Despliegue de Aplicaciones Web

Introducción a Arquitecturas Web

Tabla de Contenidos

Introducción

La arquitectura **World Wide Web (WWW)** de Internet provee un modelo de programación potente y flexible. Las aplicaciones y los contenidos son presentados en formatos de datos estándar y son localizados por aplicaciones conocidas como **web browsers (navegadores)**, que envían requerimientos de objetos a un servidor y este responde con el dato codificado según un formato estándar.

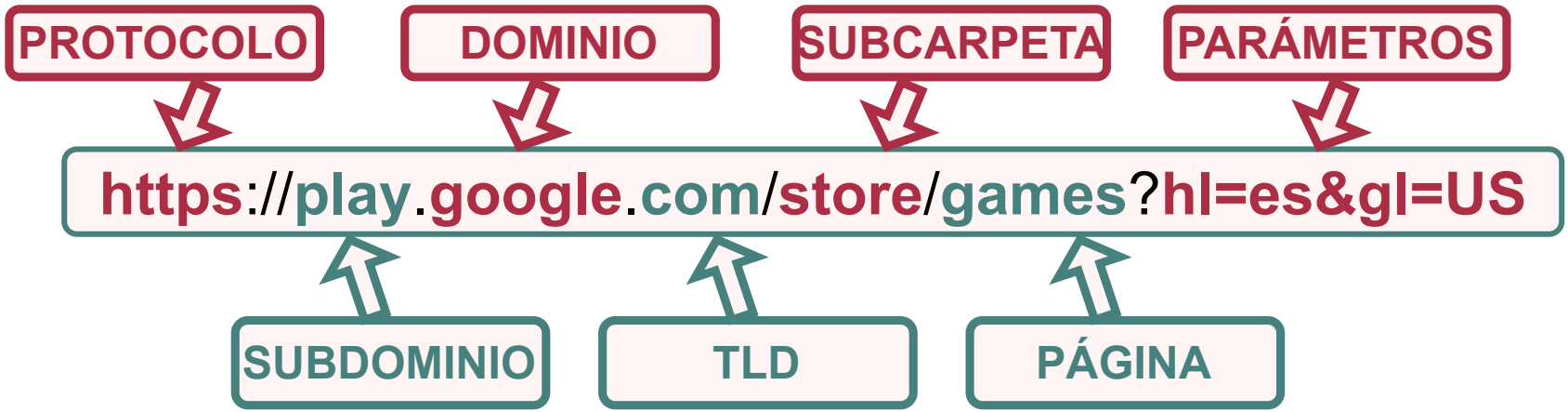
World Wide Web Consortium

En el año 1994, Berners-Lee fundó el **World Wide Web Consortium (W3C)**, una organización cuyo objetivo fundamental es **regular las recomendaciones y los estándares utilizados, asegurando la compatibilidad y la evolución de los servicios web**.

Modelo estándar de nombres

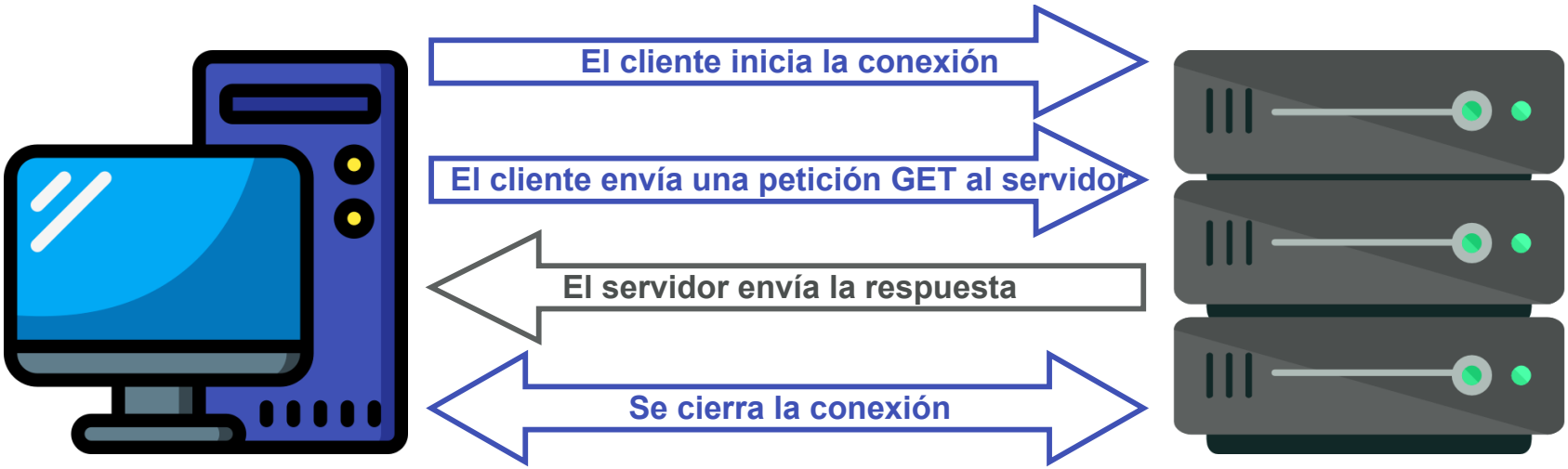
La nomenclatura utilizada para hacer referencia a objetos y poder localizar información de manera rápida y sencilla es mediante la **Uniform Resorce Locator (URL)**.

La estructura de una URL es la siguiente:



Protocolos estándar

Existen una serie de protocolos que permiten que cualquier navegador pueda comunicarse con cualquier servidor web. El protocolo **Hypertext Transfer Protocol (HTTP)** está diseñado para **transferir documentos HTML**, utilizando por defecto el puerto **TCP 80** para establecer las conexiones. Se sitúa en la capa 7 del modelo OSI (**capa de aplicación**). Otra alternativa que se ha popularizado en la actualidad sería el protocolo **HTTPS, una versión del protocolo HTTP en la que se utiliza un certificado SSL para conectarse a sitios de forma segura**.



Formatos de contenidos estándar

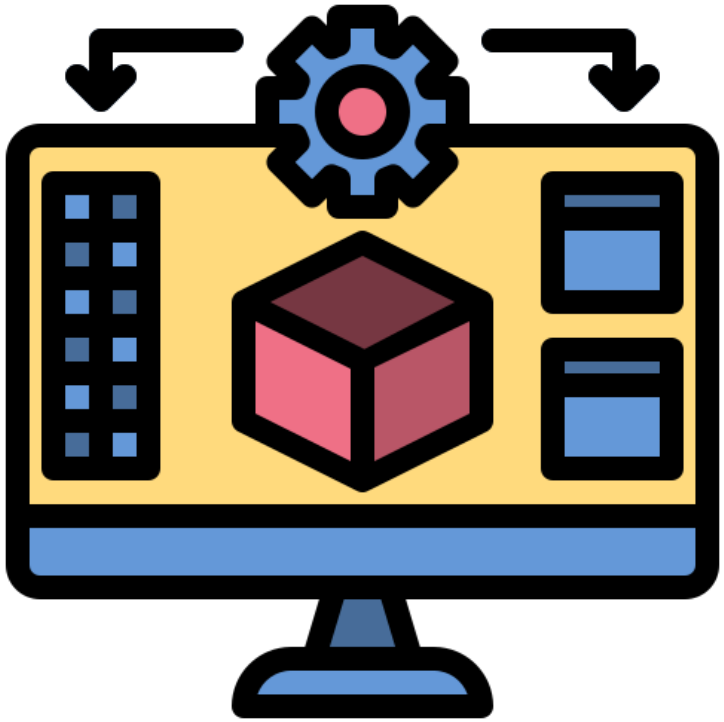
HTML (HyperText Markup Language El estándar fundamental para la creación de páginas web. **Define la estructura y el contenido de una página web**, utilizando etiquetas para marcar elementos como títulos, párrafos, enlaces, imágenes y más.

CSS (Cascading Style Sheets) Utilizado junto con HTML para controlar la presentación y el diseño de una página web. **Permite definir estilos**, como colores, fuentes, márgenes y tamaños de elementos.

JavaScript (JS) Un lenguaje de programación ampliamente utilizado para agregar interactividad y dinamismo a las páginas web. **Permite realizar acciones como validación de formularios, animaciones y manipulación del DOM (Document Object Model)**.

Aspectos generales a destacar en una arquitectura web

1. Escalabilidad.
2. Separación de responsabilidades.
3. Portabilidad.
4. Utilización de componentes en los servicios de infraestructura.
5. Gestión de las sesiones del usuario.
6. Aplicación de patrones de diseño.



Modelos de arquitectura web

Existen varias arquitecturas y modelos utilizados en el diseño y desarrollo de aplicaciones web para satisfacer diversas necesidades y requisitos:

- Modelo Cliente-Servidor.
- Arquitectura Modelo-Vista-Controlador (MVC).
- Arquitectura de Microservicios.
- Arquitectura Serverless.
- Arquitectura de aplicaciones de una sola página (SPA).
- Arquitectura P2P.

Modelo Cliente-Servidor

Cómo Funciona: El cliente realiza solicitudes al servidor, el cual procesa estas solicitudes y devuelve una respuesta.
Ejemplo: Cuando abres tu navegador y entras en una página web, tu navegador (cliente) envía una solicitud HTTP al servidor web que aloja el sitio. El servidor procesa la solicitud, genera la página solicitada y la envía de vuelta al navegador, que la muestra.
Ventajas:

- Separación clara entre cliente y servidor.
- Escalabilidad a nivel de servidor.

Desventajas:

- Dependencia de la red para la comunicación.
- Escalabilidad puede ser limitada en servidores únicos.

Arquitectura Modelo-Vista-Controlador (MVC)

Cómo Funciona: Se divide en tres componentes:

- **Modelo:** Gestiona los datos y la lógica de la aplicación.
- **Vista:** Presenta la interfaz de usuario.
- **Controlador:** Maneja las entradas del usuario y actualiza el modelo y la vista.

Ejemplo: En una aplicación web de tienda en línea, cuando un usuario realiza una búsqueda de productos, el controlador recibe la solicitud, consulta el modelo para obtener los datos de los productos y luego actualiza la vista para mostrar los resultados de búsqueda.

Ventajas:

- Mejora la organización y mantenibilidad del código.
- Facilita la reutilización y el testing de componentes.

Desventajas:

- Puede ser complejo de implementar en aplicaciones pequeñas.
- Requiere una estructura rígida que puede ser restrictiva.

Arquitectura de Microservicios

Cómo Funciona: La aplicación se divide en múltiples servicios independientes que se comunican entre sí mediante APIs.
Ejemplo: Imagina un servicio de streaming de música. Una API de microservicio podría manejar la gestión de usuarios, otra podría gestionar la biblioteca de música, y otra podría manejar las recomendaciones. Cuando un usuario inicia sesión, el microservicio de autenticación verifica sus credenciales y proporciona un token, que luego es usado por otros microservicios para personalizar la experiencia.

Ventajas:

- Escalabilidad individual de servicios.
- Despliegue y desarrollo más ágil.

Desventajas:

- Complejidad en la gestión y comunicación entre servicios.
- Requiere una infraestructura robusta y una buena estrategia de orquestación.

Arquitectura Serverless

Cómo Funciona: Las aplicaciones se ejecutan en una infraestructura gestionada por un proveedor de servicios en la nube, que maneja la ejecución de funciones específicas.
Ejemplo: Supón que tienes una función en Azure Functions que se activa cuando se sube un archivo a un contenedor de Azure Blob Storage. Cuando se sube un archivo, el contenedor de Blob Storage envía un evento que invoca la función en Azure. La función procesa el archivo (por ejemplo, analiza el contenido y lo guarda en una base de datos) y luego almacena el resultado en otro contenedor de Blob Storage.
Ventajas:

- Reducción de costos operativos.
- Escalabilidad automática.

Desventajas:

- Dependencia del proveedor de la nube.
- Limitaciones en el tiempo de ejecución y recursos.

Arquitectura de Aplicaciones de Una Sola Página (SPA)

Cómo Funciona: La aplicación carga una sola página HTML y actualiza dinámicamente el contenido mediante JavaScript.
Ejemplo: En una aplicación de correo electrónico como Gmail, al hacer clic en un mensaje, la aplicación no recarga toda la página; en su lugar, utiliza JavaScript para actualizar dinámicamente el contenido del mensaje sin recargar el navegador.
Ventajas:

- Experiencia de usuario fluida y rápida.
- Reducción de la carga en el servidor.

Desventajas:

- Problemas de SEO (optimización para motores de búsqueda).
- Mayor carga inicial de recursos.

Arquitectura P2P

Cómo Funciona: Los nodos en la red actúan tanto como clientes como servidores, compartiendo recursos entre ellos.
Ejemplo: En una red de intercambio de archivos P2P como BitTorrent, cuando descargas un archivo, no solo estás recibiendo datos de un servidor central, sino que también estás recibiendo y enviando partes del archivo a otros usuarios (pares) en la red.
Ventajas:

- Alta escalabilidad y robustez.
- Reducción de la carga en servidores centralizados.

Desventajas:

- Problemas con la gestión y seguridad de los datos.
- Complejidad en la implementación de la sincronización.