HTTP

Scritto da Febogamedev

HTTP, acronimo di Hypertext Transfer Protocol, è un protocollo del livello applicativo utilizzato per lo scambio di informazioni nel Web (Fielding et al., 2022c).
Nonostante il nome faccia riferimento all'ipertesto, HTTP non trasferisce soltanto documenti HTML. Può essere utilizzato per richiedere e trasmettere diverse risorse web (immagini, fogli CSS, font, ecc.) (Fielding et al., 2022c).

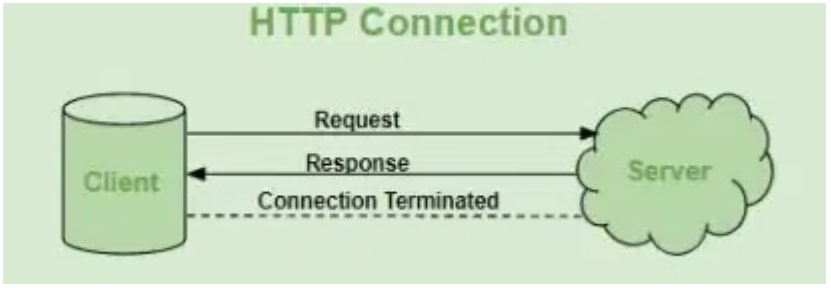
HTTP è una famiglia di protocolli basati su un modello di comunicazione a richiesta e risposta: un client invia una richiesta e un server restituisce una risposta (Fielding et al., 2022c).
Le diverse versioni mantengono gli stessi concetti fondamentali, ma rappresentano e trasportano i messaggi in modi differenti (Fielding et al., 2022a; Thomson & Benfield, 2022; Bishop, 2022).

Oggi HTTP viene normalmente utilizzato tramite *HTTPS*.
HTTPS indica una comunicazione *HTTP* protetta tramite *TLS*, un protocollo crittografico che protegge i dati trasmessi (Fielding et al., 2022c; Rescorla, 2026).
Il browser verifica l'identità del servizio remoto e i dati scambiati vengono protetti da intercettazioni e modifiche durante il trasferimento (Rescorla, 2026).

L'*HTTP* è il principale protocollo per la trasmissione d'informazioni sul web in un'architettura client-server: il client esegue una richiesta e il server restituisce la risposta (Fielding et al., 2022c).

Un servizio raggiunto tramite *HTTP* utilizza normalmente la porta predefinita 80, mentre un servizio *HTTPS* utilizza normalmente la porta 443 (Internet Assigned Numbers Authority, 2026).

Vi sono due tipi di messaggi HTTP (Fielding et al., 2022c):
- messaggi di richiesta: il client fa la richiesta al server.
- messaggi di risposta: il server risponde al client.



Le connessioni possono essere mantenute aperte e riutilizzate per più richieste (connessioni persistenti), oppure chiuse quando non servono più (Fielding et al., 2022a).

È importante notare che se una pagina web è composta da immagini, video o altre risorse esterne, Il browser interpretando tale pagina fa una richiesta *HTTP* per ogni risorsa (WHATWG, 2026).

Di seguito vengono esaminate la richiesta e la risposta HTTP. Per semplicità, faremo riferimento alla rappresentazione utilizzata in *HTTP/1.1 (Fielding et al., 2022a)*.

Richiesta HTTP

Una richiesta HTTP è un messaggio attraverso il quale un client chiede a un server di eseguire un'operazione relativa a una risorsa (Fielding et al., 2022c).

Il client può chiedere, per esempio, di:
- Ottenere una risorsa web.
- Inviare i dati di un modulo.
- Creare o modificare una risorsa.
- Eliminare una risorsa.
- Ottenere informazioni senza trasferirne il contenuto completo.

Una richiesta può essere rappresentata testualmente attraverso (Fielding et al., 2022a):
- Riga di richiesta.
- Header.
- Riga vuota (la riga vuota finale separa gli header dall'eventuale contenuto della richiesta).
- Eventuale body.

Riga di richiesta

La prima riga prende il nome di request line, o riga di richiesta (Fielding et al., 2022a).
È formata da: METODO OBIETTIVO VERSIONE (Fielding et al., 2022a).

Esempio
GET /product?id=123 HTTP/1.1

Metodo
Il metodo HTTP indica quale operazione il client chiede di applicare alla risorsa di destinazione (Fielding et al., 2022c).
I nomi dei metodi standard vengono scritti in maiuscolo (Fielding et al., 2022c).

GET
GET richiede al server una rappresentazione corrente della risorsa.

GET è considerato un metodo sicuro nella semantica HTTP: il client non sta chiedendo al server di modificare lo stato della risorsa. Questo non impedisce attività secondarie come la

registrazione della richiesta nei log, ma l'operazione richiesta deve essere essenzialmente di lettura (Fielding et al., 2022c).

HEAD
HEAD è simile a GET, ma il server non restituisce il contenuto della rappresentazione.

Può essere utilizzato per ottenere metadati come:
- media type.
- dimensione.
- data dell'ultima modifica.
- informazioni sulla cache.

Il server dovrebbe restituire, per quanto possibile,gli stessi header che avrebbe restituito in risposta a una richiesta GET (Fielding et al., 2022c).

POST
POST chiede alla risorsa di destinazione di elaborare i dati contenuti nella richiesta secondo le proprie regole.

Può essere utilizzato, per esempio, per:
- Inviare un modulo.
- Creare un nuovo commento.
- Effettuare un ordine.
- Caricare un file.
- Avviare un'elaborazione.

Il risultato dipende dalla funzione definita dall'applicazione per quello specifico *URL (Fielding et al., 2022c)*.

PUT
PUT chiede di creare o sostituire lo stato della risorsa identificata dall'URL.
In questo caso il client conosce già l'indirizzo esatto della risorsa che vuole creare o sostituire.

PUT non significa necessariamente "caricare un file": il contenuto può rappresentare qualsiasi stato riconosciuto dall'applicazione (Fielding et al., 2022c).

PATCH
PATCH viene utilizzato per applicare modifiche parziali a una risorsa.

Per esempio, anziché inviare nuovamente l'intero profilo di un utente, una richiesta PATCH potrebbe chiedere di modificarne soltanto l'immagine o il nome visualizzato (Dusseault & Snell, 2010).

DELETE
DELETE chiede di rimuovere l'associazione tra l'URL e la risorsa corrente.

Il modo in cui l'applicazione realizza questa operazione dipende dal server. Il dato potrebbe essere eliminato, archiviato o soltanto marcato come non più disponibile (Fielding et al., 2022c).

Obiettivo della richiesta
La seconda parte della riga indica il request target, cioè la parte della richiesta che identifica o individua il destinatario/bersaglio della richiesta; nella forma più comune comprende percorso ed eventuale query (Fielding et al., 2022a).

Per esempio nella richiesta "/product?id=123", possiamo distinguere:
- Percorso: /product.
- Query: id=123.

Da notare che il frammento dell'URL, cioè la parte che comincia con #, non viene inviato al server: è riservato all'elaborazione da parte del client (Berners-Lee et al., 2005).

Versione del protocollo
Nella richiesta la terza parte indica la versione del protocollo, nel nostro caso HTTP/1.1.

Header della richiesta

Gli header, o campi di intestazione, trasmettono informazioni aggiuntive sulla richiesta e sul contesto in cui viene effettuata.
Ogni campo è composto da un nome e da un valore: Nome-Campo: valore (Fielding et al., 2022a).

Host
Questo campo identifica l'host e l'eventuale porta della risorsa richiesta.
È fondamentale quando la stessa infrastruttura gestisce più nomi di dominio.
Una richiesta priva di Host deve essere considerata non valida dal server (Fielding et al., 2022a).

User-Agent
Identifica il programma client che ha inviato la richiesta.

Può contenere il nome e la versione del prodotto, ma non dovrebbe essere considerato una descrizione completa e sempre affidabile del dispositivo dell'utente (Fielding et al., 2022c).

Accept
Indica i media type preferiti dal client per la risposta.
È possibile esprimere anche preferenze relative.
Per esempio: Accept: text/html, application/json;q=0.8
Il valore q=0.8 indica una preferenza inferiore rispetto al tipo text/html, al quale, non essendo specificato q, viene applicato il valore predefinito 1 (Fielding et al., 2022c).

Content-Type

Quando la richiesta possiede un contenuto, Content-Type ne descrive il formato (Fielding et al., 2022c):
Per esempio: Content-Type: application/json.

Body della richiesta

Il body, o corpo, contiene gli eventuali dati inviati dal client al server.
Il significato del contenuto dipende dal metodo e dal media type dichiarato (Fielding et al., 2022c).
Può contenere:

- Campi di un modulo.
- Dati JSON.
- Un'immagine.
- Un documento.
- Un file.
- Istruzioni per modificare una risorsa.

Il body viene utilizzato frequentemente con POST, PUT e PATCH, ma la sua presenza non dipende soltanto dal nome del metodo: è la semantica del metodo a stabilire come deve essere interpretato (Fielding et al., 2022c).

Esempio di richiesta

```
GET /product?id=123 HTTP/1.1
Host: www.example.com
User-Agent: ExampleBrowser/1.0
Accept: text/html
```

Analisi:

- Metodo: GET, richiede una rappresentazione della risorsa.
- Percorso: /product.
- Query: id=123.
- Versione: HTTP/1.1.
- Host: www.example.com.
- User-Agent: identifica il programma client.
- Accept: indica che il client preferisce ricevere HTML.

Risposta HTTP

Una risposta *HTTP* è il messaggio con cui un server comunica al client l'esito di una richiesta (Fielding et al., 2022c).
La risposta può contenere:

- Il risultato dell'operazione.
- Una rappresentazione della risorsa richiesta.
- Informazioni utili al client.
- Un reindirizzamento.
- La descrizione di un errore.

Per esempio, il server può restituire:

- Una risorsa web.
- La conferma dell'invio di un modulo.
- Un'indicazione che la risorsa si trova a un altro URL.
- Un messaggio che segnala un problema.

Una risposta è rappresentata testualmente in questo modo (Fielding et al., 2022a):

- Riga di stato
- Header
- Riga vuota
- Eventuale body

Riga di stato

La prima riga della risposta è chiamata status line, o riga di stato.
È composta da: VERSIONE CODICE FRASE (Fielding et al., 2022a).

Esempio
HTTP/1.1 404 Not Found

Codici di stato
Il codice di stato è un numero di tre cifre che descrive il risultato della richiesta.

La prima cifra indica la categoria generale:

Categoria	Significato generale
1xx	Informazione provvisoria
2xx	Richiesta elaborata con successo
3xx	Reindirizzamento o utilizzo di una risposta precedente
4xx	Problema attribuito alla richiesta del client
5xx	Problema incontrato dal server

Il codice non descrive tutti i dettagli dell'applicazione, ma comunica un significato generale valido per i diversi sistemi HTTP (Fielding et al., 2022c).

Codici comuni:

- 200 OK: Indica che la richiesta è stata elaborata con successo. Il significato preciso del contenuto dipende dal metodo utilizzato. In risposta a una GET, normalmente il body contiene una rappresentazione della risorsa richiesta (Fielding et al., 2022c).
- 201 Created: Indica che la richiesta è stata elaborata con successo e ha portato alla creazione di una nuova risorsa. La risposta può utilizzare l'header Location per indicare l'URL della risorsa creata (Fielding et al., 2022c).
- 204 No Content: Indica che la richiesta è stata elaborata con successo, ma che la risposta non contiene un body. Può essere utilizzato, per esempio, dopo il salvataggio di una modifica quando non è necessario inviare un nuovo contenuto al client (Fielding et al., 2022c).
- 301 Moved Permanently: Indica che alla risorsa è stato assegnato un nuovo URL permanente. La nuova destinazione viene normalmente indicata attraverso l'header Location. I collegamenti futuri dovrebbero utilizzare il nuovo indirizzo (Fielding et al., 2022c).
- 302 Found: Indica che la risorsa si trova temporaneamente a un URL differente. Il client dovrebbe continuare a utilizzare l'URL originario nelle richieste future, perché il reindirizzamento potrebbe cambiare. I browser seguono normalmente il reindirizzamento in modo automatico (Fielding et al., 2022c).
- 304 Not Modified: Indica che il client può riutilizzare una risposta già conservata nella cache perché la risorsa non è cambiata. La risposta non contiene una nuova copia completa della risorsa. Questo riduce la quantità di dati trasferiti (Fielding et al., 2022b; Fielding et al., 2022c).
- 400 Bad Request: Indica che il server non può o non vuole elaborare la richiesta a causa di un problema attribuito al client, per esempio sintassi non valida, struttura del messaggio errata, dati di instradamento ingannevoli o incoerenti (Fielding et al., 2022c).
- 401 Unauthorized: Indica che la richiesta non possiede credenziali di autenticazione valide (Fielding et al., 2022c).
- 403 Forbidden: Indica che il server ha compreso la richiesta, ma rifiuta di eseguirla. L'utente potrebbe essere autenticato ma non possedere le autorizzazioni necessarie (Fielding et al., 2022c).
- 404 Not Found: indica che il server non ha trovato una risorsa disponibile per l'indirizzo richiesto oppure non vuole dichiarare che essa esista. Una possibile causa è un URL errato, ma non è l'unica (Fielding et al., 2022c).
- 429 Too Many Requests: Indica che il client ha inviato troppe richieste in un determinato intervallo di tempo. Può essere utilizzato dai servizi per limitare abusi o sovraccarichi (Nottingham & Fielding, 2012).
- 500 Internal Server Error: Indica che il server ha incontrato una condizione imprevista che gli ha impedito di completare la richiesta. Non specifica quale sia stato il problema concreto: potrebbe dipendere dal codice dell'applicazione, dalla configurazione o da un servizio utilizzato dal server (Fielding et al., 2022c).
- 502 Bad Gateway: Indica che un server che agiva come gateway o proxy ha ricevuto una risposta non valida da un altro server coinvolto nell'elaborazione (Fielding et al., 2022c).
- 503 Service Unavailable: Indica che il servizio non è temporaneamente in grado di gestire la richiesta, per esempio a causa di sovraccarico, manutenzione, indisponibilità temporanea. La risposta può contenere l'header Retry-After, che suggerisce quando riprovare (Fielding et al., 2022c).

- 505 HTTP Version Not Supported: Indica che il server non supporta, o rifiuta di supportare, la versione principale di HTTP utilizzata nella richiesta. È un codice corretto, ma decisamente meno comune nell'esperienza ordinaria degli utenti (Fielding et al., 2022c).

Frase descrittiva
La frase descrittiva, come "OK" o "Not Found", è destinata principalmente alla leggibilità umana, ma non costituisce un canale affidabile per trasmettere informazioni.
Può essere modificata, tradotta, omessa, eliminata durante la conversione tra versioni differenti di *HTTP*.
Il client deve quindi interpretare il codice numerico, non affidarsi alla frase testuale (Fielding et al., 2022a).

Header della risposta

Gli header di risposta trasmettono informazioni aggiuntive sulla risposta, sul contenuto e sulle istruzioni destinate al client.
Ogni campo è formato da un nome e da un valore: Nome-Campo: valore.

Gli header sono separati tra loro da nuove righe. Una riga vuota indica la fine della sezione degli header (Fielding et al., 2022a).

Content-Type
Descrive il media type del contenuto presente nel body.

Può indicare, per esempio:
- Content-Type: image/webp
- Content-Type: application/json
- Content-Type: application/pdf

Il browser utilizza questa informazione per capire come interpretare i dati ricevuti (Fielding et al., 2022c).

Content-Length
Quando presente e applicabile, indica in byte la lunghezza del contenuto.
Non è però obbligatorio in ogni risposta.
Può mancare, per esempio, quando la lunghezza non è nota nel momento in cui vengono inviati gli header oppure quando il protocollo utilizza altri sistemi per delimitare i dati.
In alcune risposte, come 204 No Content, non deve essere inviato (Fielding et al., 2022c).

Location
Indica un altro URL collegato al risultato della risposta.

Viene utilizzato soprattutto:
- Nei reindirizzamenti.
- Per indicare una nuova risorsa creata (Fielding et al., 2022c).

Cache-Control

Comunica regole relative alla conservazione e al riutilizzo della risposta nella cache.

Può, per esempio, indicare per quanto tempo la risposta può essere riutilizzata prima di richiedere una verifica al server (Fielding et al., 2022b).

Content-Encoding
Indica che il contenuto è stato codificato o compresso prima della trasmissione.
Il browser deve applicare la relativa decodifica prima di utilizzare il contenuto (Fielding et al., 2022c).

Set-Cookie
Chiede al browser di memorizzare un cookie secondo le condizioni indicate.
Il cookie potrà essere inviato nelle richieste successive quando le regole relative a dominio, percorso e sicurezza lo permettono (Barth, 2011).

Server
Può indicare informazioni sul software che ha gestito la richiesta.
È un campo facoltativo: il server può non inviarlo oppure fornire informazioni generiche.
Esporre il nome e la versione dettagliata del software non è necessario al funzionamento della risposta (Fielding et al., 2022c).

Body della risposta

Il body, o corpo della risposta, contiene gli eventuali dati inviati dal server al client.

Può contenere:
- codice HTML.
- CSS.
- JavaScript.
- Un'immagine.
- Dati JSON.
- Un file PDF.
- Audio o video.
- Una spiegazione dell'errore.

Il formato viene normalmente descritto dall'header Content-Type (Fielding et al., 2022c).
Non tutte le risposte contengono un body.

Per esempio:
- Una risposta a HEAD non restituisce il contenuto;
- 204 No Content non contiene contenuto;
- 304 Not Modified indica di riutilizzare la copia conservata;
- Alcune risposte informative 1xx non contengono un contenuto finale.

La presenza del body dipende quindi dal metodo della richiesta, dal codice di stato e dalle regole della specifica *HTTP (Fielding et al., 2022c)*.

Esempio di risposta

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Cache-Control: max-age=3600

<!doctype html>
<html lang="it">

<head>
  <meta charset="utf-8">
  <title>Benvenuto</title>
</head>

<body>
  <h1>Ciao, mondo!</h1>
  <p>Benvenuto nella pagina di esempio.</p>
</body>

</html>
```

Analisi:

- Versione: HTTP/1.1.
- Codice di stato: 200.
- Frase descrittiva: OK.
- Content-Type: il body contiene HTML codificato in UTF-8.
- Cache-Control: la risposta può essere riutilizzata secondo la regola indicata.
- Body: contiene il documento HTML.
- Content-Length: non è stato inserito, perché non è necessario per comprendere l'esempio e dovrebbe corrispondere esattamente ai byte trasmessi.

Materiale di supporto

Esercizi

Costruire la richiesta corretta

Un’applicazione web deve permettere a un utente di:

- Visualizzare il prodotto con ID 123.
- Ottenere soltanto i metadati della stessa risorsa.
- Inviare un nuovo commento.
- Modificare soltanto il nome visualizzato del proprio profilo.
- Sostituire completamente una risorsa già identificata da un URL preciso.
- Eliminare una risorsa.

Per ciascuna operazione:

- Scegli il metodo più appropriato tra GET, HEAD, POST, PUT, PATCH e DELETE.
- Motiva la scelta.
- Indica in quali casi è plausibile la presenza di un body nella richiesta.
- Per la visualizzazione del prodotto, costruisci una possibile riga di richiesta usando “/product?id=123”.
- Scomponi la riga ottenuta in metodo, request target e versione.
- Distingui nel request target percorso e query.
- Spiega perché un eventuale frammento come #recensioni non dovrebbe comparire nella richiesta inviata al server.
- Aggiungi almeno tre header appropriati tra Host, User-Agent, Accept e Content-Type.
- Spiega la differenza tra Accept e Content-Type.
- Spiega infine perché GET viene considerato un metodo sicuro nella semantica HTTP.

Diagnosticare le risposte del server

Un browser effettua diverse richieste e riceve queste risposte:

- Caso A: 200 OK
- Caso B: 201 Created con header Location.
- Caso C: 301 Moved Permanently.
- Caso D: 304 Not Modified.
- Caso E: 401 Unauthorized.
- Caso F: 403 Forbidden.
- Caso G: 404 Not Found.
- Caso H: 429 Too Many Requests.
- Caso I: 500 Internal Server Error.
- Caso L: 502 Bad Gateway.
- Caso M: 503 Service Unavailable.

Per ciascun caso:

- Indica a quale categoria appartiene tra 2xx, 3xx, 4xx e 5xx.
- Spiega il significato generale del codice.
- Indica quali casi rappresentano principalmente un problema attribuito al client.

- Indica quali casi rappresentano un problema lato server o infrastruttura.
- Spiega la differenza tra 401 Unauthorized e 403 Forbidden.
- Spiega la differenza tra 301 Moved Permanently e 302 Found.
- Spiega perché 304 Not Modified può ridurre la quantità di dati trasferiti.
- Spiega in quale situazione 201 Created può utilizzare l'header Location.
- Spiega la differenza concettuale tra 500, 502 e 503.
- Spiega perché un programma dovrebbe basarsi principalmente sul codice numerico e non sulla frase descrittiva come OK o Not Found.

Analizzare una comunicazione HTTP completa

Considera questa richiesta:

```
GET /catalogo?categoria=libri HTTP/1.1
Host: shop.example.com
User-Agent: ExampleBrowser/1.0
Accept: text/html, application/json;q=0.8
```

Il server risponde:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Cache-Control: max-age=3600
Set-Cookie: sessione=abc123

<!doctype html>
<html lang="it">
...
</html>
```

Analizza l'intero scambio.

- Individua nella richiesta metodo, percorso, query, versione e host.
- Spiega cosa comunica l'header Accept.
- Interpreta il valore q=0.8.
- Spiega perché la richiesta non contiene necessariamente un body.
- Individua nella risposta versione, codice di stato e frase descrittiva.
- Spiega cosa comunica Content-Type: text/html; charset=utf-8.
- Spiega quale funzione svolge Cache-Control: max-age=3600.
- Spiega cosa chiede al browser di fare Set-Cookie.
- Individua il body della risposta.
- Spiega perché non tutte le risposte HTTP devono necessariamente contenere un body, facendo almeno tre esempi tratti dalla dispensa.
- Immagina che il browser trovi nell'HTML immagini, CSS e JavaScript. Spiega perché la pagina può generare ulteriori richieste HTTP separate.
- Spiega perché una connessione HTTP può essere mantenuta aperta e riutilizzata per richieste successive.
- Concludi ricostruendo il ciclo completo: richiesta → elaborazione sul server → risposta → interpretazione del client → eventuali nuove richieste.

Bibliografia

Barth, A. (2011). *HTTP State Management Mechanism (RFC 6265)*. RFC Editor.

Berners-Lee, T., Fielding, R., & Masinter, L. (2005). *Uniform Resource Identifier (URI): Generic Syntax (RFC 3986)*. RFC Editor.

Bishop, M. (2022). *HTTP/3 (RFC 9114)*. RFC Editor.

Dusseault, L., & Snell, J. (2010). *PATCH Method for HTTP (RFC 5789)*. RFC Editor.

Fielding, R., Nottingham, M., & Reschke, J. (2022a). *HTTP/1.1 (RFC 9112)*. RFC Editor.

Fielding, R., Nottingham, M., & Reschke, J. (2022b). *HTTP Caching (RFC 9111)*. RFC Editor.

Fielding, R., Nottingham, M., & Reschke, J. (2022c). *HTTP Semantics (RFC 9110)*. RFC Editor.

Internet Assigned Numbers Authority. (2026). *Service Name and Transport Protocol Port Number Registry*. IANA.

Nottingham, M., & Fielding, R. (2012). *Additional HTTP Status Codes (RFC 6585)*. RFC Editor.

Rescorla, E. (2026). *The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 9846)*. RFC Editor.

Thomson, M., & Benfield, C. (2022). *HTTP/2 (RFC 9113)*. RFC Editor.

WHATWG. (2026). *Fetch Standard*.

Approfondimenti

- James F. Kurose e Keith W. Ross, Reti di calcolatori e Internet. Un approccio top-down, 8ª ed., Pearson, 2022 — È probabilmente il testo più direttamente adatto come prosecuzione della dispensa. Partendo dal livello applicativo, approfondisce il funzionamento del Web e di HTTP, il modello client-server, richieste e risposte, connessioni persistenti, caching, cookie e il rapporto tra HTTP e i protocolli sottostanti.
- Behrouz A. Forouzan e Firouz Mosharraf, Reti di calcolatori, 2ª ed., a cura di Gaia Maselli, McGraw-Hill Education, 2024 — Dedica un capitolo specifico al livello applicativo e adotta un approccio molto visuale. È utile per collocare HTTP all'interno della suite TCP/IP e comprendere il rapporto tra applicazioni Web, protocolli, client, server e livelli sottostanti della comunicazione.
- Andrew S. Tanenbaum, Nick Feamster e David J. Wetherall, Reti di calcolatori, 6ª ed., Pearson, 2023 — Offre una trattazione più sistematica del Web e dei protocolli applicativi, utile per approfondire HTTP dal punto di vista dell'architettura delle reti e comprendere come richieste, risposte e trasferimento delle risorse si inseriscano nel funzionamento complessivo di Internet.
- Larry L. Peterson e Bruce S. Davie, Reti di calcolatori, 3ª ed. italiana, Apogeo, 2012 — Affronta le reti attraverso un'impostazione orientata ai sistemi e ai protocolli, permettendo di approfondire il rapporto tra applicazioni, protocolli di rete e comunicazione end-to-end. Pur essendo meno recente degli altri testi, rimane utile per comprendere i principi architetturali su cui si basa anche HTTP.
- Malcolm McDonald, Web Security spiegata in modo facile. Guida illustrata per programmatori curiosi, Apogeo, 2024 — È un approfondimento complementare

particolarmente utile per la parte dedicata a HTTPS e sicurezza Web. Analizza crittografia, browser, rete e server e permette di comprendere perché proteggere la comunicazione HTTP sia fondamentale e quali vulnerabilità possano emergere nelle applicazioni Web.

Nota editoriale

Per la realizzazione di questi materiali ho utilizzato strumenti di intelligenza artificiale generativa come supporto alla scrittura, in particolare per migliorare la forma del testo, riorganizzare i contenuti, correggere la formulazione e velocizzare alcune attività redazionali.

Lavorando autonomamente alla produzione di questi materiali, cerco infatti di automatizzare tutte le attività che possono esserlo, così da poter dedicare più tempo alla ricerca, alla progettazione e allo sviluppo dei contenuti.

L'intelligenza artificiale non determina però i contenuti dell'opera: la scelta degli argomenti, la struttura, le idee, le interpretazioni, gli esempi e l'impostazione didattica sono elaborati da me. L'IA viene quindi utilizzata principalmente come strumento di supporto alla produzione e alla revisione formale, mentre la responsabilità autoriale e progettuale dei contenuti rimane mia.

Salvo diversa indicazione, questo materiale è distribuito con licenza Creative Commons Attribuzione – Non commerciale – Condividi allo stesso modo 4.0 Internazionale (CC BY-NC-SA 4.0).

È quindi possibile condividere, ridistribuire, modificare e creare opere derivate a partire dal materiale, a condizione di attribuirne correttamente la paternità, di non utilizzarlo per scopi commerciali e di distribuire eventuali versioni modificate o derivate con la stessa licenza.

