🎯 SIMULACRO EXAMEN PRÁCTICO DWES - PHP

Duración: 3 horas

SISTEMA DE GESTIÓN DE RESERVAS DE SALAS

📋 CONTEXTO

Una empresa necesita un sistema web para gestionar las reservas de sus salas de reuniones. El sistema debe permitir a los usuarios autenticados realizar reservas y consultar la disponibilidad.

📂 ESTRUCTURA DE ARCHIVOS REQUERIDA

```
proyecto/
├── public/
│   ├── index.php           (Login)
│   ├── error.php           (Página de error)
│   ├── panel.php           (Panel principal - requiere login)
│   ├── reservar.php        (Formulario + proceso de reserva)
│   ├── listar.php          (Listado de reservas)
│   └── logout.php          (Cerrar sesión)
├── src/
│   ├── Conexion.php        (Clase conexión PDO)
│   ├── GestorReservas.php  (CRUD + transacciones)
│   └── Reserva.php         (Clase entidad)
├── tools/
│   └── Mailer.php          (Clase envío correos)
└── vendor/                 (Composer)
```

🗄️ BASE DE DATOS

Crea la base de datos `reservas_salas` con la siguiente tabla:

```sql
CREATE TABLE reservas (
    id INT PRIMARY KEY AUTO_INCREMENT,
    sala VARCHAR(50) NOT NULL,
    fecha DATE NOT NULL,
    hora_inicio TIME NOT NULL,
    hora_fin TIME NOT NULL,
    usuario VARCHAR(100) NOT NULL,
    email VARCHAR(150) NOT NULL,
    confirmada TINYINT(1) DEFAULT 0,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    UNIQUE KEY unique_reserva (sala, fecha, hora_inicio)
) ENGINE=InnoDB;

-- Tabla usuarios (para login)
CREATE TABLE usuarios (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL
);

-- Usuario de prueba (password: 1234)
INSERT INTO usuarios (username, password) VALUES ('admin', '1234');
```

EJERCICIOS

EJERCICIO 1: Login con Sesiones (15 puntos)

Archivo: `public/index.php`

1. Crea un formulario de login con usuario y contraseña
2. El formulario debe enviarse por **GET** al mismo script
3. Valida las credenciales contra la tabla `usuarios` de la BBDD
4. Si el login es correcto:
    - Guarda en sesión: `usuario_id`, `username`, `autenticado = true`
    - Redirige a `panel.php`
5. Si es incorrecto:
    - Redirige a `error.php` pasando un mensaje por GET
6. Si el usuario ya está autenticado, redirige directamente a `panel.php`

Archivo: `public/error.php`

- Muestra el mensaje de error recibido por GET
- Incluye un enlace para volver a `index.php`

Archivo: `public/panel.php`

- Verifica que el usuario esté autenticado (si no, redirige a `index.php`)
- Muestra mensaje de bienvenida con el nombre de usuario
- Enlaces a: Reservar, Listar reservas, Cerrar sesión

EJERCICIO 2: Clase de Conexión PDO (10 puntos)

Archivo: `src/Conexion.php`

Crea una clase `Conexion` con:

- Namespace: `App\Src`
- Método estático `getConexion()` que devuelve un objeto PDO
- Configuración:

    - Charset: `utf8mb4`
    - Modo de errores: `PDO::ERRMODE_EXCEPTION`
- Implementa el patrón Singleton (una sola instancia)

EJERCICIO 3: Clase Entidad Reserva (10 puntos)

Archivo: `src/Reserva.php`

Crea la clase `Reserva` con:

- Namespace: `App\Src`
- Atributos privados para cada columna de la tabla
- Constructor que reciba los datos necesarios (excepto id y created_at)
- Getters y setters para todos los atributos
- Método `__toString()` que devuelva un resumen de la reserva

EJERCICIO 4: Formulario de Reserva con Cookies (20 puntos)

Archivo: `public/reservar.php`

1. **Verificar autenticación** (redirigir si no está logueado)

2. **Sistema de cookies:**

    - Al cargar la página, lee una cookie llamada `ultima_sala`
    - Si existe, pre-selecciona esa sala en el formulario
    - Al enviar el formulario, guarda la sala elegida en la cookie (duración: 7 días)

3. **Formulario con los campos:**

    - Sala (SELECT con opciones: Sala A, Sala B, Sala C)
    - Fecha (input date)
    - Hora inicio (SELECT: 08:00, 09:00, 10:00... hasta 20:00)
    - Hora fin (SELECT: 09:00, 10:00... hasta 21:00)
    - Email del usuario (input email)

4. **Validaciones en el servidor:**

    - Todos los campos son obligatorios
    - La hora de fin debe ser mayor que la hora de inicio
    - El email debe ser válido (usar `filter_var`)

5. **Método:** GET

6. **Destino:** el mismo script (recarga)

7. **Si la validación es correcta:**

    - Usa `GestorReservas` para insertar la reserva
    - Muestra mensaje de éxito encima del formulario

8. **Si hay error:**
    - Muestra mensaje de error encima del formulario
    - Mantén los valores introducidos en el formulario

EJERCICIO 5: CRUD con GestorReservas (20 puntos)

Archivo: `src/GestorReservas.php`

Crea la clase `GestorReservas` que:

1. **Namespace:** `App\Src`
2. **Implemente la interfaz** `AccionesBD` con los métodos:

```php
interface AccionesBD {
    public function insertar(array $datos): bool;
    public function eliminar(int $id): bool;
    public function actualizar(int $id, array $datos): bool;
    public function listar(): array;
    public function obtenerPorId(int $id): ?Reserva;
}
```

3. **Método adicional:** `listarPorUsuario(string $username): array`
    - Devuelve solo las reservas del usuario indicado
4. **Manejo de excepciones:**
    - Captura `PDOException` en cada método
    - Lanza excepciones propias con mensajes claros

EJERCICIO 6: Transacciones - Reservas Múltiples (15 puntos)

Añadir en: `src/GestorReservas.php`

Implementa el método:

```php
public function reservaMultiple(array $reservas): bool
```

Este método debe:

1. Recibir un array de arrays asociativos, cada uno con los datos de una reserva
2. Iniciar una transacción
3. Preparar la sentencia INSERT una sola vez
4. Ejecutar el INSERT para cada reserva
5. Si TODAS las inserciones son correctas → `commit()`

6. Si ALGUNA falla (ej: reserva duplicada por UNIQUE) → `rollBack()` y no se inserta ninguna

Prueba obligatoria: En `panel.php` añade un enlace "Test transacciones" que:

- Intente insertar 3 reservas válidas → deben insertarse todas
- Intente insertar 2 reservas donde la segunda tenga la misma sala/fecha/hora que la primera → no debe insertarse ninguna

---

EJERCICIO 7: Listado de Reservas (10 puntos)

Archivo: `public/listar.php`

1. Verificar autenticación
2. Mostrar todas las reservas del usuario actual en una tabla HTML
3. Columnas: Sala, Fecha, Hora inicio, Hora fin, Estado (Confirmada/Pendiente)
4. Cada fila debe tener:
    - Enlace "Eliminar" que pase el ID por GET
    - Enlace "Confirmar" (solo si no está confirmada)
5. Al eliminar, usa `GestorReservas->eliminar()`
6. Al confirmar, usa `GestorReservas->actualizar()` para cambiar `confirmada = 1`
7. Muestra mensaje de éxito/error tras la acción

EJERCICIO 8: Envío de Correo (BONUS - 10 puntos extra)

Archivo: `tools/Mailer.php`

Crea una clase `Mailer` que:

1. Tenga un método `enviarConfirmacion(Reserva $reserva): bool`
2. Envíe un email al usuario con los datos de su reserva
3. Use PHPMailer con configuración SMTP para Gmail
4. El email debe ser HTML con formato legible

Integración:

- Cuando el usuario hace clic en "Confirmar" en el listado, además de actualizar la BBDD, envía el email de confirmación

---

🔧 REQUISITOS TÉCNICOS

1. **Toda la navegación debe ser por GET** (excepto donde se indique)
2. **Usar PDO** con sentencias preparadas (NUNCA concatenar SQL)
3. **Usar namespaces** y autoload de Composer
4. **Validar siempre** que las variables existen antes de usarlas (`isset`)
5. **session_start()** al principio de cada script que use sesiones
6. **setcookie()** antes de cualquier salida HTML
7. **header() + exit** para redirecciones
8. **Capturar excepciones** en operaciones de BBDD

CRITERIOS DE EVALUACIÓN

Criterio	Puntos
Login y sesiones funcionan correctamente	15
Clase Conexion PDO correcta	10
Clase Reserva con todos los métodos	10
Formulario con cookies y validaciones	20
CRUD completo y funcional	20
Transacciones implementadas correctamente	15
Listado con acciones de eliminar/confirmar	10
BONUS: Envío de correo	+10
TOTAL	100 (+10)

⚠ PENALIZACIONES

- -5 puntos: SQL sin sentencias preparadas
- -5 puntos: No validar existencia de variables ($_GET, $_POST, $_SESSION)
- -3 puntos: No usar namespaces
- -3 puntos: Errores PHP visibles (warnings, notices)
- -2 puntos: No seguir la estructura de archivos indicada

CONSEJOS

1. **Empieza por la BBDD** - Crea las tablas primero
2. **Luego Conexion.php** - La necesitarás para todo
3. **Después el login** - Sin esto no puedes probar el resto
4. **GestorReservas** - Implementa primero `insertar()` y `listar()`
5. **El formulario** - Prueba que inserta bien antes de añadir cookies
6. **Transacciones al final** - Es lo más complejo
7. **Guarda frecuentemente** - No pierdas el trabajo

¡Buena suerte!