MODELO-VISTA-CONTROLADOR

ESTRUCTURA GENERAL

Este patrón separa la aplicación en tres componentes principales.

- ***Modelo (Model):*** Gestiona la lógica de datos y el acceso a la base de datos.
- ***Vista (View):*** Se encarga de la presentación de la información, es decir, lo que el usuario ve.
- ***Controlador (controller):*** Maneja la lógica de la aplicación, es decir, es el puente entre el modelo y la vista.

CARPETAS Y ARCHIVOS

Obligatorias:

✧ ***Controllers*** -> Contiene los controladores que gestionan la lógica de la aplicación. Estos reciben las solicitudes del usuario (a través de una URL), interactúan con el modelo para obtener datos, y luego cargan la vista adecuada para mostrar los datos al usuario.

- mainController.php: Es el controlador principal, que selecciona qué controlador específico debe cargar según la petición del usuario.
- otroController.php: Controlador especializado en gestionar alguna entidad del programa como puede ser productos, usuarios...etc.

✧ ***Models*** -> Representan y gestionan los datos de la aplicación. Generalmente cada tipo de entidad que maneja la aplicación tiene su propio modelo. Interactúan con la base de datos.
Sin los modelos, los controladores no tendrían una forma estructurada de manejar y procesar los datos.

- ejemploModel.php: Clase que representa un modelo de una entidad.

✧ ***Views*** -> Son las responsables de mostrar la información al usuario. Suele haber un archivo de vista para cada acción o tipo de datos que se quieren presentar.

✧

- listView.phtml: Vista que representa una lista y puede ser llamada por diferentes controladores para mostrar listas de distintos tipos de datos.

Se pueden tener tantos controladores, modelos o vistas como sea necesario.

- ***db.php:*** Este archivo gestiona la conexión a la base de datos.

- ***index.php:*** Es el punto de entrada de la aplicación. Carga el controlador principal y se encarga de enrutar las solicitudes.

No obligatorias, pero sí recomendadas:

- ***Public*** -> Esta carpeta contiene todos los recursos estáticos y archivos accesibles públicamente, como archivos de estilo, imágenes y archivos JavaScript.
  Dentro de esta carpeta, puede haber distintas subcarpetas, como **css** que almacenará "*style.css*", **img**, que almacenara todo el contenido multimedia...etc

- ***Helpers*** -> Los helpers contienen funciones y clases auxiliares que no pertenecen a un modelo, vista o controlador específico, pero que son necesarias para el funcionamiento de la aplicación.
  Cada helper puede realizar funciones diferentes, como manipulación de archivos, validación de formularios...etc. Esto permite encapsular utilidades que pueden ser compartidas entre diferentes controladores y vistas.

  Ejemplo de archivo helper

```php
<?php
class FileHelper {
    // Método para mover un archivo subido a una carpeta específica
    public static function uploadFile($file, $destination) {
        if (is_uploaded_file($file['tmp_name'])) {
            move_uploaded_file($file['tmp_name'], $destination .
basename($file['name']));
            return true;
        }
        return false;
    }
}
```

OPERADORES Y FUNCIONES

require() -> Esta función incluye y evalúa el archivo especificado. Si el archivo no se encuentra o hay un error al cargarlo, el script genera un "fatal error" y se detiene.
Se utiliza para incluir archivos que son necesarios para el funcionamiento de la aplicacioón, como archivos de configuración o clases esenciales.

include() -> También carga archivos como require(), pero include() devuelve una advertencia en caso de error y el script sigue su ejecución.

Require_once() -> Similar a require(), pero con la diferencia de que este asegura que el archivo solo se cargue una vez. Si se llama a require_once() varias veces para el mismo archivo, se ignoran las llamadas posteriores.
Se suele utilizar para evitar problemas de múltiples inclusiones del mismo archivo. Require() incluirá el archivo cada vez que se llame, sin importar si ya fue incluido anteriormente.

Operador de Resolución de Ámbito (::) -> Se utiliza para acceder a métodos y propiedades estáticas, constantes y métodos de clase. El uso más común es para métodos estáticos (que no necesitan instancias de la clase).

isset() -> Determina si una variable está definida y no es null.

Empty() -> Comprueba si una variable está vacía. Devuelve true si la variable no está definida o su valor es 0, “”, null, false, o [ ].

Die() o exit() -> Terminan la ejecución del script inmediatamente.

header() -> Envía encabezados HTTP al navegador. Se suele utilizar para redireccionar a otras páginas o controlar caché. Ejemplo: header(“Location: otraPagina.php”).

$this -> Se usa dentro de una clase para hacer referencia a la instancia actual de esa clase. También se usa dentro de métodos de una clase, para acceder a propiedades y métodos de la misma instancia.