Guía Completa: Crear un Nuevo Endpoint en MVC

Autor: Guía de Estudio para Examen
Última actualización: 2 de Febrero de 2026

Objetivo

Aprender a crear un **endpoint completo** en una arquitectura MVC (Modelo-Vista-Controlador), incluyendo:

- ✅ Base de datos (si es necesaria)
- ✅ Modelo (acceso a datos)
- ✅ Controlador (lógica de negocio)
- ✅ Vista (interfaz HTML)
- ✅ Pruebas (validación)

Tabla de Contenidos

Paso 0: Análisis del Requisito

Ejemplo de enunciado de examen:

> "Crear un sistema de reseñas de productos donde los usuarios puedan:
>
> - Ver todas las reseñas
> - Ver detalle de una reseña
> - Crear una nueva reseña
> - Mostrar las reseñas en una vista web"

¿Qué necesito identificar?

1. **Entidad principal:** Reseñas
2. **Operaciones CRUD:**
   - ✅ Crear (POST)
   - ✅ Leer/Listar (GET)
   - ✅ Leer/Ver una (GET con ID)
   - ❌ Actualizar (no pedido)

- ❌ Eliminar (no pedido)

3. **Relaciones:** Reseña pertenece a un Producto
4. **Campos necesarios:**
   - ID único
   - Producto relacionado
   - Usuario que comenta
   - Comentario
   - Puntuación
   - Fecha de creación

Paso 1: Base de Datos

Archivo: `bd/bd.sql`

Acción: MODIFICAR (agregar al final)

```
-- ============================================================
-- Tabla: resenas
-- Descripción: Reseñas de productos por usuarios
-- ============================================================
CREATE TABLE resenas (
    CodRes VARCHAR(36) PRIMARY KEY COMMENT 'UUID de la reseña',
    Producto VARCHAR(36) NOT NULL COMMENT 'FK a productos',
    Usuario VARCHAR(100) NOT NULL COMMENT 'Nombre del usuario',
    Comentario TEXT COMMENT 'Texto de la reseña',
    Puntuacion INT NOT NULL COMMENT 'De 1 a 5 estrellas',
    FechaCreacion DATETIME DEFAULT CURRENT_TIMESTAMP,

    -- Relaciones
    FOREIGN KEY (Producto) REFERENCES productos(CodProd) ON DELETE CASCADE,

    -- Índices
    INDEX idx_producto (Producto),
    INDEX idx_fecha (FechaCreacion)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ============================================================
-- DATOS DE PRUEBA - Reseñas
-- ============================================================
INSERT INTO resenas (CodRes, Producto, Usuario, Comentario, Puntuacion) VALUES
('res-001', 'prod-001', 'Juan Pérez', 'Excelente producto, muy recomendado', 5),
('res-002', 'prod-001', 'María García', 'Buena calidad pero tardó en llegar', 4),
('res-003', 'prod-002', 'Carlos López', 'No cumplió mis expectativas', 2),
('res-004', 'prod-003', 'Ana Martínez', 'Perfecto, justo lo que buscaba', 5);

-- Verificar
SELECT * FROM resenas;
```

⚠ Consideraciones

- **UUID vs Autoincremental:** Usa UUID si es una API pública
- **VARCHAR(36)** para UUIDs: `550e8400-e29b-41d4-a716-446655440000`
- **ON DELETE CASCADE:** Si borras un producto, se borran sus reseñas
- **Siempre agrega datos de prueba** para validar

Paso 2: Crear el Modelo

Archivo: `app/modelos/GestorResenas.php`

Acción: CREAR nuevo archivo

```php
<?php

namespace Mrs\ApiServer\modelos;

use Mrs\ApiServer\librerias\Db;

/**
 * GestorResenas - Gestiona operaciones CRUD de reseñas
 */
class GestorResenas
{
    /**
     * Obtiene todas las reseñas con información del producto
     *
     * @return array Lista de reseñas con JOIN a productos
     */
    public static function getResenas()
    {
        $pdo = Db::getConexion();

        $sql = 'SELECT r.CodRes, r.Usuario, r.Comentario, r.Puntuacion,
                       r.FechaCreacion, r.Producto,
                       p.Nombre as ProductoNombre,
                       p.Precio as ProductoPrecio
                FROM resenas r
                LEFT JOIN productos p ON r.Producto = p.CodProd
                ORDER BY r.FechaCreacion DESC';

        $stmt = $pdo->query($sql);
        return $stmt->fetchAll();
    }

    /**
     * Obtiene las reseñas de un producto específico
     *
     * @param string $codProd Código del producto
     * @return array Lista de reseñas del producto
     */
    public static function getResenasPorProducto($codProd)
    {
```

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

        $sql = 'SELECT r.CodRes, r.Usuario, r.Comentario, r.Puntuacion,
                       r.FechaCreacion
                FROM resenas r
                WHERE r.Producto = :prod
                ORDER BY r.FechaCreacion DESC';

        $stmt = $pdo->prepare($sql);
        $stmt->execute(['prod' => $codProd]);

        return $stmt->fetchAll();
    }

    /**
     * Obtiene una reseña por su ID
     *
     * @param string $codRes Código de la reseña
     * @return array|null Datos de la reseña o null si no existe
     */
    public static function getResena($codRes)
    {
        $pdo = Db::getConexion();

        $sql = 'SELECT r.*, p.Nombre as ProductoNombre
                FROM resenas r
                LEFT JOIN productos p ON r.Producto = p.CodProd
                WHERE r.CodRes = :id
                LIMIT 1';

        $stmt = $pdo->prepare($sql);
        $stmt->execute(['id' => $codRes]);

        $row = $stmt->fetch();
        return $row ?: null;
    }

    /**
     * Crea una nueva reseña
     *
     * @param string $codRes UUID de la reseña
     * @param string $producto UUID del producto
     * @param string $usuario Nombre del usuario
     * @param string $comentario Texto de la reseña
     * @param int $puntuacion Puntuación de 1 a 5
     * @return bool True si se creó correctamente
     */
    public static function crearResena($codRes, $producto, $usuario, $comentario,
$puntuacion)
    {
        $pdo = Db::getConexion();

        $sql = 'INSERT INTO resenas (CodRes, Producto, Usuario, Comentario,
Puntuacion)
```

```
                VALUES (:cod, :prod, :user, :com, :punt)';

        $stmt = $pdo->prepare($sql);

        return $stmt->execute([
            'cod' => $codRes,
            'prod' => $producto,
            'user' => $usuario,
            'com' => $comentario,
            'punt' => $puntuacion
        ]);
    }

    /**
     * Actualiza una reseña existente
     *
     * @param string $codRes UUID de la reseña
     * @param string $comentario Nuevo comentario
     * @param int $puntuacion Nueva puntuación
     * @return bool True si se actualizó correctamente
     */
    public static function actualizarResena($codRes, $comentario, $puntuacion)
    {
        $pdo = Db::getConexion();

        $sql = 'UPDATE resenas
                SET Comentario = :com, Puntuacion = :punt
                WHERE CodRes = :cod';

        $stmt = $pdo->prepare($sql);

        return $stmt->execute([
            'com' => $comentario,
            'punt' => $puntuacion,
            'cod' => $codRes
        ]);
    }

    /**
     * Elimina una reseña
     *
     * @param string $codRes UUID de la reseña
     * @return bool True si se eliminó correctamente
     */
    public static function eliminarResena($codRes)
    {
        $pdo = Db::getConexion();

        $sql = 'DELETE FROM resenas WHERE CodRes = :cod';

        $stmt = $pdo->prepare($sql);
        return $stmt->execute(['cod' => $codRes]);
    }
```

```
    /**
     * Obtiene el promedio de puntuación de un producto
     *
     * @param string $codProd UUID del producto
     * @return float|null Promedio o null si no hay reseñas
     */
    public static function getPromedioProducto($codProd)
    {
        $pdo = Db::getConexion();

        $sql = 'SELECT AVG(Puntuacion) as promedio, COUNT(*) as total
                FROM resenas
                WHERE Producto = :prod';

        $stmt = $pdo->prepare($sql);
        $stmt->execute(['prod' => $codProd]);

        return $stmt->fetch();
    }
}
```

📝 Notas del Modelo

- **Métodos estáticos:** No necesitas instanciar la clase
- **Prepared statements:** Siempre usa `:parametros` para prevenir SQL Injection
- **JOINs:** Incluye datos relacionados para evitar múltiples queries
- **Retorno consistente:** `fetchAll()` para listas, `fetch()` para uno, `null` si no existe
- **PHPDoc:** Documenta parámetros y retornos

🎮 Paso 3: Crear el Controlador (API)

🗂 Archivo: `app/controladores/ControladorResenas.php`

Acción: CREAR nuevo archivo

```
<?php

namespace Mrs\ApiServer\controladores;

use Mrs\ApiServer\librerias\Controlador;
use Mrs\ApiServer\modelos\GestorResenas;
use Ramsey\Uuid\Uuid;

/**
 * ControladorResenas - API REST para gestión de reseñas
 */
class ControladorResenas extends Controlador
{
    /**
     * GET /controladorresenas/listar
```

```
     * Lista todas las reseñas
     */
    public function listar(): void
    {
        // Autenticación
        $this->requireBasicAuth();

        try {
            $resenas = GestorResenas::getResenas();

            $this->jsonResponse([
                'success' => true,
                'total' => count($resenas),
                'resenas' => $resenas
            ], 200);

        } catch (\Exception $e) {
            $this->jsonResponse([
                'error' => 'Error al obtener reseñas',
                'detail' => $e->getMessage()
            ], 500);
        }
    }

    /**
     * GET /controladorresenas/producto/{codProd}
     * Lista reseñas de un producto específico
     */
    public function producto(string $codProd = ''): void
    {
        $this->requireBasicAuth();

        if (empty($codProd)) {
            $this->jsonResponse(['error' => 'Código de producto requerido'], 400);
        }

        try {
            $resenas = GestorResenas::getResenasPorProducto($codProd);
            $promedio = GestorResenas::getPromedioProducto($codProd);

            $this->jsonResponse([
                'success' => true,
                'producto' => $codProd,
                'total' => count($resenas),
                'promedio' => $promedio['promedio'],
                'resenas' => $resenas
            ], 200);

        } catch (\Exception $e) {
            $this->jsonResponse([
                'error' => 'Error al obtener reseñas del producto',
                'detail' => $e->getMessage()
            ], 500);
        }
```

```
    }

    /**
     * GET /controladorresenas/ver/{id}
     * Obtiene una reseña específica
     */
    public function ver(string $id = ''): void
    {
        $this->requireBasicAuth();

        if (empty($id)) {
            $this->jsonResponse(['error' => 'ID requerido'], 400);
        }

        try {
            $resena = GestorResenas::getResena($id);

            if (!$resena) {
                $this->jsonResponse(['error' => 'Reseña no encontrada'], 404);
            }

            $this->jsonResponse([
                'success' => true,
                'resena' => $resena
            ], 200);

        } catch (\Exception $e) {
            $this->jsonResponse([
                'error' => 'Error al obtener reseña',
                'detail' => $e->getMessage()
            ], 500);
        }
    }

    /**
     * POST /controladorresenas/crear
     * Crea una nueva reseña
     */
    public function crear(): void
    {
        $this->requireBasicAuth();

        // Solo permitir POST
        if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
            $this->jsonResponse(['error' => 'Método no permitido'], 405);
        }

        // Leer datos del body
        $data = $this->readJsonBody();

        // Validación de campos requeridos
        $camposRequeridos = ['producto', 'usuario', 'comentario', 'puntuacion'];
        foreach ($camposRequeridos as $campo) {
            if (!isset($data[$campo]) || trim($data[$campo]) === '') {
```

```
                $this->jsonResponse([
                    'error' => "Campo '$campo' es requerido"
                ], 400);
            }
        }

        // Validación de puntuación
        $puntuacion = (int)$data['puntuacion'];
        if ($puntuacion < 1 || $puntuacion > 5) {
            $this->jsonResponse([
                'error' => 'La puntuación debe estar entre 1 y 5'
            ], 400);
        }

        try {
            // Generar UUID
            $codRes = Uuid::uuid4()->toString();

            // Crear en BD
            $resultado = GestorResenas::crearResena(
                $codRes,
                $data['producto'],
                $data['usuario'],
                $data['comentario'],
                $puntuacion
            );

            if ($resultado) {
                $this->jsonResponse([
                    'success' => true,
                    'message' => 'Reseña creada correctamente',
                    'id' => $codRes
                ], 201);
            } else {
                $this->jsonResponse(['error' => 'Error al crear reseña'], 500);
            }

        } catch (\Exception $e) {
            $this->jsonResponse([
                'error' => 'Error al crear reseña',
                'detail' => $e->getMessage()
            ], 500);
        }
    }

    /**
     * PUT /controladorresenas/actualizar/{id}
     * Actualiza una reseña existente
     */
    public function actualizar(string $id = ''): void
    {
        $this->requireBasicAuth();

        if ($_SERVER['REQUEST_METHOD'] !== 'PUT') {
```

```
            $this->jsonResponse(['error' => 'Método no permitido'], 405);
        }

        if (empty($id)) {
            $this->jsonResponse(['error' => 'ID requerido'], 400);
        }

        $data = $this->readJsonBody();

        if (!isset($data['comentario'], $data['puntuacion'])) {
            $this->jsonResponse(['error' => 'Datos incompletos'], 400);
        }

        try {
            $resultado = GestorResenas::actualizarResena(
                $id,
                $data['comentario'],
                (int)$data['puntuacion']
            );

            if ($resultado) {
                $this->jsonResponse([
                    'success' => true,
                    'message' => 'Reseña actualizada'
                ], 200);
            } else {
                $this->jsonResponse(['error' => 'Reseña no encontrada'], 404);
            }

        } catch (\Exception $e) {
            $this->jsonResponse([
                'error' => 'Error al actualizar reseña',
                'detail' => $e->getMessage()
            ], 500);
        }
    }

    /**
     * DELETE /controladorresenas/eliminar/{id}
     * Elimina una reseña
     */
    public function eliminar(string $id = ''): void
    {
        $this->requireBasicAuth();

        if ($_SERVER['REQUEST_METHOD'] !== 'DELETE') {
            $this->jsonResponse(['error' => 'Método no permitido'], 405);
        }

        if (empty($id)) {
            $this->jsonResponse(['error' => 'ID requerido'], 400);
        }

        try {
```

```
            $resultado = GestorResenas::eliminarResena($id);

            if ($resultado) {
                $this->jsonResponse([
                    'success' => true,
                    'message' => 'Reseña eliminada'
                ], 200);
            } else {
                $this->jsonResponse(['error' => 'Reseña no encontrada'], 404);
            }

        } catch (\Exception $e) {
            $this->jsonResponse([
                'error' => 'Error al eliminar reseña',
                'detail' => $e->getMessage()
            ], 500);
        }
    }
}
```

📝 Notas del Controlador API

- **Un método público = Un endpoint:** `listar()` → `/controladorresenas/listar`
- **Parámetros de URL:** `ver(string $id)` → `/controladorresenas/ver/res-001`
- **Siempre validar:** Datos de entrada, métodos HTTP, autenticación
- **Códigos HTTP correctos:**
  - `200` - OK
  - `201` - Created
  - `400` - Bad Request (error del cliente)
  - `404` - Not Found
  - `405` - Method Not Allowed
  - `500` - Internal Server Error
- **Try-catch:** Captura errores de BD

🖼️ Paso 4: Crear las Vistas (HTML)

🗂️ Archivo: `app/vistas/resenas/listar.php`

Acción: CREAR nueva carpeta y archivo

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Reseñas - <?php echo NOMBRESITIO; ?></title>
    <style>
        * {
            margin: 0;
```

```
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #f5f5f5;
            padding: 20px;
        }

        .container {
            max-width: 1200px;
            margin: 0 auto;
            background: white;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }

        h1 {
            color: #333;
            margin-bottom: 30px;
            border-bottom: 3px solid #007bff;
            padding-bottom: 10px;
        }

        .btn {
            display: inline-block;
            padding: 10px 20px;
            background-color: #007bff;
            color: white;
            text-decoration: none;
            border-radius: 5px;
            margin-bottom: 20px;
            transition: background-color 0.3s;
        }

        .btn:hover {
            background-color: #0056b3;
        }

        .resena-card {
            border: 1px solid #ddd;
            border-radius: 8px;
            padding: 20px;
            margin-bottom: 20px;
            background-color: #fafafa;
            transition: box-shadow 0.3s;
        }

        .resena-card:hover {
            box-shadow: 0 4px 15px rgba(0,0,0,0.1);
        }
```

```
        .resena-header {
            display: flex;
            justify-content: space-between;
            align-items: center;
            margin-bottom: 10px;
        }

        .usuario {
            font-weight: bold;
            font-size: 1.1em;
            color: #333;
        }

        .fecha {
            color: #666;
            font-size: 0.9em;
        }

        .producto-info {
            color: #007bff;
            font-size: 0.95em;
            margin-bottom: 10px;
        }

        .estrellas {
            color: #ffc107;
            font-size: 1.2em;
            margin-bottom: 10px;
        }

        .comentario {
            color: #555;
            line-height: 1.6;
            margin-top: 10px;
        }

        .no-resenas {
            text-align: center;
            padding: 40px;
            color: #999;
            font-size: 1.1em;
        }

        .error {
            background-color: #f8d7da;
            color: #721c24;
            padding: 15px;
            border-radius: 5px;
            margin-bottom: 20px;
        }
    </style>
</head>
<body>
    <div class="container">
```

```
        <h1>📝 Reseñas de Productos</h1>

        <a href="<?php echo RUTA_URL; ?>/resenas/crear" class="btn">➕ Nueva
Reseña</a>

        <?php if (isset($datos['error'])): ?>
            <div class="error">
                ⚠️ <?php echo htmlspecialchars($datos['error']); ?>
            </div>
        <?php endif; ?>

        <?php if (!empty($datos['resenas'])): ?>
            <?php foreach ($datos['resenas'] as $resena): ?>
                <div class="resena-card">
                    <div class="resena-header">
                        <span class="usuario">
                            👤 <?php echo htmlspecialchars($resena['Usuario']); ?
>
                        </span>
                        <span class="fecha">
                            📆 <?php echo date('d/m/Y H:i',
strtotime($resena['FechaCreacion'])); ?>
                        </span>
                    </div>

                    <div class="producto-info">
                        🎮 Producto: <strong><?php echo
htmlspecialchars($resena['ProductoNombre']); ?></strong>
                    </div>

                    <div class="estrellas">
                        <?php
                        $puntuacion = (int)$resena['Puntuacion'];
                        for ($i = 0; $i < 5; $i++) {
                            echo $i < $puntuacion ? '☆' : '☆';
                        }
                        ?>
                        (<?php echo $puntuacion; ?>/5)
                    </div>

                    <div class="comentario">
                        "<?php echo htmlspecialchars($resena['Comentario']); ?>"
                    </div>
                </div>
            <?php endforeach; ?>
        <?php else: ?>
            <div class="no-resenas">
                📭 No hay reseñas disponibles
            </div>
        <?php endif; ?>
    </div>
</body>
</html>
```

Archivo: app/vistas/resenas/crear.php

Acción: CREAR nuevo archivo

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Nueva Reseña - <?php echo NOMBRESITIO; ?></title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #f5f5f5;
            padding: 20px;
        }

        .container {
            max-width: 800px;
            margin: 0 auto;
            background: white;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }

        h1 {
            color: #333;
            margin-bottom: 30px;
            border-bottom: 3px solid #28a745;
            padding-bottom: 10px;
        }

        .form-group {
            margin-bottom: 20px;
        }

        label {
            display: block;
            font-weight: bold;
            color: #555;
            margin-bottom: 8px;
        }

        input[type="text"],
```

```
        select,
        textarea {
            width: 100%;
            padding: 12px;
            border: 1px solid #ddd;
            border-radius: 5px;
            font-size: 1em;
            font-family: inherit;
        }

        textarea {
            resize: vertical;
            min-height: 120px;
        }

        .rating {
            display: flex;
            gap: 10px;
        }

        .rating input[type="radio"] {
            display: none;
        }

        .rating label {
            font-size: 2em;
            cursor: pointer;
            color: #ddd;
            transition: color 0.2s;
        }

        .rating input[type="radio"]:checked ~ label,
        .rating label:hover,
        .rating label:hover ~ label {
            color: #ffc107;
        }

        .btn {
            padding: 12px 30px;
            background-color: #28a745;
            color: white;
            border: none;
            border-radius: 5px;
            font-size: 1em;
            cursor: pointer;
            transition: background-color 0.3s;
        }

        .btn:hover {
            background-color: #218838;
        }

        .btn-secondary {
            background-color: #6c757d;
```

```
            margin-left: 10px;
        }

        .btn-secondary:hover {
            background-color: #5a6268;
        }

        .error {
            background-color: #f8d7da;
            color: #721c24;
            padding: 15px;
            border-radius: 5px;
            margin-bottom: 20px;
        }

        .success {
            background-color: #d4edda;
            color: #155724;
            padding: 15px;
            border-radius: 5px;
            margin-bottom: 20px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>➕ Nueva Reseña</h1>

        <?php if (isset($datos['error'])): ?>
            <div class="error">
                ⚠️ <?php echo htmlspecialchars($datos['error']); ?>
            </div>
        <?php endif; ?>

        <?php if (isset($datos['success'])): ?>
            <div class="success">
                ✅ <?php echo htmlspecialchars($datos['success']); ?>
            </div>
        <?php endif; ?>

        <form method="POST" action="<?php echo RUTA_URL; ?>/resenas/guardar">
            <div class="form-group">
                <label for="producto">Producto *</label>
                <select id="producto" name="producto" required>
                    <option value="">-- Selecciona un producto --</option>
                    <?php if (!empty($datos['productos'])): ?>
                        <?php foreach ($datos['productos'] as $prod): ?>
                            <option value="<?php echo
htmlspecialchars($prod['CodProd']); ?>">
                                <?php echo htmlspecialchars($prod['Nombre']); ?>
                                (€<?php echo number_format($prod['Precio'], 2); ?
>)
                            </option>
                        <?php endforeach; ?>
```

```
                    <?php endif; ?>
                </select>
            </div>

            <div class="form-group">
                <label for="usuario">Tu Nombre *</label>
                <input type="text"
                       id="usuario"
                       name="usuario"
                       placeholder="Ej: Juan Pérez"
                       required
                       maxlength="100">
            </div>

            <div class="form-group">
                <label>Puntuación *</label>
                <div class="rating">
                    <input type="radio" id="star5" name="puntuacion" value="5"
required>
                    <label for="star5">☆</label>

                    <input type="radio" id="star4" name="puntuacion" value="4">
                    <label for="star4">☆</label>

                    <input type="radio" id="star3" name="puntuacion" value="3">
                    <label for="star3">☆</label>

                    <input type="radio" id="star2" name="puntuacion" value="2">
                    <label for="star2">☆</label>

                    <input type="radio" id="star1" name="puntuacion" value="1">
                    <label for="star1">☆</label>
                </div>
            </div>

            <div class="form-group">
                <label for="comentario">Comentario *</label>
                <textarea id="comentario"
                          name="comentario"
                          placeholder="Cuéntanos tu experiencia con este
producto..."
                          required
                          maxlength="1000"></textarea>
            </div>

            <div class="form-group">
                <button type="submit" class="btn">💾 Guardar Reseña</button>
                <a href="<?php echo RUTA_URL; ?>/resenas/listar" class="btn btn-
secondary">↩ Cancelar</a>
            </div>
        </form>
    </div>

    <script>
```

```
        // Invertir orden de estrellas para que funcione de izquierda a derecha
        const ratingContainer = document.querySelector('.rating');
        const stars = Array.from(ratingContainer.children).reverse();
        stars.forEach(star => ratingContainer.appendChild(star));
    </script>
</body>
</html>
```

📁 Archivo: `app/vistas/resenas/ver.php`

Acción: CREAR nuevo archivo

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Detalle Reseña - <?php echo NOMBRESITIO; ?></title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #f5f5f5;
            padding: 20px;
        }

        .container {
            max-width: 900px;
            margin: 0 auto;
            background: white;
            padding: 40px;
            border-radius: 10px;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }

        h1 {
            color: #333;
            margin-bottom: 30px;
        }

        .resena-detail {
            border: 2px solid #007bff;
            border-radius: 10px;
            padding: 30px;
            background-color: #f9f9f9;
```

```
        }

        .info-row {
            margin-bottom: 20px;
            padding-bottom: 15px;
            border-bottom: 1px solid #ddd;
        }

        .info-row:last-child {
            border-bottom: none;
        }

        .label {
            font-weight: bold;
            color: #555;
            display: block;
            margin-bottom: 5px;
        }

        .value {
            font-size: 1.1em;
            color: #333;
        }

        .estrellas {
            color: #ffc107;
            font-size: 1.5em;
        }

        .comentario {
            background-color: white;
            padding: 20px;
            border-radius: 5px;
            line-height: 1.8;
            font-size: 1.05em;
        }

        .btn {
            display: inline-block;
            padding: 10px 20px;
            background-color: #007bff;
            color: white;
            text-decoration: none;
            border-radius: 5px;
            margin-top: 20px;
            transition: background-color 0.3s;
        }

        .btn:hover {
            background-color: #0056b3;
        }

        .error {
            background-color: #f8d7da;
```

```
            color: #721c24;
            padding: 15px;
            border-radius: 5px;
            margin-bottom: 20px;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>📄 Detalle de Reseña</h1>

        <?php if (isset($datos['error'])): ?>
            <div class="error">
                ⚠️ <?php echo htmlspecialchars($datos['error']); ?>
            </div>
        <?php elseif (isset($datos['resena'])): ?>
            <?php $resena = $datos['resena']; ?>
            <div class="resena-detail">
                <div class="info-row">
                    <span class="label">👤 Usuario:</span>
                    <span class="value"><?php echo
htmlspecialchars($resena['Usuario']); ?></span>
                </div>

                <div class="info-row">
                    <span class="label">🍕 Producto:</span>
                    <span class="value"><?php echo
htmlspecialchars($resena['ProductoNombre']); ?></span>
                </div>

                <div class="info-row">
                    <span class="label">☆ Puntuación:</span>
                    <div class="estrellas">
                        <?php
                        $puntuacion = (int)$resena['Puntuacion'];
                        for ($i = 0; $i < 5; $i++) {
                            echo $i < $puntuacion ? '☆' : '☆';
                        }
                        ?>
                        (<?php echo $puntuacion; ?>/5)
                    </div>
                </div>

                <div class="info-row">
                    <span class="label">📅 Fecha:</span>
                    <span class="value">
                        <?php echo date('d/m/Y H:i:s',
strtotime($resena['FechaCreacion'])); ?>
                    </span>
                </div>

                <div class="info-row">
                    <span class="label">💬 Comentario:</span>
                    <div class="comentario">
```

```
                        <?php echo nl2br(htmlspecialchars($resena['Comentario'])); ?>
                    </div>
                </div>
            </div>
        <?php endif; ?>

        <a href="<?php echo RUTA_URL; ?>/resenas/listar" class="btn">↩️ Volver al Listado</a>
    </div>
</body>
</html>
```

📝 Notas de las Vistas

- **Siempre usar** `htmlspecialchars()` para prevenir XSS
- **CSS inline** para simplicidad (en producción usa archivos .css externos)
- **Emojis** para interfaz amigable
- **Responsive** con `max-width` y `viewport`
- **Accesibilidad:** labels, placeholders, required

🌐 Paso 5: Crear el Controlador Web

📂 Archivo: `app/controladores/Resenas.php` (sin "Controlador" en el nombre)

Acción: CREAR nuevo archivo

```php
<?php

namespace Mrs\WebCliente\controladores;

use Mrs\WebCliente\librerias\Controlador;
use Mrs\WebCliente\librerias\ClienteAPI;

/**
 * Resenas - Controlador web para vistas HTML de reseñas
 * Consume la API REST y renderiza vistas
 */
class Resenas extends Controlador
{
    private $api;

    public function __construct()
    {
        $this->api = new ClienteAPI();
    }

    /**
     * GET /resenas/listar
     * Muestra listado de todas las reseñas
```

```
     */
    public function listar()
    {
        try {
            // Consumir API
            $respuesta = $this->api->get('/controladorresenas/listar');

            if ($respuesta['success']) {
                $datos = [
                    'resenas' => $respuesta['resenas']
                ];
            } else {
                $datos = [
                    'error' => 'No se pudieron obtener las reseñas',
                    'resenas' => []
                ];
            }

        } catch (\Exception $e) {
            $datos = [
                'error' => 'Error de conexión con la API: ' . $e->getMessage(),
                'resenas' => []
            ];
        }

        // Renderizar vista
        $this->vista('resenas/listar', $datos);
    }

    /**
     * GET /resenas/ver/{id}
     * Muestra detalle de una reseña
     */
    public function ver($id = null)
    {
        if (!$id) {
            header('Location: ' . RUTA_URL . '/resenas/listar');
            exit;
        }

        try {
            $respuesta = $this->api->get("/controladorresenas/ver/$id");

            if ($respuesta['success']) {
                $datos = [
                    'resena' => $respuesta['resena']
                ];
            } else {
                $datos = [
                    'error' => 'Reseña no encontrada'
                ];
            }

        } catch (\Exception $e) {
```

```
            $datos = [
                'error' => 'Error de conexión: ' . $e->getMessage()
            ];
        }

        $this->vista('resenas/ver', $datos);
    }

    /**
     * GET /resenas/crear
     * Muestra formulario de nueva reseña
     */
    public function crear()
    {
        try {
            // Obtener lista de productos para el select
            $respuesta = $this->api->get('/controladorproductos/productos');

            $datos = [
                'productos' => $respuesta['success'] ? $respuesta['productos'] :
[]
            ];

        } catch (\Exception $e) {
            $datos = [
                'error' => 'Error al cargar productos',
                'productos' => []
            ];
        }

        $this->vista('resenas/crear', $datos);
    }

    /**
     * POST /resenas/guardar
     * Procesa el formulario y crea la reseña
     */
    public function guardar()
    {
        if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
            header('Location: ' . RUTA_URL . '/resenas/crear');
            exit;
        }

        // Validación
        $errores = [];

        if (empty($_POST['producto'])) {
            $errores[] = 'Debes seleccionar un producto';
        }

        if (empty($_POST['usuario']) || strlen($_POST['usuario']) < 3) {
            $errores[] = 'El nombre debe tener al menos 3 caracteres';
        }
```

```
        if (empty($_POST['puntuacion']) || $_POST['puntuacion'] < 1 ||
$_POST['puntuacion'] > 5) {
            $errores[] = 'La puntuación debe estar entre 1 y 5';
        }

        if (empty($_POST['comentario']) || strlen($_POST['comentario']) < 10) {
            $errores[] = 'El comentario debe tener al menos 10 caracteres';
        }

        if (!empty($errores)) {
            $datos = [
                'error' => implode(', ', $errores),
                'productos' => []
            ];
            $this->vista('resenas/crear', $datos);
            return;
        }

        // Enviar a API
        try {
            $payload = [
                'producto' => $_POST['producto'],
                'usuario' => trim($_POST['usuario']),
                'comentario' => trim($_POST['comentario']),
                'puntuacion' => (int)$_POST['puntuacion']
            ];

            $respuesta = $this->api->post('/controladorresenas/crear', $payload);

            if ($respuesta['success']) {
                // Redirigir al listado con mensaje de éxito
                header('Location: ' . RUTA_URL . '/resenas/listar?success=1');
                exit;
            } else {
                $datos = [
                    'error' => $respuesta['error'] ?? 'Error al crear la reseña',
                    'productos' => []
                ];
                $this->vista('resenas/crear', $datos);
            }

        } catch (\Exception $e) {
            $datos = [
                'error' => 'Error de conexión: ' . $e->getMessage(),
                'productos' => []
            ];
            $this->vista('resenas/crear', $datos);
        }
    }
}
```

📝 Notas del Controlador Web

- **Consume la API REST** mediante `ClienteAPI`
- **No accede directamente a BD** (eso es trabajo de la API)
- **Renderiza vistas HTML** con `$this->vista()`
- **Valida datos del formulario** antes de enviar a API
- **Maneja redirecciones** con `header('Location: ...')`

🧪 Paso 6: Pruebas

🗁 Archivo: `tests_resenas.http`

Acción: CREAR nuevo archivo

```
### ============================================================
### TESTS DE RESEÑAS - API REST
### ============================================================

### Variables
@baseUrl = http://mywww/EjercicioRepaso_Modificado/api-server
@auth = admin admin123

### ============================================================
### TEST 1: Listar todas las reseñas
### Debe retornar 200 OK con array de reseñas
### ============================================================
GET {{baseUrl}}/controladorresenas/listar
Authorization: Basic {{auth}}

### ============================================================
### TEST 2: Ver reseña específica (existe)
### Debe retornar 200 OK con datos de la reseña
### ============================================================
GET {{baseUrl}}/controladorresenas/ver/res-001
Authorization: Basic {{auth}}

### ============================================================
### TEST 3: Ver reseña inexistente
### Debe retornar 404 Not Found
### ============================================================
GET {{baseUrl}}/controladorresenas/ver/res-999
Authorization: Basic {{auth}}

### ============================================================
### TEST 4: Crear nueva reseña (datos correctos)
### Debe retornar 201 Created con ID generado
### ============================================================
POST {{baseUrl}}/controladorresenas/crear
Authorization: Basic {{auth}}
Content-Type: application/json
```

```
{
  "producto": "prod-001",
  "usuario": "María García",
  "comentario": "Excelente producto, superó mis expectativas. Lo recomiendo
totalmente.",
  "puntuacion": 5
}

### ================================================================
### TEST 5: Crear reseña con datos incompletos
### Debe retornar 400 Bad Request
### ================================================================
POST {{baseUrl}}/controladorresenas/crear
Authorization: Basic {{auth}}
Content-Type: application/json

{
  "producto": "prod-001",
  "usuario": "Juan"
}

### ================================================================
### TEST 6: Crear reseña con puntuación inválida
### Debe retornar 400 Bad Request
### ================================================================
POST {{baseUrl}}/controladorresenas/crear
Authorization: Basic {{auth}}
Content-Type: application/json

{
  "producto": "prod-001",
  "usuario": "Pedro López",
  "comentario": "Comentario de prueba",
  "puntuacion": 10
}

### ================================================================
### TEST 7: Reseñas de un producto específico
### Debe retornar 200 OK con reseñas del producto
### ================================================================
GET {{baseUrl}}/controladorresenas/producto/prod-001
Authorization: Basic {{auth}}

### ================================================================
### TEST 8: Actualizar reseña existente
### Debe retornar 200 OK
### ================================================================
PUT {{baseUrl}}/controladorresenas/actualizar/res-001
Authorization: Basic {{auth}}
Content-Type: application/json

{
  "comentario": "He actualizado mi opinión: sigue siendo excelente",
  "puntuacion": 5
```

```
}

### ================================================================
### TEST 9: Eliminar reseña
### Debe retornar 200 OK
### ================================================================
DELETE {{baseUrl}}/controladorresenas/eliminar/res-004
Authorization: Basic {{auth}}

### ================================================================
### TEST 10: Acceso sin autenticación
### Debe retornar 401 Unauthorized
### ================================================================
GET {{baseUrl}}/controladorresenas/listar
```

✅ Checklist Final

Antes del Examen

- ☐ Entiendo la estructura MVC
- ☐ Sé crear tablas SQL con relaciones
- ☐ Puedo escribir consultas con JOIN
- ☐ Entiendo métodos estáticos vs instancias
- ☐ Sé validar datos de entrada
- ☐ Conozco los códigos HTTP (200, 201, 400, 404, 500)
- ☐ Entiendo namespaces en PHP
- ☐ Sé usar `htmlspecialchars()` para prevenir XSS
- ☐ Entiendo prepared statements para prevenir SQL Injection

Durante el Examen

Orden recomendado:

1. ☐ **Leer todo el enunciado** 2 veces
2. ☐ **Identificar** entidades, campos y relaciones
3. ☐ **Crear tabla SQL** con datos de prueba
4. ☐ **Crear modelo** con métodos CRUD
5. ☐ **Crear controlador API** con validaciones
6. ☐ **Crear vistas HTML** si es necesario
7. ☐ **Crear controlador web** si es necesario
8. ☐ **Probar endpoints** uno por uno
9. ☐ **Revisar** código, sintaxis, seguridad

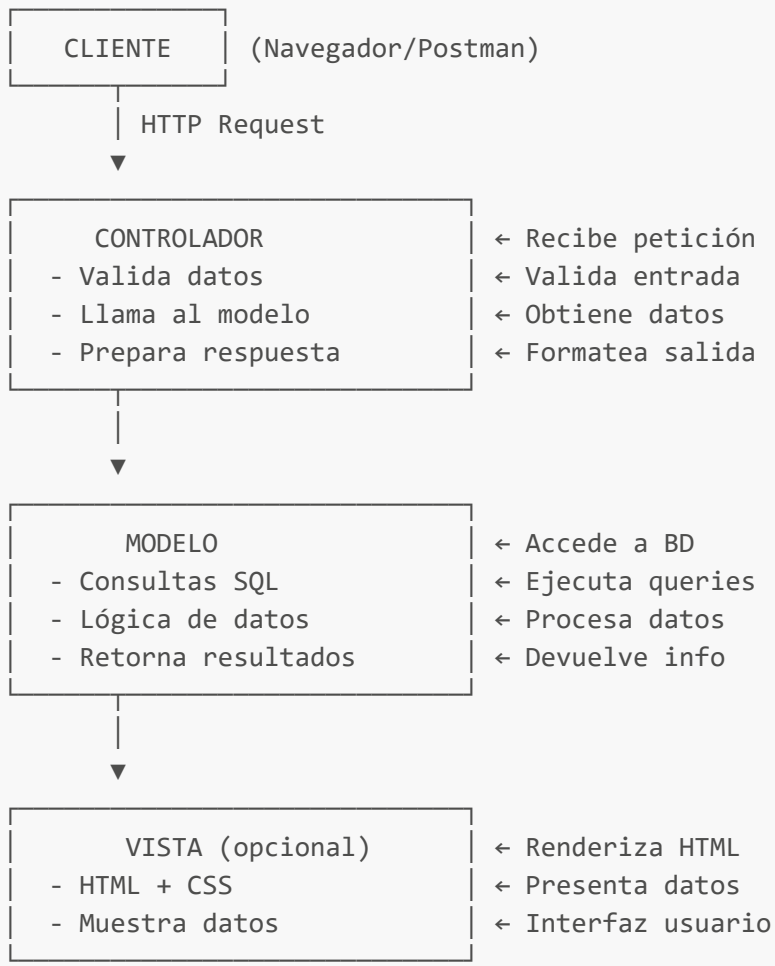
Archivos Mínimos para un Endpoint

Archivo	¿Obligatorio?	Propósito
`bd/bd.sql`	⚠ Solo si hay nueva tabla	Crear estructura de datos

Archivo	¿Obligatorio?	Propósito
app/modelos/GestorXXX.php	✅ SÍ	Acceso a base de datos
app/controladores/ControladorXXX.php	✅ SÍ (API)	Lógica de negocio API
app/controladores/XXX.php	⚠️ Solo si hay vistas	Controlador web
app/vistas/xxx/*.php	⚠️ Solo si se pide	Interfaz HTML
tests_xxx.http	☆ Recomendado	Validar funcionamiento

🎓 Conceptos Clave para Recordar

MVC Architecture



Flujo de Datos

```
URL: /controladorresenas/listar
  ↓
Core.php (enrutador)
  ↓
ControladorResenas->listar()
```

```
  ↓
GestorResenas::getResenas()
  ↓
Base de Datos
  ↓
[Datos]
  ↓
jsonResponse() o vista()
  ↓
Respuesta al Cliente
```

📖 Recursos Adicionales

Comandos SQL Útiles

```sql
-- Ver estructura de tabla
DESCRIBE resenas;

-- Contar registros
SELECT COUNT(*) FROM resenas;

-- Buscar por texto
SELECT * FROM resenas WHERE Comentario LIKE '%excelente%';

-- Agrupar por producto
SELECT Producto, AVG(Puntuacion) as promedio, COUNT(*) as total
FROM resenas
GROUP BY Producto;
```

Debugging en PHP

```php
// Ver contenido de variable
var_dump($datos);
die(); // Detener ejecución

// Error log
error_log(print_r($datos, true));

// Try-catch siempre
try {
    // código
} catch (\Exception $e) {
    error_log($e->getMessage());
}
```

🏆 Consejos para el Examen

1. **Lee TODO primero** - No empieces a codear sin entender el requisito completo
2. **Empieza por la BD** - Sin datos no hay nada que mostrar
3. **Datos de prueba** - Siempre inserta ejemplos para poder probar
4. **Valida SIEMPRE** - Nunca confíes en datos del usuario
5. **htmlspecialchars()** - En TODAS las salidas HTML
6. **Prepared statements** - En TODAS las queries SQL
7. **Try-catch** - En TODOS los métodos que accedan a BD
8. **Códigos HTTP correctos** - 200, 201, 400, 404, 500
9. **Prueba cada endpoint** - Antes de pasar al siguiente
10. **Revisa sintaxis** - Un `;` puede costarte puntos

🎯 Ejemplo Rápido (Memorizar)

Crear endpoint completo en 5 pasos:

```
1. BD: CREATE TABLE xxx + INSERT datos
2. Modelo: GestorXXX con métodos estáticos
3. Controlador: ControladorXXX con métodos públicos
4. Vista (opcional): xxx/listar.php, xxx/crear.php
5. Probar: tests_xxx.http
```

Plantilla de método del controlador:

```php
public function listar(): void
{
    $this->requireBasicAuth();

    try {
        $datos = GestorXXX::getDatos();
        $this->jsonResponse(['success' => true, 'datos' => $datos], 200);
    } catch (\Exception $e) {
        $this->jsonResponse(['error' => $e->getMessage()], 500);
    }
}
```

¡Buena suerte en tu examen! 🍀

Recuerda: **Práctica, práctica, práctica.** Crea varios endpoints de ejemplo antes del examen.

Documento creado para estudio - Desarrollo de Aplicaciones Web - 2026