivos:**
   - Configurar Site Manager con IP, usuario, contraseña
   - Conectar al servidor remoto
   - Navegar a carpeta public_html
   - Arrastrar carpeta Gym System con index.html
   - Sobrescribir archivo existente (página de construcción)
4. **Verificar en navegador:**
   - Acceder a gymystem.shop
   - Sistema online y funcional

🧩 PARTE III: AUTENTICACIÓN Y SEGURIDAD

Sesión 11: Módulo de Autenticación (Login + Recuperación) (33 min)

Archivos fundamentales:

1. **index.html:** Interfaz visual con formularios de login y recuperación
2. **auth.js:** Sistema nervioso con JavaScript, validaciones y comunicación Ajax
3. **auth.php:** Backend en PHP que procesa peticiones y verifica credenciales
4. **database.php + .env:** Capa de seguridad que protege credenciales de base de datos

Conceptos esenciales explicados:

HTML (el esqueleto):

```
<form id="loginForm" class="d-none">
    <input type="text" id="username" required>
    <input type="password" id="password" required>
</form>
```

JavaScript (el sistema nervioso):

```
// Verificar sesión al cargar
$.ajax({
    url: '../PHP/auth.php?action=check_session',
    method: 'GET',
    success: function(data) {
        if (data.status !== 'active') {
            window.location.href = '../auth/index.html';
        }
    }
});

// Enviar login al servidor
$.ajax({
    url: '../PHP/auth.php?action=login',
    method: 'POST',
    contentType: 'application/json',
    data: JSON.stringify({ username, password }),
    success: function(data) {
        window.location.href = data.redirect;
    }
});
```

PHP (ejecuta en servidor):

```
// Recibir acción por GET
$action = isset($_GET['action']) ? $_GET['action'] : null;

switch ($action) {
```

```
    case 'login':
        handleLogin();
        break;
    case 'logout':
        handleLogout();
        break;
    case 'request_reset':
        handlePasswordResetRequest();
        break;
    default:
        http_response_code(400);
        echo json_encode(['status' => 'error', 'message' => 'Acción no válida']);
}
```

Password Hashing:

```
// Almacenar contraseña segura
$hashedPassword = password_hash($password, PASSWORD_DEFAULT);

// Verificar contraseña al login
if (password_verify($password, $hashedPassword)) {
    // Contraseña correcta
}
```

Variables de Entorno (.env):

```
# Entorno local
DB_HOST_DEV=localhost
DB_USER_DEV=root
DB_PASS_DEV=
DB_NAME_DEV=gym_management

# Entorno producción
DB_HOST_PROD=localhost
DB_USER_PROD=user_db
DB_PASS_PROD=clave_produccion_123#
DB_NAME_PROD=user_db_gym_system
```

PHPMailer para envío de correos:

```
$mail = new PHPMailer(true);
$mail->isSMTP();
$mail->Host = 'smtp.gmail.com';
$mail->SMTPAuth = true;
$mail->Username = 'tu_email@gmail.com';
$mail->Password = 'contraseña_aplicacion'; // Generada en Gmail
```

```
$mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;
$mail->Port = 465;
```

Sesión 12: Depuración en Frío (Debugging Profesional) (32 min)

Herramientas de desarrollo:

1. **Activar errores de PHP:**

```
error_reporting(E_ALL);
ini_set('display_errors', 1);
```

2. **Console.log en JavaScript:**

```
console.log('Datos enviados:', username, password);
```

3. **Network Tab en navegador:**

- Ver peticiones Ajax
- Inspeccionar request y response
- Verificar content-type application/json

4. **Var_dump y error_log en PHP:**

```
// Mostrar en navegador (solo desarrollo)
<pre>
<?php
var_dump($input);
?>
</pre>
<?php die(); ?>

// Guardar en archivo de log
error_log("Input recibido: " . print_r($input, true));
```

5. **PHPMyAdmin para consultas SQL:**

- Simular consultas directamente
- Verificar resultados antes de código

6. **Variables de sesión:**

```
<pre>
<?php
```

```
echo "Login exitoso, sesión creada\n";
var_dump($_SESSION);
?>
</pre>
<?php die(); ?>
```

Flujo de debugging:

1. Preparar entorno (activar errores)
2. Depurar login desde JavaScript (console.log)
3. Ver qué recibe PHP (var_dump + Network tab)
4. Verificar consulta SQL en PHPMyAdmin
5. Ver variables de sesión creadas
6. Probar flujo completo sin SweetAlert
7. Depurar recuperación de contraseña
8. Configurar PHPMailer en modo debug

Sesión 13: Migración a Producción + PHPMailer con Gmail (31 min)

Pasos para migrar a servidor remoto:

1. **Subir archivos con FileZilla:**
   - Conectar al sitio FTP creado
   - Arrastrar carpeta Gym System a public_html
   - Sobrescribir index.html existente
2. **Crear base de datos en Hostinger:**
   - Panel VPS → Web → Database
   - Nombre: user_db_gym_system
   - Usuario: user_db
   - Contraseña: clave_produccion_123#
3. **Importar esquema SQL:**
   - Acceder a PHPMyAdmin: gymystem.shop/phpmyadmin
   - Pestaña SQL → Pegar esquema completo
   - Ejecutar → Tablas creadas con datos de prueba
4. **Configurar archivo .env para producción:**

```
DB_HOST=localhost
DB_USER=user_db
DB_PASS=clave_produccion_123#
DB_NAME=user_db_gym_system

SMTP_HOST=smtp.gmail.com
SMTP_PORT=465
```

```
SMTP_USER=tu_email@gmail.com
SMTP_PASS=contraseña_aplicacion_gmail
```

5. **Crear contraseña de aplicación en Gmail:**
    - Gestionar cuenta Google → Seguridad
    - Verificación en dos pasos → Contraseñas de aplicación
    - Nombre: Gym System → Generar
    - Copiar contraseña de 16 caracteres
6. **Subir archivos modificados:**
    - database.php actualizado
    - .env con credenciales producción
    - auth.php corregido (ruta gym_system removida)
7. **Probar envío de correo:**
    - Olvidar contraseña → Ingresar email
    - Recibir correo con enlace de recuperación
    - Token incluido en URL
    - Enlace expira en 1 hora

Sesión 14: Reset Password con Tokens Expirables (33 min)

Archivo reset_password.php:

Flujo completo:

1. Usuario olvida contraseña → Ingresa email
2. Sistema genera token seguro con `bin2hex(random_bytes(32))`
3. Guarda token en tabla `password_resets` con fecha expiración (+1 hora)
4. Envía email con enlace: `reset_password.php?token=abc123`
5. Usuario hace clic → Llega a página con formulario
6. Ingresa nueva contraseña → Sistema valida token
7. Verifica expiración → Actualiza contraseña en tabla `users`
8. Elimina token usado → Redirige al login

Código clave:

```
// Validar token y obtener user_id
$stmt = $conn->prepare("SELECT user_id, expires_at FROM password_resets WHERE
token = ?");
$stmt->bind_param("s", $token);
$result = $stmt->get_result();
$reset = $result->fetch_assoc();

// Verificar expiración
$expires = strtotime($reset['expires_at']);
```

```
if ($expires < time()) {
    $message = "El enlace ha expirado. Solicita uno nuevo.";
    $message_type = "error";
}

// Actualizar contraseña
$hash_password = password_hash($password, PASSWORD_DEFAULT);
$stmt = $conn->prepare("UPDATE users SET password_hash = ? WHERE user_id = ?");
$stmt->bind_param("si", $hash_password, $reset['user_id']);
$stmt->execute();

// Eliminar token usado
$stmt = $conn->prepare("DELETE FROM password_resets WHERE token = ?");
$stmt->bind_param("s", $token);
$stmt->execute();
```

Seguridad implementada:

- Password hashing con `password_hash()`
- Consultas preparadas para evitar SQL injection
- Validación de token y expiración
- Eliminación automática del token tras uso
- Sanitización con `htmlspecialchars()`
- Longitud mínima de contraseña: 6 caracteres

🧩 PARTE IV: MÓDULOS DE NEGOCIO

Sesión 15: Dashboard - Métricas + Gráficos Chart.js (25 min)

Archivos del módulo:

1. **index.html:** Estructura visual con tarjetas métricas, tabla y gráfico
2. **dashboard.js:** Cerebro dinámico que conecta frontend con backend
3. **dashboard.php:** Motor en servidor que procesa consultas y devuelve datos

Métricas clave:

- Socios activos (membresías vigentes)
- Asistencias de hoy (COUNT con CURDATE)
- Membresías por vencer (BETWEEN con DATE_ADD +3 días)
- Pagos últimos 7 días (WHERE paid_at >= DATE_SUB -7 días)

Gráfico de asistencia:

```
// Cargar gráfico de barras
function loadAttendanceChart() {
    $.ajax({
        url: '../PHP/dashboard.php?action=get_attendance_trend',
        method: 'GET',
```

```
        success: function(data) {
            const ctx =
document.getElementById('attendanceChart').getContext('2d');
            new Chart(ctx, {
                type: 'bar',
                data: {
                    labels: data.labels,  // Fechas últimos 7 días
                    datasets: [{
                        label: 'Asistencias',
                        data: data.data,      // Números de asistencias
                        backgroundColor: 'rgba(54, 162, 235, 0.7)'
                    }]
                }
            });
        }
    });
}
```

Popular datos (seed.sql):

- Crea registros realistas en todas las tablas
- Incluye fotos de socios en carpeta image_members
- Permite visualizar sistema completo desde el primer momento
- Evita pantallas vacías en demostraciones

Sesión 16: CRUD de Socios con Subida de Fotos (42 min)

Archivos del módulo:

1. **index.html:** Interfaz con tabla DataTables y formulario modal
2. **members.js:** Cerebro que gestiona CRUD con Ajax
3. **members.php:** Backend seguro con validaciones y subida de fotos

Funcionalidades:

- ✅ Listar socios con DataTables (búsqueda, paginación, responsive)
- ✅ Crear nuevo socio con foto (FormData + Ajax)
- ✅ Editar socio existente (cargar datos al modal)
- ✅ Eliminar socio con SweetAlert de confirmación
- ✅ Vista previa de foto antes de subir
- ✅ Validación en frontend y backend
- ✅ Sentencias preparadas MySQLi
- ✅ Sanitización con htmlspecialchars
- ✅ Eliminación de fotos al borrar registro

Subida segura de fotos:

```
function upload_photo($file) {
    // Validar tipo MIME
```

```
    $allowed_types = ['image/jpeg', 'image/png'];
    if (!in_array($file['type'], $allowed_types)) {
        return null;
    }

    // Validar tamaño (máximo 2MB)
    if ($file['size'] > 2097152) {
        return null;
    }

    // Crear nombre único
    $extension = pathinfo($file['name'], PATHINFO_EXTENSION);
    $filename = uniqid() . '.' . $extension;

    // Mover archivo al servidor
    $upload_dir = '../images/image_members/';
    if (!is_dir($upload_dir)) {
        mkdir($upload_dir, 0755, true);
    }

    $upload_path = $upload_dir . $filename;
    if (move_uploaded_file($file['tmp_name'], $upload_path)) {
        return $filename;
    }

    return null;
}
```

Sesión 17: Control de Asistencia (Registro Diario Único) (22 min)

Archivos del módulo:

1. **index.html:** Formulario para registrar asistencia + tabla historial
2. **attendance.js:** Carga socios activos y registra asistencia
3. **attendance.php:** Backend con validaciones y restricción UNIQUE

Características clave:

- ✅ Selector de socios con membresía activa
- ✅ Registro diario único gracias a restricción UNIQUE (member_id + date)
- ✅ Validación de membresía activa antes de registrar
- ✅ Fechas automáticas con CURDATE() y CURRENT_TIMESTAMP
- ✅ Tabla historial con DataTables (búsqueda + paginación)
- ✅ Mensajes de error amigables (socio ya registrado hoy)

Validación en PHP:

```
function register_attendance($member_id) {
    global $conn;
```

```
    // Verificar membresía activa
    $stmt = $conn->prepare("
        SELECT COUNT(*) as count
        FROM members_memberships
        WHERE member_id = ? AND status = 'active' AND end_date >= CURDATE()
    ");
    $stmt->bind_param("i", $member_id);
    $stmt->execute();
    $result = $stmt->get_result();
    $row = $result->fetch_assoc();

    if ($row['count'] == 0) {
        return ['status' => 'error', 'message' => 'El socio no tiene una membresía
activa'];
    }

    // Verificar que no se haya registrado ya hoy
    $stmt = $conn->prepare("
        SELECT COUNT(*) as count
        FROM attendance
        WHERE member_id = ? AND date = CURDATE()
    ");
    $stmt->bind_param("i", $member_id);
    $stmt->execute();
    $result = $stmt->get_result();
    $row = $result->fetch_assoc();

    if ($row['count'] > 0) {
        return ['status' => 'error', 'message' => 'El socio ya registró su
asistencia el día de hoy'];
    }

    // Registrar asistencia
    $stmt = $conn->prepare("INSERT INTO attendance (member_id, date, time) VALUES
(?, CURDATE(), CURRENT_TIMESTAMP)");
    $stmt->bind_param("i", $member_id);
    $stmt->execute();

    return ['status' => 'success', 'message' => 'Asistencia registrada
correctamente'];
}
```

Sesión 18: Membresías + Inscripciones (Relaciones 1 a Muchos) (43 min)

Dos módulos separados y didácticos:

Módulo 1: Memberships (CRUD de planes)

- Planes: mensual, trimestral, anual, VIP
- Campos: nombre, duración (días), precio, beneficios
- Validación: nombre obligatorio, duración ≥ 1, precio ≥ 0

- DataTables con búsqueda y paginación

Módulo 2: Enrollments (Asignar socios a planes)

- Vincula socio + plan + fechas inicio/vencimiento
- Estado: active, expired, cancelled
- Validación: sin duplicados activos (un socio un plan activo)
- Fechas automáticas calculadas por servidor

Relación uno a muchos:

- Un plan puede tener muchos socios
- Un socio puede tener múltiples inscripciones (histórico)
- Solo una inscripción activa por socio

Cálculo automático de fechas:

```
// Calcular fecha de vencimiento
$start_date = date('Y-m-d');
$duration_days = $membership['duration']; // Desde tabla memberships
$end_date = date('Y-m-d', strtotime($start_date . " +{$duration_days} days"));
```

Validación de duplicados:

```
// Verificar que socio no tenga membresía activa
$stmt = $conn->prepare("
    SELECT COUNT(*) as count
    FROM members_memberships
    WHERE member_id = ? AND status = 'active'
");
$stmt->bind_param("i", $member_id);
$stmt->execute();
$result = $stmt->get_result();
$row = $result->fetch_assoc();

if ($row['count'] > 0) {
    return ['status' => 'error', 'message' => 'El socio ya tiene una membresía
activa'];
}
```

Sesión 19: Módulo de Pagos + Filtros + Exportación CSV (22 min)

Archivos del módulo:

1. **index.html:** Formulario para registrar pago + filtros + tabla historial
2. **payments.js:** Gestiona registro, filtros y exportación CSV
3. **payments.php:** Backend con validaciones y reportes

Funcionalidades:

- ✅ Registrar pago vinculado a inscripción específica
- ✅ Selector de socios con membresías activas
- ✅ Selector de membresías activas al elegir socio
- ✅ Monto, método (efectivo/tarjeta/transferencia), notas
- ✅ Filtros dinámicos: rango de fechas, socio, método de pago
- ✅ Exportación a CSV con datos filtrados
- ✅ Tabla historial con DataTables

Exportación CSV en JavaScript:

```
function exportToCSV(data, filename) {
    // Convertir datos a CSV
    const csv = data.map(row =>
        Object.values(row).map(value =>
            `"${value.toString().replace(/"/g, '""')}"`
        ).join(',')
    ).join('\n');

    // Crear enlace de descarga
    const blob = new Blob([csv], { type: 'text/csv' });
    const url = window.URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = filename;
    a.click();
    window.URL.revokeObjectURL(url);
}
```

Reporte con filtros dinámicos:

```
function get_payments_report($filters) {
    global $conn;

    $sql = "SELECT p.*, CONCAT(m.first_name, ' ', m.last_name) as member_name,
            me.name as membership_name
            FROM payments p
            INNER JOIN members_memberships mm ON p.mm_id = mm.id
            INNER JOIN members m ON mm.member_id = m.member_id
            INNER JOIN memberships me ON mm.membership_id = me.membership_id
            WHERE 1=1";

    $params = [];
    $types = '';

    // Filtro por fecha desde
    if (!empty($filters['from'])) {
        $sql .= " AND p.paid_at >= ?";
        $params[] = $filters['from'];
```

```
            $types .= 's';
        }

        // Filtro por fecha hasta
        if (!empty($filters['to'])) {
            $sql .= " AND p.paid_at <= ?";
            $params[] = $filters['to'];
            $types .= 's';
        }

        // Filtro por socio
        if (!empty($filters['member_id'])) {
            $sql .= " AND mm.member_id = ?";
            $params[] = $filters['member_id'];
            $types .= 'i';
        }

        // Filtro por método
        if (!empty($filters['method'])) {
            $sql .= " AND p.method = ?";
            $params[] = $filters['method'];
            $types .= 's';
        }

        $stmt = $conn->prepare($sql);
        if (!empty($params)) {
            $stmt->bind_param($types, ...$params);
        }
        $stmt->execute();
        $result = $stmt->get_result();

        return $result->fetch_all(MYSQLI_ASSOC);
    }
```

Sesión 20: CRUD de Entrenadores (Módulo Final) (31 min)

Archivos del módulo:

1. **index.html:** Tabla de entrenadores + formulario modal
2. **trainers.js:** Cerebro dinámico con Ajax
3. **trainers.php:** Backend seguro con CRUD completo

Funcionalidades:

- ✅ CRUD completo: crear, listar, editar, eliminar
- ✅ Subida segura de fotos con vista previa
- ✅ Campos: nombre, apellido, email, teléfono, especialidad, foto
- ✅ DataTables con búsqueda, paginación y responsive
- ✅ Validación en frontend y backend
- ✅ Sentencias preparadas MySQLi

- ✅ Sanitización con htmlspecialchars
- ✅ Eliminación de fotos al borrar registro

Arquitectura consistente:

- Patrón CRUD idéntico al módulo de socios (members)
- Reutilización de código: copiar y adaptar
- Misma estructura de carpetas y archivos
- Consistencia en toda la aplicación

Cierre del sistema:

- Socios asisten y pagan
- Entrenadores los guían y motivan
- Sistema completo y funcional listo para uso real

💠 PARTE V: DESPLIEGUE Y PROFESIONALISMO

Sesión 21: Pruebas de API con Postman (Testing Profesional) (26 min)

Colección de Postman:

Módulo Auth:

- POST /auth.php?action=login → Iniciar sesión
- POST /auth.php?action=logout → Cerrar sesión
- GET /auth.php?action=check_session → Verificar sesión

Módulo Members:

- GET /PHP/members.php?action=get_all → Listar socios
- POST /PHP/members.php?action=create → Crear socio
- POST /PHP/members.php?action=update → Actualizar socio
- POST /PHP/members.php?action=delete → Eliminar socio

Módulo Enrollments:

- POST /PHP/enrollments.php?action=assign → Asignar membresía
- GET /PHP/enrollments.php?action=get_all → Listar inscripciones

Módulo Attendance:

- POST /PHP/attendance.php?action=register → Registrar asistencia
- GET /PHP/attendance.php?action=get_all → Listar asistencias

Módulo Payments:

- POST /PHP/payments.php?action=register → Registrar pago
- GET /PHP/payments.php?action=get_report → Reporte con filtros

Módulo Dashboard:

- GET /PHP/dashboard.php?action=get_metrics → Métricas clave
- GET /PHP/dashboard.php?action=get_recent_payments → Últimos pagos
- GET /PHP/dashboard.php?action=get_attendance_trend → Tendencia asistencia

Beneficios de usar Postman:

- ✅ Probar endpoints sin frontend
- ✅ Aislar errores (backend vs frontend)
- ✅ Verificar respuestas JSON y códigos de estado
- ✅ Simular comunicación apps móviles/web
- ✅ Reforzar buenas prácticas de API
- ✅ Acelerar desarrollo (pruebas en segundos)
- ✅ Estándar en la industria

Sesión 22: Despliegue con FileZilla (FTP/SFTP + Permisos) (7 min)

Pasos para subir sistema a producción:

1. **Conectar a servidor FTP:**
    - Abrir FileZilla → Site Manager
    - Host: dirección IP del servidor
    - Usuario: user-entrenador
    - Contraseña: [tu_contraseña]
    - Puerto: 21 (FTP) o 22 (SFTP)
2. **Sincronizar carpetas:**
    - Lado izquierdo: carpeta local Gym System
    - Lado derecho: carpeta remota public_html
    - Botones de sincronización para navegar simultáneamente
3. **Transferir archivos:**
    - Seleccionar todos los archivos
    - Arrastrar a lado derecho
    - Confirmar sobrescritura
    - Esperar transferencia completa
4. **Configurar permisos de imágenes:**
    - Carpeta images → Botón derecho → Permisos de archivo
    - Propietario: leer, escribir, ejecutar (7)
    - Grupo: leer, escribir (6)
    - Público: leer, ejecutar (5)
    - Total: 765 o 775
    - Aplicar recursivamente a subdirectorios
5. **Proteger archivo database.php:**

- Nunca compartir ni subir a repositorios públicos
- Contiene credenciales sensibles
- Usar .gitignore para excluirlo de GitHub

Sesión 23: Control de Versiones con GitHub y Git (9 min)

Pasos para subir proyecto a GitHub:

1. **Crear cuenta en GitHub:**
   - github.com → Sign up
   - Verificación en dos pasos recomendada
2. **Crear nuevo repositorio:**
   - Nombre: curso-fullstack-gym-system
   - Descripción: Sistema de gestión para gimnasios
   - Visibilidad: pública o privada
   - No inicializar con README
3. **Inicializar repositorio local:**

```
# Abrir terminal en carpeta del proyecto
cd C:\xampp\htdocs\GymSystem

# Inicializar repositorio Git
git init

# Configurar identidad
git config --global user.email "tu@email.com"
git config --global user.name "Tu Nombre"

# Agregar todos los archivos
git add .

# Primer commit
git commit -m "first commit"

# Crear rama principal
git branch -M main

# Conectar con repositorio remoto
git remote add origin https://github.com/gustabin/curso-fullstack-gym-system.git

# Subir cambios
git push -u origin main
```

4. **Autenticación con GitHub:**
   - Visual Studio Code detecta repositorio

- Abrir navegador para autenticación automática
- Conexión establecida

5. **Archivos .gitignore:**

```
# Archivos sensibles
.env
config/database.php

# Archivos del sistema
.DS_Store
Thumbs.db

# Logs
*.log
```

Beneficios de usar GitHub:

- ✅ Guardar código fuente de forma segura
- ✅ Historial completo de cambios
- ✅ Colaborar con otros desarrolladores
- ✅ Respaldo automático
- ✅ Control de versiones profesional

🏁 CONCLUSIÓN Y PRÓXIMOS PASOS

Sistema completo construido:

✅ **Autenticación segura:** Login, logout, recuperación con tokens
✅ **Dashboard:** Métricas en tiempo real + gráficos Chart.js
✅ **CRUD socios:** Gestión completa con subida de fotos
✅ **Control asistencia:** Registro diario único + validación
✅ **Membresías:** Planes con duración, precio, beneficios
✅ **Inscripciones:** Vincular socio + plan + fechas automáticas
✅ **Pagos:** Registro con filtros + exportación CSV
✅ **Entrenadores:** CRUD completo con especialidades
✅ **Pruebas API:** Colección Postman para testing profesional
✅ **Despliegue:** Subido a dominio real con Hostinger
✅ **Control versiones:** Código en GitHub con Git

Próximas mejoras posibles:

- 📅 **Sistema de reservas:** Clases grupales, sesiones personalizadas
- 💳 **Integración pasarelas:** Stripe, PayPal, MercadoPago
- 📧 **Notificaciones automáticas:** Recordatorios vencimiento, newsletters
- 📊 **Reportes avanzados:** Análisis fidelidad, tendencias, métricas negocio
- 📱 **Aplicación móvil:** React Native o Flutter

- 🔔 **Notificaciones tiempo real:** WebSockets para alertas instantáneas
- 📸 **Galería fotos:** Subida masiva, álbumes, likes, comentarios
- 🎯 **Objetivos progreso:** Seguimiento metas personales, gráficos evolución

Recursos adicionales recomendados:

Libros de Gustavo Arias:

1. **Azure Synapse Analytics desde cero** - Analítica empresarial, data warehouse y pipelines en la nube
2. **C# desde cero** - Desarrollo orientado a objetos, .NET y buenas prácticas
3. **Data Architect en Acción** - Diseño de modelos de datos, gobernanza y arquitectura escalable
4. **Node.js desde cero** - Construye servidores, APIs y aplicaciones backend con JavaScript
5. **PHP desde cero hasta experto** - Desarrollo web moderno con Laravel, MySQL y APIs REST
6. **Programación Full-Stack** - Del frontend al backend: HTML, CSS, JS, PHP y MySQL integrados
7. **Python desde cero** - Automatización, scripting y aplicaciones prácticas
8. **React y otras hierbas** - Interfaces dinámicas, componentes y manejo de estado
9. **Metodologías Ágiles con Scrum** - Planificación, sprints y gestión eficiente de proyectos IT
10. **SharePoint: La plataforma de colaboración empresarial** - Portales, flujos de trabajo y gestión documental
11. **SQL y otras hierbas** - Consultas avanzadas, diseño de tablas y optimización de rendimiento
12. **Visual Paradigm explicado** - UML, diagramas de sistema y modelado para arquitectos
13. **La arquitectura de sistemas y otras hierbas** - Patrones, microservicios y toma de decisiones técnicas
14. **Java y otras hierbas** - Programación orientada a objetos, Spring y ecosistema enterprise
15. **IA en Ciberseguridad** - Detección de amenazas, análisis predictivo y automatización de defensas

Descuento exclusivo: Escribeme para recibir el código de un 20% de descuento

---

📖 APÉNDICES

Checklist de despliegue:

- ☐ Base de datos creada en servidor remoto
- ☐ Esquema SQL importado con datos de prueba
- ☐ Archivo .env configurado con credenciales producción
- ☐ Archivo database.php actualizado
- ☐ Todos los archivos subidos con FileZilla
- ☐ Permisos de carpeta images configurados (775)
- ☐ Certificado SSL activado
- ☐ Dominio apuntando correctamente
- ☐ Prueba de login funcionando
- ☐ Prueba de registro de pago
- ☐ Verificación de envío de email
- ☐ Archivos sensibles excluidos de GitHub (.gitignore)

Comandos Git útiles:

```
# Ver estado del repositorio
git status

# Agregar cambios específicos
git add archivo.php

# Commit con mensaje descriptivo
git commit -m "Agregar módulo de pagos con filtros"

# Subir cambios a GitHub
git push origin main

# Actualizar desde repositorio remoto
git pull origin main

# Ver historial de commits
git log --oneline

# Crear nueva rama
git branch nueva-funcionalidad

# Cambiar de rama
git checkout nueva-funcionalidad

# Fusionar ramas
git merge nueva-funcionalidad
```

Recursos en línea:

- **GitHub:** github.com/gustabin/curso-fullstack-gym-system
- **Documentación PHP:** php.net/manual/es
- **MySQL Reference:** dev.mysql.com/doc
- **Bootstrap Docs:** getbootstrap.com/docs
- **jQuery API:** api.jquery.com
- **Chart.js:** chartjs.org/docs
- **DataTables:** datatables.net/manual
- **SweetAlert2:** sweetalert2.github.io
- **PHPMailer:** github.com/PHPMailer/PHPMailer

📧 SOPORTE Y COMUNIDAD

¿Tienes preguntas o necesitas ayuda?

- 📧 **Email:** tabindev@gmail.com
- 🌐 **Sitio web:** stackcodelab.com

🏆 ¡FELICITACIONES!

Has completado el curso completo de **Programación Full Stack con JavaScript, HTML, PHP y MySQL**.

Ahora tienes:

- ✅ Un sistema profesional listo para tu portafolio
- ✅ Conocimientos sólidos en desarrollo web full stack
- ✅ Experiencia con buenas prácticas de seguridad
- ✅ Habilidad para desplegar aplicaciones en producción
- ✅ Base para seguir aprendiendo y creciendo

Recuerda: El verdadero valor no está en el sistema terminado, sino en el **proceso de construcción** y las **decisiones técnicas** que tomaste en cada paso. Eso es lo que te convertirá en un desarrollador Full Stack profesional.

Ing de Sistemas • Instructor Full Stack
"Enseñando a construir sistemas reales, no solo código"