RAP 15\_GA7\_AA1\_EV02\_RM\_DEFINIR ESTÁNDARES DE CODIFICACIÓN DE ACUERDO A PLATAFORMA DE DESARROLLO ELEGIDA

LUZ ANGELA GUTIÉRREZ

ESNEYDER ALEXANDER QUEVEDO GARCÍA

DIEGO ALEXANDER TOLOZA CASTRO

DANIEL VALENCIA SOTO SERVICIO

NACIONAL DE APRENDIZAJE SENA 2627044 PROGRAMA DE ANÁLISIS Y DESARROLLO DE SOFTWARE ADSO ILMER

CUELLO GARCIA

19 DE NOVIEMBRE DEL 2023

Contenido

[INTRODUCCIÓN 3](#_Toc151455449)

[OBJETIVOS GENERALES 4](#_Toc151455450)

[Objetivo especifico 4](#_Toc151455451)

[¿QUÉ ES PHP? 5](#_Toc151455452)

[CONCLUSIONES 13](#_Toc151455453)

[BIBLIOGRAFÍA 14](#_Toc151455454)

# INTRODUCCIÓN

En el apasionante mundo del desarrollo web, la elección del lenguaje de programación adecuado desempeña un papel fundamental en el éxito de un proyecto. Cada lenguaje tiene sus características únicas y capacidades específicas que pueden influir en la eficiencia, la escalabilidad y la seguridad de una aplicación. En este ensayo, nos enfocaremos en la elección de PHP como lenguaje principal en nuestro proyecto y exploraremos las razones convincentes detrás de esta decisión.

PHP, acrónimo de "Hypertext Preprocessor", es un lenguaje de programación ampliamente utilizado en el desarrollo web. Desde su creación en 1994, PHP ha evolucionado hasta convertirse en un pilar crucial en el ecosistema de desarrollo web, impulsado por una gran comunidad de desarrolladores y una amplia adopción en todo el mundo. Su popularidad se debe en gran parte a su sintaxis sencilla y flexible, que permite a los programadores crear aplicaciones web dinámicas y funcionales con relativa facilidad.

# OBJETIVOS GENERALES

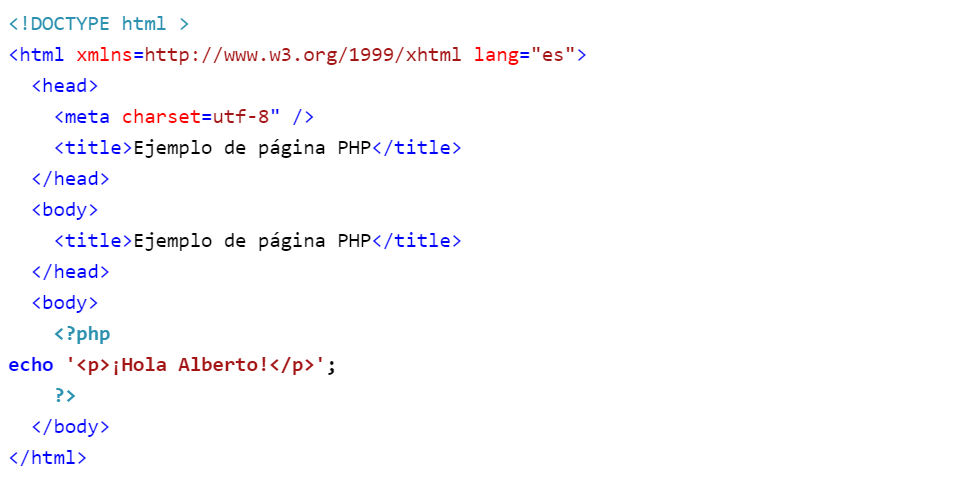
Demostrar cómo PHP, como lenguaje de programación, ofrece ventajas significativas en términos de facilidad de uso, flexibilidad y soporte comunitario, y cómo su integración en nuestro proyecto mejorará la eficiencia del desarrollo web y brindará un rendimiento óptimo para satisfacer las necesidades del negocio.

## Objetivo especifico

* Analizar las características y funcionalidades clave de PHP, como su sintaxis sencilla y flexible, su amplia compatibilidad con diferentes sistemas operativos y servidores web, y su capacidad para interactuar con bases de datos, demostrando cómo estas características benefician nuestro proyecto en términos de desarrollo ágil y eficiente.
* Investigar y presentar ejemplos de proyectos exitosos que han utilizado PHP como lenguaje principal, destacando casos reales donde PHP ha demostrado ser una elección sólida para desarrollar aplicaciones web escalables, seguras y de alto rendimiento, lo que respalda nuestra decisión de utilizarlo en nuestro proyecto.
* Comparar PHP con otros lenguajes de programación web populares, como Python o Java, evaluando las fortalezas y debilidades de cada uno en relación con nuestro proyecto específico. Esto nos permitirá argumentar de manera sólida por qué PHP es la mejor opción en términos de facilidad de uso, comunidad de desarrollo activa y recursos disponibles.
* Examinar las tendencias actuales en el mercado de desarrollo web y la demanda laboral relacionada con PHP. Mediante el análisis de datos y estadísticas actualizadas, podremos respaldar nuestra elección de PHP al demostrar su relevancia continua y la disponibilidad de profesionales capacitados en este lenguaje, lo que facilitará el mantenimiento y la evolución futura de nuestro proyecto.

# ¿QUÉ ES PHP?

PHP es un lenguaje de script que se ejecuta en el lado del servidor, cuyo código se incluye en una página HTML clásica.



La parte en negrita es código PHP incluido en la página HTML dentro de las marcas **<?php y ?>**

**Framework Interoperabiliy Group (FIG)** quienes tienen la tarea de hacer una recopilación de los puntos en común de los proyectos en PHP

El FIG cuenta con una serie de recomendaciones las cuales hace llamar PSR o PHP Standar Recomendation las cuales son elegidas basándose en votaciones realizadas por los miembros del grupo y hasta la fecha cuentan con 4 PSR aceptadas y una que ha quedado obsoleta

PHP Standards Recommendations

**PSR-0 (obsoleto): Estandar de autocarga de clases.**

Habla sobre de como se deberían cargar las clases.

El PSR-0 era un estándar de autoloading obsoleto en PHP que fue reemplazado por el **PSR-4**. El PSR-0 establecía convenciones para la carga automática de clases basadas en una estructura de directorios específica.

Cada namespace debe tener un namespace superior ("Vendor name").

Cada namespace puede tener tantos sub-namespaces como se quiera.

Los nombres de los namespaces o clases deben ser separados por un guion bajo.

Los nombres de los namespaces o clases deben ser ordenadas alfabéticamente.

Todos los archivos deben tener la extensión php.

**PSR-1: Codificación estandar básica.**

Habla sobre de como se deberían cargar las clases.

Los ficheros deben tener codificación UTF-8 sin bom.

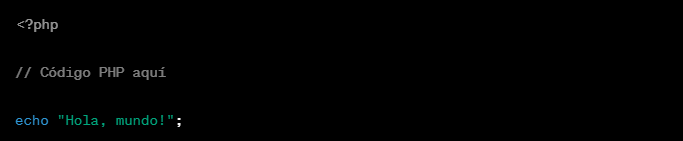
Los nombres de las clases deben utilizar la notación StudlyCaps (MyNameSpace/MiClase

El nombre de las constantes debe estar en mayúsculas.

Los nombres de los métodos deben declararse en notación camelCase (getUserOrders()).

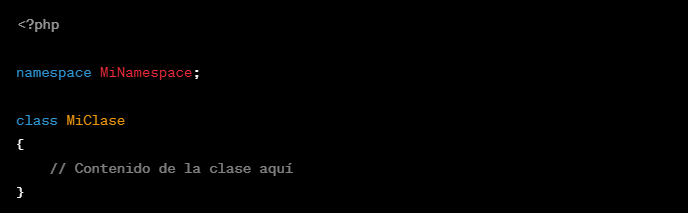
**Uso de tags PHP:**

Siempre utiliza <?php para abrir las secciones de código PHP.



**Declaración de namespaces y clases:**

Se debe utilizar namespace y class para organizar y definir la estructura del código.



**Nombres de archivos:**

El nombre del archivo debe coincidir con el nombre de la clase que contiene.

Ejemplo: Si la clase se llama MiClase, el archivo debería ser MiClase.php.

**PSR-2: Guía de estilo de codificación.**

El PSR-2 es un estándar de estilo de codificación para PHP que establece reglas para la legibilidad y el formato del código.

Es una extensión del PSR-1.

No se debe usar tabulaciones. En su lugar, se deben usar cuatro espacios para indentación.

No debe existir un límite obligatorio para la longitud de una línea. Aunque lo ideal es que midan ochenta columnas o menos y se considerará error de estilo sobrepasar las 120 columnas. Esto mejora la legibilidad del código al tener un menor recorrido que realizar en horizontal

Debe haber una línea en blanco después de la declaración del espacio de nombres y una línea después de todos los use. Esto separa de una manera lógica diferentes secciones de la cabecera de una clase.

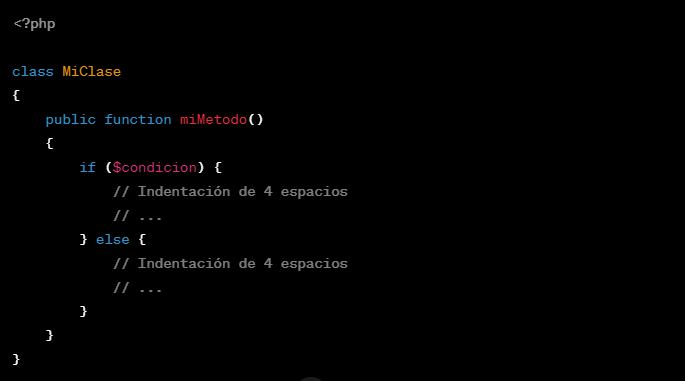
La apertura de llaves debe estar en la siguiente línea de la declaración de una clase, de una función o de un método, y el cierre justo la línea después del cuerpo. Hacer esto da importancia a la declaración de la clase y hace que se leer sin confusión entre líneas.

El paréntesis de apertura de las estructuras de control no deben tener espacio después de él. Del mismo modo, el paréntesis de cierre no debe tener espacio antes de él.

**Indentación y Espacios:**

Usar espacios para la indentación (4 espacios por nivel).

No mezclar espacios y tabulaciones para la indentación.



**Líneas en blanco:**

Dejar líneas en blanco para mejorar la legibilidad del código, pero no excederse en su uso.

No dejar líneas en blanco al final del archivo.

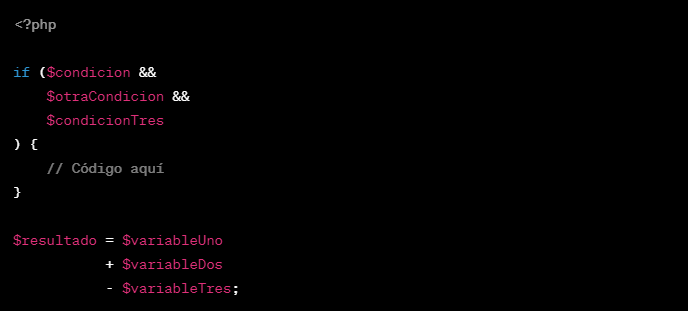


**Longitud de línea y Wrapping:**

Limitar las líneas a un máximo de 80 caracteres.

Si una línea excede este límite, debería envolverse de manera adecuada.

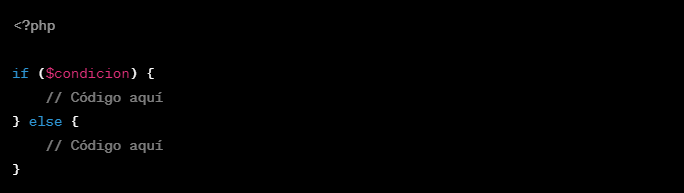
Operadores deben ir al inicio o final de línea, no mezclados.



**Uso de Llaves:**

Las llaves de apertura se colocan en la misma línea que la estructura de control.

Las llaves de cierre se colocan en una nueva línea.

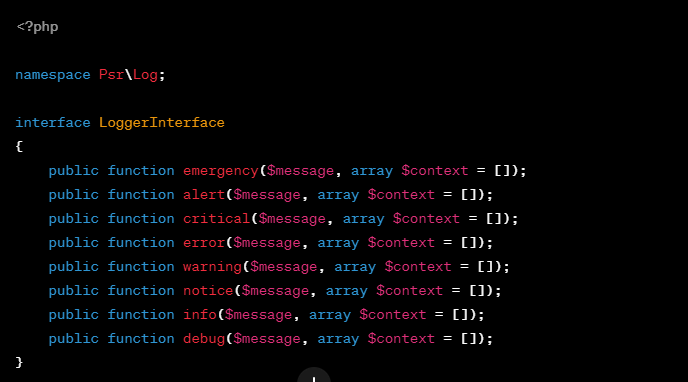


**PSR-3 - Interfaz de log**.

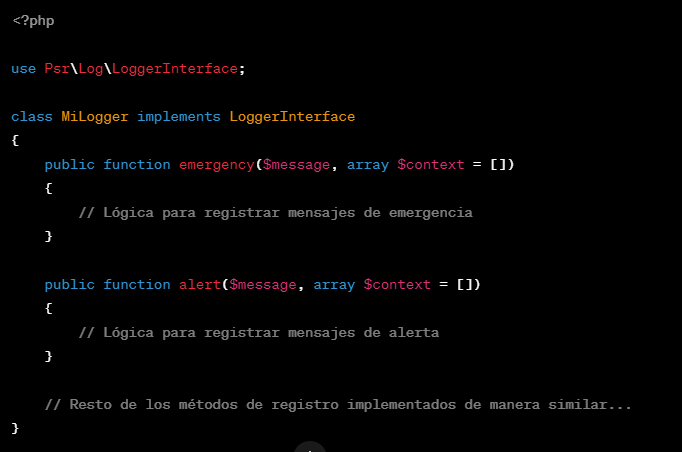
**El PSR-3** no es un estándar de estilo de codificación como el PSR-2, sino que se refiere a un estándar de registro de mensajes (logging) para PHP. El PSR-3 define una interfaz común para la implementación de sistemas de registro (logging) en aplicaciones PHP. Esta interfaz está diseñada para permitir la interoperabilidad entre diferentes bibliotecas de registro.

**El PSR-3** proporciona una interfaz LoggerInterface que define varios métodos comunes para registrar mensajes de registro con diferentes niveles de severidad, como debug, info, notice, warning, error, critical, alert y emergency.

Ejemplo de cómo se vería la interfaz **LoggerInterface:**



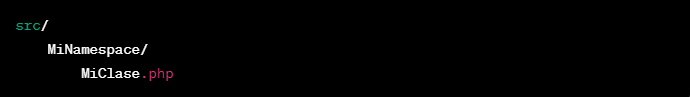
Implementación sencilla de un registrador (logger) que cumple con el estándar PSR-3 podría ser así:



**PSR-4 - Estándar de autocarga de clases.**

**El PSR-4** **(PHP Standard Recommendation 4)** define un estándar de autoloading que proporciona directrices para la carga automática de clases basadas en namespaces en PHP. Este estándar ayuda a organizar la estructura de directorios y archivos de una manera coherente y predecible, lo que facilita la ubicación de las clases por parte del autoloader.

**Estructura de directorios y archivos según PSR-4:**



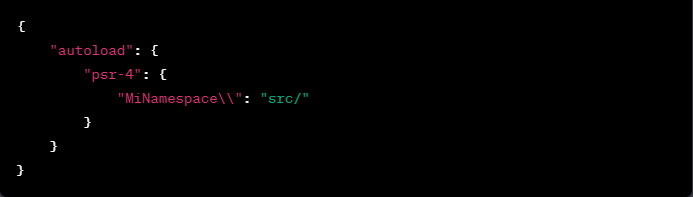
**Contenido del archivo MiClase.php siguiendo PSR-4:**



**Autoloading PSR-4:**

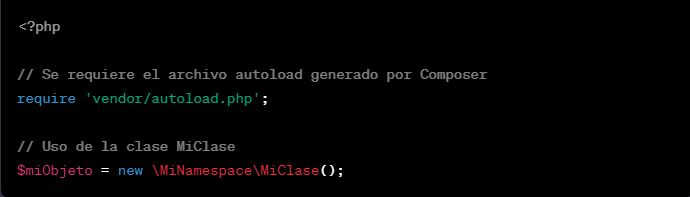
Utilización del autoloader que sigue las reglas de PSR-4 para cargar las clases automáticamente.

Para ello, se puede utilizar una función de autoloading o usar librerías como Composer, que facilitan la implementación del autoloading PSR-4. Por ejemplo, utilizando Composer, se podría tener un archivo composer.json similar a este:



Después de definir la configuración del autoloading en el archivo composer.json, se debe ejecutar el comando composer dump-autoload para que Composer genere el archivo de carga automática basado en las reglas PSR-4.

Siguiendo estas convenciones, el autoloader cargará automáticamente la clase MiClase cuando se intente utilizar en el código de la siguiente manera:



Esto permite que las clases sean cargadas automáticamente sin necesidad de requerir manualmente los archivos en el código, siguiendo la estructura de namespaces y directorios definida por PSR-4.

# CONCLUSIONES

* PHP ofrece una sintaxis sencilla y flexible que facilita el desarrollo web ágil y eficiente. Su estructura intuitiva permite a los programadores escribir y mantener código de manera más eficiente, lo que se traduce en un ahorro de tiempo y recursos en el desarrollo de nuestro proyecto.
* La amplia comunidad de desarrolladores de PHP es una ventaja significativa. Existen numerosos recursos, bibliotecas y frameworks disponibles que pueden acelerar el desarrollo y brindar soluciones probadas para desafíos comunes en el desarrollo web. Además, la comunidad activa proporciona soporte y actualizaciones constantes, lo que garantiza la seguridad y la evolución continua del proyecto.
* PHP es compatible con una amplia gama de sistemas operativos y servidores web, lo que lo hace altamente versátil y adaptable a diferentes entornos de implementación. Esto nos brinda flexibilidad y nos permite aprovechar infraestructuras existentes sin limitaciones significativas, facilitando la escalabilidad y el despliegue de nuestro proyecto en diversas plataformas.
* La capacidad de PHP para interactuar con bases de datos es una característica crucial para muchos proyectos web. PHP ofrece una amplia gama de extensiones y APIs para interactuar con sistemas de gestión de bases de datos populares, lo que nos permite almacenar y recuperar datos de manera eficiente y segura. Esta capacidad integrada de manejar la persistencia de datos es esencial para aplicaciones web dinámicas y con requisitos complejos.
* Las tendencias actuales en el mercado de desarrollo web muestran que PHP sigue siendo una opción relevante y demandada. A pesar de la aparición de nuevos lenguajes y tecnologías, PHP mantiene una base sólida de usuarios y goza de una gran demanda laboral. Esto significa que contar con profesionales capacitados en PHP y acceso a una amplia comunidad de soporte será más fácil, lo que garantiza el mantenimiento y la evolución futura de nuestro proyecto

# BIBLIOGRAFÍA

<https://coppeldev.github.io/php/standards/>

<https://www.php-fig.org/>

<https://blogprog.gonzalolopez.es/articulos/estandares-de-programacion-psr-en-php.html>

<https://styde.net/leccion-estandares-para-php/>