Logotipo

Descripción generada automáticamente

**Universidade de Santiago de Compostela**

**Escola Politécnica Superior de Enxeñaría**

**Máster Universitario en Dirección de Proyectos**

Estándar de Codificación PHP

**Sistema de Gestión de Candidatos TICs (SIGECA)**

|  |  |
| --- | --- |
| **Asignatura:** | Gestión de Calidad |
| **Profesor:** | Manuel Marey Pérez |
| **Equipo No. 1** | |
| **Integrantes:** | Maylin Vega Angulo |
|  | Erio Gutierrez Llorens |
| **Curso:** | 2024/2025, Lugo, Galicia, España |

**Información de control del documento**

|  |  |
| --- | --- |
| **Descripción** | **Valor** |
| **Título del Documento:** | Estándar de Codificación PHP |
| **Autor del documento:** | Ing. Erio Gutierrez Llorens |
| **Propietario del Proyecto:** | CEO Empresa de Soluciones Informáticas SIVSA |
| **Director del Proyecto:** | Ing. Maylin Vega Angulo |
| **Versión del Documento:** | 1.0.1 |
| **Confidencialidad:** | Limitada |
| **Fecha:** | 10/03/2025 |

**Aprobación y revisión del documento:**

NOTA: Se requieren todas las aprobaciones. Se deben mantener registros de cada aprobación.

Todos los revisores de la lista se consideran necesarios a menos que se indique explícitamente como Opcionales.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Rol** | **Acción** | **Fecha** |
| Ing. Maylin Vega Angulo | Director de Proyecto | Aprobar | 11/03/2025 |
|  |  |  |  |
|  |  |  |  |

**Historial del documento:**

El Autor del Documento está autorizado a hacer los siguientes tipos de cambios al documento sin requerir que el documento sea aprobado nuevamente:

* *Edición, formato y ortografía.*
* *Aclaraciones.*

Para solicitar un cambio en este documento, póngase en contacto con el Autor del documento o el Propietario del proyecto.

Las modificaciones de este documento se resumen en la siguiente tabla en orden cronológico inverso (primero la última versión).

|  |  |  |  |
| --- | --- | --- | --- |
| **Revisión** | **Fecha** | **Creada por** | **Breve descripción de los cambios** |
| 1.0 | 11/03/2025 | Ing. Erio Gutierrez Llorens | Formato del documento y ajustes en la descripción del estándar. |
|  |  |  |  |

**Gestión de la configuración: Localización del documento**

La última versión de este documento está guardada en:

https://github.com/sivsa/proyectos\_en\_curso/2024/sigeca/2 Planificación/Estándares

**TABLA DE CONTENIDOS**

1. Descripción General 4

2. Reglas Generales 4

3. Archivos PHP 4

3.1. Etiquetas PHP 4

3.2. Codificación de Caracteres 4

4. Clases y Objetos 4

4.1. Nombres de Clases 4

4.2. Declaración de Clases 4

4.3. Propiedades de Clase 5

4.4. PHPDoc en Clases y Métodos 5

5. Tipos de Datos 6

6. Manejo de Cadenas 6

7. Sentencias de Control 6

8. Convenciones de Espaciado 6

9. Comentarios y Documentación 6

10. Manejo de Errores y Excepciones 7

# Descripción General

Este documento establece las reglas y mejores prácticas para la codificación en PHP dentro del equipo de desarrollo. Se basa en la norma **PSR-12**, con algunas adaptaciones para garantizar uniformidad y legibilidad en el código.

# Reglas Generales

* Los archivos DEBEN usar etiquetas <?php o <?=.
* DEBE haber una nueva línea al final del archivo.
* Los archivos DEBEN utilizar codificación UTF-8 sin BOM y saltos de línea Unix (LF).
* DEBE usarse 4 espacios para sangría, NO tabulaciones.
* No se permiten espacios en blanco al final de las líneas.
* Se recomienda un límite de 80 caracteres por línea.
* Solo se permite un espacio antes y después de los operadores (=, +, etc.) y la doble flecha (=>).
* Los nombres de clases DEBEN seguir el estilo StudlyCaps.
* Las constantes de clase DEBEN escribirse en mayúsculas con guiones bajos (CONST\_VALOR).
* Los nombres de métodos y propiedades DEBEN seguir el estilo camelCase.
* Los nombres de propiedades privadas DEBEN comenzar con un guion bajo (\_).
* Se debe usar elseif en lugar de else if.

# Archivos PHP

## Etiquetas PHP

* El código DEBE usar <?php ?> o <?=.
* Si el archivo contiene solo PHP, no debe cerrar con ?>.
* No se deben agregar espacios finales en líneas de código.
* Todos los archivos PHP deben tener la extensión .php.

## Codificación de Caracteres

* Todos los archivos deben estar en UTF-8 sin BOM.

# Clases y Objetos

## Nombres de Clases

* Se usa el estilo StudlyCaps:

class MiClaseEjemplo {}

## Declaración de Clases

* Las clases deben cumplir con lo siguiente:
  + La llave de apertura **DEBE** ir en una nueva línea.
  + Cada clase debe tener un bloque **PHPDoc** adecuado.
  + Una clase por archivo.
  + Usar **espacios de nombres** (namespaces) adecuados.

namespace App\Models;

/\*\*

\* Representa un modelo de usuario.

\*/

class Usuario

{

private string $\_nombre;

public function obtenerNombre(): string

{

return $this->\_nombre;

}

}

## Propiedades de Clase

* Se deben declarar antes de los métodos.
* Las propiedades DEBEN declararse en el siguiente orden:
* public
* protected
* private
* Se deben usar nombres descriptivos en **camelCase**.
* Separar grupos de visibilidad con una línea en blanco.

class Usuario

{

public string $nombre;

protected int $edad;

private string $\_email;

}

## PHPDoc en Clases y Métodos

* Debe especificarse el tipo de dato en @var, @param, @return.
* Para arrays con tipos, se usa Tipo[].

/\*\*

\* Obtiene la lista de usuarios activos.

\*

\* @return Usuario[] Lista de usuarios activos.

\*/

public function obtenerUsuariosActivos(): array

{

return [];

}

# Tipos de Datos

* Usar nombres en minúsculas: bool, int, string, array, null.
* Evitar cambiar el tipo de una variable una vez definida.
* Usar conversiones explícitas: (int), (string), etc.

$precio = (int) $valor;

# Manejo de Cadenas

* Usar comillas simples si no hay variables dentro.

$mensaje = 'Hola Mundo';

* Si contiene comillas simples, usar comillas dobles.

$mensaje = "It's a great day!";

# Sentencias de Control

* Siempre usar llaves {} en estructuras de control.
* La llave de apertura está en la misma línea.
* Los operadores dentro de paréntesis deben estar separados por espacios.

if ($usuario !== null) {

return $usuario->nombre;

}

* Se recomienda evitar la anidación excesiva y usar return temprano.

# Convenciones de Espaciado

* Se permite un solo espacio antes y después de operadores (=, +, etc.).
* Una línea en blanco antes de return en funciones largas.
* Separar bloques lógicos con una línea en blanco.

# Comentarios y Documentación

* Se recomienda el uso de **PHPDoc** en clases, métodos y propiedades.
* Para comentarios dentro del código:

// Este es un comentario de una línea

/\* Este es un comentario de varias líneas \*/

# Manejo de Errores y Excepciones

* Usar try-catch para capturar excepciones.
* Lanzar excepciones en lugar de die() o exit().

try {

$resultado = dividir(10, 0);

} catch (Exception $e) {

echo "Error: " . $e->getMessage();

}