

Ciclo de vida del software

Requerimientos – ¿Qué necesita el cliente?

Análisis – ¿Qué datos se usarán? ¿Qué procesos?

Diseño – Pantallas, diagramas, arquitectura.

Desarrollo – Código.

Testing – Ver si funciona.

Implementación – Subirlo a producción.

Mantenimiento – Correcciones y mejoras.

Dominio y DNS

- **Dominio:** Es un nombre fácil de recordar (ej: lacochoa.gob.ar) que apunta a un servidor en Internet.
- **DNS (Domain Name System):** Es como la “guía telefónica” de Internet: traduce nombres de dominio a direcciones IP.
 - Ejemplo: cuando alguien entra a expedientes.lacochoa.gob.ar, el DNS resuelve a una **IP pública** que corresponde a tu servidor.

IP Pública vs IP Privada

- **IP Pública:** Dirección única en Internet, accesible desde cualquier lugar del mundo. Ejemplo: 186.33.211.175
- **IP Privada:** Dirección interna, solo visible dentro de la red privada/local. Ejemplo: 192.168.150.115.

HTTP y HTTPS

- **HTTP (HyperText Transfer Protocol):** Es el protocolo base de la web. Permite que navegadores y servidores se comuniquen.
 - Inseguro: la información viaja “en texto plano”.
- **HTTPS (HTTP Secure):** Es HTTP pero cifrado con un **certificado SSL/TLS**.
 - Seguro: protege credenciales, datos personales, y evita que se “espíe” la comunicación.

Certificado SSL

- **Qué es:** Un archivo digital emitido por una entidad certificadora (ej: Let's Encrypt) que valida la identidad del servidor y cifra la comunicación con el cliente.
- **Para qué sirve:**
 - Asegura que el usuario realmente se conecta al servidor correcto.
 - Cifra la información (ej: usuarios, contraseñas, datos personales).

Virtualización

- **Qué es:** Una tecnología que permite crear varias “máquinas virtuales” (VMs) independientes dentro de un mismo servidor físico.
- **Ventajas:**
 - Mejor aprovechamiento de recursos.
 - Aislamiento (cada VM puede tener su propio sistema operativo y aplicaciones).
 - Escalabilidad: se pueden crear/eliminar VMs según necesidad.

Servidor Web

Un **servidor web** es un software que atiende las peticiones de los navegadores (clientes). Recibe solicitudes HTTP/HTTPS y responde con páginas HTML, APIs o archivos.

Ejemplos:

- **Apache HTTP Server** □
 - Muy popular, flexible, usado en la mayoría de hostings.
 - Bueno para PHP, sitios dinámicos.
 - Basado en módulos.

- **Nginx** □
 - Ligero, rápido, eficiente en concurrencia.
 - Ideal como **proxy inverso** y para aplicaciones modernas (APIs, microservicios).
 - En tu caso, usás **Nginx Proxy Manager**.
- **Tomcat** □
 - Orientado a aplicaciones Java (JSP, Servlets).
 - Usado en entornos empresariales que corren sistemas Java.

Frontend (lo que llega al navegador)

“Todo lo que ven en la pantalla —botones, formularios, colores, textos, animaciones— es el **frontend**.

Es lo que el servidor entrega al navegador.”

Se ocupa de:

- Diseño visual
- Interfaz de usuario
- Experiencia de usuario
- Animaciones
- Interacción (botones, formularios, clics)

Es “la cara” del sistema.

Backend (la lógica que NO se ve)

Explicación:

“Detrás de cada página, hay lógica que procesa datos, valida usuarios, guarda información, envía correos.

Eso es el **backend**, que vive en el servidor.”

Se ocupa de:

- Autenticación
- Procesos de negocio
- Validación
- Seguridad
- Conexiones con bases de datos
- APIs
- Lógica interna

Es “el cerebro” del sistema.

Bases de Datos

Una **base de datos** guarda, organiza y permite consultar la información de forma estructurada.

Tipos:

- **Relacionales (SQL):**
 - Ejemplos: MySQL, PostgreSQL, Oracle, SQL Server.
 - Organizan los datos en tablas con filas y columnas.
 - Lenguaje: SQL.
 - En tu caso: **MySQL** (para Expedientes, Patrimonio).
- **NoSQL:**
 - Ejemplos: MongoDB, Redis, Cassandra.
 - Flexibles, pensadas para datos grandes o poco estructurados.

Dónde encajan IP, dominio, servidor, DNS?

1. El usuario escribe un dominio en el navegador.
2. DNS lo traduce a una IP pública.

3. El navegador se conecta al servidor web (Apache/Nginx).
4. El servidor web manda:
 - El **frontend** (HTML, CSS, JS).
 - Y cuando el frontend necesita datos, hace pedidos al **backend**.
5. El backend pregunta a la **base de datos**.
6. El backend responde al frontend.
7. El frontend lo muestra al usuario.

¿Qué es un Hosting?

- **Definición:**

Es un servicio que te alquila un espacio en un **servidor físico** para alojar tu página web.

Ejemplo: Hostinger, GoDaddy, Donweb, etc.
- **Características principales:**
 - Está pensado para **webs simples** (blogs, tiendas online, páginas institucionales).
 - El proveedor administra el servidor, vos solo subís tus archivos (PHP, WordPress, etc.).
 - Recursos compartidos (CPU, RAM, ancho de banda).
 - Limitado en personalización (no podés instalar cualquier software).
 - Incluye paneles fáciles (cPanel, hPanel).
- **Ventajas:**
 - Fácil de usar, ideal para principiantes.
 - Bajo costo.
 - Incluye correo, SSL básico, backups.
- **Desventajas:**
 - Poca flexibilidad.
 - Compartís recursos con otros clientes (menos rendimiento).
 - No sirve para proyectos grandes o complejos.

Qué es un Cloud (Computación en la nube)?

- **Definición:**

Es un servicio que te permite crear **máquinas virtuales** (VMs) dentro de un **data center**, usando virtualización.

Ejemplo: AWS, Azure, Google Cloud, ARSAT Nube.
- **Características principales:**
 - Podés elegir sistema operativo, instalar lo que quieras.
 - Escalabilidad: creás, eliminás o ampliás VMs según la demanda.
 - Acceso a recursos dedicados (CPU, RAM, discos, redes).
 - Mayor seguridad: firewall, VPN, IPs públicas propias.
- **Ventajas:**
 - Total flexibilidad (vos administrás todo).
 - Escalable a nivel empresarial.
 - Ideal para correr **sistemas complejos** (como los tuyos de Expedientes, Patrimonio).
- **Desventajas:**
 - Requiere más conocimientos técnicos.
 - Puede ser más costoso que un hosting compartido.
 - Sos responsable de la configuración, seguridad y mantenimiento.

Comparación simple (Hosting vs Cloud)

Característica	Hosting Tradicional	Cloud Computing (Nube)
Dónde corre	Servidor compartido	Máquinas virtuales en la nube
Flexibilidad	Baja (solo PHP, WordPress)	Alta (instalás lo que quieras)
Control	Mínimo	Total (sos admin de tus VMs)
Escalabilidad	Cambiando de plan	Creando/ajustando VMs
Costo	Bajo	Variable (según recursos)
Ideal para	Webs pequeñas, blogs	Sistemas críticos, empresas, apps municipales

Arquitectura Básica de un sistema en la nube bajo un cloud

Cloudflare

- **Qué es:** Una empresa que ofrece **DNS, seguridad y CDN** a nivel global.
- **Funcionalidades:**
 - DNS avanzado.
 - Proxy inverso (oculta la IP real del servidor).
 - Seguridad (WAF, protección contra DDoS).
 - Optimización (caché y aceleración de contenidos).

ARSAT (Nube Básica)

- **Qué es:** Empresa estatal argentina que brinda **data center, internet y nube soberana**.
- **En general:**
 - Permite contratar servidores virtuales en Argentina.
 - Incluye IPs públicas, firewall, VPN, etc.
 - ARSAT te da la **IP pública (ejemplo 186.33.211.175)** que conecta tu dominio al servidor.

Nginx Proxy Manager

- **Qué es:** Una herramienta que simplifica la gestión de **Nginx** (servidor web) con interfaz gráfica.
- **Funciones:**
 - Proxy inverso → recibe las peticiones y las envía al servidor correspondiente.
 - Manejo de certificados SSL (Let's Encrypt).
 - Gestión de accesos y redirecciones.

Comparativa: Infraestructura Cloud vs Hosting Tradicional (ej. Hostinger)

- Si contratás **Hostinger**:
 - Te dan un Apache con PHP y MySQL listo.
 - Solo subís tu web.
 - No podés instalar Nginx ni Tomcat, ni elegir otra base.
- En tu **Cloud con ARSAT**:

- Tenés VMs propias.
- Instalás Nginx Proxy Manager en una, Laravel + Nginx/Apache en otra, MySQL en otra.
- Podés crecer agregando más VMs o cambiando tecnologías (ej: usar PostgreSQL en lugar de MySQL).

Característica	Tu Arquitectura (Cloudflare + ARSAT + ProxyManager)	Hosting Tradicional (Hostinger)
Control	Total: vos decidís todo (SO, apps, firewall)	Limitado: solo lo que permite el plan
Virtualización	Varias VMs separadas con roles distintos	No ves VMs, es un servidor compartido
IP Pública	Fija, manejada en ARSAT	Compartida o dedicada según plan
Seguridad	Cloudflare + firewall ARSAT + SSL automático	SSL básico, seguridad del proveedor
Certificados SSL	Let's Encrypt automático para cada subdominio	Gratis o pago según el plan
Escalabilidad	Podés crear nuevas VMs o ampliar recursos	Escalable solo cambiando de plan
HTTP/HTTPS	Todo forzado a HTTPS con SSL	HTTP/HTTPS básico
Ideal para	Sistemas críticos, varias apps municipales	Webs simples, blogs, tiendas pequeñas
Responsabilidad técnica	Alta: vos administrás servidores	Baja: el hosting administra todo