

Objetivos & Entregable UD4

Iremos creando las diferentes carpetas y archivos PHP, así como su documentación en formato Markdown (.md) para, al finalizar cada semana, subir al menos un commit con los cambios y archivos añadidos, comentando el código debidamente.

Proyecto UD4_PDO:

- La idea de esta unidad es aplicar las nuevas características adquiridas en un proyecto haciendo uso de la clase PDO para acceder a una base de datos.
- Estructura lo que necesites para aplicar los conceptos de la unidad.

Objetivo

El objetivo de esta actividad es desarrollar un sistema de gestión CRUD (Crear, Leer, Actualizar, Eliminar) utilizando PHP y el patrón Modelo-Vista-Controlador (MVC). Este proyecto servirá como práctica para comprender la estructura y el funcionamiento de aplicaciones web en PHP sin el uso de JavaScript ni CSS.

Requisitos

1. **Lenguaje** : PHP
2. **Base de Datos** : MySQL
3. **Patrón** : Modelo-Vista-Controlador (MVC)
4. **Sin JavaScript** : Todo el manejo de la interfaz se realizará con PHP y HTML puro.
5. **Sin CSS** : La presentación será básica, solo con HTML.

Estructura del Proyecto

Crea, al menos, la siguiente estructura de carpetas y archivos en tu entorno de desarrollo:

```
proyectoUD4/ | ├── db.php # Archivo de conexión a la base de datos |── models/ #
Carpeta para los modelos |   ├── Model.php # Modelo para la tabla de ... |── controllers/
# Carpeta para los controladores |   ├── ModelController.php # Controlador para gestionar ... |
|── views/ # Carpeta para las vistas |   ├── nombreModel/ |   ├── index.php # Vista para
```

mostrar y gestionar ...(cambia nombreModel) | |—— index.php # Archivo principal que gestiona las acciones |—— README.md # Documentación del proyecto

```
proyectoUD4/
|
|—— db.php          # Archivo de conexión a la base de datos
|—— models/         # Carpeta para los modelos
|   |—— Model.php   # Modelo para la tabla de ...
|
|—— controllers/    # Carpeta para los controladores
|   |—— ModelController.php # Controlador para gestionar ...
|
|—— views/          # Carpeta para las vistas
|   |—— nombreModel/
|       |—— index.php # Vista para mostrar y gestionar ...(cambia nombreModel)
|
|—— index.php       # Archivo principal que gestiona las acciones
|—— README.md       # Documentación del proyecto
```

Tablas Sugeridas

Para esta actividad, puedes elegir entre las siguientes 20 tablas con 4 campos cada una para implementar el CRUD:

1. **Coches** (NO USAR)
2. **Productos** : id, nombre, precio, cantidad
3. **Libros** : id, título, autor, año_publicación
4. **Películas** : id, título, director, año_lanzamiento
5. **Empleados** : id, nombre, puesto, salario
6. **Clientes** : id, nombre, dirección, teléfono
7. **Órdenes** : id, cliente_id, producto_id, cantidad
8. **Categorías** : id, nombre, descripción, fecha_creación
9. **Proveedores** : id, nombre, contacto, teléfono
10. **Inventario** : id, producto_id, cantidad, ubicación
11. **Reservas** : id, cliente_id, fecha_reserva, estado
12. **Comentarios** : id, usuario_id, contenido, fecha
13. **Facturas** : id, cliente_id, total, fecha_emisión
14. **Pagos** : id, factura_id, monto, fecha_pago

- 15. **Tareas** : id, descripción, estado, fecha_creación
- 16. **Proyectos** : id, nombre, fecha_inicio, fecha_fin
- 17. **Eventos** : id, nombre, fecha, ubicación
- 18. **Cursos** : id, título, profesor, duración
- 19. **Encuestas** : id, pregunta, respuesta_a, respuesta_b

Instrucciones

- 1. **Configuración de la Base de Datos :**
- 2. Crea una base de datos en MySQL.
- 3. Importa el esquema de la tabla que hayas elegido para gestionar (por ejemplo, la tabla de coches).
- 4. **Conexión a la Base de Datos :**
- 5. En el archivo `db.php` , establece la conexión a la base de datos utilizando PDO.
- 6. **Modelo :**
- 7. Crea el modelo correspondiente en `models/Coche.php` (o el modelo que elijas) con las funciones necesarias para realizar las operaciones CRUD.
- 8. **Controlador :**
- 9. Implementa el controlador en `controllers/CocheController.php` que gestionará las solicitudes y llamará a las funciones del modelo.
- 10. **Vista :**
- 11. Diseña la vista en `views/coches/index.php` para mostrar los registros y permitir al usuario realizar las operaciones CRUD.
- 12. **Archivo Principal :**
- 13. En `index.php` , gestiona las acciones que se realizarán (agregar, editar, eliminar, mostrar).

Documentación del Proyecto

Crea un archivo `README.md` en la raíz del proyecto que contenga la siguiente información:

```
# Proyecto CRUD en PHP
```

```
Este proyecto implementa un sistema de gestión CRUD utilizando PHP y el patrón Modelo-Vista-Controlador (MVC) para gestionar una tabla con cuatro columnas. No se utiliza JavaScript ni CSS, lo que hace que la aplicación sea sencilla y fácil de entender.
```

```
## Requisitos
```

- PHP 7.0 o superior
- Servidor web (Apache, Nginx, etc.)
- MySQL o Maria

AUTOR/A:

Ampliación del proyecto con FPDF

Una vez hecho esto, **y no antes**, amplía tu proyecto con la generación de tu tabla en PDF usando fpdf como se muestra en el "**punto 6 Ficheros y PDF**"

Ten en cuenta que el ejemplo del punto 6 usa programación estructurada y no programación orientada a objetos. Puedes implementarlo así e incrustarlo en tu proyecto mediante un enlace, ya que lo único que va a generar es un documento PDF

Mantenimiento repositorio Git Hub

Hay que subir el enlace de dicho repositorio a Moodle.

Los ejercicios que se han de subir al repositorio Github y enlace al aula Moodle serán:

1. Repositorio **Github** público.
2. **Video-presentación** presentando el código de la unidad (al final) y las nuevas características aprendidas, de entre **5-10 minutos**.
3. Lenguaje técnico y no de usuario, comentando los aspectos del lenguaje adquiridos



Entregable

Se ha de entregar en el tiempo estimado en Moodle

Duración: 5 horas de clase

Recomendaciones

- [] Usa funciones como `htmlspecialchars()` para evitar problemas de **inyección** de código al imprimir datos de usuarios.
- [] Para entornos de producción, es adecuado utilizar **contraseñas seguras** para el usuario de la base de datos.

Patron de diseño

Aunque para la tarea anterior se pide una estructura concreta, aquí tienes una **estructura de diseño** típica para el MVC que es recomendable ir afianzando ahora antes de seguir avanzando

```
project/
├── app/
│   ├── Controllers/ <- Controladores
│   ├── Models/      <- Modelos
│   └── Views/        <- Vistas
├── assets (o public)/
│   ├── css/          <- Archivos CSS
│   ├── js/           <- Archivos JavaScript
│   └── index.php      <- Punto de entrada
├── routes/           <- Rutas del proyecto
├── storage/          <- Archivos generados por el sistema (logs, caché, etc.)
└── vendor/           <- Dependencias de Composer
```

Tienes más info en el apartado [Patrón de diseño](#)