Tarea: Generador de Formularios Dinámicos con PHP

Objetivo General

Crear una clase PHP que genere formularios HTML dinámicos con funcionalidad AJAX, integrando HTML, CSS, JavaScript y PHP de manera cohesiva.

Descripción del Proyecto

Desarrollarás una clase PHP llamada `FormularioDinamico` que permita crear formularios web de manera programática, con envío asíncrono de datos usando JavaScript y una presentación visual atractiva con CSS.

📝 INSTRUCCIONES ESPECÍFICAS DE IMPLEMENTACIÓN

IMPORTANTE: DEBE SEGUIR EXACTAMENTE LA ESTRUCTURA ENSEÑADA EN CLASE

No se aceptarán trabajos que no sigan las convenciones específicas del curso, independientemente de si funcionan correctamente.

Estructura obligatoria de la clase PHP:

```
<?php
class FormularioDinamico {
    // 1. PROPIEDADES (en este orden específico)
    private $campos;
    private $urlDestino;
    private $nombreFormulario;

    // 2. CONSTRUCTOR (formato exacto como se enseñó)
    public function __construct($nombre, $url) {
        // Implementación siguiendo patrón visto en clase
    }

    // 3. MÉTODOS PÚBLICOS (orden específico)
    public function addField() { }
    public function show() { }
    public function getJS() { }
    public function getCSS() { }
    public function render() { }

    // 4. MÉTODOS PRIVADOS (si los hay)
    private function validField() { }
}
?>
```

Patrón obligatorio para métodos:

- Cada método debe iniciar con comentario descriptivo según formato visto
- Validación de parámetros como se enseñó en clase
- Return statements siguiendo la estructura específica del curso
- Manejo de errores según metodología vista

Estructura HTML obligatoria:

- Usar la plantilla de formulario específica enseñada en clase
- Atributos en el orden exacto visto: id, class, name, otros
- Indentación con el número exacto de espacios/tabs usado en clase

JavaScript según patrón del curso:

- Estructura de event listeners como se enseñó
- Manejo de FormData siguiendo el ejemplo específico de clase
- Estructura de fetch() con el formato exacto visto
- Nombres de funciones y variables según convención del curso

---

Requisitos:

1. **Crear la clase `FormularioDinamico`** con las siguientes propiedades privadas:
   - $campos (array): Almacenará la información de los campos
   - $urlDestino (string): URL donde se enviará el formulario
   - $nombreFormulario (string): Identificador único del formulario
2. **Método constructor** que reciba:
   - Nombre del formulario
   - URL de destino para el submit
3. **Método `addField()`** que reciba:
   - $nombreCampo: nombre técnico del campo (name del input)
   - $alias: texto que se mostrará como label
   - $tipo: tipo de input (text, email, password, textarea, select, etc.)
   - $requerido: boolean para campos obligatorios
   - $opciones: array opcional para campos select

Ejemplo de uso esperado:

```
$formulario = new FormularioDinamico("contacto", "procesar.php");
$formulario->addField("nombre", "Nombre Completo", "text", true);
$formulario->addField("email", "Correo Electrónico", "email", true);
```

PARTE 2: Generación de HTML (25 puntos)

Requisitos:

1. **Método `show()`** que retorne una cadena con:
   - Formulario HTML con atributos apropiados

- Campos generados dinámicamente según los datos almacenados
- Botón de submit personalizado
- Estructura semántica correcta

2. **Especificaciones técnicas:**
   - Usar <form> con atributo id único
   - Cada campo debe tener su <label> correspondiente
   - Campos requeridos deben tener el atributo required
   - Incluir clases CSS para styling posterior

Estructura HTML esperada:

```
<form id="form-[nombre]" class="formulario-dinamico">
  <div class="campo-grupo">
    <label for="campo-nombre">Alias del Campo</label>
    <input type="text" id="campo-nombre" name="nombre" required>
  </div>
  <!-- Más campos... -->
  <button type="submit" class="btn-enviar">Enviar</button>
</form>
```

PARTE 3: JavaScript con Fetch API (25 puntos)

Requisitos:

1. **Método getJS()** que genere código JS para:
   - Prevenir el comportamiento por defecto del formulario
   - Capturar el evento submit
   - Recopilar datos usando FormData
   - Enviar datos con fetch() API
   - Manejar respuestas exitosas y errores
2. **Funcionalidades requeridas:**
   - Validación básica antes del envío
   - Indicador de carga durante el envío
   - Mensajes de éxito o error
   - Limpieza del formulario tras envío exitoso

Estructura JavaScript esperada:

```
document.getElementById('form-[nombre]').addEventListener('submit', function(e) {
  e.preventDefault();

  const formData = new FormData(this);

  fetch('[url-destino]', {
    method: 'POST',
    body: formData
  })
  .then(response => response.json())
  .then(data => {
    // Manejar respuesta exitosa
  })
  .catch(error => {
```

```
    // Manejar errores
  });
});
```

PARTE 4: Estilos CSS (15 puntos)

Requisitos:

1. **Método `getCSS()`** que genere estilos para:
   - Layout responsivo del formulario
   - Estilizado de campos de entrada
   - Botones con efectos hover
   - Estados de validación (error/éxito)
   - Indicadores de carga
2. **Especificaciones de diseño:**
   - Usar Flexbox o Grid para layout
   - Colores coherentes y accesibles
   - Transiciones suaves
   - Diseño mobile-first

PARTE 5: Método Integrador (10 puntos)

Requisitos:

1. **Método `render()`** que:
   - Combine HTML, CSS y JavaScript
   - Retorne el código completo listo para usar
   - Incluya las etiquetas <style> y <script> apropiadas

Entregables

1. **Archivo principal:** FormularioDinamico.php
2. **Archivo de prueba:** test.php que demuestre el uso de la clase
3. **Documentación:** Comentarios explicativos en el código
4. **Ejemplo funcional:** Formulario de contacto completo

Criterios de Evaluación

🔍 VALIDACIÓN OBLIGATORIA DE ESTRUCTURA DE CÓDIGO (40% de la calificación)

EL CÓDIGO DEBE SEGUIR EXACTAMENTE LAS CONVENCIONES ENSEÑADAS EN CLASE:

1. **Estructura de Clase PHP:**

    - Uso correcto de modificadores de acceso (private, public, protected)
    - Declaración de propiedades al inicio de la clase
    - Orden específico: constructor → métodos públicos → métodos privados
    - Nomenclatura de métodos en camelCase como enseñado
    - Uso de $this-> para acceder a propiedades de la clase
2. **Convenciones de Nomenclatura:**
    - Variables PHP con prefijo `# Tarea: Generador de Formularios Dinámicos con PHP

Objetivo General

Crear una clase PHP que genere formularios HTML dinámicos con funcionalidad AJAX, integrando HTML, CSS, JavaScript y PHP de manera cohesiva.

Descripción del Proyecto

Desarrollarás una clase PHP llamada FormularioDinamico que permita crear formularios web de manera programática, con envío asíncrono de datos usando JavaScript y una presentación visual atractiva con CSS.

PARTE 1: Estructura de la Clase PHP (25 puntos)

Requisitos:

1. **Crear la clase FormularioDinamico** con las siguientes propiedades privadas:
    - $campos (array): Almacenará la información de los campos
    - $urlDestino (string): URL donde se enviará el formulario
    - $nombreFormulario (string): Identificador único del formulario
2. **Método constructor** que reciba:
    - Nombre del formulario
    - URL de destino para el submit
3. **Método agregarCampo()** que reciba:
    - $nombreCampo: nombre técnico del campo (name del input)
    - $alias: texto que se mostrará como label
    - $tipo: tipo de input (text, email, password, textarea, select, etc.)
    - $requerido: boolean para campos obligatorios
    - $opciones: array opcional para campos select

Ejemplo de uso esperado:

```
$formulario = new FormularioDinamico("contacto", "procesar.php");
$formulario->agregarCampo("nombre", "Nombre Completo", "text", true);
$formulario->agregarCampo("email", "Correo Electrónico", "email", true);
```

PARTE 2: Generación de HTML (25 puntos)

Requisitos:

1. **Método `generarHTML()`** que retorne una cadena con:
   - Formulario HTML con atributos apropiados
   - Campos generados dinámicamente según los datos almacenados
   - Botón de submit personalizado
   - Estructura semántica correcta
2. **Especificaciones técnicas:**
   - Usar `<form>` con atributo `id` único
   - Cada campo debe tener su `<label>` correspondiente
   - Campos requeridos deben tener el atributo `required`
   - Incluir clases CSS para styling posterior

Estructura HTML esperada:

```
<form id="form-[nombre]" class="formulario-dinamico">
  <div class="campo-grupo">
    <label for="campo-nombre">Alias del Campo</label>
    <input type="text" id="campo-nombre" name="nombre" required>
  </div>
  <!-- Más campos... -->
  <button type="submit" class="btn-enviar">Enviar</button>
</form>
```

PARTE 3: JavaScript con Fetch API (25 puntos)

Requisitos:

1. **Método `generarJavaScript()`** que genere código JS para:
   - Prevenir el comportamiento por defecto del formulario
   - Capturar el evento submit
   - Recopilar datos usando `FormData`
   - Enviar datos con `fetch()` API
   - Manejar respuestas exitosas y errores
2. **Funcionalidades requeridas:**
   - Validación básica antes del envío
   - Indicador de carga durante el envío
   - Mensajes de éxito o error
   - Limpieza del formulario tras envío exitoso

Estructura JavaScript esperada:

```
document.getElementById('form-[nombre]').addEventListener('submit', function(e) {
  e.preventDefault();

  const formData = new FormData(this);

  fetch('[url-destino]', {
    method: 'POST',
```

```
    body: formData
  })
  .then(response => response.json())
  .then(data => {
    // Manejar respuesta exitosa
  })
  .catch(error => {
    // Manejar errores
  });
});
```

PARTE 4: Estilos CSS (15 puntos)

Requisitos:

1. **Método `generarCSS()`** que genere estilos para:
   - Layout responsivo del formulario
   - Estilizado de campos de entrada
   - Botones con efectos hover
   - Estados de validación (error/éxito)
   - Indicadores de carga
2. **Especificaciones de diseño:**
   - Usar Flexbox o Grid para layout
   - Colores coherentes y accesibles
   - Transiciones suaves
   - Diseño mobile-first

PARTE 5: Método Integrador (10 puntos)

Requisitos:

1. **Método `renderizarCompleto()`** que:
   - Combine HTML, CSS y JavaScript
   - Retorne el código completo listo para usar
   - Incluya las etiquetas <style> y <script> apropiadas

Entregables

1. **Archivo principal:** FormularioDinamico.php
2. **Archivo de prueba:** test.php que demuestre el uso de la clase
3. **Documentación:** Comentarios explicativos en el código
4. **Ejemplo funcional:** Formulario de contacto completo

y snake_case o camelCase según lo visto en clase

- Nombres de métodos descriptivos siguiendo el patrón enseñado
- Nombres de clases en PascalCase

- IDs y clases HTML siguiendo la convención vista en clase

3. **Estructura HTML:**
    - Uso correcto de etiquetas semánticas vistas en clase
    - Atributos en el orden específico enseñado
    - Indentación con la metodología vista en clase (2 o 4 espacios)
    - Estructura de formularios según el patrón enseñado
4. **JavaScript:**
    - Uso de addEventListener como se enseñó en clase
    - Estructura de funciones según el patrón visto
    - Manejo de promesas con la sintaxis específica enseñada
    - Nomenclatura de variables siguiendo lo visto en clase
5. **CSS:**
    - Selectores organizados según la metodología enseñada
    - Propiedades en el orden específico visto en clase
    - Uso de unidades y valores según las convenciones del curso
    - Comentarios con el formato específico enseñado

⚠ PENALIZACIONES POR ESTRUCTURA INCORRECTA:

- **-20 puntos:** No seguir la estructura de clase PHP enseñada
- **-15 puntos:** Nomenclatura incorrecta de variables/métodos
- **-10 puntos:** Indentación diferente a la enseñada en clase
- **-10 puntos:** Orden incorrecto de elementos HTML/CSS
- **-15 puntos:** No usar los patrones de JavaScript vistos en clase

📋 OTROS CRITERIOS:

- **Funcionalidad (25%):** El código debe funcionar sin errores
- **Integración (20%):** Correcta interacción entre tecnologías
- **Documentación (10%):** Comentarios según formato visto en clase
- **Creatividad (5%):** Características adicionales manteniendo la estructura

Recursos Sugeridos

- Documentación de FormData: MDN Web Docs
- Guía de Fetch API
- Principios de diseño responsivo
- Validación de formularios HTML5

Fecha de Entrega

20-jun-2025

☑ VALIDACIÓN FINAL OBLIGATORIA

ANTES DE ENTREGAR, VERIFICAR:

1. 🔍 **Revisión de Estructura PHP:**
   - [ ] ¿Las propiedades están declaradas en el orden enseñado?
   - [ ] ¿Los métodos siguen la nomenclatura exacta vista en clase? (addField, show, getJS, getCSS, render)
   - [ ] ¿El constructor tiene la estructura específica del curso?
   - [ ] ¿Se usan correctamente private, public como se enseñó?
2. 🔍 **Revisión de HTML:**
   - [ ] ¿La estructura del formulario coincide con la plantilla de clase?
   - [ ] ¿Los atributos están en el orden específico enseñado?
   - [ ] ¿La indentación es exactamente como se vio en clase?
3. 🔍 **Revisión de JavaScript:**
   - [ ] ¿El addEventListener sigue el patrón exacto del curso?
   - [ ] ¿FormData se implementa como se enseñó específicamente?
   - [ ] ¿La estructura de fetch() coincide con los ejemplos de clase?
4. 🔍 **Revisión de CSS:**
   - [ ] ¿Los selectores siguen la metodología vista?
   - [ ] ¿Las propiedades están ordenadas como se enseñó?

⚠ ADVERTENCIA:

Trabajos que no sigan la estructura exacta enseñada en clase serán penalizados severamente, incluso si funcionan correctamente. El objetivo es demostrar que comprendieron y aplicaron las metodologías específicas del curso.
