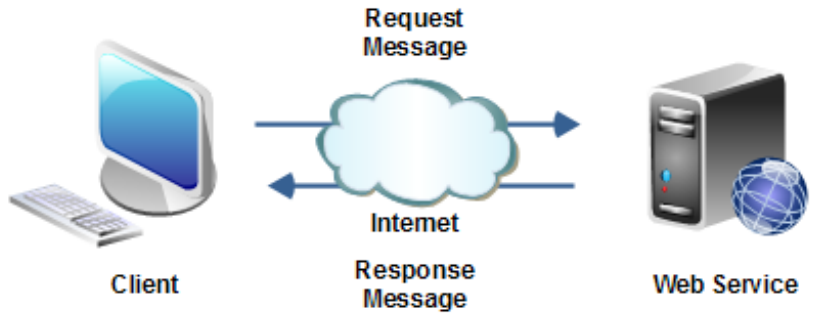
Protocolul HTTP

Introducere în Protocolul HTTP
Istoricul și evoluția HTTP
Componente de bază ale HTTP
Metode de cereri HTTP
Coduri de stare HTTP
Antete HTTP și funcțiile lor
Securitate și performanță HTTP
Viitorul protocolului HTTP

Introducere în protocolul HTTP

Definiția protocolului de transfer hipertext (HTTP)

HTTP, sau Protocolul de transfer hipertext, este protocolul fundamental de comunicare utilizat de World Wide Web. Acesta permite clienților - cum ar fi browserele web - să solicite resurse de la serverele web și să preia pagini web, imagini, videoclipuri și alte date. HTTP facilitează transferul documentelor hipertext, permițând utilizatorilor să interacționeze perfect cu conținutul web pe diferite platforme și dispozitive.



Importanța în comunicațiile Web

HTTP este esențial deoarece definește modul în care mesajele sunt formatate și transmise, asigurând o comunicare fiabilă între clienți și servere. Simplitatea, extensibilitatea și adoptarea pe scară largă îl fac protocolul principal pentru navigarea web, permițând experiențele dinamice și bogate în multimedia pe care utilizatorii le așteaptă astăzi

Rolul în architectura client-server

În modelul client-server, HTTP acționează ca protocolul care guvernează comunicarea, clienții trimițând cereri către servere, care apoi răspund cu datele solicitate. Acest schimb structurat permite site-urilor web și aplicațiilor web să funcționeze eficient, susținând întregul ecosistem de servicii online și partajare de informații.

Istoria și evoluția HTTP

Originea și versiunea inițială (HTTP/0.9)

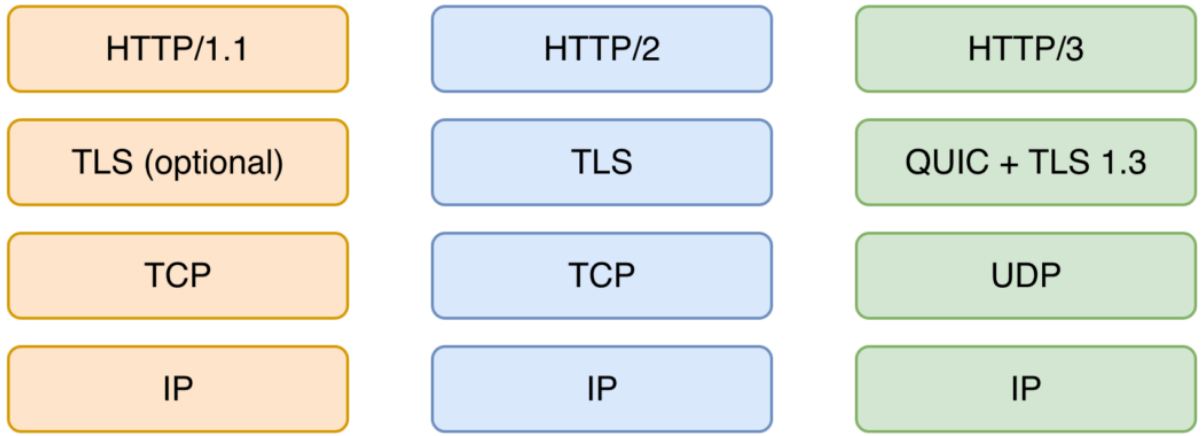
HTTP a fost dezvoltat pentru prima dată la începutul anilor 1990 ca HTTP/0.9, un protocol simplu care suporta doar recuperarea de bază a datelor brute printr-o singură cerere GET. Era inerent limitat, dar a pus bazele unor versiuni mai sofisticate.

Dezvoltarea HTTP/1.0 și HTTP/1.1

HTTP/1.0 a introdus anteturi și coduri de stare pentru un context și un control mai bun, facilitând o comunicare mai fiabilă și mai flexibilă. HTTP/1.1, lansat în 1997, a devenit versiunea cea mai utilizată, adăugând funcții precum conexiuni persistente, codificare de transfer în blocuri și mecanisme de cache, îmbunătățind semnificativ performanța și eficiența.

Introducerea HTTP/2 și HTTP/3

HTTP/2, publicat în 2015, a adus îmbunătățiri de performanță prin multiplexare, compresie a antetelor și tehnologie server push, reducând latența și timpii de încărcare. Mai recent, HTTP/3, bazat pe protocolul de transport QUIC, își propune să îmbunătățească și mai mult viteza, securitatea și rezistența conexiunilor, remodelând viitorul comunicării web.



TLS (Transport Layer Security) este un protocol criptografic conceput pentru a asigura securitatea comunicațiilor printr-o rețea de calculatoare, cum ar fi Internetul.

QUIC (Conexiuni rapide la Internet), este un protocol de nivel de transport care rulează peste UDP pentru a îmbunătăți performanța internetului prin combinarea vitezei cu securitatea încorporată. Acesta integrează funcții care erau gestionate anterior de protocoale separate, cum ar fi handshake-ul și criptarea (TLS), pentru a crea conexiuni mai rapide, mai fiabile și mai sigure pentru aplicații precum HTTP/3.

UDP (User Datagram Protocol), este un protocol neorientat pe conexiune, utilizat pentru comunicarea pe internet.Este mai rapid decat TCP dar mai puțin fiabil.

Componentele de bază ale HTTP

Rolurile clientului și serverului

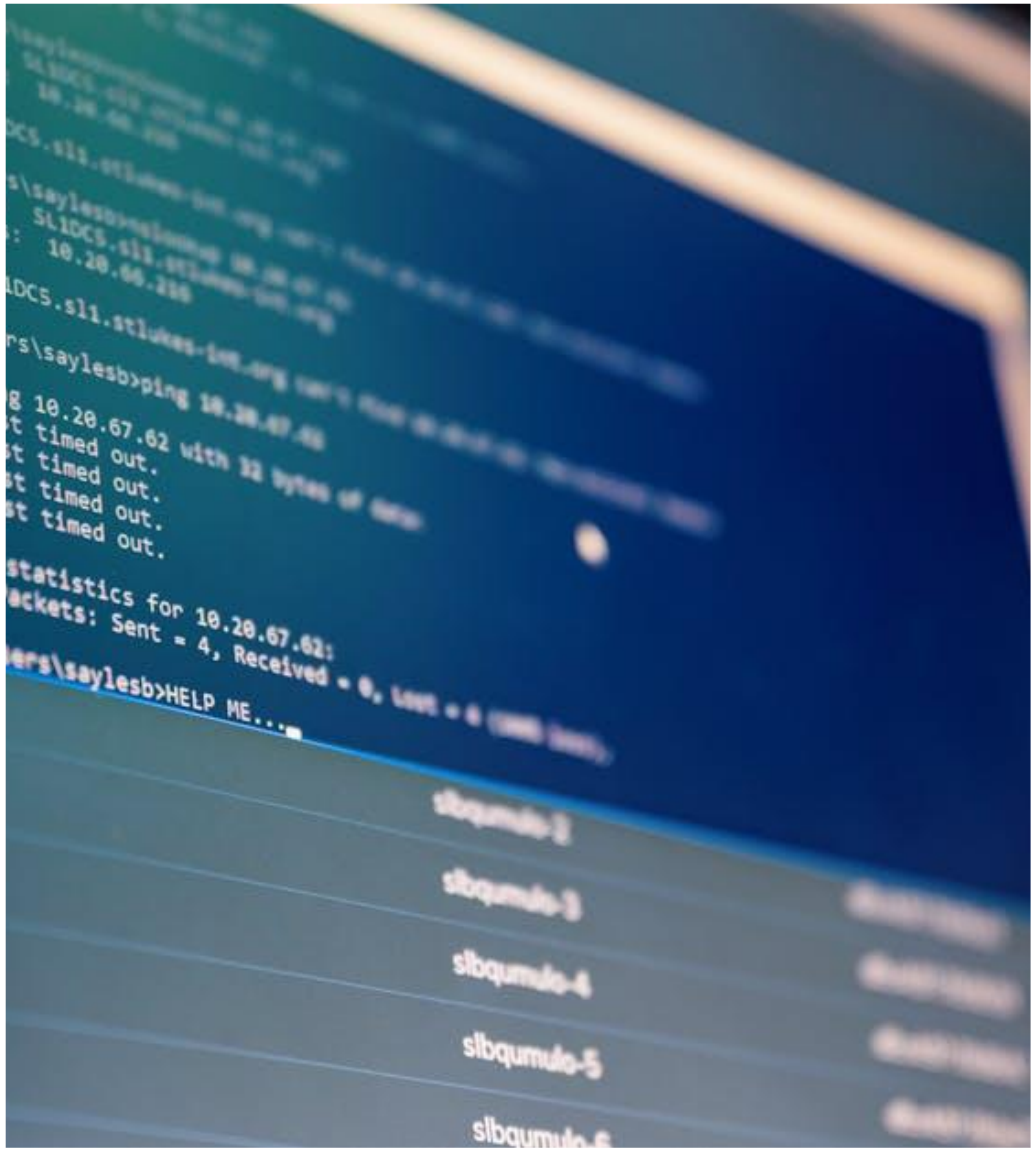
În comunicarea HTTP, clientul - de obicei un browser web - inițiază cereri pentru resurse web, în timp ce serverul răspunde prin livrarea conținutului solicitat. Această interacțiune formează coloana vertebrală a navigării și a schimbului de date online.

Mesaje de cerere și răspuns

HTTP funcționează printr-un ciclu de solicitare-răspuns în care clienții trimit solicitări care conțin metode, anteturi și conținut opțional, iar serverele răspund cu coduri de stare, anteturi și datele solicitate. Această mesagerie structurată permite o comunicare eficientă și livrarea resurselor.

Anteturi și corpul HTTP

Anteturile oferă metadate despre cerere sau răspuns, cum ar fi tipul de conținut, directivele de caching și informațiile despre agentul utilizator. Corpul conține datele efective transferate, cum ar fi conținut HTML, imagini sau formulare trimise, permițând o comunicare flexibilă și detaliată.



Metodele de cereri HTTP

GET: Obținerea datelor
Metoda GET este utilizată pentru a solicita date de la o resursă specificată de pe server. Este cea mai comună metodă, concepută pentru obținerea sigură și idempotentă, fără a afecta starea serverului. Totuși, mici cantitatăți de date dpot fi trimise la server ca parametri pentru personalizarea cererii. Când navigați pe o pagină web, browserul dvs. utilizează de obicei solicitări GET.

POST: Trimite date la server

POST este utilizat pentru a trimite date către server, cum ar fi trimiterea de formulare sau încărcarea de fișiere. Spre deosebire de GET, cererile POST pot include cantități mari de date în corpul cererii și adesea duc la o modificare a stării serverului, cum ar fi crearea sau actualizarea resurselor.

Alte metode: PUT, DELETE, HEAD, OPTIONS, PATCH

Metodele HTTP suplimentare servesc diverselor scopuri: PUT creează sau înlocuiește resurse; DELETE elimină resursele; HEAD preia doar anteturile; OPTIONS solicită informații despre capacitățile serverului; PATCH aplică modificări parțiale. Fiecare metodă ajută la facilitarea arhitecturilor RESTful și a interacțiunilor complexe.

REST (Representational State Transfer -Transfer de Stare Reprezentațional). Este un set de principii de proiectare pentru a face comunicarea în rețea mai scalabilă și flexibilă.

Exemple de cereri și răspunsuri HTTP

Sintaxa cererii

```
<METHOD> <URI> "HTTP/1.1" <crlf>
{<Header>: <Value> <crlf>}*
<crlf>
```

Exemplu

```
GET / site/index.php HTTP/1.1
Host: ie.usv.ro
Accept: text/plain
Accept: text/htm
User-Agent: curl/7.47.0
Accept: */*
```

HTTP/1.0 (1.1) suferă de câteva neajunsuri:
- Lipsa multiplexării
- Suprasarcină de conexiuni
- Ineficiență în compresia antetelor
- Ambiguitate în delimitarea cererilor
- Lipsa suportului nativ pentru criptare
- Performanță slabă pe conexiuni mobile sau instabile.

Soluția este adoptarea versiunilor mai noi ale HTTP, 2 sau 3. Până atunci se folosește HTTP/1.1 din cauza compatibilității cu infrastructura existentă.

Sintaxa răspunsului

```
< HTTP/1.1 301 Moved Permanently
< Date: Mon, 10 Nov 2025 09:46:22 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Location: https://ie.usv.ro/site/index.php
< Content-Length: 316
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://ie.usv.ro/
site/index.php">here</a>.</p>
<hr>
<address>Apache/2.4.18 (Ubuntu) Server at ie.usv.ro
Port 443</address>
</body></html>
* Connection #0 to host ie.usv.ro left intact //
```

serverul are setată o întârziere la închiderea conexiunii

Coduri de stare HTTP

Răspunsuri informative (1xx)

Aceste coduri indică faptul că a fost primită o solicitare și că procesul continuă. Exemplele includ 100 *Continue*, care semnalează că partea inițială a unei solicitări a fost primită și clientul ar trebui să continue.

Răspunsuri de succes (2xx)

Coduri precum 200 OK indică procesarea cu succes a unei cereri. Acestea confirmă că resursa solicitată a fost livrată sau că acțiunea a fost efectuată cu succes.

Mesaje de redirecționare (3xx)

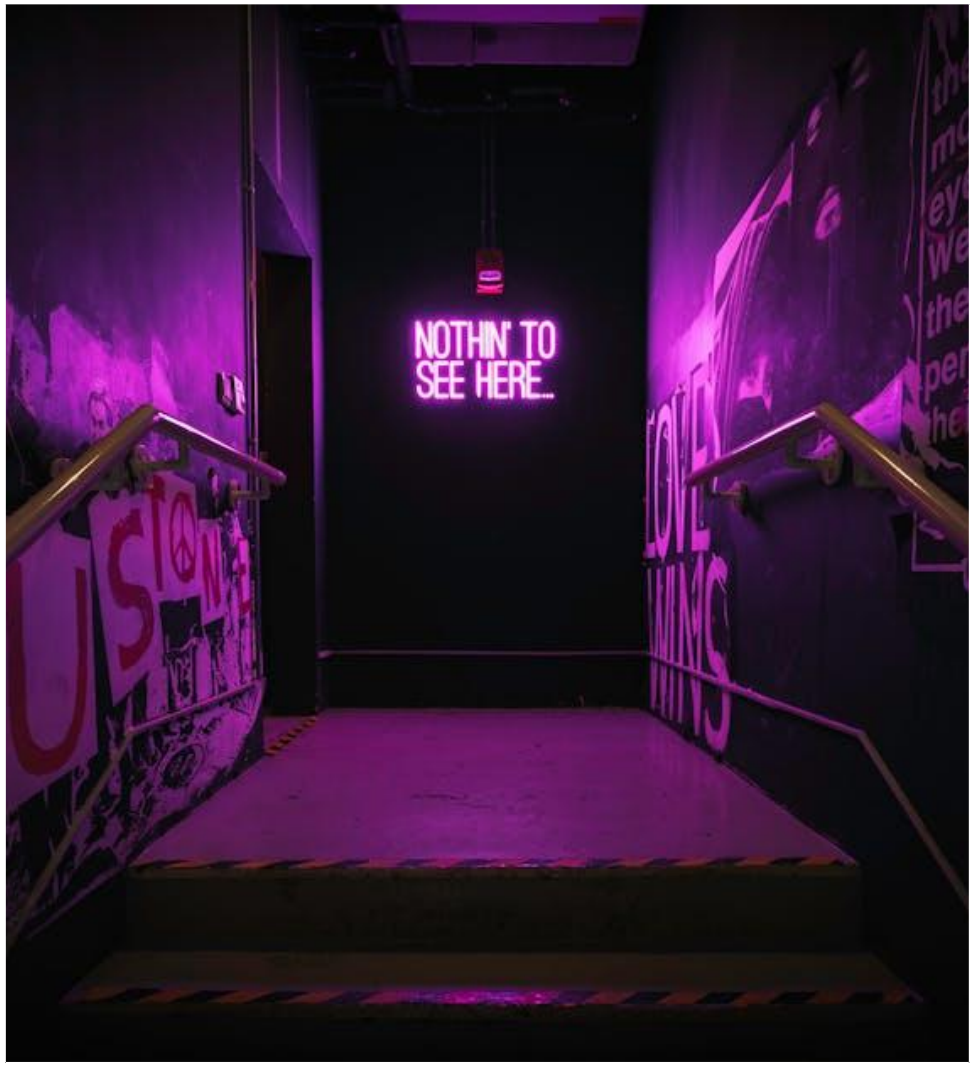
Aceste coduri, cum ar fi 301 *Mutat permanent* și 302 *Găsit, instruiesc clientul să întreprindă acțiuni suplimentare*, de obicei redirecționând către o adresă URL diferită pentru a finaliza solicitarea.

Răspunsuri la erori ale clientului (4xx)

Atunci când un client face o solicitare incorectă sau neautorizată, sunt returnate erori precum 404 Not Found sau 401 Unauthorized, semnalând probleme care necesită corectare sau autentificare.

Răspunsuri la erorile serverului (5xx)

Acestea indică probleme la nivelul serverului, cu coduri comune precum 500 *Eroare internă server* sau 503 *Serviciu indisponibil*, ceea ce înseamnă că serverul nu a reușit să îndeplinească o solicitare validă.



Antete HTTP și funcțiile lor

Antete comune (Content-Type, Accept, User-Agent)

Anteturi precum Content-Type specifică tipul de media al cererii sau al corpului răspunsului, în timp ce Accept indică ce tipuri de media poate gestiona clientul. User-Agent oferă informații despre software-ul client care face cererea.

Antete personalizate

Anteturile personalizate permit includerea extensiilor și a metadatelor suplimentare în solicitări sau răspunsuri, permițând dezvoltatorilor să implementeze funcționalități specializate sau să transmită informații specifice aplicației.

Rol în gestionarea cererilor/răspunsurilor

Anteturile joacă un rol crucial în controlul memorării în cache, negocierii conținutului, autentificării și gestionării sesiunilor, asigurând o comunicare eficientă și sigură între clienți și servere.

Fazele protocolului HTTP

1. Stabilirea conexiunii
2. Cererea
3. Procesarea cererii
4. Răspunsul
5. Întreruperea conexiunii

În procesul de stabilire a conexiunii, înainte să înceapă comunicarea HTTP, au loc trei faze numite „trei strângeri de mână”.
1. SYN (Synchronize): Clientul trimite un pachet către server pentru a iniția o conexiune.
2. SYN-ACK (Synchronize-Acknowledge): Serverul răspunde cu un pachet care confirmă solicitarea clientului și trimite, de asemenea, propria solicitare de sincronizare.
3. ACK (Acknowledge): Clientul trimite o confirmare finală înapoi către server, finalizând handshake-ul și stabilind o conexiune fiabilă.

Strângerea de mână TLS

Acesta este un proces separat, mai complex, care are loc după finalizarea strângerii de mână TCP pentru conexiunile HTTPS. Acesta stabilește o conexiune criptată și securizată între client și server.
1. Clientul și serverul își verifică reciproc identitatea, în principal prin certificate.
2. Aceștia convin asupra algoritmilor și parametrilor de criptare care vor fi utilizați pentru sesiune.
3. Odată ce această strângere de mână este efectuată, traficul HTTP criptat poate circula în siguranță între client și server.

Securitate și performanță

HTTPS (HTTP Secure) este versiunea securizată a HTTP. Acesta adaugă criptare și autentificare comunicării dintre un client și server folosind TLS (Transport Layer Security).

Când un browser sau un instrument face o solicitare HTTPS, începe cu o strângere de mână TLS. Această strângere de mână negociază parametrii criptografici și verifică certificatul serverului. Numai după finalizarea acesteia, solicitarea HTTP propriu-zisă este trimisă prin canalul securizat.

HTTPS are trei proprietăți:
Confidențialitate: Datele sunt criptate, ceea ce împiedică alte persoane să citească conținutul solicitării sau răspunsului în tranzit.
Integritate: TLS verifică dacă datele nu au fost modificate sau manipulate.
Autentificare: Certificatul serverului confirmă identitatea acestuia clientului. Dacă certificatul este invalid sau a expirat, clientul blochează solicitarea.

Performanța poate fi crescută prin eliminarea neajunsurilor existente la versiunea curenta 1.0/1.1

Avantajele HTTP/2

Multiplexare completă - Permite trimiterea mai multor cereri simultan pe aceeași conexiune TCP, eliminând blocajul de tip head-of-line și reducând semnificativ latența.

Compresia antetelor (header compression) - Folosește algoritmul HPACK pentru a comprima antetele HTTP, ceea ce reduce dimensiunea pachetelor și accelerează transferul.

Server Push - Serverul poate trimite proactiv resurse către client (ex. fișiere CSS sau JavaScript), fără ca browserul să le ceară explicit, îmbunătățind viteza de încărcare.