Normas de estandarización

Base de datos

Nomenclatura

Todas las tablas deberán tener un nombre escrito completamente en mayúsculas, y siempre en singular. No se admitirá, por ejemplo, una tabla llamada “`Ventas`”, debería ser “`VENTA`”.

Las columnas de cada una de las tablas seguirán una nomenclatura camel case, con la primera letra en mayúscula. Por ejemplo, puede existir una columna llamada “`idVenta`”, pero no se admitirá “`id_venta`” o ”`idventa`”.

Aspectos a considerar

Se prefiere la existencia de una única clave primaria en cada tabla. Es decir, que la clave primaria está conformada por tan solo una columna. No es obligatorio seguir esta norma, pero es recomendable puesto que facilitará la posterior codificación del sistema en PHP.

Se deberá hacer uso de claves secundarias siempre que sea necesario, y éstas deberán tener especificado el modo de borrado/actualización necesario cuando se modifique la tabla original (CASCADE, SET NULL…).

Se preferirá el uso de enums frente al uso de un integer para denotar un tipo. Por ejemplo, para marcar que un usuario es administrador o no, podría utilizarse un valor entero donde 0 se corresponde a usuario y 1 a administrador, en este caso se preferirá la existencia de un enum donde se definan los valores ‘administrador’ y ‘usuario’.

Código PHP

Indentación

Todo el código PHP debe estar correctamente indentado. Una indentación debe consistir de 4 espacios. El uso de tabuladores no está permitido.

Se recomienda la configuración del editor de texto utilizado acorde a esta norma, para que cuando se pulse la tecla tabulador se traduzca automáticamente a 4 espacios, y para que las indentaciones automáticas también se realicen con 4 espacios.

Máxima longitud de línea

La longitud máxima para cualquier línea de código será de 120 caracteres. La longitud recomendada es de 80 caracteres. Se deberá intentar mantener las líneas de código por debajo del límite recomendado, y sólo bajo circunstancias excepcionales se permitirá el sobrepaso del límite máximo.

Terminación de línea

La terminación de línea seguirá las convenciones Unix. Esto es, las líneas deben terminar única y exclusivamente con un único carácter de nueva línea (linefeed, LF). No se permitirá la terminación de línea de las convenciones de Apple (carriage return, CR) ni las de Windows (carriage return y linefeed, CRLF).

Se recomienda configurar el editor utilizado para que se adapte a esta norma.

Delimitación de bloques (llaves)

Las llaves de apertura y cierre de bloques deberán escribirse en sus propias líneas siempre que se correspondan con bloques de clases, métodos o funciones. En cualquier otro caso las llaves de apertura deberán estar en la misma línea que la expresión que abre el bloque (ifs, fors, whiles…).

Ejemplo:

```
class MiClase
{
    public function miFuncion($variable)
    {
        if (isset($variable)) {
            print $variable;
        }
    }
}
```

Demarcación

Todo código PHP debe estar delimitado siempre por la forma completa de los tags estándar de PHP:

```
1. <?php
2.
3. ?>
```

Los tags reducidos no están permitidos bajo ningún concepto. Debe tenerse en cuenta que, en un fichero que cuente solamente con código PHP, tras el tag de cierre `?>` no debe incluirse **nunca** y bajo ningún concepto una nueva línea en blanco.

Cadenas de texto

Toda cadena de texto en PHP debe estar delimitada siempre por comillas dobles, el uso de comillas simples no será aceptado.

```
1. $a = “Ejemplo de String”;
```

La sustitución de variables dentro de una cadena se admitirá siempre bajo estas formas:

```
1. $saludo = “Hola $nombre, bienvenido”;
2.
3. $saludo = “Hola {$nombre}, bienvenido”;
```

Y nunca, por consistencia, bajo la forma alternativa:

```
1. $saludo = “Hola ${nombre}, bienvenido”;
```

La concatenación de cadenas debe realizarse siempre con el operador “**.**”, que debe estar siempre precedido y seguido de un espacio.

```
1. $asignatura = “Interfaces” . “ de “ . “Usuario”;
```

Clases

Las clases seguirán una nomenclatura camel case con la primera letra en mayúsculas. Los métodos y atributos seguirán también una nomenclatura camel case, pero con la primera letra en minúsculas.

```
class NombreClase
{
      private $atributoUno;

      public funciton nombreFuncion()
      {
          ...
      }
}
```

Todo método y atributo dentro de una clase debe declarar su visibilidad, siendo ésta una de: public, protected o private. La declaración de atributos como var, sin especificar su visibilidad, no estará permitida.

La estructura de directorios, ficheros y namespaces seguirá la normativa PSR-0 para la correcta carga automática de las mismas. Es decir, cada clase deberá estar definida en un fichero con el mismo nombre de la clase (respetando mayúsculas), y siempre que se defina dentro de un namespace, dicho fichero .php deberá almacenarse dentro de un directorio con el nombre del namespace.

Por ejemplo, si definimos la clase `MiClase` dentro del namespace `miespacio`, definido a su vez dentro del namespace `espacios`, el fichero `.php` que define la clase deberá encontrarse, partiendo de la raíz del directorio de código, en:

```
/espacios/miespacio/MiClase.php
```