TALLER ARQUITECTURA DE APLICACIONES WEB

La arquitectura de aplicaciones web es fundamental para diseñar sistemas eficientes y escalables. Permite que los distintos componentes de una aplicación interactúen de manera coherente.

1. **¿Qué es la Arquitectura de las Aplicaciones Web?**

   La Arquitectura de Aplicaciones Web es el diseño conceptual de la estructura que define cómo se organizan, interactúan y se comunican los componentes, módulos y capas de una aplicación que se ejecuta sobre el protocolo HTTP (o HTTPS) y se accede a través de un navegador web.

   Funciona como un mapa estratégico que determina:

   - La separación de responsabilidades (capas).
   - La distribución del trabajo entre el cliente (navegador) y el servidor.
   - Cómo se gestionan los datos, la lógica de negocio y la interfaz de usuario.

2. **¿Por qué es importante la Arquitectura de las Aplicaciones Web?**

   La arquitectura web es crucial porque impacta directamente en la calidad, viabilidad a largo plazo y éxito de la aplicación. Su importancia radica en:

Aspecto	**Razón de Importancia**
Escalabilidad	Permite que la aplicación maneje un crecimiento en el número de usuarios o de tráfico sin degradar el rendimiento, ya sea aumentando recursos (escalado vertical) o añadiendo servidores (escalado horizontal).
Mantenibilidad	Una arquitectura bien definida, con separación clara de componentes, facilita que el código sea más fácil de entender, modificar y depurar.
Rendimiento	El diseño de la arquitectura (p. ej., la distribución de la carga de trabajo) influye directamente en la velocidad de respuesta y la experiencia del usuario.
Seguridad	Una estructura planificada permite implementar medidas de seguridad en distintas capas y puntos de control, protegiendo los datos y la lógica de negocio.

Reutilización	Los componentes bien delimitados pueden ser reutilizados en otras partes de la aplicación o en futuros proyectos.

3. **¿Cuáles son los Componentes de la Aplicación Web?**

Los componentes de una aplicación web se suelen dividir en tres grandes áreas o capas:

- **Cliente (Frontend):** Es la parte de la aplicación con la que el usuario interactúa directamente a través del navegador.
- **Interfaz de Usuario (UI):** Lo que el usuario ve y usa (HTML, CSS, imágenes, etc.).
- **Lógica del Cliente:** Código (generalmente JavaScript) que se ejecuta en el navegador para manejar eventos, validaciones y manipular el DOM.
- Comunicación: Realiza peticiones al servidor (generalmente vía AJAX o Fetch).

2. Servidor (Backend)

Es el corazón de la aplicación, donde reside la lógica de negocio y el procesamiento de datos.

- **Servidor Web (o Servidor HTTP):** Software que escucha las peticiones del cliente (ej. Apache, Nginx) y las dirige a la aplicación.
- **Lógica de Negocio:** El código de la aplicación propiamente dicho (ej. en Python, Java, Node.js, PHP) que procesa las peticiones, realiza cálculos, gestiona la autenticación, y decide qué hacer.
- **Capa de Acceso a Datos (DAL):** Componente que interactúa directamente con la base de datos para leer o escribir información.

3. Base de Datos (Data Store)

Es donde se almacena de forma persistente la información de la aplicación (usuarios, productos, transacciones, etc.).

- Sistema Gestor de Base de Datos (SGBD): Software (ej. MySQL, PostgreSQL, MongoDB) que administra el almacenamiento, recuperación y seguridad de los datos.

4. **Mencione las Mejores Prácticas para el desarrollo web**

- **Separación de Preocupaciones (SoC):** Dividir la aplicación en capas (Presentación, Lógica de Negocio, Datos) para que los cambios en una no afecten a las demás. El patrón MVC (Modelo-Vista-Controlador) es un ejemplo clave.
- **Desarrollo Orientado a Pruebas (TDD) / Integración Continua (CI):** Escribir pruebas unitarias y de integración para asegurar la calidad y automatizar la compilación y pruebas del código.
- **Diseño Responsivo (Responsive Design):** Asegurar que la interfaz de usuario se vea y funcione correctamente en cualquier dispositivo (móvil, tablet, escritorio).
- **Optimización del Rendimiento**:

    - ✓ Optimizar consultas a la base de datos.
    - ✓ Minificar y comprimir recursos (CSS, JS, imágenes).
    - ✓ Usar Caching (en el servidor, navegador y CDN) para reducir los tiempos de respuesta.

- **Seguridad por Diseño:** Implementar defensas contra vulnerabilidades comunes (como inyección SQL, XSS, CSRF) desde las fases iniciales del diseño.
- **Uso de Contenedores (Docker/Kubernetes):** Empaquetar la aplicación y sus dependencias para asegurar que se ejecute de manera uniforme en cualquier entorno (desarrollo, pruebas, producción).
- **Documentación Clara:** Mantener la arquitectura, la API y el código bien documentado para facilitar el trabajo del equipo actual y futuro.

5. **Escriba y esplique brevemente los Tipos de Arquitectura de Aplicaciones Web, sus ventajas y desventajas**

Existen varios modelos arquitectónicos, siendo los más comunes:

A. Arquitectura de 2 Capas (Cliente-Servidor)

La lógica se divide principalmente entre el cliente (interfaz) y un único servidor que maneja la lógica de negocio y los datos (SGBD).

- Ventajas: Simplicidad, rápida implementación para aplicaciones pequeñas.
- Desventajas: Poca escalabilidad, el servidor puede convertirse en un cuello de botella.

B. Arquitectura de 3 Capas (N-Capas)

Separa la aplicación en tres capas lógicas distintas: Presentación (Frontend), Lógica de Negocio (Backend), y Datos (Base de Datos). Cada capa puede ejecutarse en servidores separados.

- Ventajas: Alta escalabilidad y rendimiento, mejor mantenibilidad, fácil de aislar y actualizar componentes, mejor seguridad.
- Desventajas: Mayor complejidad inicial en la configuración y desarrollo. Es el estándar para aplicaciones empresariales medianas a grandes.

C. Arquitectura de Microservicios

La aplicación se descompone en una colección de servicios pequeños, independientes y desplegables por separado, cada uno ejecutando un proceso único de negocio y comunicándose a través de APIs ligeras (usualmente HTTP/REST).

- Ventajas: Máxima escalabilidad y flexibilidad (cada servicio puede usar tecnologías diferentes y escalarse independientemente), alta resiliencia.
- Desventajas: Gran complejidad operativa (manejar muchos servicios, despliegue, comunicación y monitoreo), requiere herramientas de orquestación (Kubernetes).

D. Arquitectura *Serverless* (Sin Servidor)

El proveedor de la nube (ej. AWS Lambda, Azure Functions) gestiona la infraestructura, y el código se ejecuta en funciones efímeras que solo se activan en respuesta a eventos (p. ej., una petición HTTP).

- Ventajas: Costo optimizado (solo se paga por el tiempo de ejecución), escalado automático masivo, no requiere gestión de servidores.

- Desventajas: Puede generar "vendor lock-in" (dependencia del proveedor), dificultad en el monitoreo y pruebas locales, latencia de inicio ("cold start").

6. **Escriba Ejemplos de Tecnologías aplicada a:**

   a. **Frontend:**

- React
- Angular
- Vue.js
- Svelte

   b. **Backend:**

- ASP.NET
- Node.js
- Django (Python)
- Spring Boot (Java)

   c. **Base de Datos:**

- SQL Server
- MySQL
- PostgreSQL
- MongoDB

   d. **Almacenamiento de Imágenes:**

- Amazon S3
- Google Cloud Storage
- Microsoft Azure Blob Storage
- Cloudinary

7. **Que es un protocolo de comunicación**

   Un **protocolo de comunicación** es un conjunto de reglas y estándares que permiten que dos o más sistemas o dispositivos se comuniquen entre sí a través

de una red. Estos protocolos definen cómo se envían, reciben, procesan y validan los datos para garantizar que la información llegue de manera correcta, segura y entendible entre los participantes.

8. **Explique los protocolos http, https**

HTTP (HyperText Transfer Protocol):

- Es un protocolo que permite la transferencia de información en la web, como páginas HTML, imágenes, archivos y otros recursos.
- Es **rápido y sencillo**, pero la comunicación **no está cifrada**, lo que significa que los datos pueden ser interceptados fácilmente.
- Se utiliza en sitios donde la seguridad no es crítica.
-

HTTPS (HyperText Transfer Protocol Secure):

- Es la versión segura de HTTP, ya que utiliza un **certificado SSL/TLS** para cifrar la comunicación entre el cliente (navegador) y el servidor.
- Protege la **confidencialidad**, **integridad** y **autenticidad** de los datos.
- Se usa en la mayoría de los sitios actuales, especialmente en plataformas de pagos, bancos y cualquier aplicación que maneje datos sensibles.

9. **Que es hosting, investigue los tipos de hosting y haga una tabla comparativa de mínimo 4 proveedores de este servicio, elija uno apropiado para su proyecto**

Hosting es el servicio que permite a una persona u organización almacenar y poner a disposición pública una aplicación o sitio web en Internet. Un proveedor de hosting proporciona el espacio en un servidor y los recursos necesarios para que el sitio web sea accesible 24/7.

Tipo de Hosting	Descripción	Ideal Para
Hosting Compartido	Varios sitios web comparten los recursos de un único servidor. Es el más económico.	Sitios web pequeños, blogs personales, startups con bajo tráfico inicial.

Servidor Virtual Privado (VPS)	Un servidor físico se divide virtualmente, asignando recursos dedicados (CPU, RAM) a cada usuario. Ofrece más control.	Sitios web de tráfico moderado, aplicaciones que requieren configuración de servidor personalizada.
Hosting Dedicado	El cliente alquila un servidor físico entero con todos sus recursos. Máximo rendimiento y control.	Grandes empresas, aplicaciones de alto tráfico, proyectos que manejan datos sensibles.
Cloud Hosting (Nube)	Utiliza una red de servidores interconectados. Los recursos se escalan bajo demanda.	Proyectos con picos de tráfico impredecibles, alta disponibilidad y escalabilidad.

Proveedor	**Tipo de Hosting Principal**	**Precio de Entrada (Aprox.)**	**Facilidad de Uso**	**Escalabilidad**	**Soporte Técnico**
Hostinger	Compartido, VPS, Cloud	Bajo (Desde ~$2.99/mes)	Muy Alta (hPanel intuitivo)	Media (Compartido) a Alta (Cloud)	Chat 24/7
SiteGround	Compartido, Cloud	Medio (Desde ~$3.99/mes)	Alta (Herramientas optimizadas)	Media-Alta	Excelente y especializado
Amazon Web Services (AWS)	Cloud (EC2, S3, Lambda)	Variable (Pago por uso)	Baja (Curva de aprendizaje alta)	Máxima	Estándar a Dedicado (Costo extra)
Bluehost	Compartido, VPS, Dedicado	Bajo (Desde ~$2.95/mes)	Alta (Recomendado por WordPress)	Media	24/7 por teléfono/chat

Cloud Hosting (AWS/GCS): Ideal si se prioriza la máxima escalabilidad y se tienen conocimientos técnicos para administrar la infraestructura.

10. **Que es un servidor de Dominio, escojan un dominio para su proyecto e investiga si está disponible, agregue capturas para comprobar su investigación**

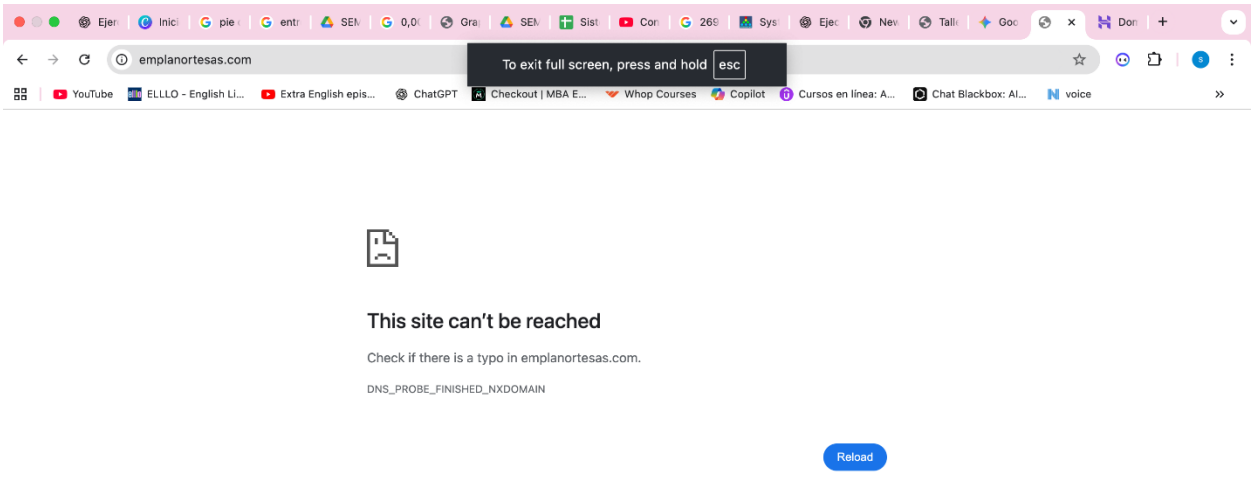
El término Servidor de Dominio se refiere comúnmente a un Servidor DNS (Domain Name System).

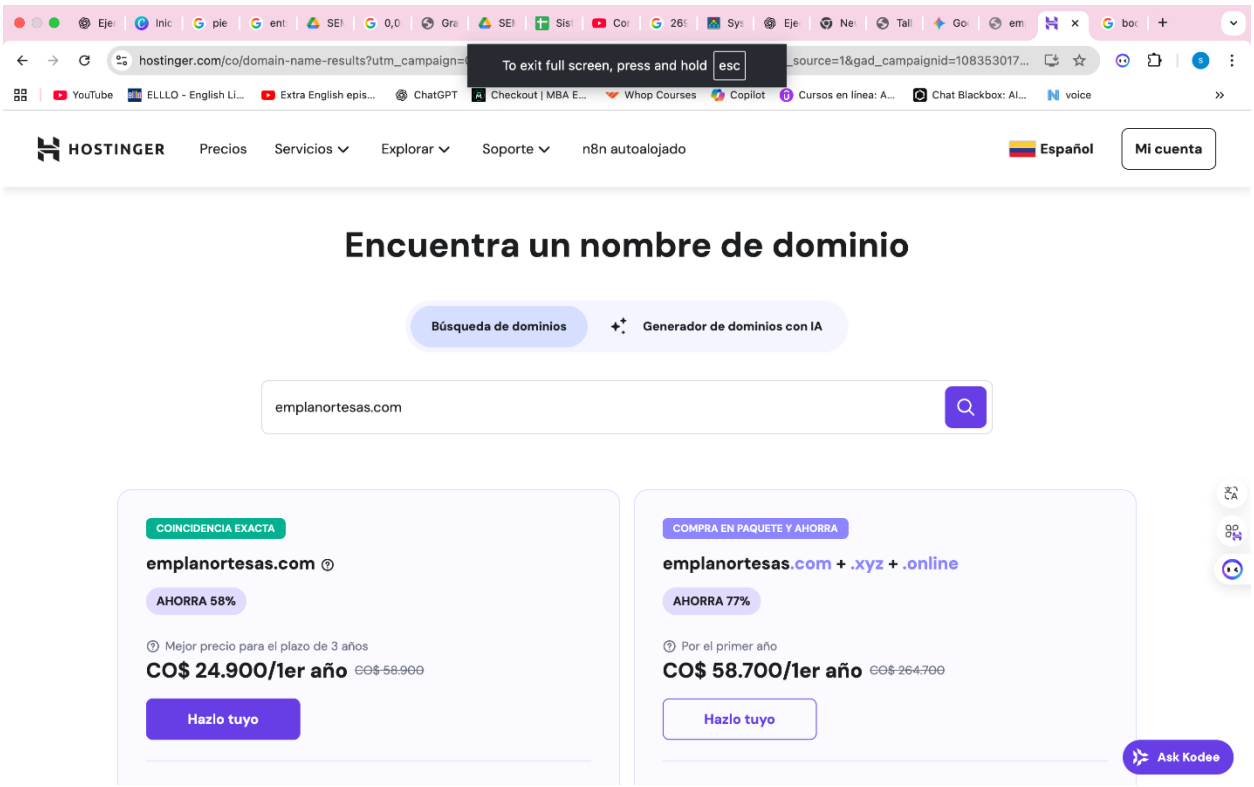
Un Servidor DNS es un componente esencial de la infraestructura de Internet que actúa como el "directorio telefónico" de la red. Su función principal es traducir los nombres de dominio (que son fáciles de recordar para los humanos, como google.com) a sus correspondientes Direcciones IP (que son necesarias para que las máquinas se comuniquen, como 142.250.72.100).

Cuando un usuario escribe un nombre de dominio en su navegador, el Servidor DNS es consultado para obtener la dirección IP del servidor donde está alojado el sitio web, permitiendo así la conexión.

Nombre de Dominio Elegido (Ejemplo Ficticio): emplanortesas.com

Investigación de Disponibilidad:




To exit full screen, press and hold esc
HOSTINGER
Precios
Servicios
Explorar
Soporte
n8n autoalojado
Español
Mi cuenta
Encuentra un nombre de dominio
Búsqueda de dominios
Generador de dominios con IA
emplanortesas.com
COINCIDENCIA EXACTA
emplanortesas.com
AHORRA 58%
Mejor precio para el plazo de 3 años
CO$ 24.900/1er año
CO$ 58.900
Hazlo tuyo
COMPRA EN PAQUETE Y AHORRA
emplanortesas.com + .xyz + .online
AHORRA 77%
Por el primer año
CO$ 58.700/1er año
CO$ 264.700
Hazlo tuyo
Ask Kodee