Reglas para una óptima implementación.

**Tabs o Espacios.**

En el contenido dentro de corchetes, siempre se nombrara este contenido con tabs. Cualquier editor considerable puede configurarse para poner tabs en vez de espacios en la asignación de nombres.

**Cabecera del archivo**

Siempre es importante que todos los archivos con la extensión .php inicien con una cabecera específica que indique información de la versión, autor de los últimos cambios, etc. Es de cada equipo decidir si se quiere o no agregar más datos.

**Comentarios en las funciones**

Todas las funciones deben tener un comentario, antes de su declaración, explicando que hacen. Ningún programador debería tener que analizar el código de una función para conocer su utilidad. Tanto el nombre como el comentario que acompañe a la función deben bastar para ello.

**Clases**

Las clases serán colocadas en un archivo .php aparte, donde sólo se colocará el código de la clase. El nombre del archivo será el mismo del de la clase y siempre empezará en mayúscula. En lo posible, procurar que los nombres de clase tengan una sola palabra.

Las clases siguen las mismas reglas de las funciones, por tanto, debe colocarse un comentario antes de la declaración de la clase explicando su utilidad.

**Hacks**

Los hacks que sea necesario colocar en el código deben, como las clases o funciones, ser comentados y en lo posible animar a otros programadores a reemplazarlos o mejorarlos por soluciones mejores.

**Ubicación de archivos**

En proyectos web o aplicaciones, generalmente se tendrán las siguientes carpetas:

* **/ Carpeta raiz:** Aquí irán los archivos .php a los que accede el usuario directamente, interfaz, etc.
* **clases:** Una carpeta conteniendo exclusivamente las clases usadas en el proyecto
* **includes:** Todos los archivos que sean llamados por otros .php en forma de módulos o de librerías de funciones.
* **db:** En caso de tener la posibilidad de usar varias bases de datos, aquí colocaremos los .php que manejen esas características multicapa para cada sistema de datos soportado.
* **templates:** En caso de usar un sistema de plantillas (Como smarty o el de phpBB), aquí guardaremos todos los archivos .tpl

**Estilo y reglas de escritura de código PHP**

**Nombres de variables**

Por más que parezca lo más "cool", se recomienda **no** adoptar la **notación hungara**en el código. Esta es aquella donde colocamos el tipo de datos antes del nombre de variable: *strNombre* para un string. En lo posible **NO** la usen, muchos grandes proyectos creen firmemente que es una de las tecnicas de ofuscación de código más ampliamente usadas en la actualidad.

Los nombres deben ser descriptivos y concisos. No usar ni grandes frases ni pequeñas abreviaciones para las variables. Siempre es mejor saber qué hace una variable con sólo conocer su nombre. Esto aplica para los nombres de **variables**, **funciones**, **argumentos de funciones** y **clases**.

Todos los nombres deben estar en minúscula (Excepto con las clases, donde la primera letra ha de ser mayúscula). En caso de usar más de una palabra, ésta será separada por un signo de underscore "**\_**".

En las **funciones**, es importante que el nombre denote su función inmediatamente. Cosas como *imprimir\_datos* están bien, pero estaría mejor*imprimir\_datos\_usuario*. De igual manera, en los **argumentos** de las funciones queremos saber inmediatamente que estamos usando. Es mejor*crear\_usuario($nick, $email)* que *crear($n, $e)*.

La filosofía es sencilla. No dañes la legibilidad del código por pereza. Por supuesto, aplica el sentido común y no crees funciones de más de 4 palabras.

**Corchetes o llaves. Donde colocarlas**

Aunque esto sea motivo de peleas constantes en los equipos de trabajo, lo mejor es seguir el camino que permita mayor claridad en el desarrollo. Para ponerlo en pocas palabras, todos los corchetes van en una línea propia.

**Poner espacios entre signos**

Otra cosa simple. Si tienes un signo binario, pon espacios a ambos lados. Tienes un signo unario, pon espacios a uno de sus lados. O en términos más simples, programa como si escribieras (bien) en español. Es algo muy sencillo que puede ayudar de gran manera en la lectura del código.

**Precedencia de operadores**

Puedes ser un programador *hardcore*, pero, ¿En serio te sabes la precedencia de operadores en PHP? Como creemos que no, lo mejor es siempre usar paréntesis para estar seguro. Básicamente, la idea es no dejar operaciones complejas a freaks matemáticos y estar seguros que nuestros compañeros en el equipo con menos “habilidad” comprendan todo sin problemas:

**Cadenas de texto entre comillas**

PHP tiene dos formas de poner strings o cadenas de texto. Con comillas simples y con comillas dobles. La diferencia es que si usas comillas dobles y metes dentro del texto un nombre de variable, el compilador lo interpretará y reemplazará por su valor. Por ésta razón **siempre** has de usar comillas simples **a menos** que necesites hacer la interpolación de variables que permiten las dobles. Es difícil acostumbrarse porque ocurre sólo en PHP (Y algunos otros más malignos, Perl4Life) pero por eso es PHP.

Por supuesto hay casos especiales donde es mejor usar dobles comillas (Como cuando usas caracteres de escape **\** intensivamente) así que siéntete con libertad de romper ésta regla cuando sea en pro de mejorar la lectura del código.

**Números dentro del código**

A veces ponemos números especiales dentro de nuestro código para situaciones especiales. **NO LO HAGAS**. Si necesitas poner un número especial, conviértelo en constante y entonces impleméntalo. Ejemplo:

**Operadores unarios de suma y resta.**

Es sencillo, úsalos en una sola línea y a la derecha, ejemplo:

**Condicionales de una sola línea**

Úsalas sólo para hacer cosas simples. En lo posible siempre utiliza if

**No uses variables sin inicializar**

Si no tienes control sobre el valor de una variable, verifica que esté inicializada. Esto lo permite PHP de la siguiente manera:

Pero **sólo** usa está opción cuando **no** tengas el control o no estés seguro que valor pueda tener la variable (Como en variables que llegan por parámetro o por GET, etc)

**Instrucción “switch”**

Esta es una de las cosas más feas de los lenguajes con estilo de C. Cuando debas usarla.

Fuente: <http://www.cristalab.com/tutoriales/reglas-de-codificacion-y-lineamientos-de-codigo-php-c190l/>