Buenas Prácticas

Claudio Flores Sapiain

Ingeniero Informático

Scrum Master

Ingeniero DevOps

2020

Introducción

Para el desarrollo de un proyecto de software, se recomienda seguir con una serie de buenas prácticas que a la larga facilitarán entender y mantener el software.

Buenas Prácticas

1.- Escribir un programa lo más simple posible.

2.- Documentar el código fuente las funciones de entrada y salida, todo esto debe ser visible sólo en el ambiente de desarrollo y certificación.

3.- Mantener un control de versiones, para ellos se debe definir un programador líder.

Buenas Prácticas

4.- Se debe utilizar el sistema de versionamiento GIT, SubVersion, Bitbucket, etc.

5.- El versionamiento de un proyecto solo debe ser visible en los ambientes de desarrollo y certificación.

6.- No hacer uso de código redundante.

Buenas Prácticas

7.- El tamaño de las sangrías deben ser consistentes y no debe variar a lo largo del código, esto facilitará la lectura del código.

8.- No hacer uso de variable que no contengan algún significado descriptivo.

9.- Se recomienda declarar las variables en líneas separadas y utilizar comentarios descriptivos para cada una.

Buenas Prácticas

11.- Evitar el código comentado para que no se ejecute, debido a que la lectura puede ser engorrosa.

12.- Se recomienda que al inicio de cada función, un bloque de comentarios que explique el comportamiento general de la función.

Buenas Prácticas

13.-Si utiliza operadores (+, -, &&) agregar espacio al inicio y fin de su uso.

14.-Se recomienda evitar la incorporación de más de una instrucción por línea.

15.- Se recomienda priorizar el uso de librerías locales por sobre las librerías remotas. *EJ.: JQuery*

Buenas Prácticas

16.-Todas las aplicaciones creadas deben pasar por un chequeo básico de seguridad y desempeño.

17.-Utilizar como estándar desarrollos responsive.

18.-Utilizar el manejo de excepciones en todos los lenguajes (try-catch).

Buenas Prácticas

19.-Se debe considerar el registro de todas las acciones en archivos de log, esta información podrá ser utilizada en la recuperación de información, incluyendo las instancias en donde no se almacenó información en la base de datos.

20.-Se debe controlar el crecimiento y la propagación de los archivos log, para evitar colapso en el disco.

21.- Si se trabaja con un compilador, ajustar sus opciones para que arroje la mayor cantidad de errores y advertencias posibles al compilar.

Buenas Prácticas

22.-Para desarrollos web, se debe realizar la integración Google Analitics.

23.-También se debe considerar la integración de técnicas de SEO.

24.- Definición de variables locales al inicio de la implementación de cada función

Buenas Prácticas PHP

25.-No utilizar etiquetas cortas, esto evitará posible problemas de portabilidad.(<?php ?>)

26.-Echo y Print realizan la misma función, aunque se debe tener en consideración que echo es más rápida que print, ya que echo puede imprimir más de una cadena separada por comas, en cambio print solo imprime un cadena.

```
print 'Hola';
echo 'Hola', 'Hola de nuevo';
```

Buenas Prácticas PHP

27.-Cuando se trabaje con cadenas, se debe evitar el uso de comillas dobles, debido que esto aumenta el tiempo de ejecución, para realizar esto, se recomienda utilizar echo y concatenar las cadenas con comas.

EJ.: *echo 'Hola', $nombre, ', ¿qué te trae por aquí?*

28.-La codificación de caracteres se debe utilizar codificación UTF8.

Buenas Prácticas PHP

30.- Las constantes de las clases deben declararse en mayúscula con guiones bajos como separadores. *EJ: CONSTANTE_DE_CLASE.*

31.-Se recomienda la programación orientada a objetos, esto permitirá separar las aplicaciones en secciones y también ayudará a simplificar el código evitando la redundancia y haciéndolo más eficiente.

Buenas Prácticas PHP

32.- Se recomienda prevenir ataques en los formularios de login, para lograr eso se debe inicializar la variable, así se evitan ataques del tipo XSS (Cross Site Scripting).

```php
<?php
if (correct_user($_POST['user'], $_POST['password']) {
    $login = true;
}

if ($login) {
    forward_to_secure_environment();
}
?>
```

Buenas Prácticas PHP

33.- Encriptar contraseñas, se recomienda el uso del método **bcrypt.**

34.-Minimizar el número de variables globales.

35.- Optimizar consultas a bases de datos(DBA).

36.- Toda variable traspasada entre PHP debe ser en POST y recuperadas en POST.

Buenas Prácticas PHP

33.- Utilizar un único archivo de configuración que contenga todos los datos configurables que puedan variar de un ambiente de desarrollo a uno de producción. Ej: las rutas base, conexiones a bases de datos, url de WS, etc.

34.-Utilizar un usuario de base de datos específico para la aplicación.

35.- Agregar y utilizar librerías de protección sobre XSS.

Buenas Prácticas JAVA

36.- Evitar la creación innecesaria de objetos, esto debido al gran uso de memoria, esto se puede evitar inicializando objetos solo cuando serán útiles en el código.

37.-No hacer variables de instancias públicas, ya que son más propensas a ser modificadas por otros y generar errores, para esto una de las mejores prácticas es declarar variables como privadas y crear accesos set y get.

38.- Considerar siempre desarrollar teniendo en cuenta los principios básicos de desarrollo orientado a objetos: Abstracción, encapsulado, herencia, polimorfismo.

Buenas Prácticas JAVA

39.- Tratar de usar librerías estándares en lugar de hacer una desde cero, esto permite optimizar tiempo, reducir los errores, y además de hacer el código más legible y fácil de mantener.

40.-Siempre regresar arreglos vacíos y no nulos.

41.- Usar New solo cuando sea necesario, ya que si el objeto creado no llega a utilizarse, consume espacio de memoria sin ser necesario.

Buenas Prácticas JAVA

42.- Evitar la creación de objetos temporales innecesarios, ya que estos consumen recursos y aumentan el trabajo de limpieza al Garbage Collector.

43.-Utilizar librería de logs log4j en donde se registre los principales eventos para información de debug en ambientes de desarrollo y producción.

44.- Verificar un correcto uso de excepciones, utilizando el tipo adecuado y especificando la mayor cantidad de información para determinar un posible error no considerado en el flujo.

Buenas Prácticas JAVA

45.- Ojo con los bucles de gran manejo de información, evita la sobre carga de control de flujo en cada iteración.

46.-Documentar funciones en ambiente de desarrollo y certificación.

47.- Hacer uso de los versionamiento en desarrollo y certificación.

Preguntas...