UD2 Introducción a las arquitecturas web y servidores

6. Recursos necesarios para ejecutar una aplicación web

Desplegar una aplicación web no consiste únicamente en escribir código: también se necesitan diversos recursos, tanto **materiales** (hardware, red) como **lógicos** (software, servicios), que permitan poner en marcha el sistema de forma funcional, segura y escalable. En este apartado analizaremos los elementos imprescindibles que hacen posible que una aplicación web esté disponible para los usuarios finales.

6.1. Un entorno de ejecución

Toda aplicación necesita un entorno donde pueda ejecutarse. Esto puede ser:

- **Servidor físico** (bare metal): un ordenador dedicado exclusivamente al alojamiento.
- **Servidor virtual**: una máquina virtual dentro de un sistema de virtualización (VMware, VirtualBox, etc.).
- **Contenedor** (Docker, Podman…): un entorno ligero que empaqueta la aplicación con todas sus dependencias.
- **Plataforma cloud** (PaaS): proveedores como Heroku, Vercel o AWS que gestionan la infraestructura automáticamente.

Este entorno debe contar con el **sistema operativo adecuado**, generalmente Linux en entornos de producción web, aunque también se puede usar Windows Server para aplicaciones específicas.

6.2. Servidor web

Es el software encargado de recibir las peticiones HTTP/HTTPS y responder con los recursos solicitados. Actúa como punto de entrada principal de la aplicación. Algunos ejemplos comunes son:

- **Apache HTTP Server**: muy extendido, altamente configurable.
- **Nginx**: muy eficiente y moderno, ideal para alto tráfico y proxy inverso.
- **Caddy**: menos conocido, pero destaca por su configuración sencilla y certificados HTTPS automáticos.

El servidor web puede:

- Servir archivos estáticos (HTML, imágenes, CSS).
- Redirigir o reenviar peticiones a otros servicios (backend, API).
- Implementar seguridad, compresión, caché, etc.

6.3. Lenguaje y motor de ejecución

Si la aplicación es dinámica, necesita un **lenguaje de programación del lado servidor** y su correspondiente motor o intérprete. Ejemplos:

- **PHP**: ejecutado mediante `php-fpm` o integrado en Apache con mod_php.
- **Java**: desplegado en un servidor de aplicaciones como Tomcat o WildFly.
- **Python**: ejecutado con WSGI en combinación con servidores como Gunicorn.
- **Node.js**: servidor propio, basado en JavaScript y orientado a aplicaciones en tiempo real.
- **Ruby (on Rails)**: requiere un servidor de aplicaciones como Puma o Passenger.

Además del lenguaje, la aplicación puede requerir:

- Librerías o frameworks (Laravel, Express.js, Django, etc.).
- Gestores de dependencias (npm, pip, Composer, etc.).

6.4. Base de datos

Muchas aplicaciones web necesitan persistir datos. Para ello utilizan un **sistema de gestión de bases de datos (SGBD)**, que puede ser:

- **Relacional (SQL)**: como MySQL, PostgreSQL o MariaDB.
- **No relacional (NoSQL)**: como MongoDB, Redis o Firebase.

Es importante gestionar correctamente:

- La creación y migración de esquemas.
- El acceso concurrente.
- La autenticación de usuarios y permisos.
- Las copias de seguridad y la recuperación ante fallos.

6.5. Servicios de red

Una aplicación web depende también de otros servicios de red auxiliares, como:

- **DNS**: para que el dominio sea resolvible.
- **Firewall / NAT**: para proteger el servidor y enrutar correctamente las conexiones.
- **Certificados SSL/TLS**: imprescindibles para habilitar HTTPS (HTTP seguro).
- **SMTP** (correo saliente): si la aplicación envía correos electrónicos.
- **FTP/SFTP**: en algunos casos, para subir archivos al servidor.

6.6. Dominio y configuración del DNS

Para que una aplicación sea accesible desde Internet, es necesario registrar un **nombre de dominio** (por ejemplo, `midominio.com`) y configurarlo en un sistema de **DNS público**, de forma que apunte a la IP del servidor.

También puede configurarse un **subdominio** (por ejemplo, `app.midominio.com`) para distinguir diferentes servicios o entornos (producción, pruebas, etc.).

6.7. Seguridad y usuarios

Una aplicación en producción debe estar **protegida**:

- Mediante **autenticación** y control de acceso.
- Usando **HTTPS para cifrar las comunicaciones**.
- Con **firewalls** y herramientas de detección de intrusos.
- Actualizando regularmente el software.
- Aplicando principios como el de **mínimo privilegio** (solo dar los permisos estrictamente necesarios).

6.8. Monitorización y logs

Es esencial poder saber cómo está funcionando la aplicación y detectar errores. Para ello se utilizan herramientas de:

- **Logs de servidor**: acceso, errores, peticiones.
- **Monitorización**: recursos del sistema (CPU, RAM, red), estado del servidor.
- **Alertas**: notificaciones automáticas ante caídas o problemas.

Plataformas como Grafana, Prometheus o ELK Stack permiten crear sistemas avanzados de observabilidad.

6.9. Herramientas de despliegue y automatización

En entornos modernos, se utilizan herramientas que facilitan el despliegue:

- **Git**: para llevar el control de versiones.
- **CI/CD** (integración y entrega continuas): como GitHub Actions o GitLab CI.
- **Docker / Docker Compose**: para aislar entornos y simplificar dependencias.
- **Ansible / Terraform**: para definir infraestructura como código.

Estas herramientas ayudan a mantener entornos consistentes, reproducibles y escalables.

6.10. Conclusión

Como has podido comprobar, una aplicación web necesita mucho más que código fuente para funcionar. Desde el sistema operativo hasta los certificados SSL, pasando por el DNS, la base de datos y el servidor web, todo debe estar correctamente configurado y mantenido.

Este conjunto de recursos constituye lo que se conoce como el **entorno de ejecución** o **infraestructura de despliegue** de la aplicación, y dominarlo es esencial para cualquier profesional del desarrollo y la administración de sistemas web.