1. INTRODUCCIÓN AL DESARROLLO DE PLUGINS

* Ejemplo: "Desarrollar un plugin para WordPress es una manera poderosa de extender la funcionalidad del CMS. Basándome en mi experiencia, comencé creando un plugin sencillo para añadir un widget personalizado usando las funciones básicas de WordPress. En la documentación no está, pero descubrí que un buen método es estructurar el código utilizando un patrón de diseño como el Modelo-Vista-Controlador (MVC) para separar la lógica y mejorar la escalabilidad del plugin."

2. LO BÁSICO DE UN PLUGIN

* Ejemplo: "Un plugin en WordPress es un conjunto de funciones agrupadas que pueden ser añadidas a un sitio web para ampliar sus capacidades. Mi primer plugin constaba de un archivo principal PHP y un archivo README. En la documentación no está, pero descubrí que un buen método es utilizar un cargador de clases (autoloader) para organizar mejor el código y facilitar su mantenimiento a medida que el plugin crece."

3. SEGURIDAD DE PLUGIN

* Ejemplo: "La seguridad es crucial en el desarrollo de plugins. Inicialmente, usé sanitize\_text\_field() para sanitizar datos de formularios, pero en la documentación no está cómo manejar entradas JSON de forma segura. Descubrí que un buen método es usar wp\_json\_encode() y wp\_json\_decode() para asegurar que los datos se procesen correctamente, minimizando riesgos de inyecciones."

4. HOOKS

* Ejemplo: "Los hooks permiten a los desarrolladores interactuar con WordPress sin modificar el núcleo. Usé add\_action para insertar contenido adicional en el footer de un sitio. En la documentación no está, pero descubrí que un buen método es usar has\_action() para verificar si un hook ya está ocupado antes de añadir la acción, lo que ayuda a evitar conflictos con otros plugins."

5. PRIVACIDAD

* Ejemplo: "Los plugins deben manejar los datos del usuario de manera ética y legal. En un plugin que desarrollé, añadí un checkbox de consentimiento usando wp\_create\_nonce. En la documentación no está, pero descubrí que un buen método es implementar una página dedicada a la gestión de la privacidad dentro del área de administración del plugin, donde los usuarios puedan configurar cómo se manejan sus datos."

6. MENÚS DE ADMINISTRACIÓN

* Ejemplo: "Crear menús personalizados en el área de administración mejora la usabilidad de un plugin. Usé add\_menu\_page para agregar un menú, pero en la documentación no está cómo manejar permisos avanzados. Descubrí que un buen método es utilizar current\_user\_can() para validar las capacidades de los usuarios antes de mostrarles opciones específicas, mejorando la seguridad y personalización."

7. SHORTCODES

* Ejemplo: "Los shortcodes permiten a los usuarios insertar funciones complejas con facilidad. Desarrollé un shortcode para mostrar una galería de imágenes usando add\_shortcode. En la documentación no está, pero descubrí que un buen método es combinar shortcodes con bloques de Gutenberg usando register\_block\_type() para ofrecer una experiencia de usuario más moderna y visual."

8. OPCIONES DE PLUGINS

* Ejemplo: "Las opciones permiten a los usuarios personalizar los plugins sin tocar el código. Implementé una página de opciones usando register\_setting, pero en la documentación no está cómo gestionar múltiples opciones relacionadas de manera eficiente. Descubrí que un buen método es agrupar opciones en arrays y guardarlas como una única entrada en la base de datos, lo que reduce las consultas y mejora el rendimiento."

9. METADATA

* Ejemplo: "Los metadatos enriquecen los posts y páginas. En un plugin de eventos, añadí campos personalizados usando add\_post\_meta. En la documentación no está cómo optimizar la carga de metadatos. Descubrí que un buen método es utilizar get\_post\_meta() con la opción de caché activada para reducir la carga en la base de datos y mejorar la velocidad del sitio."

10. POST TYPES

* Ejemplo: "Custom Post Types permiten organizar contenido de manera específica. Utilicé register\_post\_type para crear un nuevo tipo de contenido para testimonios. En la documentación no está, pero descubrí que un buen método es usar post\_type\_supports() para asegurar que el Custom Post Type incluya todas las características necesarias, como campos personalizados o revisiones."

11. TAXONOMIA

* Ejemplo: "Las taxonomías personalizadas son útiles para clasificar contenido. Implementé una taxonomía personalizada en un plugin de recetas usando register\_taxonomy. En la documentación no está cómo relacionar taxonomías de manera dinámica. Descubrí que un buen método es utilizar register\_taxonomy\_for\_object\_type() para asociar dinámicamente taxonomías con diferentes tipos de contenido según sea necesario."

12. ROLES DE USUARIOS

* Ejemplo: "Los roles de usuario y sus capacidades deben ser gestionados cuidadosamente. Añadí un nuevo rol con permisos específicos usando add\_role, pero en la documentación no está cómo manejar la actualización de roles. Descubrí que un buen método es utilizar add\_cap() y remove\_cap() en un script de actualización para ajustar las capacidades de los roles existentes sin necesidad de eliminarlos y recrearlos."

13. API - REST/HTTP

* Ejemplo: "La API REST de WordPress permite interactuar con el CMS de manera programática. Utilicé register\_rest\_route para desarrollar un plugin que sincroniza productos entre un sitio WooCommerce y una app móvil. En la documentación no está cómo manejar errores personalizados. Descubrí que un buen método es usar rest\_ensure\_response() para devolver respuestas de error con códigos HTTP específicos y mensajes detallados, mejorando la depuración y la experiencia del usuario."

14. JAVASCRIPT

* Ejemplo: "JavaScript permite crear interfaces de usuario interactivas. Añadí un script en un plugin para validar formularios antes de enviar los datos al servidor. En la documentación no está cómo manejar dependencias de scripts eficientemente. Descubrí que un buen método es utilizar wp\_register\_script() y luego wp\_enqueue\_script() con la opción de dependencias, asegurando que todos los scripts se carguen en el orden correcto."

15. TAREAS CRON

* Ejemplo: "Las tareas cron automatizan procesos repetitivos. En un plugin de newsletter, configuré una tarea cron usando wp\_schedule\_event para enviar correos automáticamente a suscriptores cada semana. En la documentación no está cómo manejar tareas fallidas. Descubrí que un buen método es implementar un sistema de registro (logging) que almacene los resultados de cada tarea cron, lo que facilita la identificación y resolución de problemas."

16. INTERNACIONALIZACIÓN

* Ejemplo: "La internacionalización permite que los plugins sean traducidos a múltiples idiomas. En mi último plugin, añadí soporte para la traducción usando \_\_() y load\_plugin\_textdomain(). En la documentación no está cómo manejar la localización de formatos de fecha y hora. Descubrí que un buen método es utilizar wp\_date() en lugar de date() para que los formatos se ajusten automáticamente a la configuración regional del usuario."

17. USO DE SVN Y WORDPRESS DIRECTORY

* Ejemplo: "SVN es el sistema de control de versiones utilizado por el repositorio de WordPress. Mi experiencia subiendo un plugin al repositorio me enseñó a manejar las ramas y los tags. En la documentación no está cómo automatizar la liberación de versiones. Descubrí que un buen método es crear scripts bash para automatizar el etiquetado y la subida de archivos, lo que agiliza el proceso y reduce errores."

18. HERRAMIENTAS DE DESARROLLO

* Ejemplo: "Existen diversas herramientas que facilitan el desarrollo de plugins. Utilicé WP-CLI para automatizar la creación de un entorno de desarrollo y PHPUnit para realizar pruebas unitarias. En la documentación no está cómo integrar estas herramientas en un flujo de trabajo CI/CD. Descubrí que un buen método es configurar un pipeline en herramientas como GitHub Actions o GitLab CI para automatizar las pruebas y el despliegue, asegurando una mayor calidad y consistencia en el desarrollo."