# Mejores prácticas y mejoras recomendadas para plugins de WordPress

Este documento resume las principales mejoras que deben aplicarse a un plugin de WordPress para cumplir con los estándares de calidad, seguridad y compatibilidad. Está pensado como una guía de referencia para futuros proyectos.

## 1. Verificación de seguridad en cada archivo PHP

**Objetivo:** Evitar el acceso directo a archivos PHP.

**Cómo:** Incluir la siguiente línea al inicio de cada archivo .php del plugin:

defined('ABSPATH') or die('Acceso no permitido.');

**Dónde:** Al principio de todos los archivos PHP, incluyendo controladores, modelos, vistas y archivos de funciones.

## 2. Verificación de nonce en formularios (protección CSRF)

**Objetivo:** Asegurar que los formularios sólo puedan ser enviados desde el panel del administrador por un usuario con sesión activa.

**Cómo:**

1. Incluir un campo de seguridad en el formulario:

wp\_nonce\_field('mi\_plugin\_nonce\_action', 'mi\_plugin\_nonce\_field');

1. Verificar en el procesamiento:

if (!isset($\_POST['mi\_plugin\_nonce\_field']) || !wp\_verify\_nonce($\_POST['mi\_plugin\_nonce\_field'], 'mi\_plugin\_nonce\_action')) {

die('Petición no válida');

}

**Dónde:** En todos los formularios del plugin (en la vista) y en los controladores que procesan la información.

## 3. Archivos index.php en carpetas internas

**Objetivo:** Prevenir el listado del contenido de los directorios del plugin si el servidor no tiene deshabilitada la opción de autoindex.

**Cómo:** Crear un archivo index.php en cada subdirectorio con este contenido:

<?php

// Silencio de seguridad

**Dónde:** En todas las carpetas del plugin, como:

* /includes
* /model
* /controller
* /views
* /assets/css
* /assets/js

## 4. Archivo de desinstalación (uninstall.php)

**Objetivo:** Limpiar los datos creados por el plugin al ser desinstalado.

**Cómo:** Crear un archivo uninstall.php en la raíz del plugin:

if (!defined('WP\_UNINSTALL\_PLUGIN')) {

die;

}

global $wpdb;

$wpdb->query("DROP TABLE IF EXISTS {$wpdb->prefix}mi\_tabla\_personalizada");

**Dónde:** Raíz del plugin, junto al archivo principal.

**Nota:** Asegurarse de que la tabla esté prefijada correctamente y que no se eliminen datos sensibles innecesariamente.

## 5. Escapado de datos de entrada y salida

**Objetivo:** Evitar XSS e inyecciones de HTML o JavaScript.

**Cómo:**

* Al mostrar datos en HTML: esc\_html(), esc\_attr(), esc\_url()
* Al almacenar datos: sanitize\_text\_field(), intval(), sanitize\_email(), según el tipo de dato

**Dónde:**

* En la salida de datos en las vistas (HTML)
* En la validación de datos antes de guardar en la base de datos

## 6. Uso correcto de capacidades de usuario

**Objetivo:** Restringir el acceso a la configuración del plugin a usuarios autorizados.

**Cómo:**

add\_menu\_page(

'Mi Plugin',

'Mi Plugin',

'manage\_options', // Solo administradores

'mi-plugin',

'mi\_plugin\_admin\_page'

);

**Dónde:** En la función que registra el menú de administración.

## 7. Internacionalización (i18n)

**Objetivo:** Permitir traducción del plugin a otros idiomas.

**Cómo:** Envolver textos en funciones \_\_() o \_e():

echo \_\_('Texto traducible', 'mi-plugin');

**Dónde:** En cualquier lugar donde se muestren cadenas al usuario.

**Adicional:** Crear archivo de texto .pot con Poedit o WP-CLI para facilitar traducciones.

## 8. Archivo README.md informativo

**Objetivo:** Informar al usuario sobre el uso, instalación y funcionalidades del plugin.

**Cómo:** Estructura recomendada:

# Nombre del Plugin

## Descripción

Breve explicación del objetivo del plugin.

## Instalación

Pasos para instalar manualmente.

## Uso

Shortcodes disponibles, configuraciones, etc.

## Licencia

Tipo de licencia (GPL v2 o superior).

**Dónde:** En la raíz del plugin.

## 9. Cabecera del plugin

**Objetivo:** Permitir que WordPress detecte y registre el plugin.

**Cómo:** Al inicio del archivo principal PHP:

/\*

Plugin Name: Nombre del Plugin

Description: Descripción breve.

Version: 1.0

Author: Tu Nombre

\*/

**Dónde:** En el archivo principal, como mi-plugin.php en la raíz.

## 10. Estilo modular de organización

**Objetivo:** Separar responsabilidades y facilitar el mantenimiento.

**Cómo:** Crear subcarpetas:

* /controller: para la lógica de procesamiento
* /model: para el acceso a la base de datos
* /views: para formularios y plantillas HTML
* /includes: funciones auxiliares y constantes
* /assets/css, /assets/js: recursos estáticos

**Dónde:** Dentro de la carpeta principal del plugin.

Este conjunto de mejoras garantiza que cualquier plugin desarrollado sea más seguro, estructurado y profesional. Se recomienda revisar esta guía al inicio y al final de cada desarrollo para validar que se cumplan estos puntos.