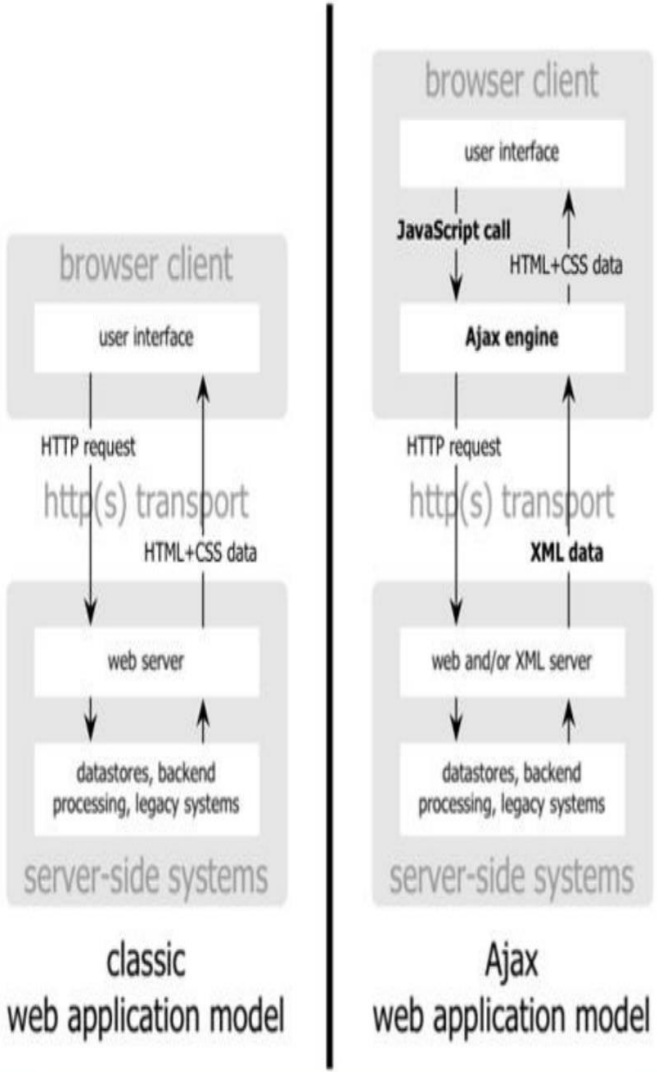
UNIVERSIDAD D CÓRDOBA

PROGRAMACIÓN WEB – SEMINARIO 2

Fundamentos de Internet

Dr. José Raúl Romero Salguero
jrromero@uco.es



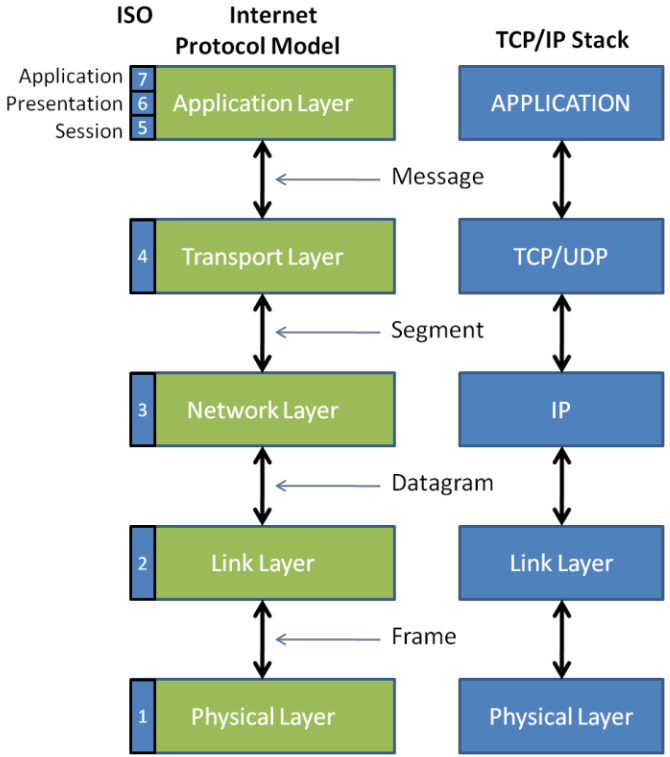
1.

Internet como red_ Protocolos

Empecemos por las entrañas…

Internet

- ¡**Recordemos**! Internet es un conjunto de redes físicas heterogéneas, interconectadas mediante protocolo TCP/IP
- El uso de TCP/IP permite componer una red lógica de nodos a escala global
- Múltiples capas de protocolos de comunicación: IP→ TCP/UDP → HTTP/FTP/POP3/SMTP/SSH…



TCP: *Transmission Control Protocol*
IP: *Internet Protocol*

Internet __ Elementos influyentes: ISP

Además de routers e **ISPs**, actores como los puntos de intercambio de tráfico (IXPs) y las CDNs tienen un papel crucial en la experiencia de usuario.

- Un **ISP** (*Internet Service Provider*) es la entidad que proporciona acceso a Internet a usuarios finales (hogares, empresas, universidades, móviles). Gestiona asignación de direcciones IP, DNS recursivo, ancho de banda y políticas de tráfico.

  Influye en: asignación de IPs, latencia (calidad variable no medible), filtrado o bloqueo de puertos, cachés no controladas

Internet __ Elementos influyentes: IXP

- Un **IXP** (*Internet Exchange Point*) es un punto físico donde distintos ISPs, redes académicas y proveedores de contenido intercambian tráfico directamente, en lugar de pasarlo por transitarios internacionales.

  Esto reduce latencia (no solo depende de tu servidor o del usuario, también del camino entre ISP donde los IXP son clave), coste y congestión.

  Grandes plataformas como Netflix, Google o Cloudfare están presentes en los IXP para reducir distancias con sus usuarios – Influye en la experiencia de usuario.

  Es crítico elegir un proveedor *cloud* con presencia en los IXP relevantes para tu aplicación.

Ejemplos: DE-CIX (Frankfurt, Madrid), LINX (Londres), AMS-IX (Ámsterdam).

Internet __ Elementos influyentes: CDN-1

- Una **CDN** (red de distribución de contenidos) es un conjunto de servidores repartidos por numerosas ubicaciones que almacenan copias duplicadas de los datos, de modo que pueden satisfacer las solicitudes de datos en función de cuáles sean los servidores más cercanos a los respectivos usuarios finales.

  Las CDN permiten un servicio rápido menos afectado por el tráfico elevado: menor **carga** y mayor **escalabilidad**.

  Las CDNs están repartidas geográficamente (cerca de los usuarios), lo que afecta a la **latencia**, **fiabilidad** y **disponibilidad** de recursos.

  Suelen incluir terminación TLS, además de estar protegidos frente a **DDoS**. Ofertan HTTP/2 y HTTP/3/QUIC (la CDN negocia la versión, transparente al programador).

  Se utilizan ampliamente para la entrega de archivos estáticos, lo que reduce la carga de solicitudes en los servidores de la propia organización → Son muy utilizados para alojar sitios estáticos completos.

Internet __ Elementos influyentes: CDN-2

Cómo funciona:

- Cuando un usuario solicita contenido de una aplicación web, la solicitud se enruta al servidor CDN más cercano (también conocido como "***edge server***") en función de factores como la latencia de la red y la carga del servidor.
- A continuación, el "*edge server*" comprueba si el contenido solicitado ya está almacenado en la caché.
- Si es así, el contenido se sirve directamente desde la caché; de lo contrario, el servidor perimetral obtiene el contenido del servidor de origen, lo almacena en la caché y lo sirve al usuario.
- Las solicitudes posteriores del mismo contenido pueden servirse desde la caché, lo que reduce la latencia y descarga el tráfico del servidor de origen.

Ejemplos: Cloudflare, Akamai, Fastly, Amazon CloudFront.

Internet __ Elementos influyentes

Los inicios del *streaming* (2007-2010):

- Netflix servía contenido desde *datacenters* distribuidos en EEUU, alquilando proveedores *cloud*
- El video se transmitía atravesando IXP e ISP internacionales hasta el usuario final
- **Problemas**: altísima latencia, congestión en hora punta y elevados costes de ancho de banda

Éxito implica escalabilidad (2010-2012):

- Netflix pasa de pocos millones a decenas de millones de suscriptores en 5 años
- Se generan disputas con los ISP (p.ej. Comcast en EEUU limitaba la calidad del video para no congestionar sus redes)
- **Problema**: Netflix dependía de terceros

Internet __ Elementos influyentes NETFLIX

Implantación de su CDN propio (2012-):

- Netflix anuncia **Open Connect**, su propio CDN
- Construye multitud de OCA (*Open Connect Appliances*) con discos duros llenos de su contenido (series, películas) y los instala directamente dentro del ISP (red local)
- Algoritmos de predistribución de contenido:
  - Los OCA se descargan en horas valle de forma anticipada.
  - Contenido personalizado por país (según gustos de usuarios).
  - Al pulsar "Play", el contenido ya está en el servidor propio.
- Desarrollo de algoritmos de enrutamiento (DNS) inteligente hacia el OCA más cercano.

Datos actuales (2025):

- +18.000 OCA con ISP en 100 países.
- La mayoría del tráfico de Netflix nunca llega a Internet global.
- Un usuario de Córdoba verá su serie desde un OCA en Madrid (ISP: Telefónica)
- Netflix cuenta con la red de CDN privada mayor del mundo, equiparable a las redes de Cloudfare (proveedor de CDN).

Internet __ Elementos influyentes

Lecciones aprendidas

Escalar requiere proximidad

CDNs son críticos para recursos pesados

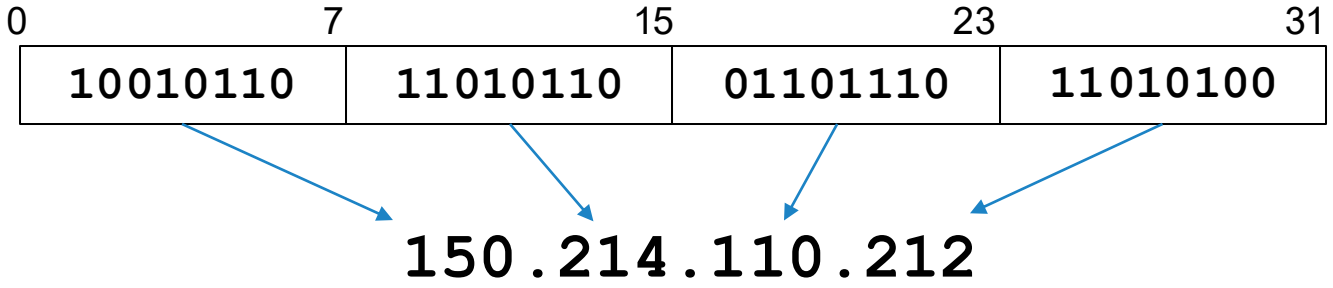
Si controlas la entrega, controlas la experiencia

Mejor protocolos más modernos
(se adaptaron rápidamente a HTTP/3/QUIC)

El diseño de un sistema web no es solo programar

Internet Protocol (IP)

- Sencillo protocolo de envío de datos entre dos máquinas
- Cada dispositivo tiene una dirección de 32 bits, separada en 4 bloques de 8 bits (0-255) en IPv4



- **IETF** (*Internet Engineering Task Force*) e **IANA** (*Internet Assigned Numbers Authority*) tienen reservadas direcciones IP para fines concretos

Internet Protocol (IP)

- **IPv4** (32 bits) es todavía predominante frente a **IPv6** (128 bits), a pesar del agotamiento de direcciones, y las ventajas de IPv6 (autoconfiguración, espacio y seguridad nativa)
- Algunas direcciones IP reservadas de interés:
  - 0.0.0.0 – 0.255.255.255 : Rango de direcciones de origen
  - 10.0.0.0 – 10.255.255.255 : Direcciones de red privada
  - 127.0.0.0 – 127.255.255.255 : Direcciones de *loopback*

    .1 localhost

Otras IP reservadas para TEST-NET, multidifusión, multicast, etc.

Protocolo TCP

- TCP garantiza la entrega de los mensajes sobre IP, en contraposición a UDP (*User Datagram Protocol*)
- Algunos programas (juegos, *streaming*) utilizan el protocolo UDP, que es más sencillo y rápido, ya que puede resultar asumible la pérdida de mensajes
- TCP/IP utiliza multiplexado__ múltiples aplicaciones utilizando la misma dirección IP
  - Se asigna un número de puerto a cada programa o servicio, que en general es accedido mediante *socket* de red
  - RFC 6335 "*Procedures for the Management of the Service Name and Transport Protocol Port Number Registry*": https://tools.ietf.org/html/rfc6335

Protocolo TCP__ Puertos

- Gestionados por la *Internet Assigned Numbers Authority* (IANA), fundación privada de EEUU perteneciente a la ICANN, que asigna cada puerto a aplicaciones concretas
- Puertos de 0 a 1023 (0 a 2^{10}-1) se denominan puertos de sistema:
  - 21: FTP (file transfer protocol)
  - 22: SSH (secure shell)
  - 23: Telnet
  - 25: email
  - 53: DNS (domain name server)
  - 80: HTTP (hypertext transfer protocol)
  - 443: HTTP sobre TLS/SSL (HTTPS)
  - 990: FTPS (FTP sobre TLS/SSL)
  - 992: Telnet sobre TLS/SSL

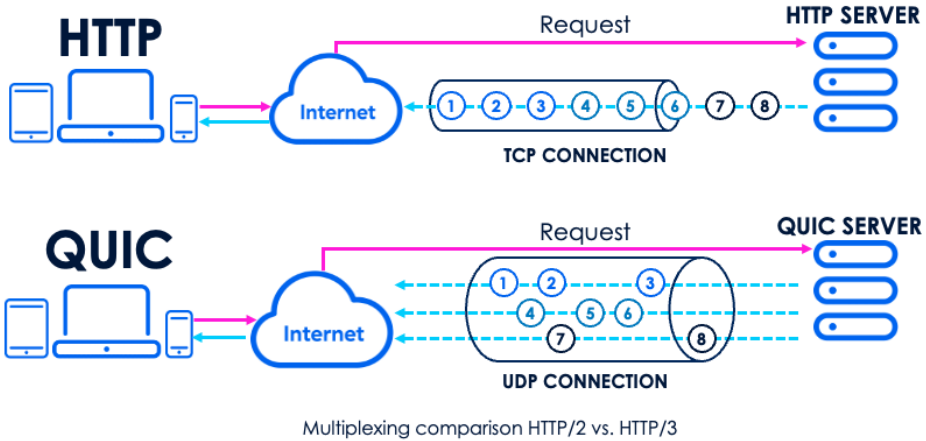
Protocolo TCP__ Puertos

- Puertos de 1024 a 49151 (2^{10} a $2^{14}+2^{15}-1$) son puertos registrados a demanda de las entidades o empresas que los reservan:
  - 1433: MS SQL Server
  - 1521: Oracle listener
  - 3306: MySQL
  - 3690: Subversion
  - 5400,5500,5600,5700,5800,5900: VNC (escritorio remoto sobre HTTP)
  - 8000: iRDMI (*Intel Remote Desktop Management Interface*) – Utilizado erróneamente como puerto HTTP, lo que supone un riesgo de amenaza
  - 8080: HTTP alternativo

  Lista completa en Wikipedia:
  https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

Protocolo QUIC

- **QUIC** (*Quick UDP Internet Connections*) es un protocolo multiplexado a nivel de transporte (como TCP o UDP) que se construye sobre UDP.
- **Características principales**:
  - Establecimiento de conexión más rápido (100-300ms en HTTP/2 frente a 0-100ms en HTTP/3)
  - Un problema de TCP es HOL (***head-of-line***), es decir, si se pierde un solo paquete, todos los siguientes paquetes están bloqueados hasta que el paquete perdido se detecta y se retransmite. QUIC utiliza flujos independientes, permitiendo varios flujos (*streams*) en una única conexión (multiplexión)



Multiplexing comparison HTTP/2 vs. HTTP/3

Protocolo QUIC

- QUIC incluye la encriptación de forma nativa. TCP la debe incluir con TLS.
- El cambio entre redes móviles y fijas puede provocar interrupciones en la conexión end-to-end (cliente-servidor), lo que provoca reconexiones y repeticiones en TCP. QUIC incorpora mecanismos para reinicio rápido y migración de la conexión (p.ej. cambio de dirección IP de una red a otra)
- Latencia más corta en QUIC (conexiones se identifican con un ID único, no por la IP)

- Más apropiado para tráfico de *streaming* y *gaming*, también para aplicaciones móviles, dispositivos IoT (*real-time*) e IoV (*Internet of Vehicles*), *cloud computing* y aplicaciones *eCommerce* (seguridad y privacidad)
- **Resistencia a QUIC**: Los administradores tienen reticencia a implantar QUIC debido a su imposibilidad de inspeccionar el tráfico de la red

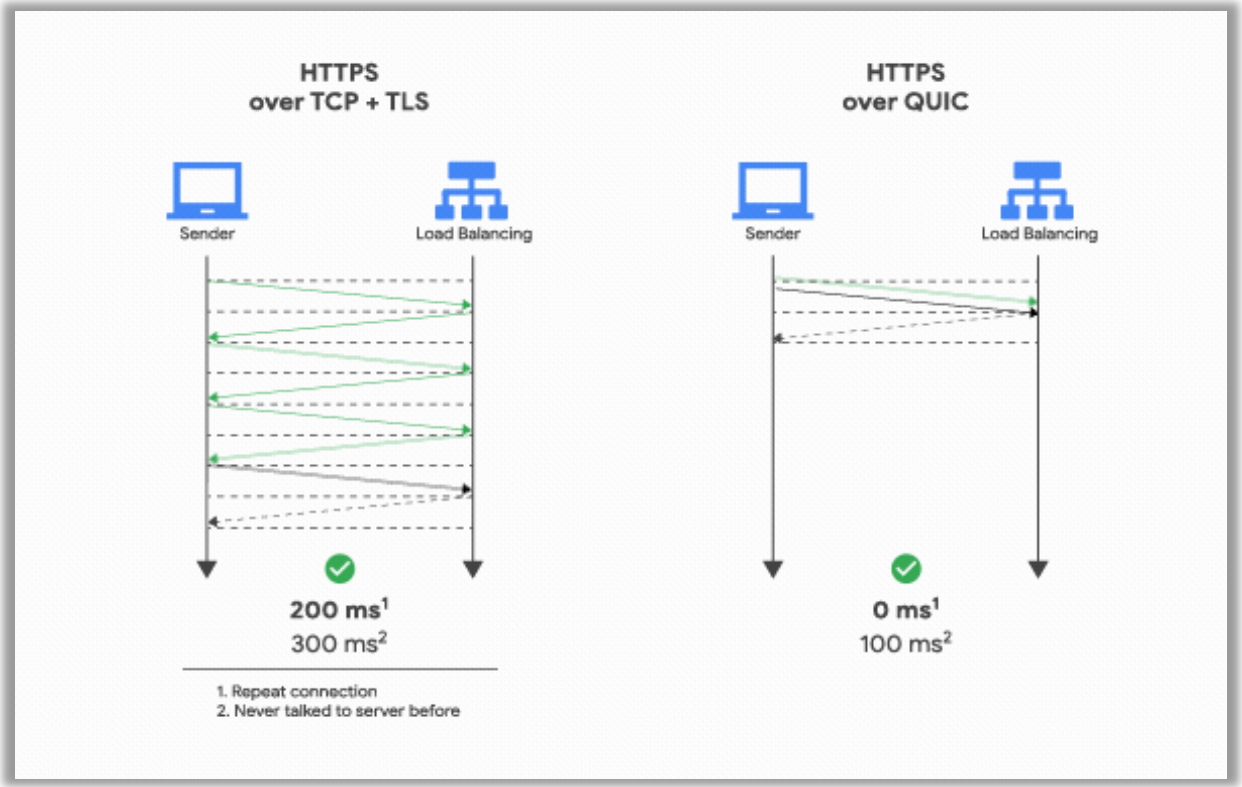
Protocolo QUIC

- Se utiliza mayoritariamente TCP+TLS para que los firewalls puedan realizar inspecciones de tráfico
  - Los firewalls bloquean QUIC
  - Los navegadores bloquean QUIC
- IETF QUIC Working Group versión 1

  https://quicwg.org/
- IETF RFC 9000

  https://datatracker.ietf.org/doc/html/rfc9000

Protocolo QUIC

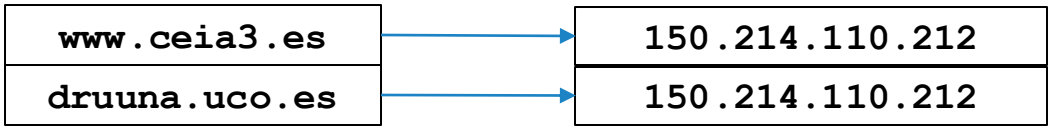


Fuente: Google

1. **Cliente envía un paquete UDP** al servidor con la petición inicial (incluyendo el handshake TLS 1.3).
2. **Servidor responde** confirmando la conexión y el cifrado.
3. En la primera conexión: 1-RTT (ida y vuelta).
4. En reconexiones: 0-RTT (sin ida y vuelta extra).
5. Una vez establecida, se pueden abrir **múltiples flujos paralelos** dentro de la misma conexión QUIC.
6. Si hay pérdida de un paquete en un flujo, solo ese flujo espera la retransmisión → los demás siguen avanzando.

Domain Name System (DNS)

- Implementación de la transparencia de nombrado ofrecida por sistemas distribuidos
- Conjunto de servidores que mapean nombres a direcciones IP
- Sistema jerárquico; seguridad **DNSSEC** para garantizar autenticidad
- Privacidad: *DNS over HTTPS* (DoH) y *DNS over TLS* (DoT)

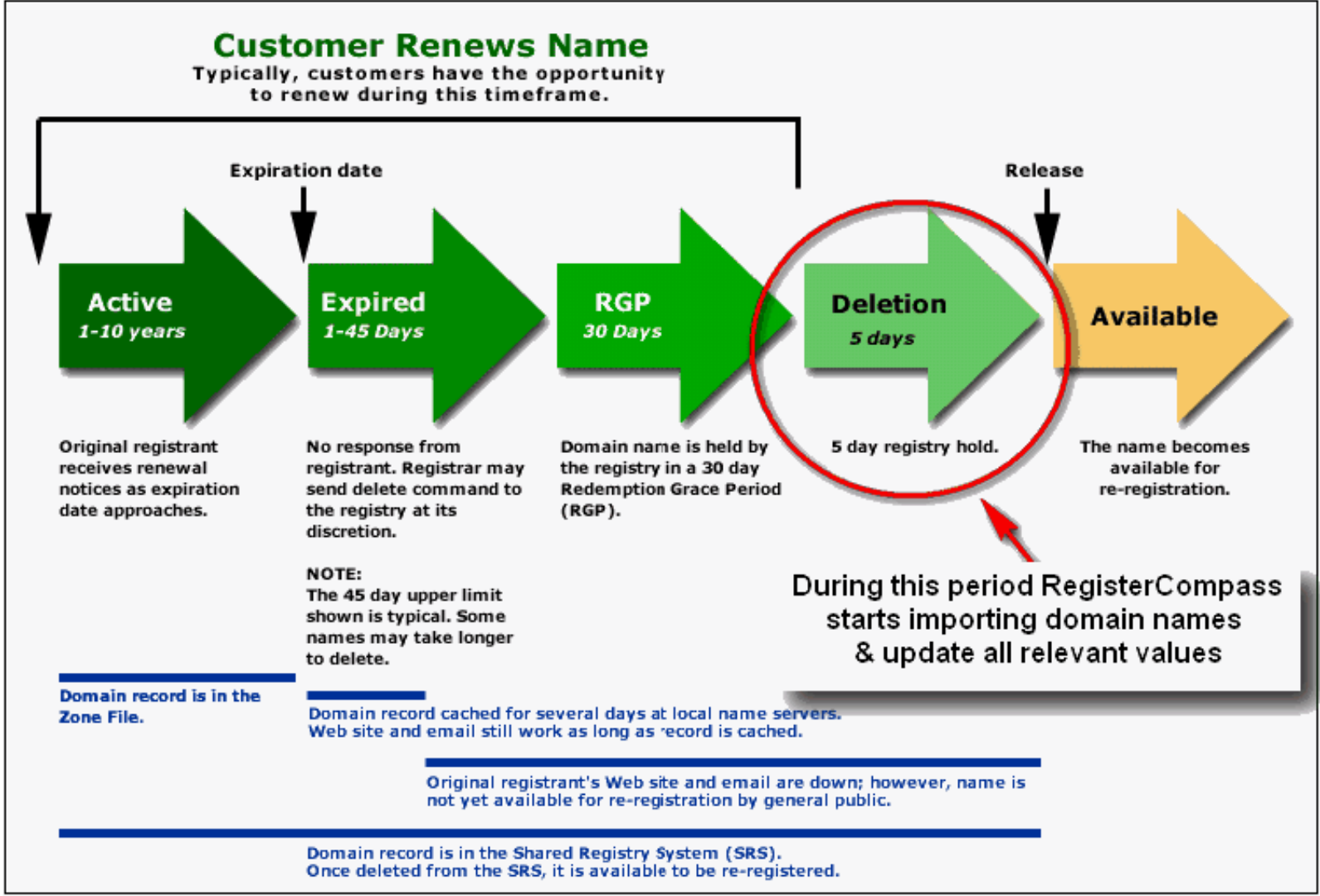


- El servicio de registros WhoIs permite consultar esta correspondencia. Por ejemplo: https://whois.domaintools.com/

Domain Name System (DNS)

- Nuestras máquinas pueden guardar copias locales (caché) del fichero *hosts*
  - Sistemas Windows: `C:\Windows\System32\drivers\etc\hosts`
  - Sistemas MacOS: `/private/etc/hosts`
  - Sistemas Unix/Linux: `/etc/hosts`
  - Sistemas Android: `/system/etc/hosts`
- Conocer estos ficheros (texto plano) es útil para redireccionar dominios a local mientras se están desarrollando
  - ¡**Mucha precaución con la edición de estos ficheros**!
- Sistema susceptible a ataques (p.ej. redireccionando a *phishing*)

Domain Name System (DNS)



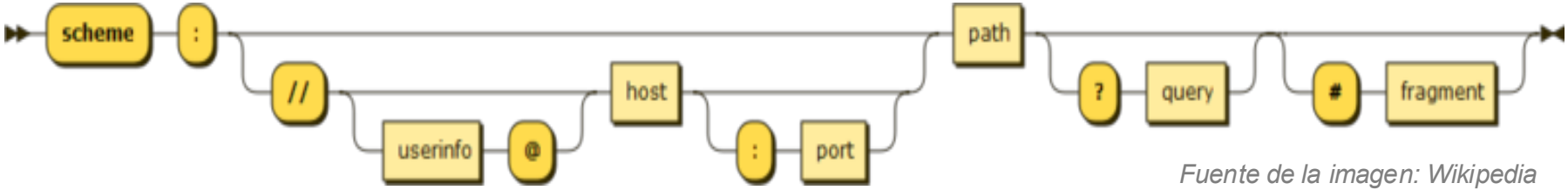
Uniform Resource Identifier (URI)

- Cadena de caracteres que sirve como localizador, nombre, o ambos
- Identificación no ambigua de un recurso concreto, definiendo un espacio de nombres separados de forma jerárquica
- RFC 3986 "*Uniform Resource Identifier(URI): Generic Syntax*": https://tools.ietf.org/html/rfc3986
- Utilizada para definir el acceso único y no ambiguo de cualquier recurso: direcciones de páginas web, datos o ítems de la web semántica, servicios y APIs, etc.

Uniform Resource Identifier (URI)

```
ftp://ftp.is.co.za/rfc/rfc1808.txt
ldap://[2001:db8::7]/c=GB?objectClass?one
mailto:John.Doe@anexample.com
telnet://192.0.2.16:80/
http://www.jrromero.net
```

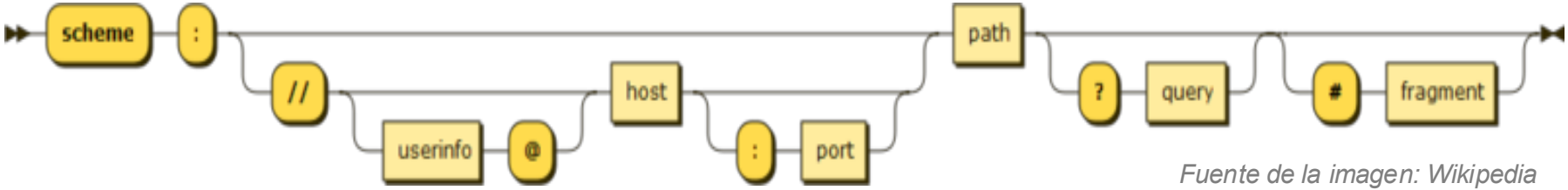
Uniform Resource Identifier (URI)



Fuente de la imagen: Wikipedia

- El esquema se especifica como cadena de caracteres no vacía y minúscula (http, https, ftp, telnet, mailto, etc.)
- Componente de autoridad compuesto por:
  - usuario + ":" + *password*
  - *Host*: nombre de servidor o dirección IP en notación decimal con puntos
  - Número de puerto
- Ruta (*path*) separada por segmentos "/". Puede ser vacía y debe corresponderse con la ruta del sistema de ficheros (no necesariamente 1:1)

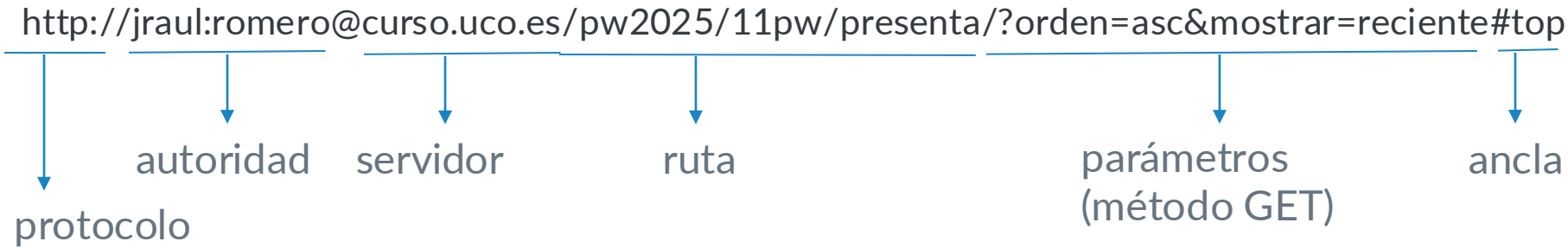
Uniform Resource Identifier (URI)



Fuente de la imagen: Wikipedia

- Un componente de consulta (*query*) con una cadena de consulta sobre datos no jerárquicos. No está estandarizado su formato, si bien es habitual la secuencia de tipo "`?clave1=valor1&clave2=valor2`"
  - Utilizado en el caso de paso de parámetros por método GET.
  - Los caracteres están reservados como delimitadores: `: / ? # [ ] @`
- Un componente fragmento, que contiene un identificador de un recurso secundario, como anclas (*anchors*) en secciones de HTML

Uniform Resource Locator (URL)



- El protocolo determina el resto de la estructura
- El dominio del servidor es mapeado por el DNS a su dirección IP
- Al ser protocolo HTTP, el puerto por defecto (según ICANN) es 80
- La ruta finaliza en un directorio, por lo que el recurso invocado por defecto es determinado en la parte de servidor (`index.php`, `index.jsp`, `index.html`, ...)

2.

Protocolo HTTP

¡No sólo es lo que escribimos en el navegador!

Hypertext Transport Protocol (HTTP)

- Establece el conjunto de comandos interpretados por el servidor web
- Creado inicialmente por Tim Bernes-Lee en el CERN (1989)
- Actualmente, desarrollado por IETF
- Versión inicial HTTP/1.1 y actualmente HTTP/2
- La versión más reciente – HTTP/3 – se liberó en Junio de 2022
  - RFC 9114 "*Hypertext Transfer Protocol Version 3 (HTTP/3)*": https://datatracker.ietf.org/doc/html/rfc9114
- La mayoría de los navegadores ya lo soportaban desde hace algún tiempo
- Trabaja sobre TLS (*Transport Layer Security*) 1.3 o superior + UDP
  - RFC 7301 "*Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension*": https://tools.ietf.org/html/rfc7301

Hypertext Transport Protocol (HTTP)

Comparativa HTTP/1.1 vs. HTTP/2 vs. HTTP/3

- **HTTP/1.1:** No tiene multiplexación; cada solicitud requiere una nueva conexión TCP, lo que genera sobrecarga y ralentiza la carga de páginas.

- **HTTP/2:** Introduce multiplexación sobre una única conexión TCP, enviando múltiples flujos de datos simultáneamente y evitando que una solicitud bloquee a otras. Puede causar bloqueo HoL (*head-of-line*) por pérdida de algún paquete.

- **HTTP/3:** Ofrece multiplexación a través de QUIC, que opera sobre la capa de aplicación, permitiendo flujos de datos múltiples, independientes y sin bloqueos, incluso si la conexión subyacente tiene problemas.

La **multiplexión** permite la transmisión simultánea de múltiples solicitudes y respuestas a través de una **única conexión TCP** o **QUIC**, en lugar de requerir una conexión separada para cada petición

Hypertext Transport Protocol (HTTP)

Comparativa HTTP/1.1 vs. HTTP/2 vs. HTTP/3

HTTP/1.1: Requiere múltiples *handshakes* TCP y TLS (para HTTPS), aumentando la latencia inicial de la conexión.

HTTP/2: El *handshake* inicial de TCP sigue siendo necesario, pero se beneficia de la multiplexación para reducir el número de conexiones establecidas.

HTTP/3: Es mucho más rápido ya que QUIC es un protocolo diseñado para minimizar el handshake, reduciendo el número de rondas necesarias para establecer una conexión segura.

Nota: SSL está obsoleto, a favor de TLS

El ***handshake*** se realiza en el contexto de HTTPS usando TLS/SSL, donde ambas partes negocian los parámetros de la comunicación, autentican sus identidades y acuerdan un cifrado común para intercambiar datos de forma privada y segura.

Hypertext Transport Protocol (HTTP)

Comparativa HTTP/1.1 vs. HTTP/2 vs. HTTP/3

HTTP/1.1: La latencia es alta debido a las conexiones por solicitud y la falta de optimizaciones.

HTTP/2: Reduce la latencia gracias a la multiplexación y la compresión de cabeceras, pero aún sufre de los bloqueos de cabecera TCP.

HTTP/3: Ofrece latencia significativamente menor en comparación con sus predecesores, especialmente en conexiones lentas o con pérdida de paquetes, al usar QUIC.

La **latencia** es el tiempo de retardo que transcurre desde que el navegador de un usuario envía una solicitud hasta que recibe una respuesta del servidor.

Directamente relacionada con la experiencia de usuario, aunque afecta también a *aplicaciones en tiempo real* y *rendimiento SEO*.

Hypertext Transport Protocol (HTTP)

- HTTP permite diferentes tipos de peticiones (**métodos**)

  https://datatracker.ietf.org/doc/html/rfc7231#section-4

Hypertext Transport Protocol (HTTP)

- Peticiones seguras (sólo lectura) – no alteran el estado del servidor:
  - GET – Devuelve el recurso identificado por el *Request-URI*
  - OPTIONS – Retorna las opciones de comunicación disponibles por el servidor (*server capabilities*)
  - HEAD – Inspecciona la cabecera del recurso (no devuelve el *BODY*)
- Peticiones idempotentes – una solicitud idéntica puede realizarse múltiples veces, devolviendo siempre el mismo resultado:
  - PUT – Solicita al servidor guardar el cuerpo de la solicitud en la ubicación dada por la URL
  - DELETE – Solicita al servidor que elimine el recurso en la URL dada
  - Métodos seguros (GET, OPTIONS, HEAD)

Hypertext Transport Protocol (HTTP)

- Peticiones no idempotentes – realizar múltiples peticiones idénticas podría causar efectos adicionales (p.ej. enviar varias veces una orden):
  - ❑ POST – Envío de información al servidor (p.ej. información de un formulario)

```
1  POST /cgi-bin/process.cgi HTTP/1.1
2  User-Agent: Mozilla/4.0 (compatible; MSIE5.01;
3  Windows)
4  Host: www.ceia3.es
5  Content-Type: text/xml; charset=utf-8
6  Content-Length: 88
7  Accept-Language: es-es
8  Accept-Encoding: gzip, deflate
   Connection: Keep-Alive
```

Hypertext Transport Protocol (HTTP)

- Peticiones no idempotentes – (cont.)
  - PATCH – Solicita un conjunto de cambios (parciales) descritos en la entidad identificada por la *Request-URI*

    Muy utilizado en el desarrollo de REST/APIs – no es necesario enviar la entidad completa, por lo que permite la actualización de un único campo, con el consecuente ahorro de ancho de banda

    Definido en RFC 5789 "*PATCH Method for HTTP*": https://tools.ietf.org/html/rfc5789

Hypertext Transport Protocol (HTTP)

Method	Safe	Idempotent	Cacheable
`GET`	Yes	Yes	Yes
`HEAD`	Yes	Yes	Yes
`OPTIONS`	Yes	Yes	No
`TRACE`	Yes	Yes	No
`PUT`	No	Yes	No
`DELETE`	No	Yes	No
`POST`	No	No	Conditional*
`PATCH`	No	No	Conditional*
`CONNECT`	No	No	No

Fuente: developer.mozilla.org

Hypertext Transport Protocol (HTTP)

- Muy importante para el programador web conocer las diferencias entre GET y POST:

GET

- La cadena de consulta se envía en la URL del método (visible)
- Es cacheable
- Puede guardarse como marcador por el cliente
- Tiene restricciones de longitud
- Solicitud de solo lectura
- **NUNCA** enviar datos sensibles (p.ej. *passwords*)

POST

- Datos enviados para creación/ modificación de un recurso
- No puede guardarse como marcador
- No tiene restricciones en la longitud
- Permite adjuntos de distintos tipos MIME (*Multipurpose Internet Mail Extension*)

Hypertext Transport Protocol (HTTP)

- Otras peticiones:
  - CONNECT – Utilizado por cliente para establecer una conexión sin cierre con el servidor
  - TRACE – Realiza una prueba de eco de retorno (muy utilizado para depuración, medición de latencias y desarrollo)

Hypertext Transport Protocol (HTTP)__ Cabeceras

- Las **cabeceras** son fragmentos de información que se transmiten junto con las peticiones (*request*) y respuestas (*response*).
- Aunque se presta atención a los métodos (GET, POST, etc.), las cabeceras son igual o más importantes, ya que definen **cómo debe comportarse la comunicación entre cliente y servidor**.
- Cabeceras modernas: `Authorization`, `Cache-Control`, `Accept`, `Origin`
  - `Authorization`: permite al cliente enviar credenciales para acceder a un recurso protegido. Se utiliza en la autenticación con tokens de las APIs (p.ej., OAuth2) Sin esta cabecera, muchas APIs devuelven el código 401 (Unauthorized) directamente.
  - `Cache-Control`: indica cómo puede un recurso ser almacenado en cachés intermedias (navegador, CDN, proxy). Controla la duración, validez y uso de cachés compartidas.
  - `Accept`: permite al cliente indicar qué formatos espera del servidor (tipos de medios). En API REST es habitual devolver JSON.
  - `Origin`: identifica el origen (dominio) desde el que se realiza la petición (solo dominios autorizados acceden a los recursos).

Hypertext Transport Protocol (HTTP)__ Códigos de estado

- Los códigos de estado son identificadores y metainformación asociados a la respuesta (*Response*) HTTP recibida por el servidor:
  - Definidos en RFC 9110 "*HTTP Semantics*" (Sección 15): https://datatracker.ietf.org/doc/html/rfc9110
  - Tipos de códigos:
    - 1*xx* – Códigos informativos
    - 2*xx* – Códigos de éxito
    - 3*xx* – Códigos de redirección
    - 4*xx* – Códigos de error en cliente
    - 5*xx* – Códigos de error en servidor

```
100 – Continue
200 – OK
203 – Non-Authoritative Information
301 – Moved Permanently
404 – Not Found
408 – Request Timeout
500 – Internal Server Error
503 – Service Unavailable
```

Hypertext Transport Protocol (HTTP)__ Tipos de medios (antes MIME)

- Forma estándar de indicar la naturaleza y formato de un recurso
- IANA es el organismo responsable de realizar el control y seguimiento de los tipos estándar
  - RFC 6838 "*Media Type Specifications and Registration Procedures*": https://tools.ietf.org/html/rfc6838
  - RFC 2045 "*Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*": https://tools.ietf.org/html/rfc2045
  - RFC 2046 "*Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*": https://tools.ietf.org/html/rfc2046

Hypertext Transport Protocol (HTTP)__ Tipos de medios (antes MIME)

- Los tipos de medios se dividen en categorías:
  - ❑ *application*, *audio*, *font*, *example*, *image*, *message*, *model*, *multipart*, *text*, *video*
  - ❑ El listado completo de tipos en IANA: https://www.iana.org/assignments/media-types/media-types.xhtml
- Ejemplos:

MIME type	extensión
`text/html`	.html
`text/plain`	.txt
`image/gif`	.gif
`image/jpeg`	.jpg
`video/quicktime`	.mov
`application/octet-stream`	.exe

Recursos

- IETF RFC 791 – Internet Protocol (IPv4)
  https://www.rfc-editor.org/rfc/rfc791
- IETF RFC 8200 – Internet Protocol, Version 6 (IPv6)
  https://www.rfc-editor.org/rfc/rfc8200
- IETF RFC 768 – User Datagram Protocol (UDP)
  https://www.rfc-editor.org/rfc/rfc768
- IETF RFC 9293 – Transmission Control Protocol (TCP) (actualización moderna de TCP)
  https://www.rfc-editor.org/rfc/rfc9293
- IETF RFC 1034 y 1035 – Domain Name System (DNS)
  https://www.rfc-editor.org/rfc/rfc1034
  https://www.rfc-editor.org/rfc/rfc1035
- IETF RFC 4033–4035 – DNS Security Extensions (DNSSEC)
  https://www.rfc-editor.org/rfc/rfc4033

Recursos

- IETF RFC 9110 – HTTP Semantics (reemplaza al viejo RFC 2616)
  https://www.rfc-editor.org/rfc/rfc9110
- IETF RFC 9112 – HTTP/1.1
  https://www.rfc-editor.org/rfc/rfc9112
- IETF RFC 9113 – HTTP/2
  https://www.rfc-editor.org/rfc/rfc9113
- IETF RFC 9114 – HTTP/3
  https://www.rfc-editor.org/rfc/rfc9114
- IETF RFC 9000 – QUIC: A UDP-Based Multiplexed and Secure Transport
  https://www.rfc-editor.org/rfc/rfc9000

Recursos

- MDN Web Docs – HTTP Headers (documentación práctica)
  https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers
- MDN Web Docs – HTTP Caching (Cache-Control, ETag, etc.)
  https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching
- IETF RFC 8484 – DNS Queries over HTTPS (DoH)
  https://www.rfc-editor.org/rfc/rfc8484
- IETF RFC 7858 – DNS over TLS (DoT)
  https://www.rfc-editor.org/rfc/rfc7858
- Cloudflare 1.1.1.1 DNS over HTTPS/TLS
  https://developers.cloudflare.com/1.1.1.1/dns-over-https/
- Netflix Tech Blog – Open Connect (CDN propia de Netflix)
  https://netflixtechblog.com/open-connect-overview-414e801bfb87

Hazlo tú mismo – 1

La **caché web** es un componente fundamental para el rendimiento de las aplicaciones modernas, pero también puede generar problemas si no se gestiona bien.

Tu tarea es investigar en dos tipos de fuentes:

- Fuentes no oficiales: busca algún artículo en la plataforma medium.com u otro blog técnico sobre caché en programación web.
- Fuentes oficiales o contrastadas:
  - https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching
  - RFC de la IETF, como RFC 7234 – HTTP Caching: https://www.rfc-editor.org/rfc/rfc7234

Con esa información, responde en el foro de Moodle:

- Definición breve de caché en la Web (en tus propias palabras).
- Un uso correcto de la caché en programación web (ejemplo práctico, como Cache-Control: max-age=3600). ¿Dónde y cómo se indica? ¿Para qué sirve hacerlo?
- Un posible problema o mal uso de la caché, según lo que hayas leído.
- Incluye las referencias que consultaste.

Programación Web

Introducción a la programación Web__ **Curso 2025/26**