CUNEF UNIVERSIDAD | 5 años

G-174. Programación de Servidores Web

Tema 1: Introducción a la Programación de Servidores Web

Por Andres Gorostidi - Andres.Gorostidi@cunef.edu

Índice

01

Historia y Evolución de la web

Historia y evolución de la web

Internet nació a finales de los años 60 con ARPANET, que conectaba universidades. Esto allanó el camino para la comunicación global y la creación de redes.

En 1989, Tim Berners-Lee propuso la World Wide Web, que revolucionó el acceso a la información a través del hipertexto. El primer sitio web vio la luz en 1991.

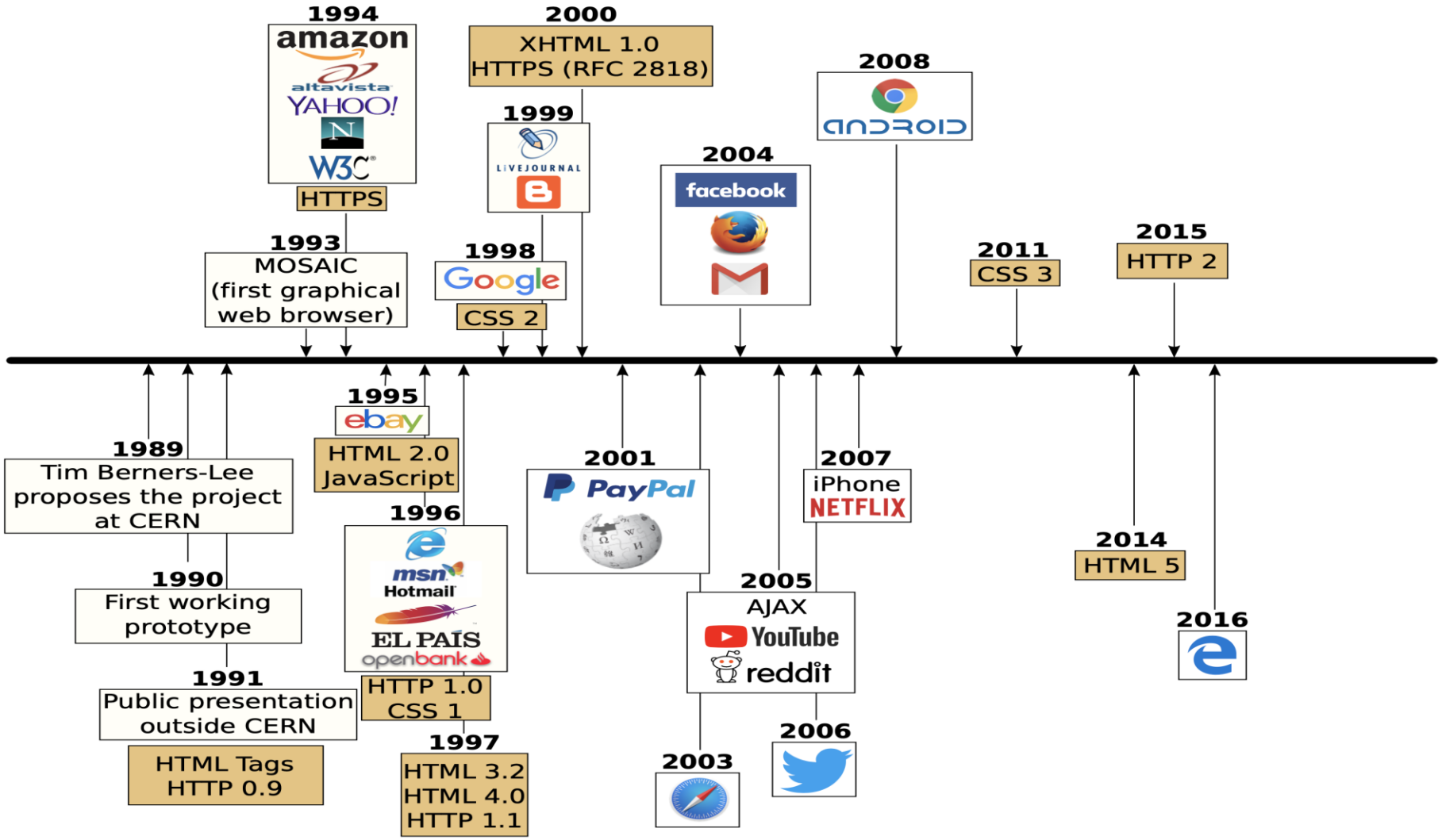
A finales de los 90 se produjo un crecimiento explosivo de los negocios basados en la web. Este boom cambió las economías y llevó a la creación de los gigantes tecnológicos que conocemos hoy.

1. **Web 1.0 (1991 - 2004)** - Se considera que la Web 1.0 comenzó en 1991 con la creación de la primera página web por Tim Berners-Lee. Web estática y de solo lectura. Los usuarios solo podían consumir contenido sin interactuar demasiado. Se trataba de páginas web simples con enlaces, textos alguna imagen. Las páginas eran mayormente de texto con imágenes simples, y no había interacción dinámica entre usuarios y el contenido. Los sitios eran administrados solo por los propietarios y no había posibilidad de retroalimentación o contribuciones de los usuarios.

2. Web 2.0 (2004 - presente). A partir de 2004 comenzó la transición hacia la Web 2.0, aunque el término fue popularizado en 2005 por Tim O'Reilly. La web se volvió más interactiva, con la participación activa de los usuarios a través de redes sociales, blogs, wikis y plataformas colaborativas. Se introdujeron tecnologías como AJAX, lo que permitió crear aplicaciones web más dinámicas y fluidas. Ejemplos significativos: Facebook, Twitter, Wikipedia, YouTube

3. Web 3.0 (2010 - presente). Aunque el concepto se empezó a discutir en 2006, la implementación y el uso real comenzaron alrededor de 2010. Conocida como la web semántica. La Web 3.0 trata de hacer que los datos sean más comprensibles tanto para humanos como para máquinas, aprovechando tecnologías como la inteligencia artificial, la cadena de bloques (blockchain), y una mayor interconexión entre dispositivos. El objetivo principal es que la web "entienda" los datos, y no solo los muestre. Además, **IoT** (Internet de las Cosas) comienza a ganar terreno, conectando dispositivos y objetos a la web

4. Web 4.0 (Teórica). Aunque todavía no está completamente implementada, se discute que podría comenzar en la década de 2030. Se refiere a una web que está completamente interconectada con el mundo físico, donde las máquinas y los dispositivos inteligentes interactúan de manera automática y continua con los humanos. Algunos la llaman la “Web inteligente”, con una integración profunda de tecnologías como la inteligencia artificial avanzada, la realidad aumentada y la realidad virtual.

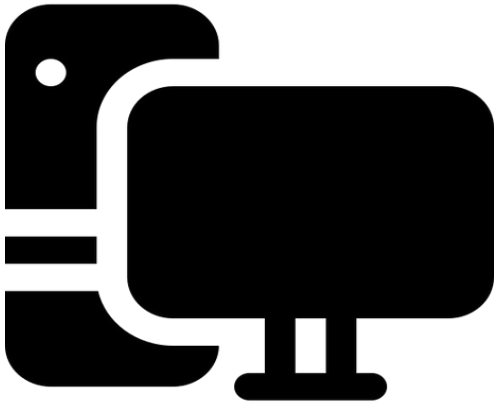

1989
Tim Berners-Lee proposes the project at CERN
1990
First working prototype
1991
Public presentation outside CERN
HTML Tags
HTTP 0.9
1993
MOSAIC (first graphical web browser)
HTTPS
1994
amazon
altavista
YAHOO!
W3C
1995
ebay
HTML 2.0
JavaScript
1996
msn
Hotmail
EL PAÍS
openbank
HTTP 1.0
CSS 1
1997
HTML 3.2
HTML 4.0
HTTP 1.1
1998
Google
CSS 2
1999
LIVEJOURNAL
2000
XHTML 1.0
HTTPS (RFC 2818)
2001
PayPal
2003
2004
facebook
2005
AJAX
YouTube
reddit
2006
2007
iPhone
NETFLIX
2008
ANDROID
2011
CSS 3
2014
HTML 5
2015
HTTP 2
2016

02

Cliente vs Servidor

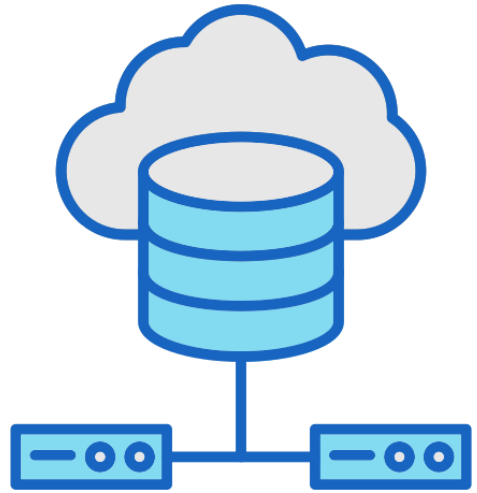
CLIENTE

Cuando un sistema requiere acceder a la información proporcionada por servidores este debe ejecutar software de cliente (aplicaciones, navegadores web) y comunicarse con los servidores mediante un protocolo de comunicación construido sobre TCP/IP para luego transformar la información recibida de manera que sea comprensible para el usuario.



SERVIDOR

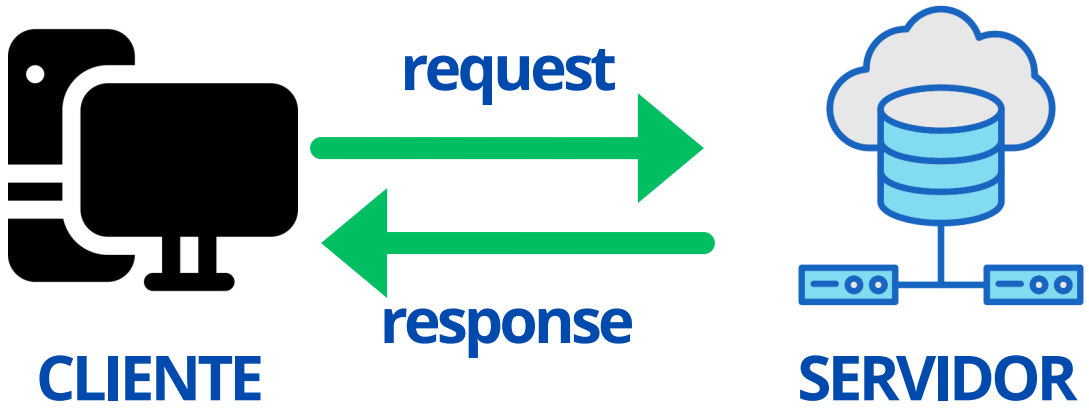
El servidor es un sistema que almacena y proporciona información a otros dispositivos en la red. Este maneja las solicitudes del cliente y responde la información solicitada.



CLIENTE vs SERVIDOR

El proceso consiste en que el cliente envía un mensaje al servidor a través de la red, esto mediante un protocolo de comunicación. El cliente espera un mensaje de respuesta.

Cuando el servidor recibe la petición, realiza el trabajo solicitado o busca los datos solicitados y devuelve una respuesta.

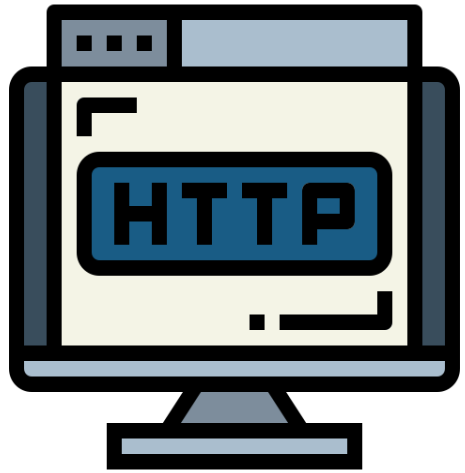


03

HTTP vs HTTPS

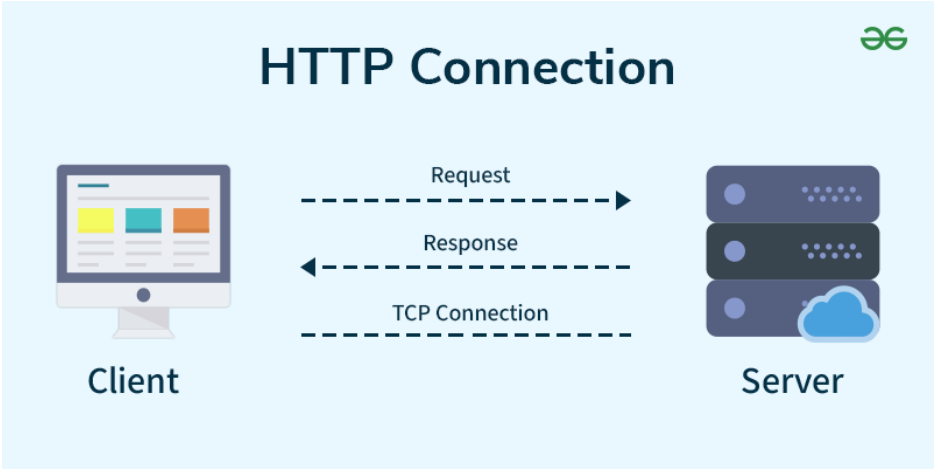
HTTP (HyperText Transfer Protocol)

HTTP es un protocolo de comunicación basado en TCP/IP, es una especificación detallada de cómo deben comunicarse los clientes y servidores web. La estructura básica de la comunicación HTTP sigue lo que se conoce como modelo petición-respuesta.



La mayoría de las páginas web se escriben utilizando HTML el Lenguaje de Marcado de Hipertexto, que junto con HTTP es una tecnología web fundamental.

HTTP (HyperText Transfer Protocol)



- Una interacción HTTP se inicia cuando un cliente envía una solicitud al servidor.
- El cliente espera que el servidor genere una respuesta.
- El formato de los mensajes de solicitud y respuesta lo dicta HTTP.
- HTTP no dicta el protocolo de red que debe utilizarse para la comunicación, pero sí espera que tanto la solicitud como la respuesta se envíen dentro de una conexión de tipo TCP. Así que la mayoría de las implementaciones HTTP envían estos mensajes usando TCP.
- TCP garantiza la entrega ordenada y fiable de los datos entre el cliente y el servidor.

HTTP: Versiones de protocolo

Las versiones más utilizadas actualmente son:

- HTTP/1.1: versión utilizada mayoritariamente por mucho tiempo desde finales de los 90.
- HTTP/2: mejora la eficiencia mediante codificación binaria de mensajes, compresión de cabeceras, multiplexación de múltiples peticiones/respuestas sobre una única conexión TCP, peticiones iniciadas por el servidor, etc.
- HTTP/3: presenta algunas mejoras de eficiencia similares a las de HTTP/2 y utiliza el protocolo QUIC sobre UDP en vez de TCP.

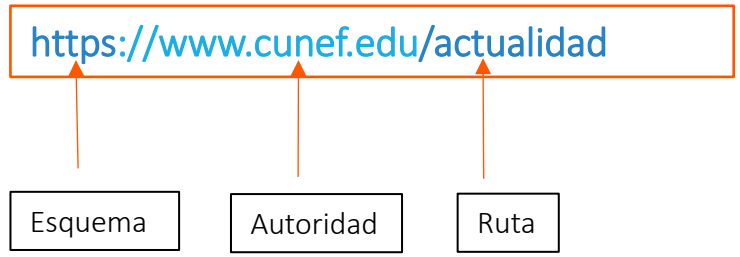
Las tres versiones del protocolo son compatibles en términos de semántica y estructura de los mensajes, cambiando principalmente la codificación de los mensajes y la forma en que estos se transportan sobre sus protocolos de transporte subyacentes.

HTTP: Recursos

Los recursos se identifican en HTTP mediante identificadores uniformes de recurso (URI, Uniform Resource Identifier).

Un URI es una secuencia de caracteres compacta que identifica a un recurso abstracto o físico, utilizado en múltiples protocolos y aplicaciones.

Un URI que además proporciona la información necesaria para localizar y acceder al recurso se denomina localizador uniforme de recurso (URL, Uniform Resource Locator).



HTTP: la URI (Unify Resource Indicator)

Partes de la URI

- Esquema: hace referencia al nombre de un esquema, que define como se asignan los identificadores en su ámbito. Los esquemas usados en HTTP son **http y https.** Otros esquemas pueden ser ftp, o mailto:
- Autoridad: elemento de una autoridad jerárquica de asignación de nombres, típicamente basado en un nombre de dominio de DNS o una dirección de red (IP, IPv6) y, opcionalmente, un número de puerto. Opcionalmente puede incluir tambien información de autenticacion
- Ruta: identifica un recurso en el ámbito del esquema y autoridad proporcionados, típicamente organizado jerárquicamente en fragmentos separados por barras (“/”).

HTTP: la URI (Unify Resource Indicator)

Partes de la URI

- Consulta (query): Es una cadena de datos opcional que envía información adicional al servidor. Se coloca después de un signo de interrogación (?). Si hay varios parámetros de consulta, se separan por el carácter &.
- Identificador de fragmento: identifica un recurso secundario en el contexto del recurso primario como, por ejemplo, un fragmento concreto (sección) de una página Web. Va después del símbolo #.

Métodos HTTP

METHOD	SOLICITA AL SERVIDOR QUE...	USO PRINCIPAL
GET	Devuelva el recurso especificado por la URL de solicitud sea devuelto como cuerpo en un mensaje de respuesta.	Obtener recursos, páginas
HEAD	Devuelva solo la cabecera del recurso especificado por la URL de solicitud.	Realizar pruebas
POST	El cuerpo del request sean procesados por el recurso especificado por la URL de solicitud.	Envío de formularios
PUT	El cuerpo del request actualice o reemplace un recurso específico en el servidor.	Escribir datos en el servidor
DELETE	Elimine un recurso específico.	Eliminar recursos (La autenticación y la autorización desempeñan un papel fundamental)
TRACE	Devuelva una copia del mensaje de petición HTTP completo, los campos de cabecera y el cuerpo, recibido por el servidor.	Realizar pruebas

HTTP

```
$ telnet www.example.org 80
Trying 192.0.34.166...
Connected to www.example.com
(192.0.34.166).
Escape character is '^]'.
GET / HTTP/1.1
Host: www.example.org
```

```
HTTP/1.1 200 OK
Date: Thu, 09 Oct 2003 20:30:49 GMT
Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html
<HTML>
<HEAD>
<TITLE>Example Web Page</TITLE>
</HEAD>
<body>
<p>You have reached this web page by typing
&quot;example.com&quot;,
&quot;example.net&quot;,
or &quot;example.org&quot; into your web browser.</p>
<p>These domain names are reserved for use in documentation
and are
not available for registration. See
<a href="http://www.rfc-editor.org/rfc/rfc2606.txt">RFC
2606</a>,
Section 3.</p>
</BODY>
</HTML>
```

REQUEST

RESPONSE

HTTP

Una respuesta HTTP incluye:

- **Un estado:** código numérico indicando el resultado del procesado de la petición.
- **Cabeceras de la respuesta:** datos adicionales acerca de cómo debe ser procesada la respuesta.
- **Cuerpo de la respuesta:** representación de la respuesta a la petición, típicamente una página HTML, una imagen, etc.

HTTP

Existen cinco tipos de código de estado:

- 1XX: de información.
- 2XX: petición procesada con éxito.
- 3XX: redirección a otro recurso.
- 4XX: error en la petición.
- 5XX: error en el servidor.

Código	Razón	Significado
200	**Ok**	Petición procesada con éxito.
301	**Moved Permanently**	Recurso movido a otro URL (cabecera Location), que el cliente debe usar siempre a partir de ahora.
302	**Found**	Recurso movido temporalmente a otro URL (cabecera Location).
303	**See other**	Se debe cargar otro recurso (Pagina de confirmación, progreso, etc.) con método GET.
304	**Not Modified**	El cliente puede usar su versión del recurso en cache.
400	**Bad Request**	El cliente envió una petición HTTP invalida (sintaxis, etc.).
403	**Forbidden**	El cliente no puede acceder al recurso.
404	**Not Found**	No existe un recurso con la ruta dada.
405	**Method Not Allowed**	El recurso no admite el método indicado en la petición.
500	**Internal Server Error**	Error en el servidor al procesar la petición.

HTTP: cookies

Las cookies permiten mantener estado, consisten en pequeñas cantidades de datos asociados a un nombre que el servidor genera y envía al cliente en mensajes de respuesta, para que este las incluya en sucesivas peticiones.

HTTP es un protocolo sin estado, esto significa que cada petición es independiente de las demás.

HTTP: cookies

Una cookie se representa como una pequeña cadena de texto que contiene:

- Un nombre: un servidor puede establecer varias cookies con distintos nombres.
- Un valor: los datos de la cookie en sí mismos.
- Atributos:
  - Domain y Path: definen en qué peticiones, por autoridad y ruta, el cliente enviara la cookie al servidor.
  - Expires y Max-Age: definen cuándo la cookie debe dejar de ser utilizada por el cliente. Si no se especifica ninguno, se elimina al cierre del navegador.
  - Secure: la cookie solo puede ser enviada por canales seguros (HTTPS típicamente).
  - HttpOnly: solo se debe enviar o permitir acceso a la cookie a través de HTTP o HTTPS. Por ejemplo, no debe ser accesible a código JavaScript (por motivos de seguridad).

HTTP: cookies

Algunos usos típicos de las cookies son los siguientes:

- Gestión de sesiones: el usuario se autentica al principio para crear una sesión (el servidor envía una cookie con un token de sesión). El servidor identifica peticiones subsiguientes como parte de la misma sesión porque incluyen este mismo token de sesión.
- Almacenamiento de preferencias en el lado del cliente: se pueden almacenar en cookies las preferencias del usuario para el sitio Web en su navegador Web.
- Rastreo de usuarios: los sitios Web pueden utilizar cookies para rastrear el comportamiento de los usuarios (cuando terceros hacen este rastreo, por ejemplo, con fines comerciales, se puede considerar que su uso es abusivo)

HTTPS: SSL (Secure Sockets Layer)

Al principio sólo se utilizaba la web para distribuir páginas estáticas. Pero al surgir la idea de realizar transacciones financieras a través de la web, fue necesario asegurar el manejo de los datos bancarios de los usuarios al momento de realizar la comunicación entre el ciente y el servidor.

HTTPS: SSL (Secure Sockets Layer)

En 1995, Netscape Communications Corp., introdujo un paquete de seguridad llamado SSL (Secure Sockets Layer). Este software y su protocolo son usados por los navegadores web modernos.

HTTPS: SSL (Secure Sockets Layer)

SSL construye una conexión segura entre dos sockets, incluyendo:

1. Negociación de parámetros entre cliente y servidor.
2. Autenticación del servidor por parte del cliente.
3. Comunicación secreta.
4. Protección de la integridad de los datos.

HTTPS: SSL (Secure Sockets Layer)

Cuando se utiliza HTTP sobre SSL, se denomina HTTPS (HTTP seguro), aunque se trate del protocolo HTTP estándar.

SSL actua como una capa entre la aplicación y el transporte, esta acepta las peticiones del navegador y las envía a TCP para transmitirse al servidor.

Una vez establecida la conexión segura, SSL se encarga de gestionar la compresión y el cifrado.

HTTPS: SSL (Secure Sockets Layer)

Aplicación (HTTP)
Seguridad (SSL)
Transporte (TCP)
Red (IP)
Enlace de datos (PPP)
Física (módem)

Capas (y protocolos) para un usuario doméstico que navega con SSL.

HTTPS

Propiedades del protocolo HTTPS:

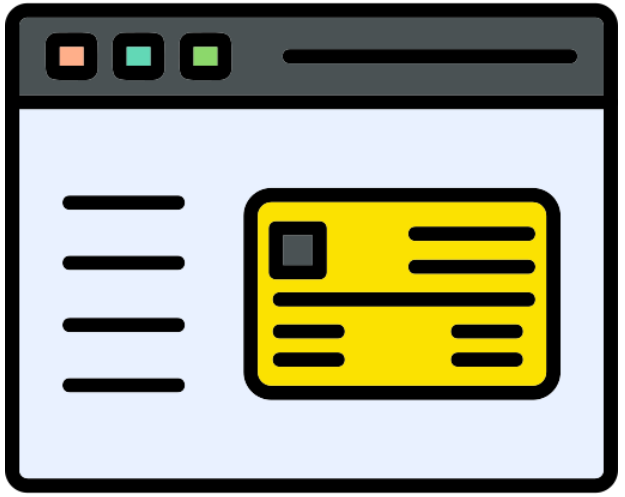
- Autenticación: se autentica siempre al servidor y, opcionalmente, al cliente.
- Confidencialidad: los datos enviados por el canal seguro una vez este se ha establecido son solo visibles por los dos extremos del mismo.
- Integridad: cualquier modificación de los datos enviados por el canal seguro una vez se ha establecido este serán detectada.

04

Páginas estáticas vs dinámicas

Páginas estáticas

Una página estática es una página web que es devuelta tal cuál está almacenada en el servidor, por lo tanto la página web se ve igual para cualquier usuario que solicita acceso a la misma, su contenido solo puede ser editado cambiando el contenido de los archivos en el servidor.



Páginas dinámicas

Es una página web que varia según la interacción con el usuario además de la programación por el lado del servidor.

Páginas estáticas vs Páginas dinámicas

Página web estática	Página web dinámica
Realizadas principalmente en XHTML o HTML.	Muchas posibilidades en diseño y desarrollo.
Enfocada principalmente a mostrar una información.	Utiliza varios lenguajes y técnicas de programación en su desarrollo.
Portabilidad, funciona en cualquier servidor.	Permite un gran número de funcionalidades tales como bases de datos, foros, etc.
Muy rápidas, acceso óptimo.	El usuario puede alterar el diseño, contenidos o presentación de la página a su gusto.
Máximo desempeño y funcionalidad.	Suele estar almacenada en bases de datos, de las cuales se extrae una parte según las acciones que realiza el visitante.
Facilitan el posicionamiento.	
Costos de alojamientos menores.	
No se requiere ninguna instalación ni configuración de software.	

05

CMS y DXP

CMS (Content Management System)

Un **CMS** (Content Management System o Sistema de Gestión de Contenidos) es una aplicación de software que permite la generación de contenido en un sitio web sin necesidad de tener conocimientos avanzados sobre programación.

Algunos de los más populares son Wordpress , Joomla y Drupal.

También los hay especializados en eCommerce como Shopify, Magento o ZenCart.

CMS (Content Management System): ejemplo WORDPRESS

Es un CMS de código abierto que inicialmente estuvo enfocado como una plataforma para blogs, pero debido a la flexibilidad y la comunidad de desarrolladores con las que cuenta, se pueden añadir plugins y aplicaciones para modificar o añadir funcionalidades, por lo que puede ser usado para realizar todo tipo de páginas web.

DXP (Digital Experience / Portales)

Un **DXP** (Digital Experience Platform o Plataforma de Experiencia Digital) es una solución más avanzada que un CMS, diseñada para ofrecer experiencias personalizadas y coherentes en múltiples canales (sitios web, aplicaciones móviles, redes sociales, etc.)

Mientras que un CMS se enfoca principalmente en la gestión de contenido, un DXP abarca aspectos más amplios como la personalización, análisis de usuarios, y la integración con otras herramientas empresariales, permitiendo una experiencia digital más rica y personalizada.

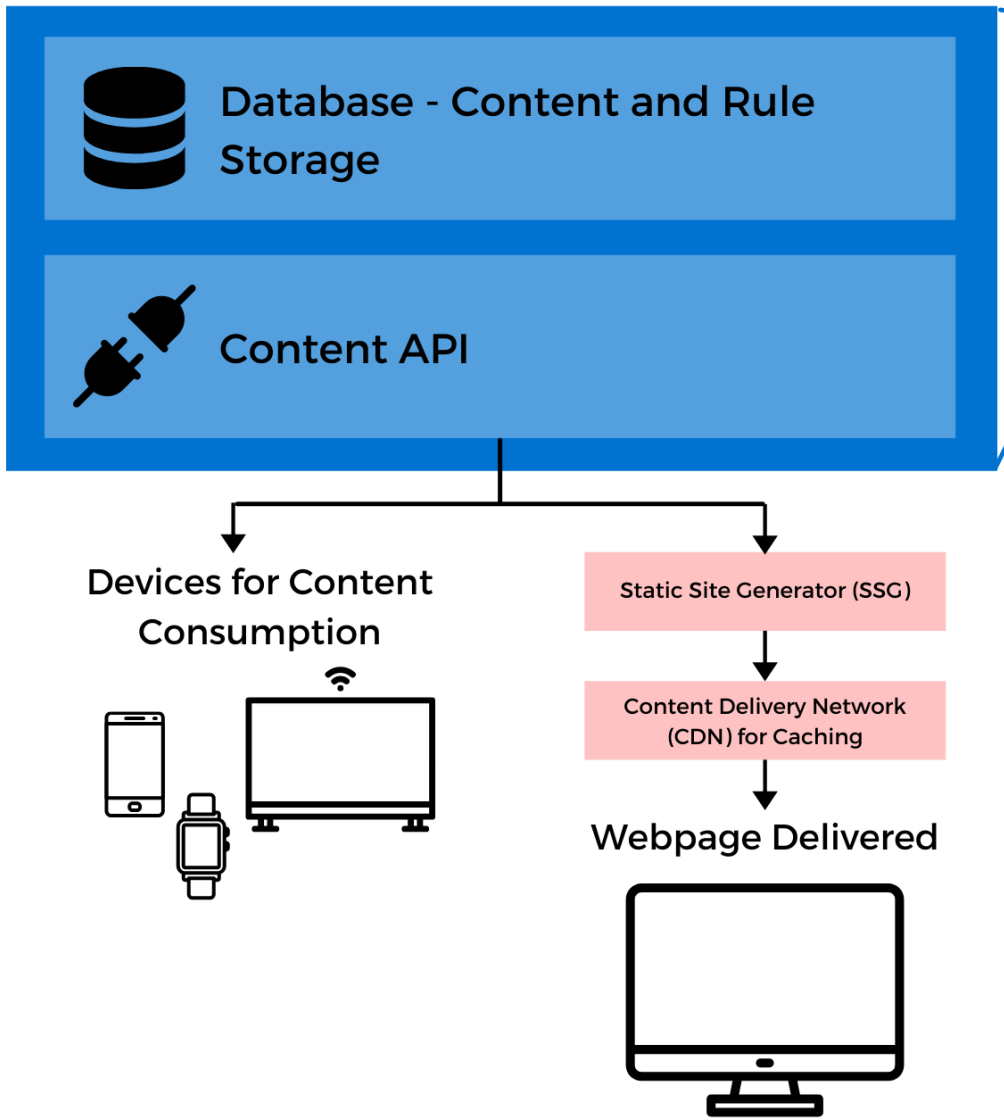
HCL Digital Experience

Liferay

Adobe Experience Manager

Como ejemplos de DXP podemos nombrar **Adobe Experience Manager**, **Liferay**, o **HCL DXP**, que permiten la gestión del contenido junto con herramientas avanzadas de marketing y análisis.

Headless Content Management System (CMS)



Digital Experience Platform (DXP)

- Content Management System (CMS)
- Customer Relationship Manager (CRM)
- Digital Asset Manager (DAM)
- Commerce Engine
- Personalization Engine
- Portals
- Other Services

06

Redes Sociales y Marca Personal

Redes sociales y marca personal

Para poder destacar en el mercado laboral actual es necesario saber usar de manera efectiva las redes sociales para presentar el perfil y aptitudes laborales, para esto se pueden usar redes sociales como:

- LinkedIn: Contar con un perfil actualizado y publicar artículos de manera regular en esta red social puede ayudar a construir una red de contactos profesional.
- Github: Publicar proyectos personales o contribuir a proyectos ya existentes contribuye en aumentar la credibilidad y visibilidad.

07

Roles en un equipo de desarrollo

Roles en un equipo de desarrollo

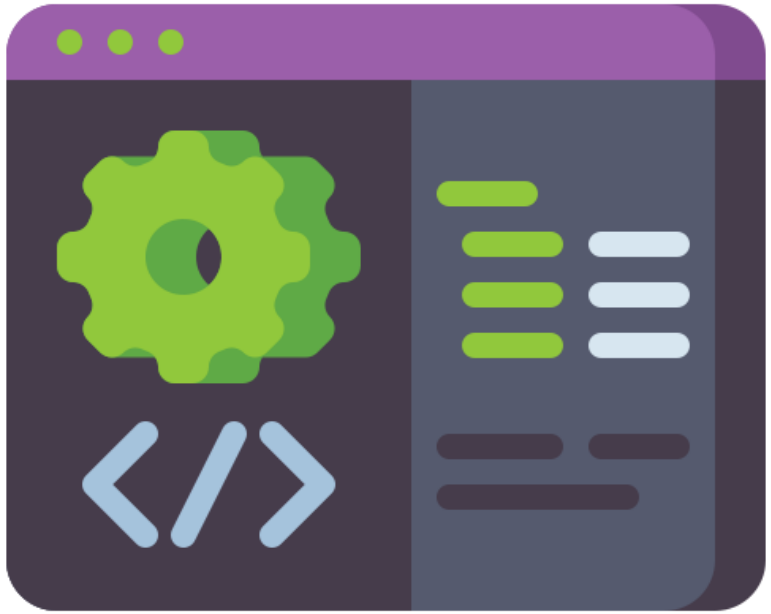
1) Desarrollador Backend
2) Desarrollador Frontend
3) Ingeniero QA
4) UX/UI Designer

1) Desarrollador Backend

Desarrolla y mantiene la lógica del servidor, generando APIs que permiten que las aplicaciones de los usuarios interactúen con la información almacenada en la base de datos.

Las páginas web dinámicas obtienen datos en tiempo real a través de las APIs alojadas en el backend.

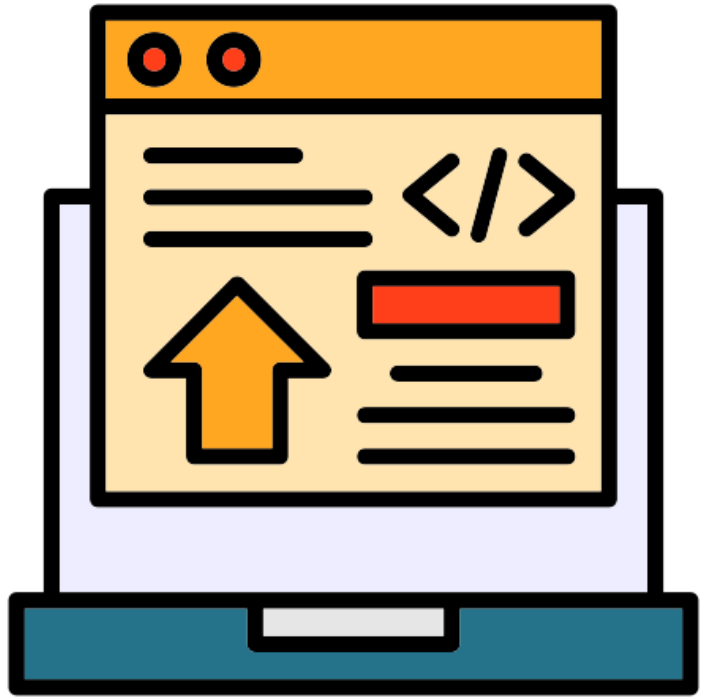
La sintaxis y el entorno de desarrollo varían según el lenguaje de programación elegido, los lenguajes principalmente usados son: Java, PHP, Python, Javascript



2) Desarrollador Frontend

Construye la interfaz de usuario, asegurando que las páginas web sean atractivas para el usuario .

Además implementa las funcionalidades necesarias para que los datos enviados por el backend se muestren de manera efectiva en la interfaz de usuario.



Lenguajes y Herramientas Principales:

- HTML
- CSS
- Javascript

3) Ingeniero QA

Se encarga de realizar distintas pruebas para identificar y corregir errores en la aplicación, asegurandose de esta manera que el software funcione correctamente cumpliendo con los estándares del usuario

Lenguajes y Herramientas Principales:

- Selenium
- Cypress
- Git

4) UX/UI Designer

Diseña y mejora continuamente la interfaz de la aplicación para asegurar la mejor experiencia para el usuario, todo esto lo hace en base a investigación y análisis del comportamiento de los usuarios.

Herramientas principales:

- Figma, Zeplin
- Adobe XD

08

Introducción a lenguajes del lado de servidor

¿Qué es un lenguaje del lado del servidor?

Es un lenguaje de programación utilizado para manejar la lógica del negocio y realizar operaciones en el servidor. Genera contenido dinámico para luego ser enviado al cliente. De esta manera la aplicación a la que accede el usuario se personaliza según la interacción que este realice con ella

JAVA

Java es un lenguaje de programación orientado a objetos utilizado principalmente en el desarrollo de aplicaciones empresariales

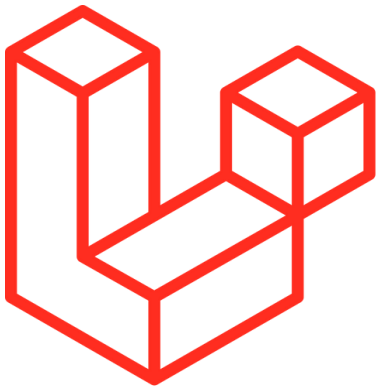
Las ventajas que presenta son su robustez, seguridad y ecosistema de desarrollo.

Su framework principal es Spring, este permite el desarrollo de aplicaciones empresariales

PHP

PHP es un lenguaje de scripting del lado del servidor utilizado principalmente en el desarrollo web. Permite a los desarrolladores crear contenido dinámico y gestionar interacciones con bases de datos.

Las principales ventajas que presenta son su facilidad de uso y flexibilidad para adaptarse a las necesidades de desarrollo.



Sus frameworks y herramientas más populares son:

- Laravel (Desarrollo web)
- CakePHP (Desarrollo web)

Python

Python es un lenguaje de programación de alto nivel, interpretado, y orientado a objetos, conocido por su sintaxis clara y legible.

La principal ventaja que presenta es su versatilidad y escalabilidad para construir aplicaciones.

Sus casos de uso más comunes son:

- Desarrollo Web
- Análisis de datos
- Aprendizaje Automático e Inteligencia Artificial

Sus frameworks y herramientas más populares son:

- Django (Desarrollo web)
- Flask (Desarrollo web)
- Pandas (Análisis de datos)
- TensorFLow (Redes neuronales)

JavaScript

JavaScript es un lenguaje de programación interpretado y orientado a objetos que permite la creación de contenido interactivo y dinámico en las páginas web.

Aunque principalmente es usado para el desarrollo frontend, también puede ser ejecutado en el servidor mediante la plataforma de Node.js.

Debido a su caracteristica “asincrona” y no bloqueante, es muy apropiado para ejecutar operaciones i/o.

Las ventajas que presenta son su gran compatibilidad con los navegadores web además de la variedad de bibliotecas y frameworks disponibles para el desarrollo.

JavaScript

Sus casos de uso más comunes son:

- Desarrollo Web
- Desarrollo de Aplicaciones Móviles

Sus frameworks y herramientas más populares son:

- React (Desarrollo web frontend)
- Node.js (Desarrollo web backend)
- Express.js (Desarrollo web backend)
- Nest.js (Desarrollo web backend)

CUNEF UNIVERSIDAD

5 años

GRACIAS POR VUESTRA ATENCIÓN