Unidad 3

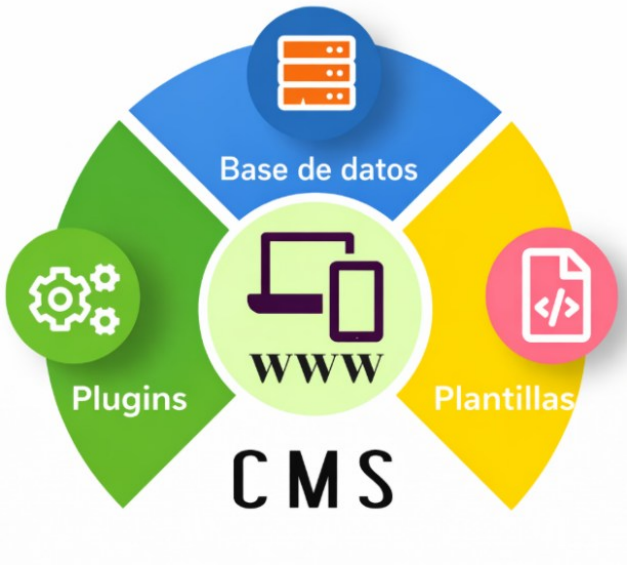
Despliegue y administración de aplicaciones web

Una vez preparado el entorno, el siguiente paso consiste en desplegar y administrar aplicaciones web reales. En el ámbito profesional, muchas organizaciones utilizan gestores de contenidos y plataformas corporativas que permiten implementar soluciones web sin necesidad de programación avanzada. La correcta instalación y configuración de estas herramientas exige conocimientos técnicos precisos y una metodología organizada.

En esta unidad se estudian los sistemas de gestión de contenidos, con especial atención a su proceso de instalación, configuración, ampliación mediante extensiones y mantenimiento. Asimismo, se analizan otros tipos de aplicaciones web corporativas, como plataformas educativas o herramientas de gestión empresarial. El objetivo es capacitar para intervenir en la implantación, adaptación y administración básica de este tipo de soluciones en entornos reales.

1. CMS: concepto y tipologías.

El despliegue de aplicaciones web en entornos profesionales no siempre implica el desarrollo desde cero de una solución personalizada. En muchos casos, las organizaciones utilizan **sistemas de gestión de contenidos (CMS, Content Management System)** que permiten crear, administrar y actualizar sitios web y aplicaciones con una estructura predefinida y extensible. Estos sistemas facilitan la publicación de contenidos, la gestión de usuarios y la integración de funcionalidades sin necesidad de programación avanzada.



Componentes estructurales de un CMS: base de datos, plantillas y extensiones (plugins) que permiten ampliar funcionalidades.

En el contexto del perfil de Sistemas Microinformáticos y Redes, el interés principal no reside en desarrollar un CMS, sino en saber **instalarlo, configurarlo, administrarlo, asegurar su funcionamiento y mantenerlo actualizado**. La correcta comprensión del concepto y de las distintas tipologías existentes permite seleccionar la herramienta adecuada según el objetivo del proyecto.

Un **CMS** es una aplicación web que permite gestionar contenidos digitales —textos, imágenes, documentos, productos, cursos, noticias— mediante una interfaz administrativa accesible desde navegador. Se basa normalmente en una arquitectura en tres capas:

- Presentación (interfaz pública y panel de administración).
- Lógica de negocio (gestión de usuarios, publicación, permisos).
- Base de datos (almacenamiento de contenidos y configuraciones).

Los CMS separan el contenido de la estructura técnica, lo que permite que usuarios con perfil no técnico puedan actualizar la información sin modificar directamente archivos del servidor.

Entre sus funcionalidades habituales se encuentran:

- Gestión de usuarios y roles.
- Edición de contenido mediante panel visual.
- Gestión de categorías y etiquetas.
- Soporte para extensiones o módulos.
- Gestión de plantillas visuales.
- Publicación programada.
- Sistema de comentarios o interacción.

El uso de un CMS aporta ventajas significativas frente a la gestión manual de archivos HTML:

- Centralización de la administración.
- Actualización dinámica sin intervención técnica directa.
- Control de versiones.
- Separación entre diseño y contenido.
- Escalabilidad mediante extensiones.
- Gestión estructurada de permisos.

Según el punto de vista del técnico en sistemas, estas ventajas deben equilibrarse con la necesidad de mantener el sistema actualizado y protegido frente a vulnerabilidades.

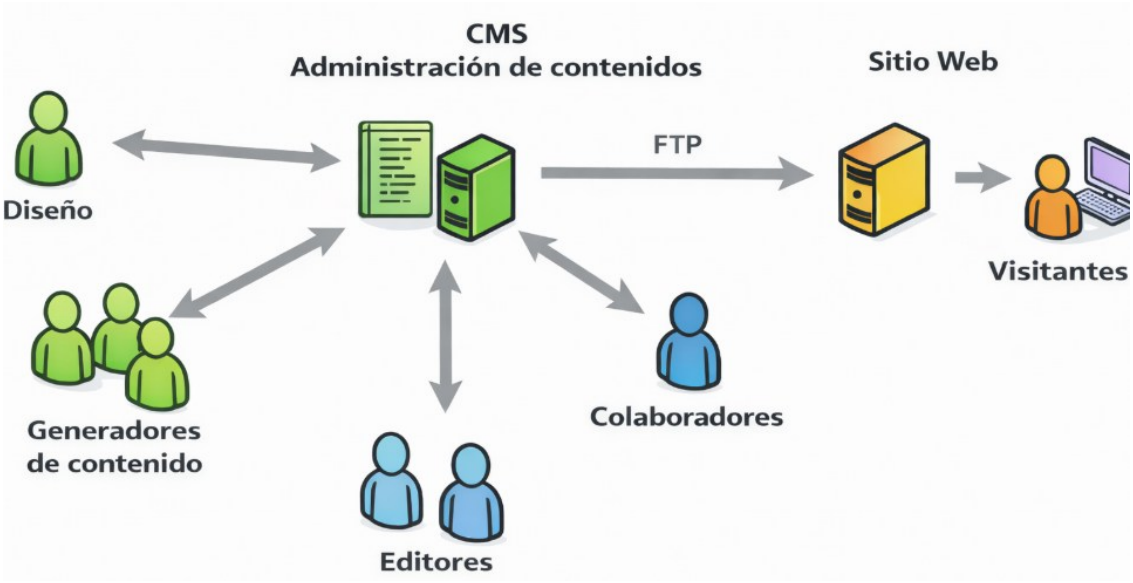

Nota

Cuanto más extendido está un CMS, mayor es la probabilidad de que existan intentos automatizados de explotación de vulnerabilidades conocidas. Por ello, el mantenimiento periódico es una obligación técnica.

Los CMS pueden clasificarse según el tipo de aplicación para la que están diseñados. Esta clasificación facilita su elección en entornos reales:

Tipología	Finalidad principal	Ejemplos de uso
CMS generalista	Gestión de sitios corporativos y blogs	Web institucional
CMS de comercio electrónico	Gestión de catálogo y ventas	Tienda online
CMS educativo (LMS)	Gestión de cursos y alumnos	Plataforma formativa
CMS documental	Gestión de archivos internos	Intranet corporativa
CMS empresarial integrado	Gestión combinada de procesos	Portal interno avanzado

Cada tipología responde a necesidades específicas y presenta módulos adaptados a su propósito.



Flujo de administración en un CMS: gestión de contenidos en el backend y publicación en el sitio web para los usuarios finales.

CMS generalistas

Los CMS generalistas son los más extendidos. Permiten crear:

- Sitios web corporativos.
- Blogs.
- Portales informativos.
- Páginas institucionales.

Sus características principales incluyen:

- Sistema de plantillas.
- Editor visual de contenido.
- Gestión de usuarios por roles.
- Amplio ecosistema de extensiones.
- Soporte para SEO básico.

Este tipo de CMS resulta adecuado para organizaciones que necesitan presencia web flexible y actualizable.

CMS de comercio electrónico

Estos sistemas incorporan funcionalidades específicas para venta online:

- Gestión de productos.
- Carrito de compra.
- Integración con pasarelas de pago.
- Gestión de inventario.

- Control de pedidos.
- Cálculo de impuestos y envíos.

Desde el punto de vista técnico, exigen mayor atención a:

- Seguridad.
- Integración con APIs externas.
- Protección de datos.
- Copias frecuentes.

LMS (Learning Management System)

Los sistemas de gestión de aprendizaje permiten:

- Publicar cursos.
- Gestionar matrículas.
- Evaluar alumnos.
- Generar informes.
- Controlar progreso.

Son habituales en centros educativos y empresas que implementan formación interna. Su administración técnica implica:

- Gestión de roles (profesor, alumno, administrador).
- Integración con directorios.
- Control de capacidad del servidor.
- Supervisión de almacenamiento de contenidos multimedia.

CMS empresariales integrados

Existen plataformas que combinan gestión de contenidos con funciones empresariales avanzadas, integrando módulos de:

- Gestión documental.
- Gestión de usuarios internos.
- Flujos de aprobación.
- Informes.
- Integraciones externas.

En estos casos, el técnico debe considerar:

- Escalabilidad.
- Integración con sistemas corporativos.
- Configuración avanzada de permisos.
- Mantenimiento continuo.

Clasificación según modelo de implantación

Además de su finalidad, los CMS pueden clasificarse según el modelo de despliegue:

Modelo	Características	Impacto técnico
Instalación local (on-premise)	Se instala en servidor propio	Control total, mayor responsabilidad
Hosting gestionado	Instalado en proveedor externo	Administración compartida
SaaS	Servicio completo en la nube	Gestión funcional sin control de infraestructura

Esta clasificación es relevante para determinar el alcance de la administración técnica.

Arquitectura interna de un CMS

Aunque el usuario solo percibe una interfaz visual, internamente un CMS suele incluir:

- Motor de plantillas.
- Sistema de autenticación.
- Gestor de base de datos.
- Módulos o plugins.
- Sistema de caché.
- Panel administrativo.

Esta estructura modular permite ampliar funcionalidades sin modificar el núcleo del sistema.

Ejemplo

Una empresa de servicios técnicos necesita:

- Página corporativa.
- Blog de noticias.
- Formulario de contacto.
- Área privada para clientes.
- Integración con correo corporativo.

Tras analizar requisitos, se opta por un CMS generalista con soporte para plugins de área privada.

El técnico debe:

1. Instalar el CMS en servidor web.
2. Configurar base de datos.
3. Definir roles (administrador, editor, cliente).
4. Instalar extensiones necesarias.
5. Configurar seguridad básica.
6. Programar copias de seguridad.

La elección del CMS depende no solo de funcionalidades visibles, sino también de la capacidad técnica disponible para administrarlo.

Riesgos asociados a los CMS

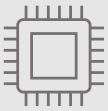
Aunque los CMS simplifican el despliegue, presentan riesgos específicos:

- Vulnerabilidades en plugins.
- Versiones desactualizadas.
- Permisos mal configurados.
- Exposición del panel de administración.
- Falta de copias de seguridad.

La administración responsable implica revisión periódica y actualización controlada.

Nota

Cada plugin instalado aumenta la superficie de ataque y el consumo de recursos. Es recomendable utilizar únicamente extensiones necesarias y provenientes de fuentes confiables.



Actividad 9

Una organización desea implantar una nueva plataforma digital y plantea los siguientes escenarios independientes:

Escenario A: Una empresa quiere renovar su web corporativa, publicar noticias, gestionar varias secciones institucionales y permitir que distintos departamentos actualicen contenidos sin conocimientos técnicos.

Escenario B: Una tienda física quiere ampliar su negocio con venta online, gestión de productos, pagos electrónicos y control de pedidos.

Escenario C: Un centro de formación necesita publicar cursos, gestionar alumnos, evaluar progresos y generar informes de seguimiento.

Para cada escenario:

Indica qué tipología de CMS es la más adecuada.

Justifica brevemente la elección desde el punto de vista funcional y técnico.

Señala al menos un aspecto de mantenimiento o seguridad que el técnico de sistemas deberá supervisar.

2. Instalación completa de WordPress.

Entre los sistemas de gestión de contenidos generalistas, **WordPress** ocupa una posición predominante en el ámbito profesional debido a su amplia adopción, flexibilidad y ecosistema de extensiones. Su instalación constituye un ejercicio práctico completo que integra todos los elementos estudiados en la unidad anterior: servidor web, base de datos, configuración de permisos, conexión dinámica y seguridad básica.

Desde el punto de vista técnico, instalar WordPress no consiste únicamente en copiar archivos y ejecutar un asistente gráfico. Implica preparar adecuadamente el entorno, configurar la base de datos con criterios de seguridad, ajustar permisos y verificar el correcto funcionamiento del sistema antes de su puesta en producción.

Antes de iniciar la instalación, deben cumplirse los siguientes requisitos técnicos:

- Servidor web operativo (Apache o Nginx).
- Motor PHP instalado y habilitado.
- Sistema gestor de base de datos (MySQL o MariaDB).
- Usuario y base de datos creados.
- Permisos adecuados en el directorio del servidor.
- Conectividad de red funcional.
- Puertos HTTP/HTTPS abiertos.

Los requisitos técnicos habituales son:

Componente	Requisito mínimo recomendado
PHP	Versión actual compatible
Base de datos	MySQL o MariaDB activo
Memoria del servidor	Adecuada al tráfico previsto
HTTPS	Certificado configurado
Espacio en disco	Suficiente para archivos y cargas

La verificación de estos requisitos reduce incidencias durante el proceso.

Creación de la base de datos

El primer paso técnico consiste en crear:

1. Una base de datos específica para WordPress.
2. Un usuario exclusivo.
3. Asignación de privilegios limitados.

El usuario asignado no debe ser el administrador global del sistema gestor.

Ejemplo

Ejemplo conceptual de configuración segura:

- Base de datos: wp_empresa
- Usuario: wp_user
- Privilegios: SELECT, INSERT, UPDATE, DELETE
- Sin acceso a otras bases

Esta separación protege el resto del entorno en caso de vulnerabilidad.

Cada aplicación web debe disponer de su propia base de datos y usuario exclusivo. Compartir credenciales entre aplicaciones aumenta el riesgo de impacto cruzado ante incidentes.

Descarga y preparación de archivos

El siguiente paso consiste en:

- Descargar la versión oficial desde la fuente autorizada.
- Descomprimir el paquete.
- Copiar los archivos al directorio DocumentRoot o al subdirectorio correspondiente.
- Verificar estructura correcta de carpetas.

El directorio debe tener permisos adecuados para que el servidor pueda leer los archivos y, si es necesario, escribir en directorios específicos (por ejemplo, carpeta de contenidos).

Configuración del archivo de conexión

WordPress utiliza un archivo de configuración donde se especifican:

- Nombre de la base de datos.
- Usuario.
- Contraseña.
- Servidor de base de datos.
- Prefijo de tablas.
- Claves de seguridad.

Recuerda

Este archivo es crítico y debe protegerse adecuadamente.

Se debe:

- Utilizar prefijo distinto al predeterminado para mayor seguridad.
- Establecer claves únicas generadas de forma aleatoria.
- No asignar permisos de escritura innecesarios tras la configuración.

Ejecución del asistente de instalación

Accediendo desde navegador a la dirección del servidor, el sistema detecta que no existe configuración completa y lanza el asistente de instalación.

Durante este proceso se definen:

- Título del sitio.
- Usuario administrador.
- Contraseña robusta.
- Dirección de correo de administración.

Es importante no utilizar credenciales triviales para el usuario administrador.

Verificación posterior a la instalación

Tras completar el asistente, deben realizarse comprobaciones técnicas:

- Acceso correcto al panel de administración.
- Funcionamiento del sitio público.
- Conexión correcta con base de datos.
- Creación adecuada de tablas.
- Revisión de permisos en archivos.

También debe verificarse que:

- No se muestren errores PHP.
- No existan advertencias de configuración.
- La instalación esté bajo HTTPS.

Ajustes iniciales recomendados

Una instalación básica no es suficiente para producción. Tras la puesta en marcha se recomienda:

- Cambiar la estructura de enlaces permanentes.
- Desactivar edición de archivos desde panel.
- Configurar idioma y zona horaria.
- Eliminar contenidos de ejemplo.
- Activar actualizaciones automáticas si procede.
- Configurar copias de seguridad.

Estas tareas forman parte de la fase de endurecimiento inicial.

Permisos de archivos y directorios

El servidor web necesita:

- Permiso de lectura en archivos principales.
- Permiso de escritura en directorio de contenidos (para cargas y actualizaciones).
- Restricción en archivos sensibles.

Recuerda

Un error común es otorgar permisos excesivos para "evitar problemas", lo que incrementa el riesgo de modificación no autorizada.

Los criterios generales de permisos son:

Elemento	Nivel de permiso recomendado
Archivos principales	Lectura para servidor
Carpeta de contenidos	Escritura controlada
Archivo de configuración	Restrictivo
Directorio raíz	Sin permisos globales

El principio de mínimo privilegio debe aplicarse de forma sistemática.

Ejemplo

Una empresa contrata un VPS Linux para alojar su web corporativa basada en WordPress.

Proceso técnico:

1. Instalación de Apache y PHP.
2. Instalación de MariaDB.
3. Creación de base de datos y usuario exclusivo.
4. Descarga de WordPress.
5. Configuración del archivo de conexión.
6. Ajuste de permisos.
7. Configuración de VirtualHost.
8. Activación de HTTPS.
9. Verificación desde red externa.

Durante el proceso se detecta que el directorio de contenidos no permite escritura. Se ajustan permisos específicos sin conceder privilegios globales.

Problemas habituales durante la instalación

Las incidencias más frecuentes incluyen:

- Error de conexión a base de datos.
- Versión de PHP incompatible.
- Permisos insuficientes.
- Conflicto de puertos.
- Certificado HTTPS mal configurado.

El diagnóstico debe apoyarse en revisión de logs del servidor web y del motor PHP.

Consideraciones para entorno productivo

En producción deben aplicarse medidas adicionales:

- Ocultar versión de WordPress.
- Limitar intentos de acceso al panel.
- Restringir acceso al panel por IP si procede.
- Programar copias automáticas.
- Documentar la instalación.

Nota

Cada instalación debe quedar registrada con versión, fecha, configuración aplicada y credenciales administrativas gestionadas de forma segura. Esta documentación facilita futuras intervenciones.

3. Configuración inicial y ajustes técnicos.

Una vez completada la instalación de WordPress, el sistema no puede considerarse listo para producción sin realizar una **configuración técnica inicial estructurada**. Esta fase tiene como objetivo optimizar rendimiento, reforzar seguridad, ajustar parámetros operativos y preparar el entorno para un uso estable y profesional.

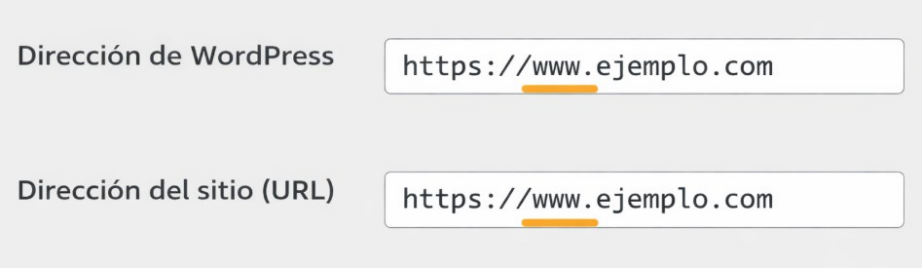
La experiencia demuestra que muchas incidencias posteriores no se deben a errores en la instalación, sino a una configuración inicial incompleta o incorrecta. Por ello, esta etapa debe abordarse de forma sistemática y documentada.

Configuración general del sistema

Desde el panel de administración deben revisarse los ajustes básicos:

- Título del sitio.
- Descripción.
- Idioma.
- Zona horaria.
- Formato de fecha y hora.
- Dirección URL correcta (con HTTPS).

Una URL mal configurada puede provocar redirecciones erróneas o problemas de acceso.



Parámetros de dirección en WordPress: correcta definición de URL para evitar redirecciones erróneas y problemas de acceso.

Entre los ajustes críticos iniciales destacan:

Parámetro	Acción recomendada	Impacto técnico
Dirección del sitio	Forzar HTTPS	Seguridad y SEO
Zona horaria	Ajustar correctamente	Publicaciones programadas
Nombre de usuario admin	Evitar “admin”	Reducción de ataques automatizados
Visibilidad en buscadores	Ajustar según fase	Control de indexación

Configuración de enlaces permanentes

WordPress permite elegir la estructura de URLs (permalinks). La configuración predeterminada basada en parámetros no es recomendable en producción.

Opciones habituales son:

- Estructura basada en nombre de entrada.
- Estructura personalizada.
- Inclusión de categorías.

Según un punto de vista técnico:

- Las URLs amigables mejoran posicionamiento.
- Requieren correcta configuración del servidor web.
- Implican reglas de reescritura activas.

Una mala configuración puede provocar errores 404 tras cambiar la estructura.

Gestión de roles y usuarios

WordPress incluye distintos roles predefinidos:

Rol	Capacidades principales
Administrador	Control total
Editor	Gestión de contenidos
Autor	Publicación propia
Colaborador	Redacción sin publicación
Suscriptor	Acceso básico

Se debe:

- Crear cuentas individuales.
- Evitar compartir usuario administrador.
- Asignar rol mínimo necesario.
- Activar autenticación robusta.

Nota

El uso indiscriminado del rol administrador incrementa el riesgo de alteraciones accidentales o maliciosas. La asignación de permisos debe ser restrictiva y justificada.

Configuración de seguridad básica

La seguridad inicial debe incluir medidas estructurales:

- **Protección del archivo de configuración:**
  - Limitar permisos.

  - Evitar acceso desde navegador.
- **Desactivar edición de archivos desde panel:**
  - Reduce riesgo ante acceso no autorizado.
- **Limitar intentos de acceso:**
  - Evita ataques de fuerza bruta.
- **Cambio de prefijo de tablas:**
  - Reduce ataques automatizados dirigidos.
- **Activar HTTPS obligatorio:**
  - Forzar redirección desde HTTP.

Ajustes de rendimiento

WordPress puede optimizarse mediante:

- Activación de caché.
- Compresión de archivos.
- Minimización de recursos estáticos.
- Configuración adecuada de memoria PHP.

Los parámetros técnicos relevantes son:

Parámetro	Función	Impacto
Límite de memoria PHP	Define capacidad de ejecución	Evita errores por falta de recursos
Caché	Reduce consultas repetidas	Mejora rendimiento
Tamaño máximo de subida	Controla archivos subidos	Previene abuso
Tiempo máximo de ejecución	Evita procesos bloqueados	Estabilidad

La optimización debe adaptarse al volumen de tráfico previsto.

Configuración del sistema de actualizaciones

WordPress recibe actualizaciones periódicas que pueden ser:

- Actualizaciones de núcleo.
- Actualizaciones de plugins.
- Actualizaciones de temas.

Las estrategias posibles son:

Estrategia	Ventaja	Riesgo
Actualización automática total	Seguridad rápida	Posible incompatibilidad
Actualización manual controlada	Mayor estabilidad	Riesgo si se retrasa
Entorno de pruebas previo	Alta fiabilidad	Requiere infraestructura adicional

En entornos profesionales se recomienda probar actualizaciones en entorno de pruebas antes de aplicarlas en producción.

Configuración de copias de seguridad

Tras la instalación, debe definirse:

- Frecuencia de copia.
- Ubicación externa.
- Inclusión de base de datos y archivos.
- Procedimiento de restauración documentado.

No debe dependerse únicamente de copias proporcionadas por proveedor de hosting sin verificación.

Configuración del correo saliente

WordPress utiliza correo para:

- Recuperación de contraseñas.
- Notificaciones administrativas.
- Confirmaciones.

Es necesario verificar:

- Que el servidor puede enviar correos.
- Que los mensajes no se clasifican como spam.
- Que la dirección remitente es válida.

Una mala configuración puede impedir recuperación de acceso en caso de incidente.

Ejemplo

Una empresa instala WordPress y realiza la configuración inicial:

1. Activa HTTPS.
2. Cambia estructura de enlaces permanentes.
3. Crea usuarios diferenciados.
4. Instala plugin de caché.
5. Limita intentos de acceso.
6. Programa copias automáticas externas.
7. Desactiva edición directa de archivos.
8. Ajusta permisos del archivo de configuración.

Tras estas acciones, el sistema pasa de estar “instalado” a estar “operativamente preparado”.

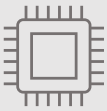
Para cerrar la configuración inicial, puede utilizarse un checklist estructurado:

Elemento	Verificado
HTTPS activo	✓
Permalinks configurados	✓
Usuario admin seguro	✓
Roles asignados correctamente	✓
Copias programadas	✓
Permisos revisados	✓
Plugins innecesarios eliminados	✓
Logs revisados	✓

Este procedimiento estandariza la puesta en producción.

Recuerda

La configuración inicial y los ajustes técnicos constituyen la fase de endurecimiento y optimización tras la instalación de un CMS como WordPress. Sin esta etapa, el sistema puede funcionar, pero no estará adecuadamente preparado para un entorno profesional.



Actividad 10

Una empresa ha instalado WordPress en producción y, tras varios cambios, comienzan a producirse errores 404 en algunas páginas. Además, todos los usuarios del equipo comparten una misma cuenta con rol de administrador y las actualizaciones están desactivadas para evitar "problemas".

Desde el punto de vista técnico, explica:

Qué decisiones de configuración pueden estar provocando los errores 404.

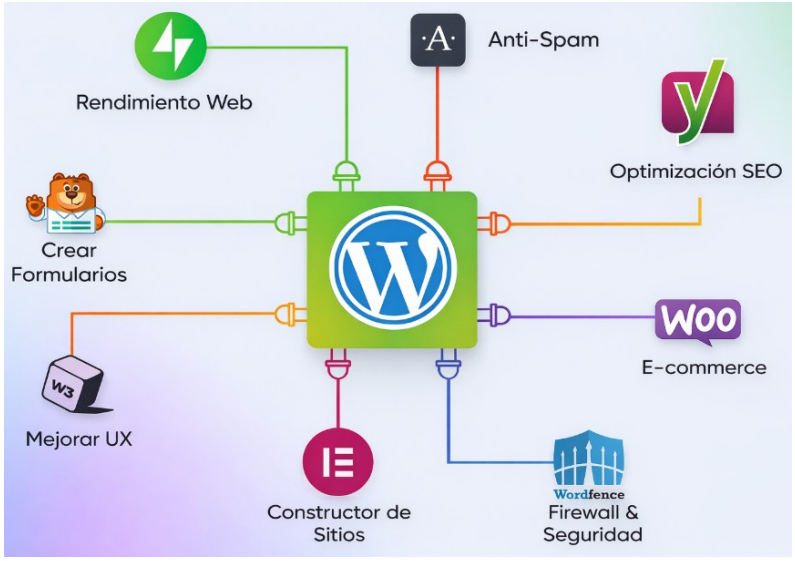
Qué riesgos existen en la gestión actual de usuarios.

Qué estrategia de actualizaciones sería más adecuada en un entorno profesional.

Indica al menos una medida adicional de seguridad que debería aplicarse.

4. Gestión de temas y plugins.

Una vez configurado técnicamente el CMS, el siguiente nivel de administración se centra en la **gestión de temas y plugins**, elementos que determinan la apariencia, las funcionalidades y, en gran medida, la seguridad del sistema. En WordPress —y en la mayoría de CMS modernos— el núcleo del sistema ofrece una base funcional estable, mientras que los temas (themes) y plugins (extensiones) permiten adaptar el sitio a necesidades concretas.



Extensión funcional de un CMS mediante plugins que amplían capacidades como seguridad, SEO, comercio electrónico o rendimiento.

Según un punto de vista técnico, la instalación de un tema o plugin no es una acción meramente estética o funcional. Cada componente adicional modifica el comportamiento del sistema, introduce código nuevo y puede afectar al rendimiento, a la seguridad y a la estabilidad. Por ello, la gestión de estos elementos debe abordarse con criterios técnicos claros, no únicamente con criterios visuales o de funcionalidad inmediata.

Un **tema** define la estructura visual y parte del comportamiento de la interfaz pública del sitio web. Incluye:

- Diseño gráfico.
- Disposición de elementos.
- Plantillas de páginas.
- Tipografías y estilos.
- Configuración visual adaptable.

El tema no debería alterar la lógica interna del sistema ni sustituir funcionalidades propias de plugins, aunque en la práctica algunos incluyen funcionalidades adicionales.

El cambio de tema no modifica el contenido almacenado en la base de datos, pero sí puede afectar a la forma en que dicho contenido se muestra.

Los temas pueden clasificarse según su enfoque y estructura:

Tipo de tema	Características	Impacto técnico
Tema estándar	Diseño predefinido con opciones básicas	Bajo impacto estructural
Tema multipropósito	Gran cantidad de opciones y constructores visuales	Mayor consumo de recursos
Tema minimalista	Diseño ligero, orientado a rendimiento	Alta eficiencia
Tema específico (nicho)	Adaptado a sector concreto (tienda, formación, portfolio)	Funcionalidad orientada

La elección del tema influye directamente en:

- Tiempo de carga.
- Compatibilidad con plugins.
- Facilidad de mantenimiento.
- Actualizaciones futuras.

Nota

Un tema visualmente atractivo pero excesivamente cargado puede aumentar el tiempo de respuesta del servidor y afectar negativamente a la experiencia de usuario y al posicionamiento.

La instalación puede realizarse:

- Desde el repositorio oficial.
- Mediante subida manual.
- A través de paquetes proporcionados por terceros.

Tras la instalación, el tema debe activarse y configurarse. Es recomendable:

- Probar en entorno de pruebas antes de aplicar en producción.
- Verificar compatibilidad con versión actual del CMS.
- Revisar fecha de última actualización.
- Comprobar reputación del desarrollador.

Un cambio de tema en producción sin pruebas previas puede generar:

- Errores visuales.
- Incompatibilidades.
- Pérdida de elementos configurados.
- Fallos en widgets o menús.

Gestión de plugins: concepto y alcance

Un **plugin** es una extensión que añade funcionalidades al núcleo del CMS. Puede incorporar:

- Formularios de contacto.
- Sistemas de seguridad.
- Herramientas SEO.
- Caché.
- Integración con APIs externas.
- Comercio electrónico.
- Gestión avanzada de usuarios.

Cada plugin introduce código adicional que se ejecuta en el servidor. Por tanto, su instalación debe evaluarse con criterio técnico.

Los plugins pueden agruparse según su finalidad:

Categoría	Función principal	Ejemplo de uso
Seguridad	Protección ante ataques	Limitación de accesos
Rendimiento	Optimización y caché	Mejora de carga
SEO	Optimización para buscadores	Metadatos estructurados
Formularios	Captura de datos	Contacto o registro
Comercio electrónico	Gestión de productos	Tienda online
Integración externa	Conexión con APIs	CRM o redes sociales

Esta clasificación facilita una selección racional y evita instalaciones innecesarias.

Antes de instalar un plugin deben evaluarse varios factores:

- Compatibilidad con la versión actual del CMS.
- Número de instalaciones activas.
- Valoraciones y reseñas.
- Frecuencia de actualizaciones.
- Soporte disponible.
- Impacto estimado en rendimiento.
- Revisión de permisos solicitados.

Un plugin desactualizado o sin mantenimiento puede convertirse en vector de vulnerabilidad.

Nota

Cada plugin añadido aumenta la superficie de ataque y el consumo de recursos. Es recomendable instalar únicamente aquellos estrictamente necesarios para el funcionamiento del sitio.

Interacción entre temas y plugins

En entornos reales, los problemas no suelen originarse por un solo componente, sino por la interacción entre ellos.

Conflictos habituales son:

- Incompatibilidad entre versiones.
- Funcionalidades duplicadas.
- Errores JavaScript.
- Conflictos en estilos CSS.
- Bloqueo de ejecución de scripts.

Recuerda

Cualquier modificación importante debe probarse en entorno controlado antes de aplicarse en producción.

Los temas y plugins requieren actualizaciones periódicas para:

- Corregir vulnerabilidades.
- Mejorar rendimiento.
- Añadir compatibilidad.
- Adaptarse a nuevas versiones del núcleo.

Las estrategias de actualización son:

Estrategia	Ventaja	Riesgo
Actualización automática	Seguridad rápida	Posibles incompatibilidades
Actualización manual tras pruebas	Mayor control	Requiere tiempo
Entorno staging previo	Alta estabilidad	Necesita infraestructura adicional

En entornos empresariales es recomendable disponer de un entorno de pruebas (staging) donde validar actualizaciones antes de aplicarlas.

No basta con desactivar un plugin; si no se utiliza, debe eliminarse completamente para:

- Reducir superficie de ataque.
- Disminuir consumo de recursos.
- Simplificar mantenimiento.

Del mismo modo, mantener varios temas instalados sin uso puede suponer riesgo innecesario.

Ejemplo

Una empresa desea añadir nuevas funcionalidades:

- Sistema de caché.
- Formulario avanzado.
- Plugin de seguridad.

Proceso técnico recomendado:

1. Analizar si el servidor ya incluye mecanismos de caché.
2. Evaluar plugins compatibles.
3. Instalar uno por uno.
4. Probar funcionamiento tras cada instalación.
5. Medir impacto en tiempo de carga.
6. Verificar ausencia de errores en consola.
7. Documentar cambios realizados.

Tras las pruebas se detecta que un plugin de caché entra en conflicto con el sistema de servidor. Se decide utilizar únicamente la caché a nivel servidor, eliminando el plugin.

La gestión inadecuada de temas y plugins puede generar:

- Lentitud en carga.
- Errores de ejecución.
- Vulnerabilidades explotables.
- Caídas del servidor.
- Conflictos tras actualizaciones.

Una administración responsable implica equilibrio entre funcionalidad y estabilidad.

Procedimiento recomendado de gestión

Para mantener control técnico, puede aplicarse el siguiente esquema operativo:

1. Evaluar necesidad real.
2. Verificar compatibilidad.
3. Instalar en entorno de pruebas.
4. Comprobar rendimiento.
5. Aplicar en producción.
6. Documentar versión instalada.
7. Programar revisión periódica.

Este procedimiento sistematiza la gestión y reduce intervenciones impulsivas.

5. Seguridad básica en CMS.

La seguridad en un CMS no puede abordarse como una tarea puntual posterior a la instalación. Constituye un **proceso continuo de prevención, supervisión y mantenimiento** que debe integrarse desde el primer momento en el ciclo de vida del sistema. La popularidad de plataformas como WordPress convierte a los CMS en objetivos frecuentes de ataques automatizados, explotación de vulnerabilidades conocidas, fuerza bruta, inyecciones de código y acceso indebido a paneles administrativos.

Seguridad en CMS: protección del núcleo del sistema frente a accesos indebidos, vulnerabilidades y ataques automatizados.

Desde el perfil técnico de administración de sistemas, la seguridad básica en un CMS implica aplicar un conjunto estructurado de medidas preventivas orientadas a reducir la superficie de ataque, limitar privilegios, mantener el software actualizado y monitorizar posibles incidentes.

Para diseñar medidas adecuadas es necesario comprender los riesgos más habituales:

Tipo de amenaza	Descripción	Impacto potencial
Fuerza bruta	Intentos automatizados de acceso	Compromiso de cuentas
Vulnerabilidades en plugins	Fallos de código explotables	Ejecución remota
Inyección SQL	Manipulación de consultas	Acceso a base de datos
Cross-Site Scripting (XSS)	Inyección de scripts en páginas	Robo de sesión
Acceso a archivos sensibles	Exposición de configuraciones	Fuga de datos
Malware por carga de archivos	Subida de archivos maliciosos	Control del servidor

La identificación de estas amenazas permite establecer medidas preventivas coherentes.

Actualización constante del núcleo, temas y plugins

Una de las medidas más eficaces consiste en mantener el CMS y sus componentes en versiones actualizadas.

Las actualizaciones corrigen:

- Vulnerabilidades conocidas.
- Errores de seguridad.
- Problemas de compatibilidad.
- Fallos críticos detectados.

La política recomendada de actualización consiste en:

Elemento	Frecuencia recomendada	Método
Núcleo CMS	Inmediata tras pruebas	Manual controlada
Plugins críticos	Tras validación en entorno de pruebas	Programada
Temas activos	Tras verificar compatibilidad	Controlada
Plugins no utilizados	Eliminación	Permanente

Actualizar sin pruebas previas puede generar incompatibilidades; no actualizar expone el sistema a riesgos conocidos. El equilibrio debe gestionarse técnicamente.

Protección del acceso administrativo

El panel de administración constituye uno de los puntos más sensibles del sistema.

Las medidas básicas recomendadas son:

- No utilizar el nombre de usuario predeterminado.
- Establecer contraseñas robustas.
- Activar autenticación multifactor si está disponible.
- Limitar intentos de acceso.
- Restringir acceso por IP en entornos corporativos.
- Forzar uso exclusivo de HTTPS.

Nota

Las credenciales administrativas no deben almacenarse en texto plano ni compartirse entre usuarios. Es recomendable utilizar gestores seguros de contraseñas en entornos profesionales.

Configuración de permisos en archivos y directorios

El sistema de archivos constituye otra capa crítica.

Se debe:

- Asignar permisos mínimos necesarios.
- Restringir escritura únicamente a directorios imprescindibles.
- Proteger el archivo de configuración.
- Evitar permisos globales de escritura.

El control de permisos consiste en:

Elemento	Riesgo si está mal configurado	Acción recomendada
Archivo de configuración	Exposición de credenciales	Permisos restrictivos
Carpeta de contenidos	Subida de archivos maliciosos	Escritura controlada
Directorio raíz	Modificación de archivos críticos	Evitar permisos amplios

Un error en permisos puede facilitar la inyección de código malicioso.

Protección frente a ataques de fuerza bruta

Los ataques automatizados buscan acceder al panel mediante múltiples intentos.

Las medidas recomendadas son:

- Limitación de intentos.
- Bloqueo temporal tras varios fallos.
- Registro de IP sospechosas.
- Implementación de CAPTCHA en formularios.
- Monitorización de intentos fallidos.

Estas medidas reducen significativamente la probabilidad de acceso no autorizado.

Seguridad en la base de datos

El CMS depende directamente de la base de datos. Las medidas básicas incluyen:

- Usuario exclusivo para la aplicación.
- Privilegios mínimos necesarios.
- Contraseña robusta.
- Restricción de acceso remoto.
- Copias de seguridad periódicas.
- Monitorización de actividad.

No debe utilizarse el usuario administrador global para la conexión de la aplicación.

Uso de HTTPS y cifrado

El uso obligatorio de HTTPS protege:

- Credenciales de acceso.
- Formularios.
- Sesiones.
- Datos transmitidos.

Debe configurarse:

- Certificado válido.
- Redirección automática desde HTTP.
- Renovación periódica del certificado.
- Eliminación de contenido mixto.

Recuerda

Un CMS sin HTTPS en producción constituye un riesgo evidente.

Monitorización y registro de actividad

La seguridad no se limita a prevenir; también implica detectar.

Debe activarse:

- Registro de accesos.
- Registro de cambios en configuración.
- Alertas ante modificaciones críticas.
- Supervisión de archivos modificados.

La revisión periódica de logs permite detectar patrones anómalos antes de que el incidente escale.

Gestión de extensiones con criterio de seguridad

Cada plugin introduce código adicional.

Criterios de seguridad son:

- Instalar únicamente plugins necesarios.
- Eliminar extensiones no utilizadas.
- Verificar procedencia y reputación.
- Revisar permisos solicitados.
- Mantener actualizados.

Un plugin desactualizado puede ser más peligroso que el núcleo del sistema.

Copias de seguridad como medida de seguridad

Aunque las copias se estudian como elemento de continuidad, también constituyen una medida de seguridad.

Ante incidentes como:

- Infección por malware

- Alteración de archivos
- Corrupción de base de datos

La restauración permite recuperar el estado anterior. La copia de seguridad no evita el ataque, pero reduce su impacto. La seguridad debe combinar medidas preventivas y capacidad de recuperación.

Ejemplo

Una empresa despliega WordPress en servidor propio.

Medidas aplicadas:

1. Cambio del usuario administrador predeterminado.
2. Activación de HTTPS obligatorio.
3. Configuración de limitación de intentos.
4. Protección del archivo de configuración.
5. Eliminación de plugins innecesarios.
6. Instalación de plugin de seguridad reconocido.
7. Configuración de copias automáticas externas.
8. Revisión de permisos de directorios.

Tras estas acciones, se ejecuta una revisión de logs para confirmar ausencia de accesos sospechosos.

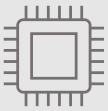
Este procedimiento reduce significativamente la superficie de ataque inicial.

Errores frecuentes en seguridad básica

Por último, las incidencias más habituales se producen por:

- Uso de contraseñas débiles.
- No actualizar el CMS.
- Mantener plugins sin soporte.
- Exponer phpMyAdmin sin protección.
- No limitar accesos administrativos.
- No revisar logs periódicamente.

La seguridad básica no requiere herramientas complejas, sino aplicación disciplinada de buenas prácticas.



Actividad 11

Un sitio web corporativo basado en un CMS comienza a mostrar comportamientos anómalos: redirecciones inesperadas, creación de usuarios desconocidos y lentitud general del sistema.

Tras una revisión preliminar se detecta que:

No se han actualizado plugins desde hace meses.

El acceso al panel administrativo no tiene limitación de intentos.

El archivo de configuración es accesible desde la web.

Existen múltiples cuentas con privilegios elevados.

Explica:

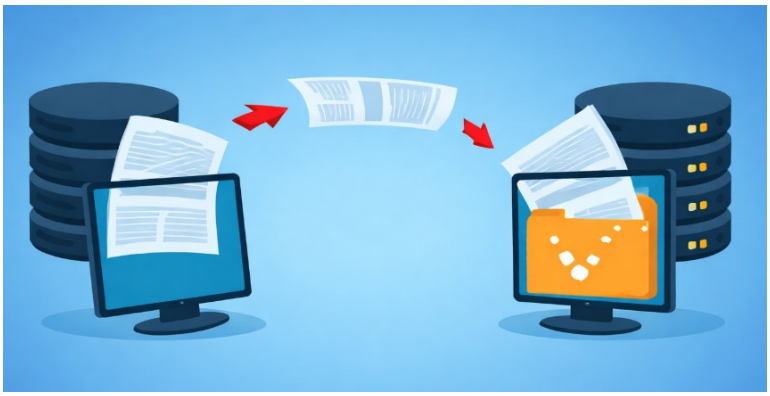
Qué tipos de amenazas podrían estar implicadas según la clasificación estudiada.

Qué medidas preventivas básicas deberían aplicarse de forma inmediata.

Qué principio general de seguridad no se ha respetado en este caso.

6. Migración de sitios web.

La **migración de un sitio web** consiste en trasladar una aplicación desde un entorno origen a un entorno destino, manteniendo su funcionalidad, contenido, configuraciones y estructura operativa. Este proceso puede implicar cambios de servidor, proveedor de hosting, dominio, protocolo (HTTP a HTTPS), infraestructura (local a cloud) o incluso versión del sistema operativo y del servidor web.



Migración de sitio web: traslado controlado de archivos y bases de datos entre servidores manteniendo la integridad del sistema.

Desde el punto de vista técnico, la migración no es una simple copia de archivos. Requiere planificación, análisis previo, control de dependencias, verificación posterior y, en muchos casos, coordinación con usuarios o departamentos implicados. Una migración mal ejecutada puede provocar pérdida de datos, caídas prolongadas del servicio, errores de conexión a base de datos o impactos negativos en posicionamiento.

Las migraciones pueden clasificarse según su alcance y complejidad:

Tipo de migración	Descripción	Nivel de complejidad
Cambio de servidor	Traslado a nuevo servidor con mismo dominio	Media
Cambio de dominio	Modificación de URL principal	Media-alta
Cambio de proveedor	Migración completa de hosting	Media
Cambio de infraestructura	Local → Cloud / VPS	Alta
Cambio de protocolo	HTTP → HTTPS	Baja-media
Cambio de versión PHP/DB	Actualización de entorno	Media

Identificar correctamente el tipo de migración permite anticipar posibles incidencias.

Una migración técnica debe abordarse siguiendo una secuencia lógica:

1. Análisis previo del entorno origen
2. Preparación del entorno destino
3. Copia de archivos
4. Exportación e importación de base de datos
5. Actualización de configuraciones internas
6. Pruebas en entorno destino
7. Cambio de DNS (si procede)
8. Verificación final y monitorización

Cada fase tiene implicaciones técnicas específicas.

Antes de iniciar el traslado, deben documentarse:

- Versión del CMS.
- Versión de PHP.
- Versión del gestor de base de datos.
- Plugins activos.
- Tamaño de base de datos.
- Tamaño de directorios.
- Configuración de servidor.
- Certificados SSL.
- Cron programados.

Un checklist previo podría ser:

Elemento	Verificado
Copia completa realizada	✓
Versiones documentadas	✓
Plugins revisados	✓
Espacio suficiente en destino	✓
Acceso administrativo disponible	✓

Este análisis evita sorpresas en el entorno de destino.

Copia de archivos

El primer paso técnico consiste en copiar:

- Directorio raíz del sitio.
- Carpeta de contenidos (uploads).
- Archivos de configuración.
- Certificados si aplica.

La copia debe realizarse asegurando:

- Integridad de archivos.
- Conservación de estructura.
- Permisos adecuados.

En entornos Linux suele realizarse mediante herramientas de sincronización seguras; en Windows mediante transferencia controlada.

Exportación e importación de base de datos

La base de datos debe exportarse desde el entorno origen e importarse en el destino.

Los pasos técnicos son:

1. Generar volcado completo.
2. Crear base de datos en destino.
3. Crear usuario con privilegios adecuados.
4. Importar volcado.
5. Verificar estructura y registros.

Es importante que la exportación e importación se realicen sin pérdida de codificación ni truncamientos.

Nota

Diferencias importantes entre versiones del gestor de base de datos pueden generar incompatibilidades. Es recomendable verificar compatibilidad antes de importar.

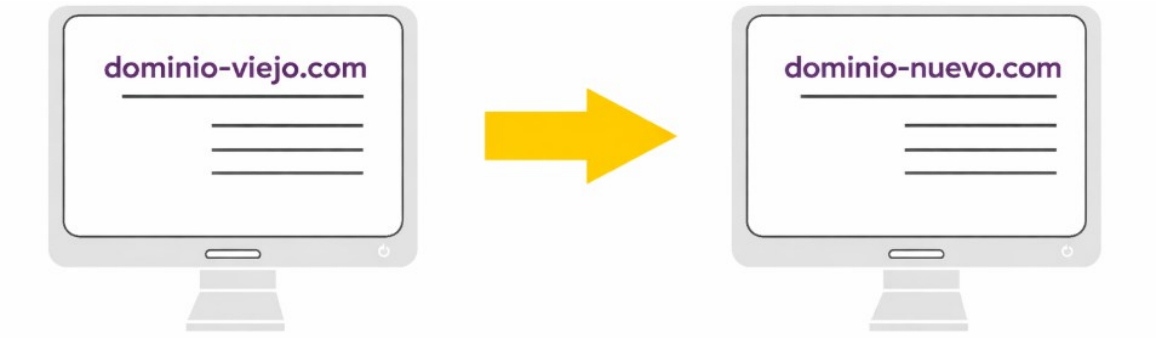
Actualización de configuraciones internas

En CMS como WordPress, pueden existir referencias internas al dominio o rutas absolutas.

Debe revisarse:

- Archivo de configuración.
- URL del sitio en base de datos.
- Configuración de correo.
- Rutas de carga.
- Ajustes de HTTPS.

En migraciones con cambio de dominio, es necesario actualizar referencias internas para evitar redirecciones incorrectas o recursos no encontrados.



Actualización de dominio durante una migración, con sustitución de referencias internas para evitar errores de redirección.

Configuración del servidor en destino

El entorno destino debe estar correctamente configurado:

- Servidor web operativo.
- PHP compatible.
- Base de datos activa.
- Permisos adecuados.
- Firewall configurado.
- Certificado SSL instalado.

El error frecuente es copiar el sitio sin preparar adecuadamente el entorno.

Pruebas previas al cambio definitivo

Antes de redirigir el tráfico real, es recomendable:

- Probar acceso mediante IP o dominio temporal.
- Verificar acceso al panel de administración.
- Comprobar enlaces internos.
- Validar envío de formularios.
- Confirmar conexión a base de datos.
- Revisar consola de navegador en busca de errores.

Esta fase reduce la probabilidad de caída visible al usuario final.

Cambio de DNS y propagación

Si la migración implica cambio de servidor manteniendo dominio, será necesario modificar los registros DNS.

Aspectos técnicos relevantes son:

- Tiempo de propagación.
- Configuración correcta de registros A o CNAME.
- TTL adecuado.
- Verificación posterior desde distintas ubicaciones.

Durante la propagación pueden coexistir accesos a servidor antiguo y nuevo.

Migración con cambio de dominio

Cuando se cambia el dominio principal, deben considerarse aspectos adicionales:

- Actualización de URLs internas.
- Configuración de redirecciones 301.

- Actualización de certificados SSL.
- Notificación a motores de búsqueda.
- Revisión de enlaces externos.

La redirección permanente (301) es esencial para preservar posicionamiento y evitar errores 404 masivos.

Ejemplo

Una empresa decide trasladar su WordPress desde hosting compartido a un VPS propio.

Proceso aplicado:

1. Generación de copia completa de archivos y base.
2. Instalación de Apache, PHP y MariaDB en VPS.
3. Creación de base y usuario exclusivo.
4. Importación de datos.
5. Ajuste de archivo de configuración.
6. Pruebas mediante acceso por IP.
7. Activación de certificado SSL.
8. Cambio de DNS.
9. Monitorización durante 48 horas.

Durante la verificación se detecta que el límite de memoria PHP era inferior al del servidor anterior. Se ajusta configuración antes de abrir al público.

Errores habituales en migraciones

Las incidencias más frecuentes incluyen:

- Olvidar actualizar URL interna.
- Permisos incorrectos en destino.
- Base de datos incompleta.
- Incompatibilidad de versiones PHP.
- Falta de redirecciones tras cambio de dominio.
- Certificado SSL no configurado correctamente.
- DNS mal apuntado.

La prevención se basa en planificación y verificación sistemática.

Plan de contingencia

Toda migración debe contemplar un plan de reversión:

- Mantener entorno antiguo activo durante transición.
- No cancelar servicio anterior hasta verificar estabilidad.
- Conservar copia original intacta.
- Documentar pasos realizados.

Esta estrategia permite recuperar rápidamente el entorno original ante fallo grave.

7. Aplicaciones web corporativas (CRM, ERP, LMS).

Más allá de los CMS orientados a la publicación de contenidos, las organizaciones utilizan **aplicaciones web corporativas** diseñadas para gestionar procesos internos, relaciones comerciales, recursos y formación. Estas aplicaciones no tienen como finalidad principal la comunicación pública, sino la **gestión estructurada de información crítica para la actividad empresarial**.

Entre las más relevantes se encuentran los **CRM (Customer Relationship Management)**, los **ERP (Enterprise Resource Planning)** y los **LMS (Learning Management System)**. Desde la perspectiva del técnico en sistemas, el interés no se centra en su desarrollo funcional, sino en su **instalación, despliegue, configuración, mantenimiento, integración y seguridad**.

CRM: Gestión de relaciones con clientes

Un **CRM** es una aplicación orientada a la gestión de contactos, oportunidades comerciales, clientes y procesos de venta. Permite centralizar la información relacionada con la actividad comercial de la empresa.

CRM como herramienta de gestión y análisis de relaciones comerciales mediante centralización de datos y métricas.

Las funciones principales de un CRM son:

- Registro de clientes y contactos.
- Seguimiento de oportunidades de venta.
- Gestión de presupuestos y contratos.
- Automatización de tareas comerciales.
- Registro de comunicaciones.
- Generación de informes.

Un CRM moderno suele estar basado en:

- Aplicación web accesible desde navegador.
- Base de datos relacional.
- Sistema de roles y permisos.
- Integración con correo electrónico.
- APIs para conexión con otros sistemas.

Según un punto de vista técnico, su despliegue implica:

- Configuración de base de datos.
- Gestión de usuarios por departamentos.
- Integración con servidor de correo.
- Control de copias de seguridad.
- Protección de datos sensibles.


Nota

Los CRM almacenan datos personales y comerciales sensibles. La configuración debe garantizar acceso restringido, cifrado en tránsito y copias de seguridad seguras.

ERP: Planificación de recursos empresariales

Un **ERP** es un sistema integral que gestiona procesos internos de la empresa, tales como:

- Contabilidad.
- Facturación.
- Inventario.
- Compras.
- Recursos humanos.
- Producción.

A diferencia de un CMS, un ERP centraliza procesos críticos y suele integrarse con múltiples módulos interdependientes.

Sus características técnicas relevantes son:

- Base de datos compleja y estructurada.
- Módulos interrelacionados.
- Control exhaustivo de permisos.
- Integración con dispositivos externos (lectores, terminales).
- Posibilidad de acceso multiusuario simultáneo.

Su clasificación según modelo de implantación es:

Modelo	Características	Implicaciones técnicas
ERP local (on-premise)	Instalado en servidor propio	Mayor control y responsabilidad
ERP en hosting gestionado	Servidor externo dedicado	Administración compartida
ERP SaaS	Servicio en la nube	Gestión funcional sin infraestructura

La elección del modelo afecta directamente a las tareas del técnico en sistemas.

LMS: Gestión de formación

Un **LMS (Learning Management System)** es una aplicación web diseñada para administrar cursos, alumnos y procesos formativos.

Sus funcionalidades habituales son:

- Creación de cursos.
- Matriculación de usuarios.
- Seguimiento del progreso.
- Evaluaciones y calificaciones.
- Generación de certificados.
- Informes de actividad.

En entornos empresariales y educativos, el LMS puede integrarse con:

- Directorios corporativos.
- Sistemas de autenticación centralizada.
- Plataformas de videoconferencia.
- Herramientas de análisis.

Desde el punto de vista técnico, el LMS requiere especial atención en:

- Capacidad del servidor (por uso multimedia).
- Gestión de almacenamiento.
- Control de accesos.
- Copias de seguridad frecuentes.

Diferencias funcionales entre CRM, ERP y LMS

Aunque comparten características técnicas comunes (aplicación web + base de datos), sus finalidades difieren claramente:

Sistema	Enfoque principal	Tipo de datos gestionados
CRM	Relación con clientes	Contactos, oportunidades, ventas
ERP	Procesos internos	Facturación, inventario, recursos
LMS	Formación	Cursos, alumnos, resultados

Comprender estas diferencias permite adaptar la infraestructura a cada caso.

Independientemente del tipo de aplicación, existen requisitos técnicos compartidos:

- Servidor web estable.
- Base de datos dimensionada.
- Copias de seguridad automatizadas.
- Control de accesos por rol.

- HTTPS obligatorio.
- Monitorización de rendimiento.
- Registro de actividad.

La criticidad de los datos gestionados suele ser mayor que en un CMS público.

Integración entre sistemas

En entornos corporativos reales, estas aplicaciones no funcionan aisladas. Es frecuente que exista integración entre:

- CRM y ERP (sincronización de clientes y facturación).
- LMS y ERP (gestión de formación interna).
- CRM y herramientas de marketing.
- ERP y sistemas contables externos.

Estas integraciones suelen realizarse mediante APIs y servicios web.

Desde el punto de vista técnico, esto implica:

- Gestión de credenciales seguras.
- Configuración de endpoints.
- Monitorización de sincronización.
- Verificación de consistencia de datos.

Ejemplo

Una empresa de servicios técnicos decide implantar:

- CRM para seguimiento comercial.
- ERP para facturación.
- LMS para formación interna.

El técnico en sistemas realiza:

1. Evaluación de requisitos de hardware.
2. Instalación en servidor Linux.
3. Creación de bases de datos independientes.
4. Configuración de roles por departamento.
5. Activación de HTTPS.
6. Integración básica entre CRM y ERP.
7. Configuración de copias diarias.
8. Documentación del despliegue.

Durante las pruebas se detecta que el ERP requiere mayor memoria de la inicialmente configurada. Se ajusta parámetro de PHP antes de abrir el sistema a usuarios.

Este caso muestra que el despliegue corporativo requiere planificación técnica más exigente que un CMS estándar.

Riesgos y buenas prácticas

Las aplicaciones corporativas gestionan datos sensibles y procesos críticos. Los riesgos incluyen:

- Acceso indebido a información financiera.
- Alteración de registros.
- Pérdida de datos formativos.
- Interrupción de procesos internos.

Buenas prácticas recomendadas son:

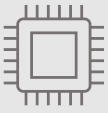
- Separación de entornos (producción y pruebas).
- Gestión de permisos por rol.
- Registro de auditoría.
- Actualizaciones controladas.
- Pruebas periódicas de restauración.
- Documentación detallada de configuración.

Recuerda

A medida que la empresa crece:

- Aumenta el número de usuarios concurrentes.
- Se incrementa el volumen de datos.
- Se requiere mayor capacidad de almacenamiento.
- Pueden necesitarse servidores dedicados o balanceo de carga.

Planificar la escalabilidad desde el inicio evita migraciones apresuradas posteriores.



Actividad 12

Una empresa tecnológica presenta las siguientes necesidades:

Control de inventario y facturación.

Gestión de nóminas y recursos humanos.

Acceso multiusuario simultáneo desde distintas sedes.

Integración con lectores de códigos de barras.

Control estricto de permisos por departamento.

Indica qué tipo de aplicación web corporativa (CRM, ERP o LMS) es la más adecuada para este caso y justifica la elección. En la justificación deben mencionarse tanto el enfoque funcional como las implicaciones técnicas del sistema elegido.

8. Entornos de desarrollo y producción.

En la administración profesional de aplicaciones web no es aceptable realizar cambios directamente sobre el sistema en producción sin validación previa. La separación entre **entorno de desarrollo**, **entorno de pruebas (staging)** y **entorno de producción** constituye una práctica fundamental para garantizar estabilidad, seguridad y continuidad del servicio.

Esta diferenciación permite trabajar con seguridad sobre nuevas funcionalidades, actualizaciones o modificaciones sin afectar a los usuarios finales. Desde la perspectiva del técnico en sistemas, comprender y administrar correctamente estos entornos es una competencia esencial.

Trabajo en entorno de desarrollo previo a la validación y despliegue en producción.

Cada entorno cumple una función específica dentro del ciclo de vida de la aplicación:

Entorno	Finalidad principal	Usuarios habituales
Desarrollo	Programación y experimentación	Desarrolladores
Pruebas / Staging	Validación antes de publicar	Técnicos y responsables
Producción	Servicio activo para usuarios finales	Clientes / empleados

El entorno de desarrollo es el espacio donde se realizan:

- Pruebas de nuevas funcionalidades.
- Instalación experimental de plugins.
- Cambios en diseño.
- Actualizaciones preliminares.

Suele estar alojado:

- En equipo local.
- En servidor interno.
- En subdominio privado.

Nota

No está destinado a usuarios finales.

Por otra parte, el entorno de producción es el entorno público o corporativo operativo. Debe caracterizarse por:

- Estabilidad.
- Seguridad reforzada.
- Control de accesos.
- Monitorización constante.
- Copias de seguridad activas.

Cualquier cambio en producción debe estar previamente validado.

Entorno intermedio: staging

Entre desarrollo y producción suele existir un entorno intermedio denominado **staging** o entorno de pruebas. Su finalidad es reproducir el entorno real con las mismas características técnicas, permitiendo validar:

- Actualizaciones del CMS.
- Cambios de versión PHP.
- Instalación de nuevos módulos.
- Migraciones.
- Modificaciones estructurales.

Este entorno debe ser lo más similar posible a producción en:

- Versión de servidor web.
- Configuración de base de datos.
- Parámetros de memoria.
- Sistema operativo.
- Estructura de directorios.

Nota

Un entorno de pruebas con configuraciones distintas a producción puede ocultar errores que aparecerán únicamente al desplegar los cambios reales.

Arquitecturas habituales de separación

La separación puede implementarse de distintas formas según recursos disponibles:

- **Separación física:**
  - Servidores distintos.
  - Bases de datos independientes.

  - Infraestructura replicada.
  - Mayor seguridad y aislamiento.
- **Separación lógica:**
  - Subdominios (ej. staging.empresa.com).
  - Carpetas independientes.
  - Bases de datos diferenciadas.
  - Usuarios distintos.
  - Menor coste, pero exige control riguroso de permisos.

Flujo de trabajo recomendado

Un procedimiento estructurado reduce riesgos en producción.

1. Desarrollo de cambios en entorno local.
2. Pruebas funcionales internas.
3. Migración a entorno staging.
4. Validación técnica y funcional.
5. Aprobación.
6. Despliegue controlado en producción.
7. Verificación post-despliegue.
8. Monitorización durante periodo inicial.

Este flujo evita intervenciones improvisadas.

Control de versiones y gestión de cambios

Aunque en entornos formativos no siempre se implementa formalmente, en entornos profesionales es recomendable utilizar sistemas de control de versiones para:

- Registrar cambios en código.
- Permitir revertir modificaciones.
- Facilitar trabajo colaborativo.
- Mantener trazabilidad.

La gestión de cambios debe documentar:

- Fecha.
- Responsable.
- Motivo.
- Versión afectada.
- Resultado.

Gestión de bases de datos entre entornos

La base de datos requiere especial atención al migrar entre entornos:

- No utilizar datos reales en entorno público de pruebas.
- Evitar sobrescribir producción accidentalmente.
- Mantener copias actualizadas para pruebas realistas.
- Controlar credenciales diferenciadas.

Se expone una tabla comparativa de gestión de datos:

Aspecto	Desarrollo	Producción
Datos reales	No recomendado	Sí
Copias frecuentes	Opcional	Obligatorio
Acceso restringido	Flexible	Estricto
Impacto de errores	Bajo	Alto

La protección de datos personales debe considerarse especialmente en entornos corporativos.

Pruebas antes de desplegar en producción

Antes de aplicar cambios definitivos deben validarse:

- Funcionamiento completo de la aplicación.
- Compatibilidad de plugins.
- Rendimiento.
- Integración con servicios externos.
- Seguridad (HTTPS activo).
- Permisos correctos.

También es recomendable revisar:

- Logs de errores.
- Consola del navegador.
- Consumo de recursos.

Ejemplo

Una empresa necesita actualizar WordPress a una versión mayor.

Procedimiento aplicado:

1. Copia completa de producción.
2. Restauración en entorno staging.
3. Actualización del núcleo en staging.
4. Verificación de compatibilidad de plugins.
5. Pruebas funcionales completas.
6. Ajustes de errores detectados.
7. Programación de ventana de mantenimiento.
8. Aplicación en producción.
9. Monitorización posterior.

Durante la prueba se detecta incompatibilidad de un plugin crítico. Se sustituye por alternativa antes de desplegar en producción. Este procedimiento evita interrupciones inesperadas.

Riesgos de no separar entornos

Cuando no existe separación adecuada, pueden producirse:

- Caídas del sitio en horario laboral.
- Exposición accidental de información.
- Errores visibles para clientes.
- Pérdida de confianza.
- Inestabilidad por pruebas improvisadas.

Fallo visible en sitio web como consecuencia de modificaciones no validadas previamente en entorno de pruebas.

La ausencia de entornos diferenciados suele asociarse a organizaciones con bajo nivel de madurez tecnológica.

Buenas prácticas operativas

Una gestión adecuada de entornos implica:

- Documentar configuraciones.
- Sincronizar versiones.
- Mantener copias independientes.
- Limitar acceso a entornos de prueba.
- Evitar indexación de entornos staging.
- Controlar credenciales diferenciadas.
- Registrar cambios aplicados.

Estas prácticas profesionalizan la administración del sistema.

La diferenciación entre entornos de desarrollo, pruebas y producción constituye una práctica esencial en el despliegue y administración de aplicaciones web. Esta separación permite minimizar riesgos, garantizar estabilidad y mantener control sobre los cambios realizados. Para el técnico en sistemas, gestionar correctamente estos entornos mejora la calidad del servicio, y refuerza la seguridad y continuidad operativa de la infraestructura web corporativa.

9. Resumen.

El despliegue de una aplicación web consiste en trasladar un sistema ya desarrollado a un entorno operativo preparado para su uso real, asegurando su correcta ejecución, integración y seguridad. No se limita a copiar archivos en un servidor, sino que implica configurar dependencias, establecer permisos, adaptar parámetros de entorno y verificar la compatibilidad entre servidor web, motor de ejecución y base de datos. Un despliegue adecuado garantiza estabilidad inicial; una mala planificación puede generar fallos estructurales desde el primer momento.

El proceso de despliegue debe contemplar la configuración de dominios, directorios virtuales o VirtualHost, certificados digitales para habilitar HTTPS, conexión con bases de datos y validación de rutas internas de la aplicación. También resulta fundamental ajustar parámetros del entorno (límites de memoria, tamaño máximo de subida de archivos, tiempos de ejecución), que influyen directamente en el rendimiento y la experiencia del usuario.

Una vez desplegada, la aplicación requiere administración continua. Esta administración incluye la gestión de usuarios y roles, la supervisión de permisos, la revisión de logs, la aplicación de actualizaciones y parches de seguridad, y la monitorización del consumo de recursos. La estabilidad de un entorno web depende más del mantenimiento constante que del despliegue inicial.

La gestión de copias de seguridad constituye un pilar esencial de la administración. Deben planificarse respaldos periódicos tanto de archivos como de bases de datos, verificando su integridad y capacidad de restauración. Una copia no comprobada no garantiza recuperación. En entornos profesionales, se establecen políticas de retención y procedimientos documentados de restauración ante incidentes.

El control de versiones y la gestión de cambios forman parte del ciclo de vida de la aplicación. Antes de aplicar actualizaciones en producción, deben realizarse pruebas en entornos controlados para evitar interrupciones del servicio. La ausencia de pruebas puede provocar incompatibilidades entre versiones de la aplicación, del servidor o de las extensiones instaladas.

La administración también implica la supervisión del rendimiento. El análisis de consumo de CPU, memoria, espacio en disco y número de conexiones activas permite anticipar problemas de saturación. En caso de aumento de demanda, pueden aplicarse medidas de escalabilidad vertical (ampliación de recursos) u horizontal (distribución de carga entre varios servidores).

En aplicaciones modernas, la integración con servicios externos y APIs exige controlar credenciales, tokens de acceso y configuraciones de conexión. Un error en la gestión de integraciones puede afectar funcionalidades críticas, como sistemas de pago o servicios de autenticación.

La seguridad durante la administración incluye la aplicación de políticas de hardening, desactivación de componentes innecesarios, control de accesos remotos y revisión de vulnerabilidades. La exposición pública de una aplicación web la convierte en un objetivo potencial, por lo que la actualización constante y la vigilancia de registros son medidas imprescindibles.

En entornos basados en CMS o plataformas de gestión de contenidos, la administración incorpora tareas adicionales como la instalación y actualización de plugins, la revisión de compatibilidad entre extensiones y la eliminación de componentes obsoletos. Cada elemento añadido amplía la superficie de riesgo si no se controla adecuadamente.

El modelo de despliegue puede variar según el contexto: infraestructura local, VPS, PaaS o SaaS. En todos los casos, aunque parte de la infraestructura esté externalizada, la responsabilidad sobre la correcta configuración funcional y la gestión de usuarios recae en la organización.

El despliegue y la administración de aplicaciones web constituyen un proceso continuo orientado a garantizar disponibilidad, integridad, seguridad y rendimiento. No se trata de una acción puntual, sino de una actividad permanente que combina configuración técnica, control de cambios, monitorización y mantenimiento preventivo para asegurar la continuidad del servicio en entornos profesionales.

10. Prueba de autoevaluación.

1. ¿Qué se entiende por despliegue de una aplicación web?

a) El proceso de instalar, configurar y poner en funcionamiento la aplicación en un entorno servidor.

b) El diseño gráfico de la interfaz de usuario.

c) La creación del código fuente inicial.

d) La eliminación de versiones anteriores del sistema operativo.

2. ¿Cuál es una práctica recomendada antes de desplegar una aplicación en producción?

a) Publicarla directamente en el servidor principal sin pruebas previas.

b) Realizar pruebas en un entorno de preproducción o pruebas.

c) Desactivar los mecanismos de autenticación para evitar errores.

d) Eliminar los registros de actividad anteriores.

3. ¿Qué función cumple un entorno de preproducción (staging)?

a) Sustituir definitivamente al entorno de producción.

b) Servir como copia de seguridad automática.

c) Permitir pruebas y validaciones antes del paso a producción.

d) Actuar como servidor DNS principal.

4. En la administración de aplicaciones web, la gestión de permisos y usuarios permite:

a) Aumentar la velocidad de conexión.

b) Controlar el acceso a funciones y recursos según roles definidos.

c) Reducir el tamaño del código fuente.

d) Evitar la necesidad de copias de seguridad.

5. ¿Qué objetivo tiene el hardening en un entorno web?

a) Incrementar el número de servicios activos por defecto.

b) Mejorar el diseño visual del panel de administración.

c) Reducir vulnerabilidades mediante configuración segura y eliminación de elementos innecesarios.

d) Automatizar el desarrollo del código.

6. ¿Qué herramienta o práctica facilita la recuperación ante fallos graves en una aplicación web?

a) Desactivar el registro de logs.

b) Implementar copias de seguridad periódicas y verificadas.

c) Eliminar cuentas de usuario inactivas únicamente.

d) Utilizar exclusivamente HTTP sin cifrado.

7. ¿Cuál es la finalidad de un balanceador de carga en entornos con alta demanda?

a) Diseñar automáticamente la base de datos.

b) Distribuir el tráfico entre varios servidores para mejorar disponibilidad y rendimiento.

c) Sustituir al servidor web principal.

d) Reducir el número de usuarios conectados.

8. En la gestión de aplicaciones web, la monitorización del sistema permite:

a) Detectar incidencias de rendimiento y posibles fallos antes de que afecten gravemente al servicio.

b) Eliminar la necesidad de actualizaciones.

c) Sustituir las políticas de seguridad.

d) Evitar el uso de certificados digitales.

9. ¿Por qué es importante documentar el proceso de despliegue y configuración?

a) Para reducir el tamaño de la base de datos.

b) Para facilitar la repetición del proceso, el mantenimiento y la resolución de incidencias.

c) Para eliminar la necesidad de formación técnica.

d) Para sustituir las pruebas de funcionamiento.

10. ¿Qué riesgo implica desplegar directamente en producción sin pruebas previas?

a) Mejora automática del rendimiento.

b) Reducción de la carga del servidor.

c) Aumento de la seguridad del sistema.

d) Posible interrupción del servicio y aparición de errores no detectados.

Unidad 4

Seguridad en aplicaciones web

Las aplicaciones web están expuestas de forma permanente a riesgos derivados de su accesibilidad a través de redes públicas. Las vulnerabilidades pueden afectar tanto al servidor como a la propia aplicación, y su explotación puede provocar pérdida de información, interrupciones del servicio o compromisos de seguridad más amplios dentro de la infraestructura.

En esta unidad se examinan las principales amenazas asociadas a entornos web y se desarrollan medidas técnicas orientadas a la protección y el fortalecimiento del sistema. Se abordan aspectos como el uso de certificados digitales, la configuración segura del servidor, la protección de gestores de contenidos y la implementación de copias de seguridad. El objetivo es integrar criterios de seguridad desde la fase de despliegue hasta el mantenimiento continuo.

1. Principales vulnerabilidades web.

Las aplicaciones web están expuestas de forma permanente a Internet o a redes corporativas, lo que las convierte en objetivos constantes de ataques automatizados y dirigidos. A diferencia de los sistemas cerrados, una aplicación web interactúa con entradas de usuario en tiempo real, procesa datos dinámicos y se comunica con bases de datos y servicios externos. Esta exposición aumenta la superficie de ataque y exige comprender las vulnerabilidades más frecuentes.

Entre las amenazas más relevantes se encuentran la **inyección SQL (SQL Injection)**, el **Cross-Site Scripting (XSS)**, los ataques de **fuerza bruta** y el **Cross-Site Request Forgery (CSRF)**. No es necesario profundizar en su explotación técnica avanzada, pero sí comprender su lógica conceptual, su impacto y las medidas generales de prevención.

Inyección SQL (SQL Injection)

La **inyección SQL** se produce cuando una aplicación web no valida correctamente los datos introducidos por el usuario antes de incorporarlos a una consulta a la base de datos. En este escenario, el atacante puede introducir código SQL malicioso que altera la consulta original.

En una aplicación web típica, un formulario de inicio de sesión puede generar una consulta que verifica usuario y contraseña en la base de datos. Si la aplicación inserta directamente los datos introducidos en la consulta sin validación ni parametrización, un atacante podría modificar la lógica de la consulta.

El problema no reside en la base de datos en sí, sino en la **construcción insegura de consultas** dentro de la aplicación.

Cuando la inyección SQL es posible, el atacante puede:

- Acceder a datos confidenciales.
- Modificar registros.
- Eliminar información.
- Crear usuarios administrativos.
- Extraer grandes volúmenes de datos.

En aplicaciones corporativas (CRM, ERP), el impacto puede ser crítico.

Una clasificación conceptual de consecuencias es:

Nivel de impacto	Ejemplo
Acceso indebido	Inicio de sesión sin credenciales válidas
Exposición de datos	Descarga de información de clientes
Manipulación	Alteración de precios o facturación
Destrucción	Eliminación de tablas completas