Guías de estilo PHP

Joel Acoran Cruz Morales - Curso 2023/2024

Proyecto UFC

Documento para las guías de estilos de PHP para el proyecto final de 2do curso del Ciclo formativo de grado superior en Diseño de Aplicaciones Web de Joel Acoran Cruz Morales

1. Nombres significativos

- Usa nombres descriptivos y significativos para variables, funciones y métodos.
- Usa camelCase para nombrar variables y funciones:

```php
You, 2 seconds ago | 1 author (You) | Codeium: Refactor | Explain
class PeleaDBController extends Controller
{
    Codeium: Refactor | Explain | Generate Function Comment | ×
    public function obtenerPeleas()
    {
        // Obtener todas las Peleas con sus relaciones
        $peleas = DB::table('Pelea')
                    ->leftJoin('Velada', 'Pelea.ID_Velada', '=', 'Velada.ID_Velada')
                    ->select('Pelea.*', 'Velada.Nombre_Vel')
                    ->get();

        return response()->json($peleas);
    }

    Codeium: Refactor | Explain | Generate Function Comment | ×
    public function obtenerPeleaPorID($ID_Pelea)
    {
        // Obtener una Pelea por su ID
        $pelea = DB::select('SELECT * FROM Pelea WHERE ID_Pelea = ?', [$ID_Pelea]);

        return response()->json($pelea);
    }
```

Ejemplo de nombres descriptivos y significativos y uso de camelCase.

2. Indentación

- Utiliza espacios en lugar de tabulaciones para la indentación.
- Usa un nivel de indentación de 2 o 4 espacios para cada nivel de anidamiento.

```php
public function show($ID_Pais)
{
    $paises = Pais::find($ID_Pais);

    if ($paises) {
        return $paises;
    } else {
        return response()->json(['message' => 'Pais no encontrado'], 404);
    }
    return response()->json($paises, 200);
}
```

Ejemplo de indentación del código haciendo uso de 4 espacios para niveles de anidamiento.

3. Espacios en blanco

- Deja un espacio antes y después de los operadores.
- Deja un espacio después de las comas en listas de argumentos y matrices,.
- No deja espacios en blanco al final de las líneas

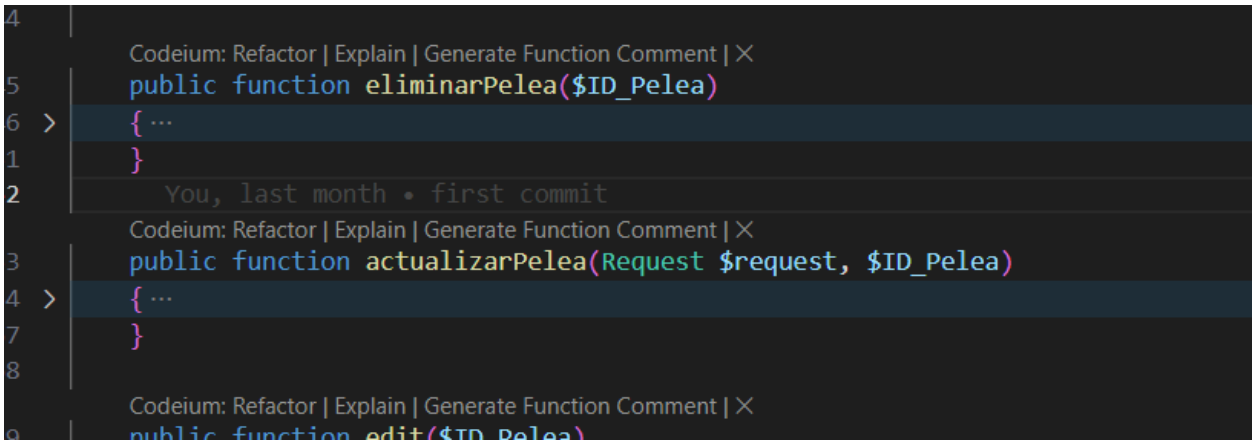
```
Codeium: Refactor | Explain | Generate Function Comment | ×
public function Velada()
{
    return $this->belongsTo('App\Models\Velada', 'ID_Velada');
}

Codeium: Refactor | Explain | Generate Function Comment | ×
public function Arbitro()
{
    return $this->belongsTo('App\Models\Participante', 'ID_Participante');
}
```

Ejemplo de no dejar espacios en blanco, dejar espacios después de cada coma en listas de argumentos y no dejar espacios en blanco al final de las líneas.

4. Líneas en blanco

- Deja líneas en blanco para separar bloques de código relacionados, como funciones y clases.
- Deja líneas en blanco al principio y al final de los bloques de código.



Ejemplo de dejar líneas en blanco antes y después de cada función

5. Comentarios

- Usa comentarios para explicar partes del código que puedan ser confusas o necesiten aclaración.
- Utiliza comentarios de una sola línea (//) para comentarios breves y comentarios de varias líneas (/* */) para explicaciones más largas.

```
Codeium: Refactor | Explain | Generate Function Comment | ×
91  public function edit($ID_Velada)
92  {
93      // Obtener todas las localizaciones
94      $localizaciones = Localizacion::all();
95
96      // Verificar si se encontraron localizaciones
97      if ($localizaciones->isEmpty()) {
98          // Manejar el caso en que no se encontraron localizaciones
99          return response()->json(['error' => 'No se encontraron localizaciones'], 404);
100     }
101
102     // Llamada al método edit del VeladaDBController
103     $velada = $this->veladaDBController->edit($ID_Velada);
104
105     // Verificar si se encontró la velada
106     if (!$velada) {
107         // Manejar el caso en que no se encontró la velada
108         return response()->json(['error' => 'No se encontró la velada'], 404);
109     }
110
111     // Pasar las localizaciones y la velada a la vista
112     return view('Velada.editarVelada', compact('velada', 'localizaciones'));
113 }
114
```

Ejemplo de utilización de comentarios con "//" en la misma línea aclarando la función.

6. Longitud de línea

- Limita la longitud de las líneas de código a 80-100 caracteres.
- Si una línea excede esta longitud, divídela en varias líneas de manera lógica y coherente.

```
Codeium: Refactor | Explain | Generate Function Comment | X
public function crearVelada(Request $request)
{
    $nombreVel = $request->input('Nombre_Vel');
    $idLocalizacion = $request->input('ID_Localizacion');
    $fechaVel = $request->input('Fecha_Vel');

    DB::insert('INSERT INTO Velada (Nombre_Vel, ID_Localizacion, Fecha_Vel) VALUES (?, ?, ?)',
    [$nombreVel, $idLocalizacion, $fechaVel]);

    return redirect('/velada');
}
```

Ejemplo de utilización de comillas dobles de manera consciente para cadena de texto y con líneas de menor longitud y dividiéndolas en caso de que se excedan.

7. Uso de llaves

- Usa llaves {} incluso para bloques de una sola línea.
- Abre las llaves en la misma línea que la declaración de control de flujo (if, else, for, while, etc.).

```
Codeium: Refactor | Explain | Generate Function Comment | X
public function show($ID_Usuario)
{
    $usuario = Usuario::find($ID_Usuario);

    if ($usuario) {
        return response()->json($usuario, 200);
    } else {
        return response()->json(['message' => 'Usuario no encontrado'], 404);
    }

}
```

*Ejemplos de utilización de **if, else***

8. Manejo de errores

- Utiliza manejo de errores adecuado y consistente para garantizar que cualquier excepción o error sea capturado y manejado de manera apropiada.
- Utiliza try...catch para manejar errores de manera segura.

```
Codeium: Refactor | Explain | Generate Function Comment | ×
public function edit($ID_Velada)
{
    try {

        $velada = DB::selectOne('SELECT * FROM Velada WHERE ID_Velada = ?', [$ID_Velada]);
        if (!empty($velada)) {
            return $velada;
        } else {
            return response()->json(['error' => 'No se encontró la Velada'], 404);
        }
    } catch (\Exception $exception) {
        return response()->json(['error' => 'Error al obtener la Velada'], 500);
    }
}
```

Uso de manejo de errores para cada uso con try catch.