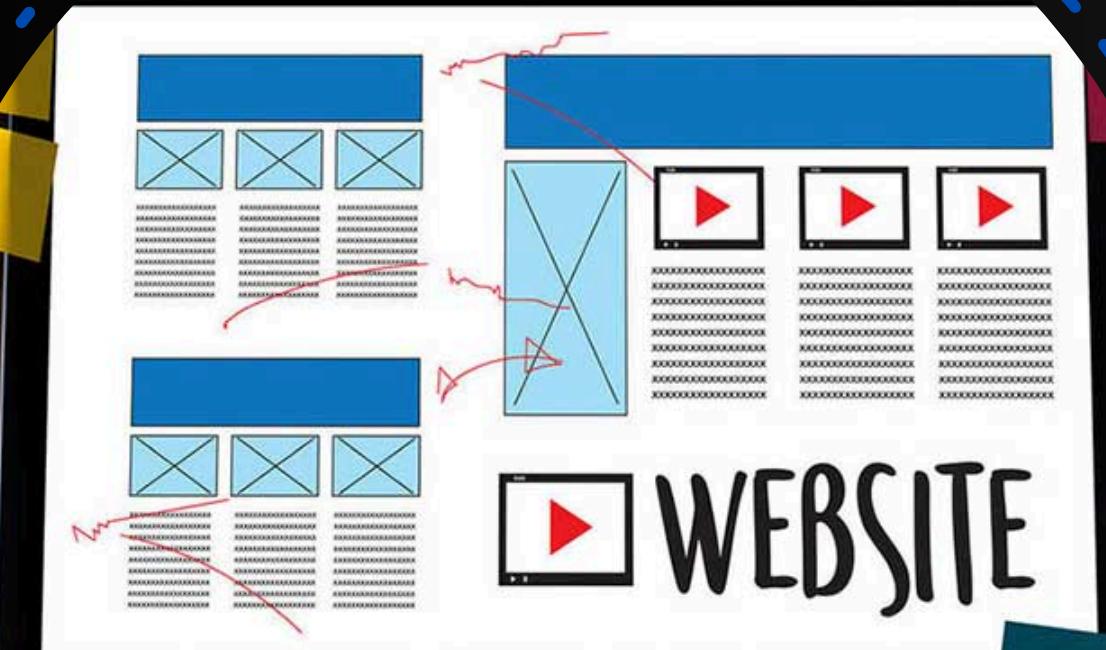


# HTTP, APACHE Y ARQUITECTURAS WEB

Miguel Fernandez Martín, Diego Rivas González  
y Angel Campo Sanchez-Rey

# ÍNDICE

1. Protocolo HTTP
2. Arquitecturas Web
3. Servidor Web Apache



# Introducción

Para que las páginas web funcionen correctamente, necesitan tres elementos fundamentales:

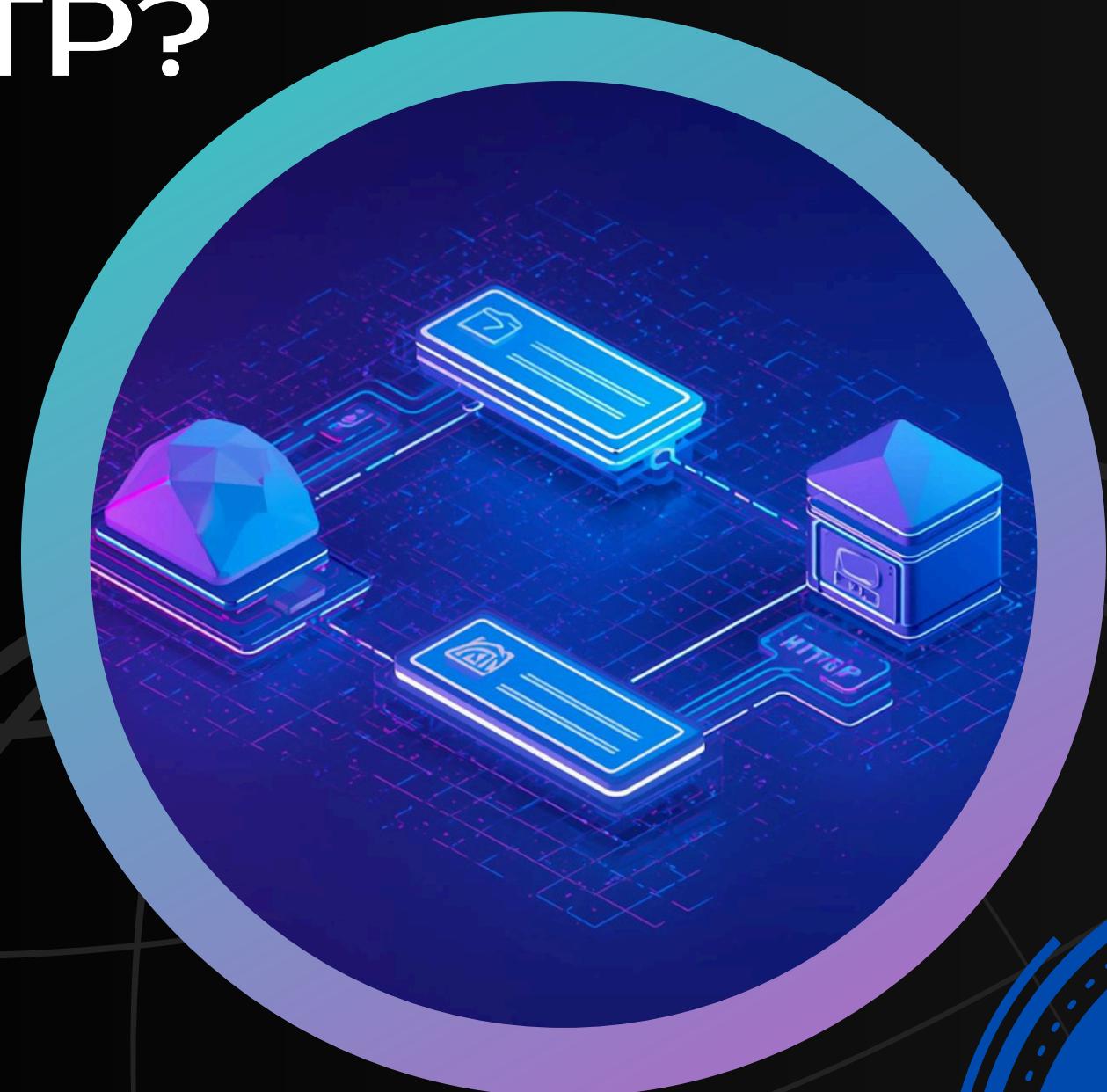
- **Protocolo de comunicación HTTP**
- **Servidor Apache**
- **Arquitectura Web**

Estos tres elementos trabajan juntos para garantizar que la información llegue de manera clara, rápida y ordenada.

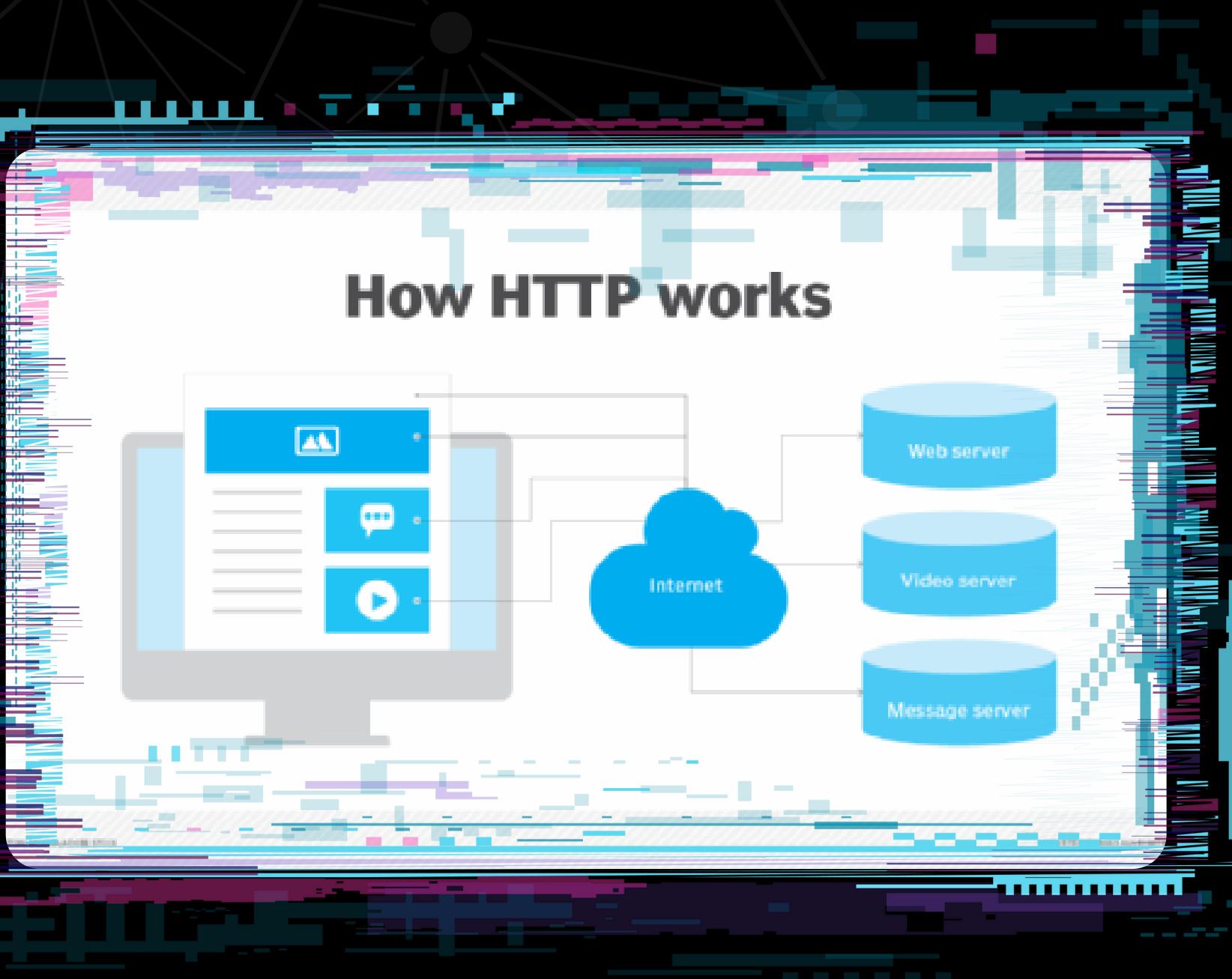


# QUE ES EL PROTOCOLO HTTP?

El protocolo HTTP (Protocolo de Transferencia de Hipertexto) es un conjunto de reglas que permite a los navegadores web solicitar y recibir recursos (como páginas HTML, imágenes o videos) desde los servidores de Internet, operando bajo un modelo cliente-servidor.



# FUNCIONAMIENTO



## PASOS DE LA COMUNICACIÓN:

- 1-🌐 El cliente envía una petición HTTP mediante la URL la cual incluye un método (GET, POST, PUT o DELETE).
- 2-💻 El servidor recibe y procesa la petición mediante una conexión TCP y procesa la solicitud.
- 3-⟳ El servidor envía una respuesta HTTP, que incluye un código de estado:
  - ✓ 2xx: éxito
  - ⚠ 4xx: error del cliente
  - ✗ 5xx: error del servidor
- 4-💻 El navegador interpreta y muestra la respuesta renderizando el código HTML o cerrando la conexión TCP.

# CARACTERÍSTICAS

## **Modelo Cliente-Servidor**

HTTP se basa en un intercambio directo de peticiones y respuestas entre un cliente y un servidor.

## **Protocolo sin estado**

Cada solicitud y respuesta es independiente lo que significa que el servidor no guarda información sobre comunicaciones anteriores haciéndolo mas eficiente.

## **Transferencia de hipertexto**

Permite el envío de información compleja como texto, imágenes y videos dando forma a una pagina web.

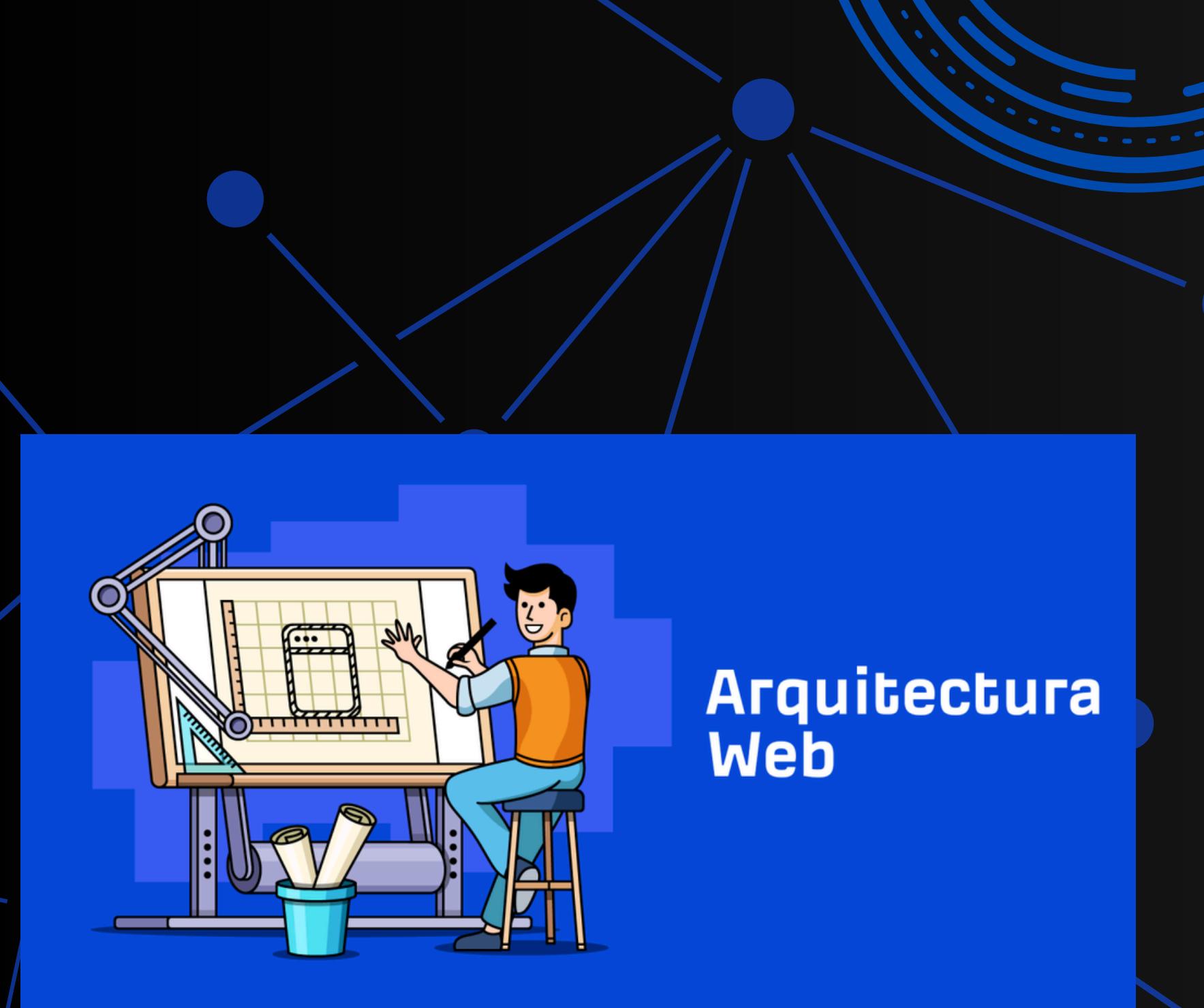
# COMPARATIVA

HTTP es el lenguaje que usan los navegadores y servidores web para comunicarse. HTTPS es una versión segura de HTTP que hace exactamente lo mismo (comunicación entre navegador y servidor), pero los datos viajan cifrados mediante un certificado SSL/TLS.

Característica	HTTP 	HTTPS 
<b>Significado</b>	<i>HyperText Transfer Protocol</i>	<i>HyperText Transfer Protocol Secure</i>
<b>Seguridad</b>	No cifra los datos; la información viaja en texto plano.	Cifra los datos mediante <b>SSL/TLS</b> , protegiendo la información.
<b>URL</b>	Comienza con http://	Comienza con https://
<b>Privacidad</b>	Los datos pueden ser interceptados fácilmente por atacantes.	La comunicación está cifrada, evitando espionaje o manipulación.
<b>Integridad de datos</b>	No garantiza que los datos lleguen sin alteraciones.	Garantiza que los datos no se modifiquen durante la transmisión.
<b>Autenticación del servidor</b>	No hay verificación de identidad.	Usa certificados digitales para autenticar al servidor.
<b>Uso recomendado</b>	Solo para sitios sin información sensible (por ejemplo, páginas informativas).	Para cualquier sitio, especialmente los que manejan datos personales, contraseñas o pagos.
<b>Impacto en SEO</b>	Menor prioridad en buscadores.	Google favorece a los sitios con HTTPS.

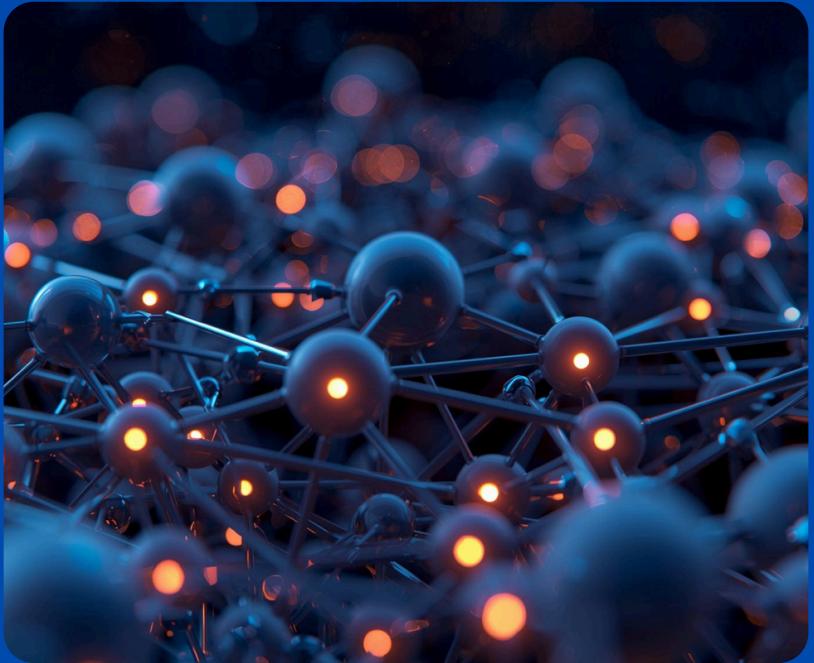
# ARQUITECTURA WEB

1. Componentes principales
2. Tipos comunes
3. Tendencias actuales



# COMPONENTES PRINCIPALES

Las arquitecturas web se refieren a cómo se distribuyen y se conectan los componentes para dar servicio al usuario.



## CLIENTE: NAVEGADOR Y APLICACIÓN MÓVIL

Los clientes son esenciales para la interacción con servicios web.

- Navegador.
- Envía peticiones HTTP.
- Muestra contenido (HTML, CSS, JavaScript).



## SERVIDOR WEB: PROVEEDOR DE RECURSOS

Gestiona las solicitudes y recursos de los clientes:

- Recibe la petición.
- Procesa la solicitud.
- Devuelve respuesta
- (página web, un archivo, datos en JSON, etc.).



## BASE DE DATOS: ALMACÉN INFORMACIÓN

Las bases de datos almacenan y proporcionan información a los servidores:

- Responde a las consultas que hace el servidor web.

# TIPOS DE ARQUITECTURAS WEB

Arquitectura	Descripción	Ventajas	Desventajas	Ejemplo
<b>1 capa (Monolítica)</b>	Interfaz, lógica y datos están en un solo servidor. Todo el sistema funciona como una única unidad.	Simple y fácil de implementar.	Difícil de escalar y mantener.	Páginas HTML estáticas servidas desde Apache.
<b>2 capas (Cliente-Servidor)</b>	El cliente se conecta directamente a un servidor que gestiona la aplicación y los datos.	Más organizada que la monolítica.	Limitada en seguridad y escalabilidad.	Aplicación que conecta directamente a una base de datos.
<b>3 capas</b>	Divide la aplicación en: presentación, lógica y datos. Cada capa tiene su función y comunicación independiente.	Mejor seguridad, mantenimiento y escalabilidad.	Mayor complejidad y sobrecarga de comunicación.	Aplicaciones modernas con servidor PHP, Java o Node.js.
<b>Modernas (Microservicios y Nube)</b>	La aplicación se divide en servicios pequeños e independientes desplegados en la nube.	Alta escalabilidad, independencia y flexibilidad.	Mayor complejidad y necesidad de coordinación entre servicios.	Aplicaciones distribuidas en AWS, Azure o Google Cloud.

# TENDENCIAS ACTUALES

## ARQUITECTURAS SIN SERVIDOR (SERVERLESS)

Permiten ejecutar funciones bajo demanda sin necesidad de administrar un servidor físico o virtual.

Ejemplo: AWS Lambda, Google Cloud Functions

## ARQUITECTURAS JAMSTACK

Basadas en JavaScript, APIs y Markup. Ofrecen alto rendimiento y seguridad al separar el frontend del backend.

Ejemplo: sitios creados con frameworks como Next.js, Gatsby...



## MAYOR SEGURIDAD



## RENDIMIENTO OPTIMIZADO

# SERVIDOR WEB APACHE

Cuando visitas una web, Apache puede ser el encargado de mostrarte lo que ves de la forma que lo haces y como lo haces.

## SERVIDOR WEB APACHE

COMUNICACIONES EN LA TERMINAL

# ¿QUÉ ES APACHE?



Apache es un software servidor web que gestiona y entrega páginas a los usuarios cuando acceden mediante un navegador.

Actualmente es uno de los servidores mas usados en el mundo

# ¿CÓMO FUNCIONA?

- Recepción de solicitudes.
- Envío de la solicitud.
- Procesamiento y respuesta.
- Entrega del contenido.



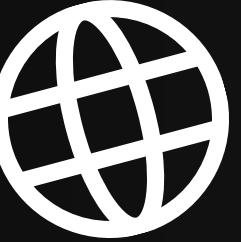
# CARACTERÍSTICAS PRINCIPALES

- Software libre y gratuito
- Multiplataforma
- Arquitectura modular
- Alojamiento virtual
- Soporte para programas de programación



# ¿DÓNDE SE USA?

- Alojamiento de sitios web
- Aplicaciones web y APIs
- Hosting de CMS y plataformas
- Entornos de desarrollo (DevOps)
- Entornos educativos y de pruebas



# VENTAJAS Y DESVENTAJAS



## Ventajas

- Estable y confiable
- Gran comunidad y soporte
- Facil de usar e instalar

## Desventajas:

- Mas lento en comparación con otros mas modernos
- Configuración manual
- Menos eficiente

# ¿POR QUÉ ES IMPORTANTE?



- Permite acceder a sitios desde cualquier parte del mundo
- Es una herramienta clave para aprender sobre servidores y redes.
- Entenderlo ayuda a comprender cómo se entrega el contenido en Internet.



 **Borracho.rar**

**¡Muchas Gracias!**