



Estándar de Codificación PHP

TEMPUS



Un Caso de Uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

Estos ilustran los requerimientos del sistema al mostrar como reacciona una respuesta a eventos que se producen en el mismo

Las Realizaciones de los Casos de Uso se llevan a cabo como resultado de un caso de uso específico. La realización del caso de uso debe cumplir con los requerimientos establecidos y debe reflejar el comportamiento de su caso de uso correspondiente. Este artefacto se halla dentro del Modelo de Diseño reflejando los productos de trabajo relacionados con el caso de uso pero que pertenecen a dicho modelo. Estos productos de trabajos relacionados consisten en los diagramas de comunicación y secuencia que expresan el comportamiento del caso del uso en términos de objetos de colaboración, y dichos diagramas deben elaborarse haciendo uso de (UML).



Tabla de contenido

[Descripción 4](#_Toc498023456)

[PSR-1 4](#_Toc498023457)

[Información General 4](#_Toc498023458)

[Archivos 4](#_Toc498023459)

[Etiquetas de PHP 4](#_Toc498023460)

[Codificación de caracteres 4](#_Toc498023461)

[Namespace y Nombres de clases 5](#_Toc498023462)

[Constantes de clase, propiedades y métodos 6](#_Toc498023463)

[Constantes 6](#_Toc498023464)

[Propiedades 6](#_Toc498023465)

[Métodos 6](#_Toc498023466)

[Precondiciones 7](#_Toc498023467)

Estándar de Codificación PHP

Descripción

El objetivo de esta guía es reducir los conflictos en el desarrollo y utilización del código generado por diferentes programadores. Se enumeran un conjunto de reglas y consejos sobre cómo generar el código en lenguaje PHP.

Las reglas de estilo que se mencionan en el presente documento derivan de PSR-2: Guía de estilo de codificación. PSR-2 se obtuvo de las similitudes de diversos proyectos. Una característica importante es que el beneficio no se encuentra en las reglas mismas, sino en el intercambio de las mismas. Es por ello, que se ha optado por este conjunto de directrices para el presente proyecto.

PSR-1

## Información General

* Los archivos deben usar solo etiquetas <?php y <?=.
* Los archivos deben usar solo UTF-8 sin BOM para código PHP.
* Los archivos deben declarar símbolos (clases, funciones, constantes, etc.) o causar efectos secundarios (generar una salida, cambiar una configuración, etc.) pero no ambas.
* Los nombres de clases deben ser declarados en StudlyCaps.
* Las constantes deben declararse en mayúsculas con separadores de subrayado.
* Los nombres de los métodos deben declararse en camelCase.

## Archivos

### Etiquetas de PHP

El código PHP debe usar las etiquetas largas <?php ?> o las etiquetas <?= ?> de “echo”, no debe usar las otras variaciones de etiquetas.

### Codificación de caracteres

EL código PHP debe usar soloUTF-8 sin BOM.

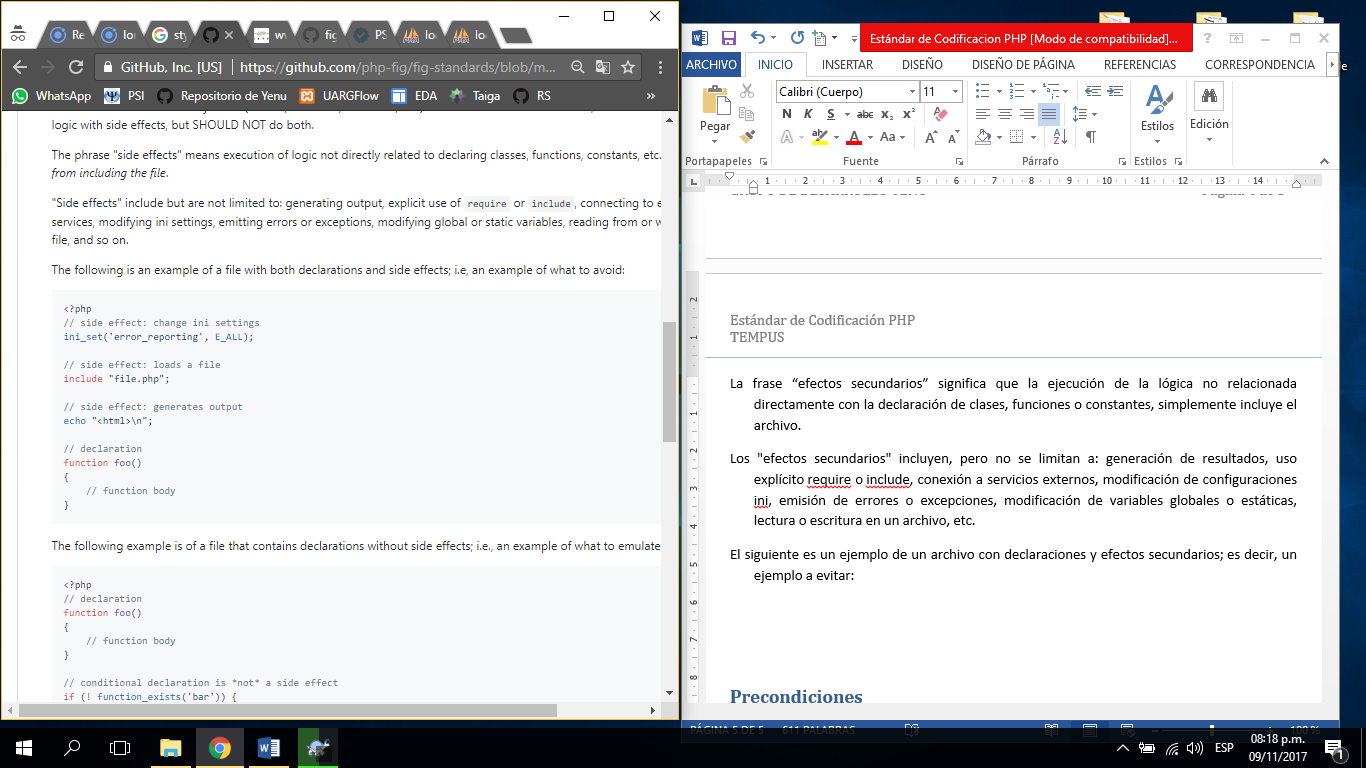
Efectos secundarios

Un archivo debería declarar nuevos símbolos como clases, funciones o constantes y no causar otros efectos secundarios. Deberia ejecutar la lógica con efectos secundarios, pero no hacer ambas cosas.

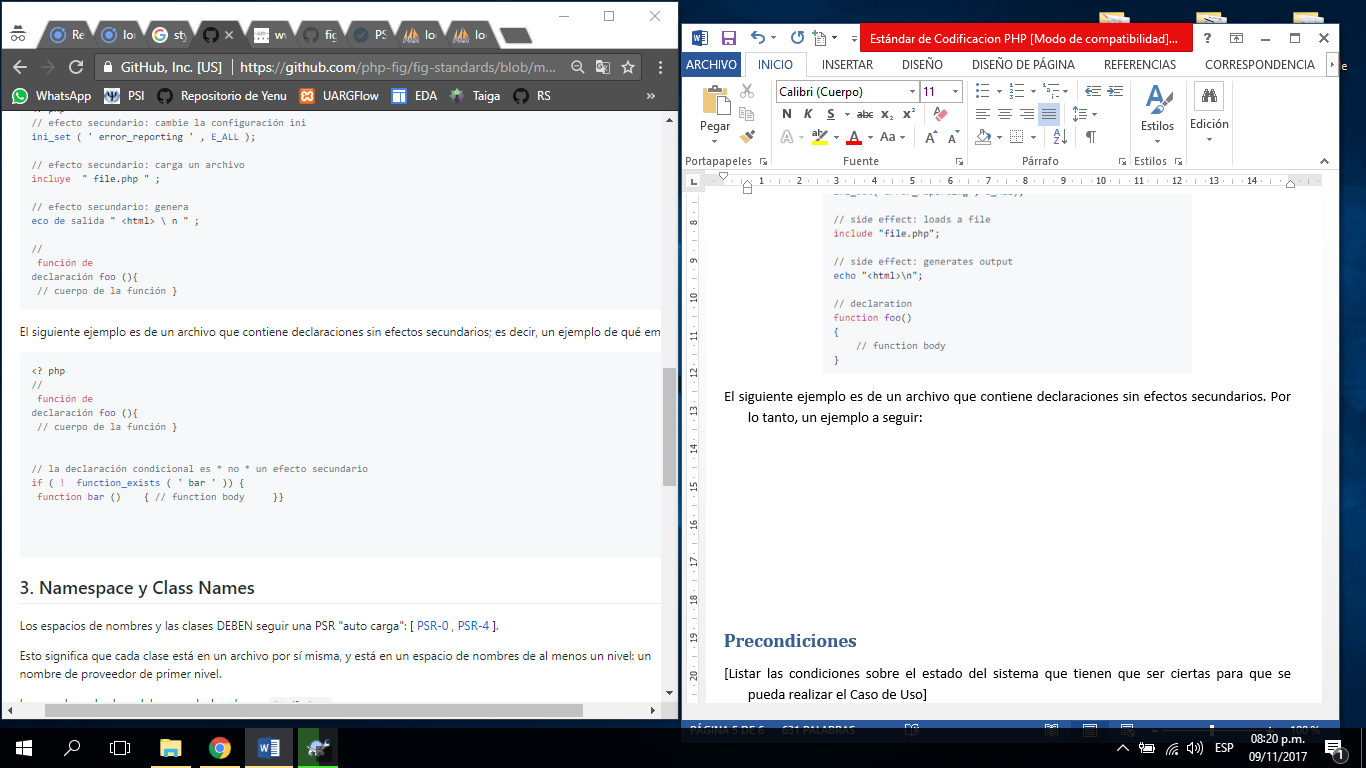
La frase “efectos secundarios” significa que la ejecución de la lógica no relacionada directamente con la declaración de clases, funciones o constantes, simplemente incluye el archivo.

Los "efectos secundarios" incluyen, pero no se limitan a: generación de resultados, uso explícito require o include, conexión a servicios externos, modificación de configuraciones ini, emisión de errores o excepciones, modificación de variables globales o estáticas, lectura o escritura en un archivo, etc.

El siguiente es un ejemplo de un archivo con declaraciones y efectos secundarios; es decir, un ejemplo a evitar:



El siguiente ejemplo es de un archivo que contiene declaraciones sin efectos secundarios. Por lo tanto, un ejemplo a seguir:



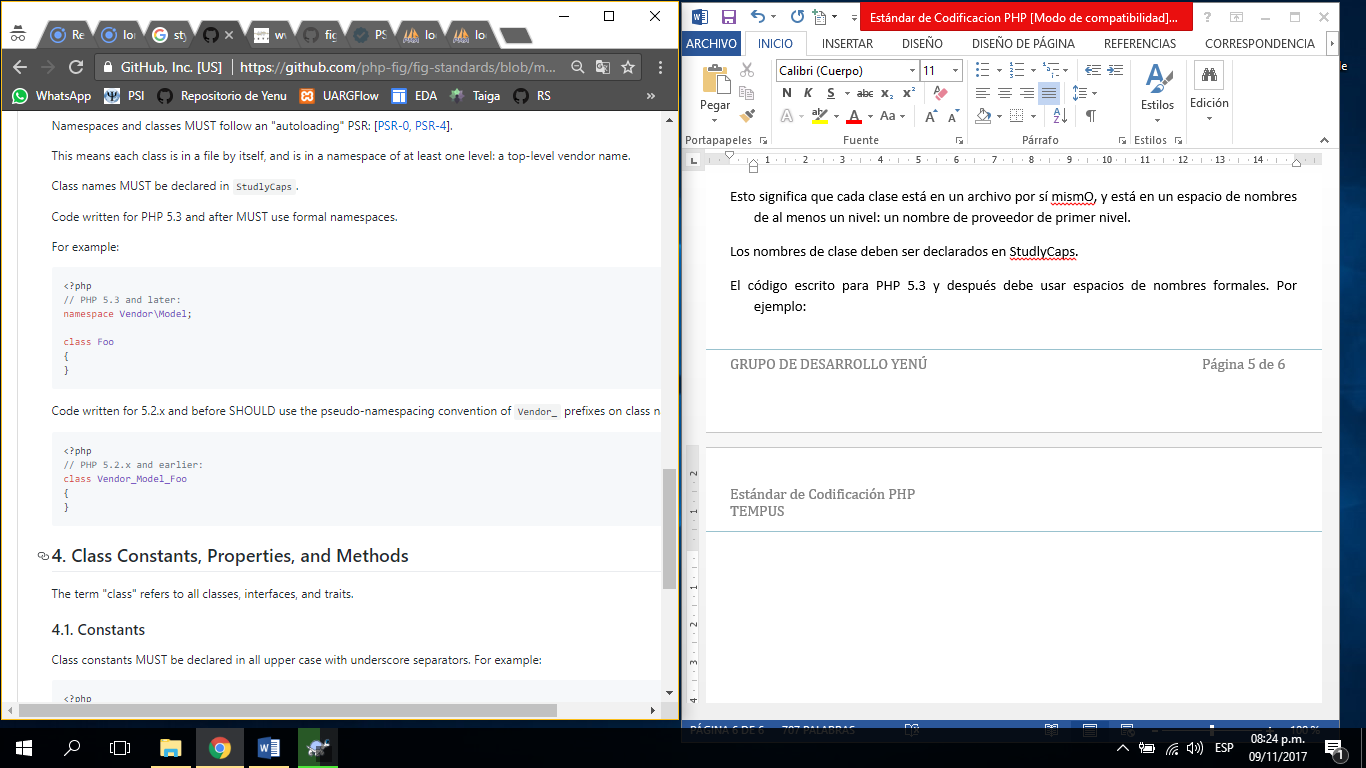
## Namespace y Nombres de clases

Los espacios de nombres y las clases deben seguir una PSR "autoloading": [[PSR-0](https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-0.md) , [PSR-4](https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-4-autoloader.md)].

Esto significa que cada clase está en un archivo por sí mismO, y ​​está en un espacio de nombres de al menos un nivel: un nombre de proveedor de primer nivel.

Los nombres de clase deben ser declarados en StudlyCaps.

El código escrito para PHP 5.3 y después debe usar espacios de nombres formales. Por ejemplo:

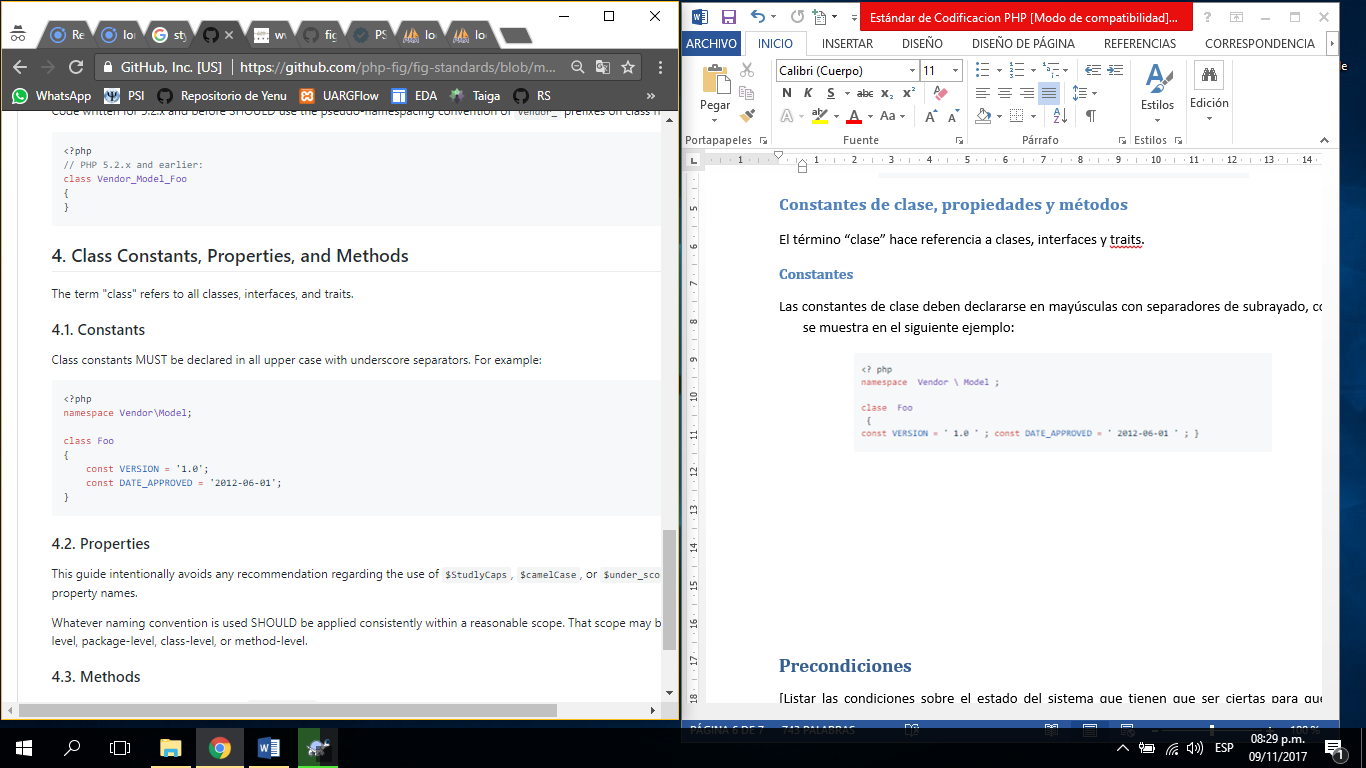


## Constantes de clase, propiedades y métodos

El término “clase” hace referencia a clases, interfaces y traits.

### Constantes

Las constantes de clase deben declararse en mayúsculas con separadores de subrayado, como se muestra en el siguiente ejemplo:



### Propiedades

En esta guía se evita intencionalmente alguna recomendación sobre el uso de $StudlyCaps, $camelCase o $under\_score para nombres de propiedades.

Cualquiera que sea la convención de nomenclatura que se use, deberá aplicarse de manera coherente dentro de un alcance razonable. Ese alcance puede ser a nivel de proveedor, a nivel de paquete, nivel de clase o nivel de método.

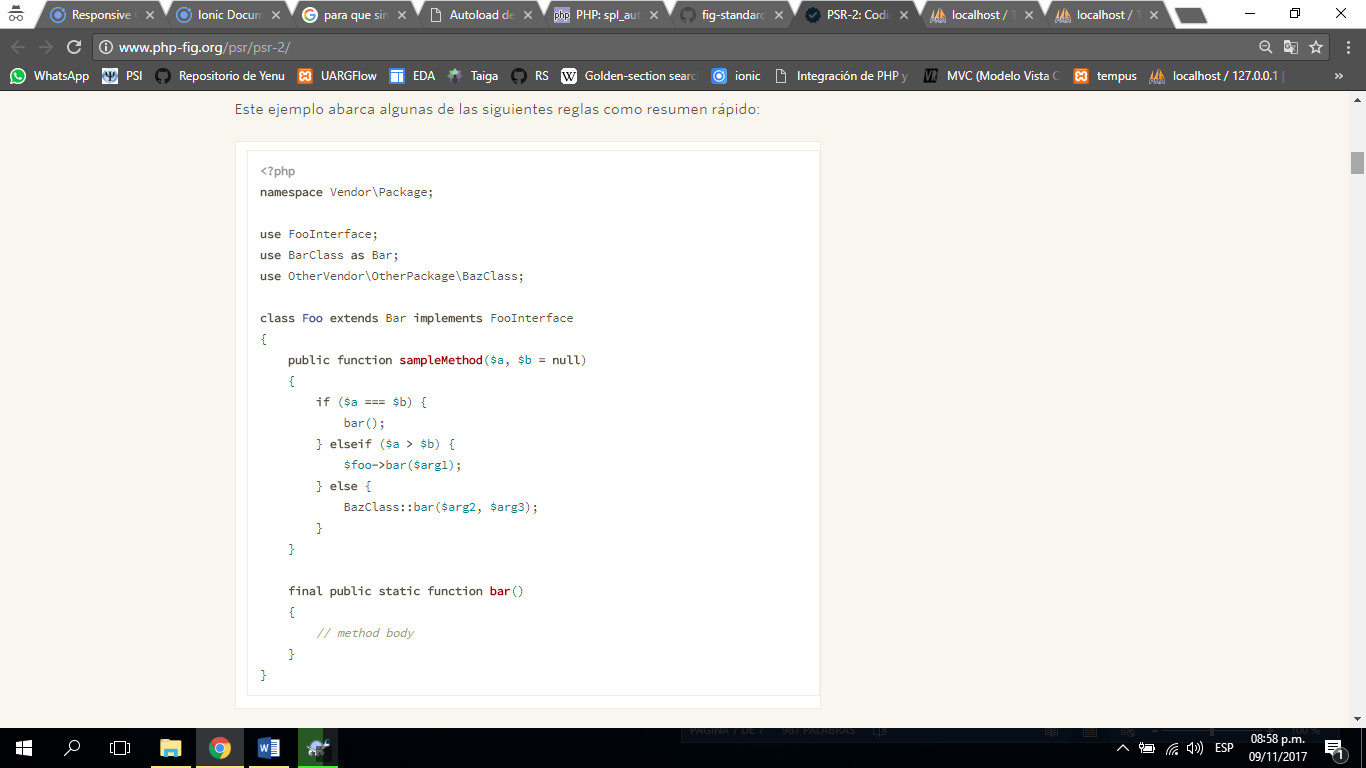
### Métodos

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

# PSR-2

## Visión de conjunto

* El código debe seguir la guía de estilo de codificación PSR-1.
* El código debe seguir 4 espacios para sangrar (sangría), no tabular.
* No debe haber un límite estricto en la longitud de la línea, el límite flexible debe ser de 120 caracteres. Las líneas deberían tener 80 caracteres o menos.
* Debe haber una línea en blanco después de la declaración del namespace. Además debe haber una línea en blanco después del bloque de declaraciones de uso.
* La apertura de las llaves para las clases deben ir en la siguiente línea, y las llaves de cierre deben pasar a la siguiente línea después del cuerpo.
* La visibilidad debe declararse en todas las propiedades y métodos. Las declaraciones ABSTRACT y FINAL deben hacerse antes de la visibilidad. STATIC debe declararse luego de la visibilidad.
* Las palabras clave de la estructura de control deben tener un espacio después de ellas, el método y llamadas de función no.
* Las llaves de apertura para estructuras de control deben seguir en la misma línea y las llaves de cierre deben pasar a la siguiente línea después del cuerpo.
* Al abrir paréntesis para las estructuras de control no deben tener un espacio después de ellos, y el cierre de paréntesis no debe tener espacio antes.



## Estándar básico de codificación

EL código debe seguir todas las reglas definidas en PSR-1

## Archivos

Todos los archivos PHP deben usar el final de línea UNIX LF.

Todos los archivos PHP deben terminar con una sola línea en blanco.

La etiqueta de cierre ?> debe omitirse de los archivos que solo contengan código PHP.

## Líneas

No debe haber un límite estricto en la longitud de la línea.

El límite flexible en la longitud debe ser de 120 caracteres. Las inspecciones automáticas deben advertir pero no deben indicar como error al límite flexible.

Las líneas no deberán tener más de 80 caracteres. Las líneas más largas deberán dividirse en múltiples líneas posteriores de no más de 80 caracteres cada una.

Se pueden agregar líneas en blanco para mejorar la legibilidad e indicar bloques de código relacionados.

No debe haber más de una declaración por línea.

## Sangría

El código debe usar una sangría de 4 espacios y no debe usar tabulación para ello.

Palabras clave

Las palabras clave de PHP deben estar en minúsculas.

Las constantes de PHP true, false y null deben estar en minúsculas.

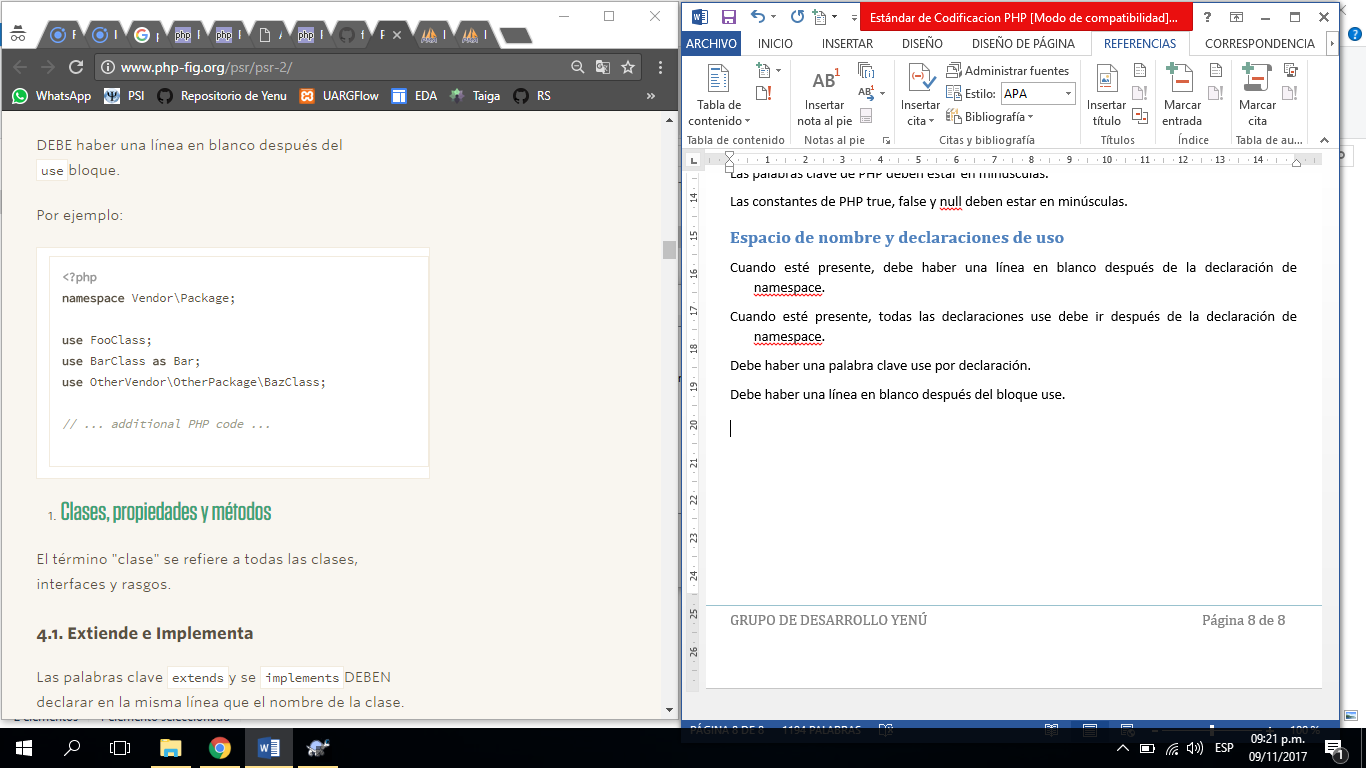
## Espacio de nombre y declaraciones de uso

Cuando esté presente, debe haber una línea en blanco después de la declaración de namespace.

Cuando esté presente, todas las declaraciones use debe ir después de la declaración de namespace.

Debe haber una palabra clave use por declaración.

Debe haber una línea en blanco después del bloque use.



## Clases, propiedades y métodos

Extiende e implementa