Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología - Concepción del Uruguay
Licenciatura en Sistemas de Información

Programación Avanzada

Arquitectura y Protocolos de la World Wide Web

Alumno: Borda Alexander

Profesor: Lic.Ernesto Ledesma

Carrera: Lic. En Sistemas de Información

Curso Lectivo: 2025

Historia de la WWW

1. ¿Como surgió el sistema de documentos distribuidos WWW?

El sistema de documentos distribuidos conocido como World Wide Web surgió a fines de los años 80 en el CERN, cuando Tim Berners-Lee propuso un sistema de hipertexto distribuido que permitiera enlazar documentos y facilitar el acceso a la información de manera sencilla. La idea era superar las limitaciones de los sistemas de hipertexto locales existentes, combinando un protocolo de comunicación (HTTP), un lenguaje de marcado (HTML) y un esquema de direccionamiento universal (URL).

2. ¿En que año se creo la www? ¿Quien la creo? ¿Cuál fue el propósito inicial?

La WWW fue creada en 1989-1990 por Tim Berners-Lee, con la colaboración de Robert Cailliau, y se hizo pública en 1991. Su propósito inicial era favorecer el intercambio de información entre científicos e investigadores de diferentes universidades y laboratorios que trabajaban en conjunto, evitando la duplicación de esfuerzos y mejorando la cooperación científica.

3. ¿Sobre que red funcionaba? ¿Cuales fueron los componentes iniciales?

Desde sus orígenes, funcionó sobre Internet, que ya utilizaba el protocolo TCP/IP. Los componentes iniciales fueron el protocolo HTTP, el lenguaje HTML, las direcciones URL, el primer servidor web denominado httpd, el primer navegador llamado WorldWideWeb (desarrollado en NeXTSTEP) y el primer sitio web accesible públicamente, alojado en el CERN en la dirección http://info.cern.ch/.

WWW en la actualidad

4. Nombre y describa las distintas etapas de evolución en la web. Realice una cronología de tecnologías que surgieron en la web hasta la actualidad.

La Web ha pasado por varias etapas:

- **Web 1.0 (1990 – 2000):** era estática y de solo lectura. Los sitios contenían documentos en HTML, imágenes y enlaces, pero los usuarios eran solo consumidores de información. Tecnologías: HTML básico, CGI, primeros navegadores (Mosaic, Netscape), HTTP/0.9 – HTTP/1.0.

- **Web 2.0 (2000 – 2010):** surgió la web social, participativa e interactiva. Los usuarios pasaron a ser creadores de contenido. Tecnologías: AJAX, JavaScript dinámico, CSS, bases de datos integradas a servidores web, redes sociales, blogs y wikis.

- **Web 3.0 (2010 – actualidad):** conocida como web semántica o inteligente, incorpora inteligencia artificial, big data y metadatos que permiten entender mejor el significado de la información. Tecnologías: RDF, OWL, JSON-LD, blockchain en algunas aplicaciones, buscadores inteligentes, asistentes virtuales.

- **Web 4.0 (futuro cercano):** orientada a la **web ubicua** y la integración total con el Internet de las Cosas (IoT). Se busca la interacción natural entre humanos y máquinas mediante IA avanzada, realidad aumentada, dispositivos inteligentes y experiencias inmersivas.

5. ¿Qué es un Servidor WWW?

Un servidor WWW es un software y hardware que almacena páginas web, aplicaciones y recursos, y los pone a disposición de los usuarios en Internet utilizando el protocolo HTTP o

HTTPS. Su función principal es recibir solicitudes de clientes web (navegadores), procesarlas y devolver como respuesta documentos en HTML, imágenes, videos o cualquier recurso disponible. Ejemplos de software servidor web son Apache, Nginx y Microsoft IIS.

6. ¿Qué es un cliente web? explique detalladamente.

Un cliente web es la aplicación que utiliza el usuario para conectarse al servidor WWW, solicitar recursos y presentarlos de manera comprensible. El ejemplo más común de cliente web es el **navegador** (Chrome, Firefox, Safari, Edge), que interpreta HTML, CSS y JavaScript para mostrar las páginas.

Además de los navegadores, existen clientes especializados como aplicaciones móviles que consumen servicios web a través de APIs. El cliente no solo muestra el contenido, también permite interacción con formularios, envío de datos y ejecución de scripts.

7. ¿Como está compuesto el sistema WWW actual?

- **Clientes web (navegadores y apps)** que realizan las solicitudes.
- **Servidores web** que almacenan los recursos.
- **Protocolos de comunicación** como HTTP/HTTPS, DNS para resolver direcciones, y TCP/IP como base de transmisión.
- **Lenguajes y tecnologías de contenido**: HTML5, CSS3, JavaScript, además de frameworks como React, Angular y Vue.
- **Servicios distribuidos**: bases de datos, microservicios, APIs, servicios en la nube (AWS, Azure, Google Cloud).
- **CDN (Content Delivery Networks)** para distribuir contenido de manera eficiente.

8. Realice un diagrama de crecimiento de host conectados a Internet desde su inicio hasta la actualidad.

9. Realiza un diagrama conceptual de composición de la web. ¿Cuáles son los Componentes de la www?

10. ¿Cuál es la Arquitectura del sistema distribuido de documentos?

La arquitectura de la WWW se basa en un modelo **cliente-servidor**. Los clientes solicitan recursos mediante HTTP y los servidores responden. Además, hay un sistema distribuido de resolución de nombres (DNS), servidores replicados y tecnologías como CDN que distribuyen contenido en distintas regiones para mejorar el rendimiento.
Hoy en día también se integra una arquitectura **multicapa**: presentación (navegador), lógica de negocio (servidor web, aplicaciones) y datos (bases de datos).

11. ¿Cómo funciona en sistema www?

El funcionamiento es el siguiente:

Un usuario introduce una dirección URL en su navegador. El cliente consulta al sistema DNS para resolver la dirección IP del servidor. Luego establece una conexión TCP/IP y envía una solicitud HTTP o HTTPS al servidor web. El servidor procesa la petición, busca el recurso solicitado (por ejemplo, un archivo HTML o una respuesta generada por una aplicación), y lo envía de regreso al cliente. Finalmente, el navegador interpreta el código recibido, renderiza el contenido y lo presenta al usuario, permitiendo la navegación mediante hipervínculos que disparan nuevas solicitudes.

Tecnologías, Servidores y Clientes.

12. ¿Que es HTML? ¿De donde proviene?

HTML es el **HyperText Markup Language**, un lenguaje de marcado diseñado para estructurar y etiquetar contenido en la Web (títulos, párrafos, enlaces, listas, imágenes, formularios, etc.). No es un lenguaje de programación sino un conjunto de marcas que describen la semántica y la estructura de un documento. HTML proviene de la idea de hipertexto y su primera versión la definió Tim Berners-Lee en el CERN a finales de 1989/1990; las primeras implementaciones estaban fuertemente influenciadas por SGML (Standard Generalized Markup Language). Con el tiempo la especificación fue adoptada y normalizada por el W3C y evolucionó hasta HTML5, que incorpora APIs y funcionalidad modernas.

13. ¿A que se le llama pagina web estática? ¿y pagina dinámica? ¿y página activa?

Una página web **estática** es un archivo (normalmente .html) que el servidor entrega tal cual al cliente: su contenido no cambia salvo que se modifique el archivo en el servidor. Una página **dinámica** es aquella cuyo contenido se genera en el momento por software del servidor (o por código en el cliente): suele acceder a bases de datos, personalizar la salida según el usuario o estado, y puede cambiar sin editar archivos estáticos. El término **página activa** suele utilizarse para referirse a páginas que contienen y ejecutan código (por ejemplo scripts del lado del cliente como JavaScript, applets, ActiveX o plugins) que permiten interactividad avanzada; a veces también se usa para referirse a tecnologías de servidor como “Active Server Pages (ASP)”, así que conviene distinguir contexto: “activo” = ejecutable/enriquecido frente a “estático” = solo marcación.

14. ¿Qué es una URL? ¿qué función cumple en la Web? ¿Qué es un servidor de archivos basado en URL?

Una URL (Uniform Resource Locator) es la dirección que localiza un recurso en la Web. Su función es identificar dónde y cómo obtener ese recurso: incluye esquema/protocolo (http, https, ftp, file), nombre de host (dominio), puerto opcional, ruta al recurso, cadena de consulta y fragmento. Gracias a la URL el navegador sabe a qué servidor conectar y qué recurso solicitar. Un “servidor de archivos basado en URL” es simplemente un servicio (por ejemplo un servidor HTTP, FTP o un servidor de archivos) que publica ficheros y los hace accesibles mediante URLs; cuando el cliente pide la URL el servidor devuelve el archivo solicitado.

15. ¿Dónde y como se ejecuta el lenguaje HTML?

HTML no “se ejecuta” como un programa, sino que **se interpreta y renderiza** en el cliente: el navegador descarga el documento HTML, el motor de parseo construye el DOM, el motor de estilo aplica CSS (CSSOM) y el navegador construye el árbol de renderizado, calcula el layout y pinta los píxeles en pantalla. En ese proceso los motores JavaScript (V8, SpiderMonkey, JavaScriptCore, etc.) pueden ejecutar scripts embebidos que modifican el DOM en tiempo real.

16. ¿Qué es un lenguaje script en la tecnología WWW?

Un lenguaje *script* en la tecnología WWW es un lenguaje usado para escribir pequeños programas (scripts) que automatizan tareas, manipulan contenidos y gestionan comportamiento. En la Web se distinguen scripts del lado del cliente (JavaScript) que controlan la interacción en el navegador y scripts del lado del servidor (PHP, Python, Perl, Ruby, etc.) que generan contenido dinámico o exponen servicios. Los scripts permiten, por ejemplo, validar formularios, hacer peticiones asíncronas (AJAX/Fetch), o producir HTML dinámico en el servidor.

17. ¿Qué es un pluggin? ¿qué es una cookie?

Un **plugin** (o complemento) es un módulo que extiende las capacidades del navegador para manejar formatos o funcionalidades que no están integradas nativamente (históricamente Flash Player, Java Plugin, Silverlight). Hoy la mayoría de plugins NPAPI fueron retirados por razones de seguridad; las funciones se sustituyen por APIs web y extensiones. Una **cookie** es un pequeño

dato que un servidor puede pedir al navegador que almacene y que luego el navegador envía de vuelta en cada petición al mismo dominio; se usa para gestionar sesiones (identificar usuario), preferencias, rastreo y personalización. Las cookies tienen atributos importantes: dominio, path, expiración, Secure, HttpOnly y SameSite, y su uso está regulado por leyes de privacidad.

18. ¿Qué es JavaScript? ¿Dónde y quien ejecuta JavaScript? ¿Cómo nació JavaScript?

JavaScript es un lenguaje de scripting dinámico y de tipado débil diseñado inicialmente para añadir comportamiento en páginas web; fue creado por Brendan Eich en Netscape en 1995 (pasó por nombres como Mocha y LiveScript). Está estandarizado por ECMAScript (ECMA-262). JavaScript se ejecuta en el navegador por el motor JavaScript (por ejemplo V8 en Chrome/Node, SpiderMonkey en Firefox, JavaScriptCore en Safari). Desde la aparición de Node.js (Ryan Dahl, 2009) JavaScript también se ejecuta en el servidor y en entornos fuera del navegador, lo que lo convirtió en un lenguaje universal en el ecosistema web.

19. ¿Qué es un Applet de Java? ¿Cómo puede un Browser ejecutar código Java?

Un **applet de Java** es un pequeño programa Java que podía incrustarse en una página web y ejecutarse dentro de la máquina virtual Java (JVM) del navegador mediante un plugin (Java Runtime Environment + Java Plugin). El navegador cargaba el applet y la JVM lo ejecutaba en un sandbox con restricciones de seguridad. Esta técnica requirió plugins (NPAPI) y con el tiempo fue considerada insegura y obsoleta; los navegadores modernos dejaron de soportar applets y Sun/Oracle y la comunidad migraron a otras soluciones (aplicaciones nativas, Java Web Start —también deprecado— y tecnologías web modernas).

20. ¿Cómo funciona la tecnología Flash de Macromedia?

La tecnología **Flash** (originalmente de Macromedia, luego Adobe) permitía crear contenido multimedia interactivo (animaciones, vídeo, juegos) embebido en páginas web mediante ficheros SWF y ejecutado por el plugin Flash Player en el navegador. Flash usaba ActionScript como lenguaje y ofrecía timeline, gráficos vectoriales, audio y streaming. Por problemas de seguridad, rendimiento y la aparición de estándares abiertos (HTML5, Canvas, WebGL, WebAudio), Flash fue eliminado progresivamente y Adobe terminó su soporte el 31 de diciembre de 2020; los navegadores ya lo bloquean por defecto.

21. ¿Para que se utiliza la tecnología CGI? ¿Qué lenguajes de programación se pueden utilizar para los scripts CGI?

CGI (Common Gateway Interface) es una especificación que define cómo un servidor web puede ejecutar programas externos (scripts o binarios) para generar respuestas dinámicas. Cuando llega una petición, el servidor crea variables de entorno (REQUEST_METHOD, QUERY_STRING, etc.), pasa datos por stdin (para POST) y espera la salida del programa con cabeceras HTTP + cuerpo. Se pueden escribir scripts CGI en cualquier lenguaje que pueda leer stdin y escribir stdout: Perl (histórico), C, Python, Ruby, PHP, shell script, etc. CGI es simple pero poco eficiente a gran escala (cada petición lanza un proceso); por eso surgieron alternativas como FastCGI, servidores de aplicaciones y frameworks persistentes.

22. ¿Qué es un “form” HTML? ¿para que se utiliza?

Un **form** en HTML es un elemento que agrupa controles (input, select, textarea, button) para que el usuario introduzca datos y los envíe al servidor. Se utiliza para búsquedas, inicios de sesión, envío de comentarios, subidas de archivos, etc. Un form especifica el atributo `action` (dónde enviar), `method` (GET ó POST) y `enctype` (por ejemplo `application/x-www-form-urlencoded` o `multipart/form-data` para archivos). Al enviar, el navegador empaqueta los valores y realiza la petición HTTP correspondiente.

23. ¿Qué son y cómo funcionan las tecnologías ASP, PHP y JSP?

ASP, PHP y JSP son tecnologías para generar páginas web dinámicas del lado del servidor. **ASP (Classic)** fue una solución Microsoft en la que se embebía código VBScript/JScript dentro de páginas .asp y se ejecutaba bajo IIS; evolucionó a **ASP.NET**, que usa el framework .NET y

lenguajes compilados. **PHP** es un lenguaje servidor que se integra directamente en HTML con delimitadores `<?php ?>`; se ejecuta en el servidor (por ejemplo en un stack LAMP) y produce HTML de salida. **JSP (JavaServer Pages)** permite escribir páginas con fragmentos Java o tags que se compilan en servlets Java y corren en un contenedor (Tomcat), integrándose con la plataforma Java EE. En todos los casos el servidor interpreta/compila el código y envía al cliente el HTML resultante; las diferencias principales están en el lenguaje, ecosistema, rendimiento y modelo de despliegue.

24. ¿Qué es DHTML? ¿y XHTML?

DHTML (Dynamic HTML) fue un término comercial/periodístico para describir la combinación de HTML, CSS y JavaScript que permite contenido interactivo y dinámico en el navegador (manipulación del DOM, efectos visuales, posicionamiento dinámico). No es una especificación formal, sino una práctica. **XHTML** es la reformulación de HTML como aplicación de XML: exige documentos bien formados, etiquetas cerradas, reglas estrictas y uso de namespaces; XHTML 1.0 fue popular entre quien buscaba rigor XML. HTML5 recuperó flexibilidad y nuevas APIs y hoy es la recomendación dominante para contenido web.

25. ¿Que es XML ?explique detalladamente.

XML (eXtensible Markup Language) es un lenguaje de marcado genérico y extensible diseñado para almacenar y transportar datos de forma legible y estructurada. A diferencia de HTML, XML no define etiquetas predefinidas sino que permite crear vocabularios propios; es auto-descriptivo: `<cliente><nombre>...</nombre></cliente>`. Tiene reglas de well-formedness (bien formado) y se puede validar contra DTD o XML Schema (XSD). Soporta espacios de nombres (namespaces) para evitar colisiones, y existen tecnologías relacionadas como XPath (selección), XSLT (transformación) y DOM/SAX para parseo. XML fue muy usado en servicios web (SOAP), feeds (RSS), configuración y documentos empresariales; en APIs públicas JSON se popularizó por ser más ligero, pero XML sigue vigente en muchas integraciones.

26. ¿Que son los Servicios Web? De un ejemplo de aplicación utilizando Servicios

Los **Servicios Web** son interfaces que permiten que aplicaciones diferentes se comuniquen a través de la red usando estándares abiertos. Pueden ser SOAP (basados en XML con WSDL y un contrato fuerte) o RESTful (arquitectura basada en recursos y verbos HTTP, respuestas típicamente en JSON). Un ejemplo práctico: una app móvil del clima que consume un servicio web REST (por ejemplo la API de OpenWeather) para obtener el pronóstico; la app hace peticiones HTTP a la API, recibe JSON con temperatura, humedad y lo muestra al usuario. Otro ejemplo en e-commerce: el sitio web puede invocar el servicio web del pasarela de pagos para procesar una transacción, o llamar a servicios de envío para calcular costos y generar etiquetas.

WEB. *URL´s, Dominios y Direcciones.*

27. ¿Qué es una URL? ¿qué función cumple en la web? ¿Cuál es la estructura de una URL?

Una URL (*Uniform Resource Locator*) es la dirección única que identifica y localiza un recurso en la Web, permitiendo al navegador saber **dónde está** y **cómo acceder a él**.

Su función principal es servir como “puente” entre el usuario y el recurso solicitado (páginas HTML, imágenes, videos, documentos, etc.).

Estructura de una URL:
esquema://usuario:contraseña@servidor:puerto/ruta/recurso?consulta#fragmento

28. ¿Qué es un servidor de archivos basado en URL?

Un servidor de archivos basado en URL es un servicio que permite acceder a recursos a través de una dirección URL. Funciona como un repositorio que entrega archivos al cliente cuando estos son solicitados.

Ejemplos:

- Un **servidor web HTTP** que entrega páginas y documentos.
- Un **servidor FTP** accesible mediante URLs `ftp://usuario@servidor/ruta`.
  En ambos casos, el cliente utiliza la URL para localizar el archivo y el servidor lo envía al navegador o aplicación.

29. ¿Cuál es el formato de una URL en IPv6?

    En IPv6, debido a los dos puntos (:) que forman parte de las direcciones, la IP debe escribirse entre corchetes dentro de la URL para evitar ambigüedad.

    Formato : http://[dirección_ipv6]:puerto/ruta/recurso

30. ¿Qué es un Dominio www? ¿Qué relación tiene con una URL?
    Un **dominio** es un nombre legible que se utiliza para identificar y acceder a un servidor en Internet, funcionando como una capa amigable sobre las direcciones IP.
    El subdominio `www` fue históricamente usado para indicar que un recurso pertenece a la **World Wide Web**, aunque hoy en día no es obligatorio y muchos sitios funcionan sin él.
    La relación con la URL es directa: dentro de la estructura de una URL, el dominio aparece en la parte del **servidor** (`www.ejemplo.com`), y es resuelto por el DNS hacia la dirección IP real donde está alojado el recurso.

31. ¿Cuál es la función del DNS en la WEB?

    El **DNS (Domain Name System)** funciona como el “sistema telefónico” de Internet. Su función es **traducir nombres de dominio legibles por humanos (ej. www.google.com) en direcciones IP (ej. 142.250.184.206 en IPv4 o 2a00:1450:4009:80b::200e en IPv6)** que identifican a los servidores en la red.

    Sin DNS, los usuarios tendrían que recordar y escribir direcciones IP en lugar de nombres de dominio. Además, el DNS facilita la escalabilidad de la Web al permitir que varios servidores puedan estar asociados a un mismo nombre de dominio (balanceo de carga, CDN, etc.).

Protocolos utilizados en la www

32. ¿Cuál es la relación entre HTML y HTTP?
    HTML (*HyperText Markup Language*) es el lenguaje de marcado usado para estructurar documentos en la Web, mientras que HTTP (*HyperText Transfer Protocol*) es el protocolo de comunicación que transporta esos documentos entre servidores y clientes.
    En otras palabras: **HTTP es el medio de transporte, HTML es el contenido transportado.**
    El navegador usa HTTP para pedir recursos a un servidor y estos recursos pueden ser páginas escritas en HTML, imágenes, hojas de estilo, scripts, etc.

33. ¿Cuáles son las versiones de HTTP y cuales sus diferencias?

    HTTP/0.9 (1991): muy simple, solo permitía transferir documentos HTML de texto plano. Una petición era solo `GET`. No soportaba cabeceras ni códigos de estado.

    HTTP/1.0 (1996): agregó cabeceras de petición y respuesta, códigos de estado (200, 404, 500), soporte para tipos de contenido (MIME), y métodos adicionales (`POST`, `HEAD`). Cada conexión servía para una sola transacción (no persistente).

    HTTP/1.1 (1997): la versión más usada durante décadas. Introdujo conexiones persistentes (mantener TCP abierto), pipeline de peticiones, soporte de caché avanzado, control de rango de bytes (descargas parciales) y nuevos métodos (`OPTIONS`, `PUT`, `DELETE`).

    HTTP/2 (2015): optimiza rendimiento. Usa multiplexación (varias solicitudes en la misma conexión), compresión de cabeceras, push del servidor (envío proactivo de recursos). Mejora velocidad y reduce latencia, especialmente en móviles.

HTTP/3 (2022 en adelante): basado en QUIC (sobre UDP, no TCP). Ofrece menor latencia, cero *head-of-line blocking* y mayor seguridad (TLS 1.3 integrado). Diseñado para la Web moderna y redes móviles.

34. ¿Qué es MIME?

MIME (*Multipurpose Internet Mail Extensions*) es un estándar que define tipos de contenido para identificar qué clase de datos se están transmitiendo. Aunque fue creado para correo electrónico, HTTP lo adoptó para describir el tipo de recurso enviado en la cabecera `Content-Type`.
Ejemplos:

- `text/html` → documento HTML
- `text/css` → hoja de estilo
- `image/png` → imagen PNG
- `application/json` → datos JSON
- `video/mp4` → archivo de video

Esto permite que el navegador sepa cómo interpretar el recurso recibido.

35. Explique cómo es una solicitud HTTP.

Una solicitud HTTP tiene tres partes principales:

Línea de petición: contiene el método, la ruta y la versión de HTTP.

Cabeceras de petición: proporcionan información adicional (tipo de navegador, cookies, idioma, autenticación, etc.).

Cuerpo de la petición (opcional): usado en métodos como `POST` o `PUT` para enviar datos (formularios, JSON, archivos).

36. ¿Cuáles son los métodos de HTTP 1? Enumere y describa el funcionamiento de cada uno.
En HTTP/1.0 y HTTP/1.1 los métodos principales son:

- **GET:** solicita un recurso específico. No tiene cuerpo en la petición y los parámetros opcionales se pasan en la URL (`?clave=valor`). Ejemplo: obtener una página o imagen.
- **HEAD:** igual que GET, pero solo devuelve cabeceras, sin el cuerpo del recurso. Se usa para verificar si un recurso existe, comprobar fecha de modificación o tamaño.
- **POST:** envía datos al servidor en el cuerpo de la petición (formularios, JSON, archivos). Generalmente modifica el estado en el servidor (crear un recurso).
- **PUT:** reemplaza o crea un recurso en el servidor con los datos enviados. Es idempotente: varias llamadas producen el mismo resultado.
- **DELETE:** solicita eliminar un recurso en el servidor. También es idempotente.
- **OPTIONS:** pide al servidor qué métodos y características soporta para un recurso dado. Muy usado en CORS (Cross-Origin Resource Sharing).
- **TRACE:** devuelve la misma petición que recibió, para diagnóstico y depuración.
- **CONNECT:** usado para crear un túnel de comunicación con un servidor, generalmente para tráfico cifrado con HTTPS.