

Web & Networking

come funziona



Chi sono

Alessia Gennari

24 anni

Studentessa del corso di Informatica presso l'Università di Verona

Programmatrice back-end da 3 anni

Step 1:

Le Basi

Quale è la differenza tra browser e motore di ricerca?

BROWSER:

Software che permette di accedere a internet e visualizzare siti web

(es. Chrome, Firefox, Brave, etc.)

MOTORE DI RICERCA:

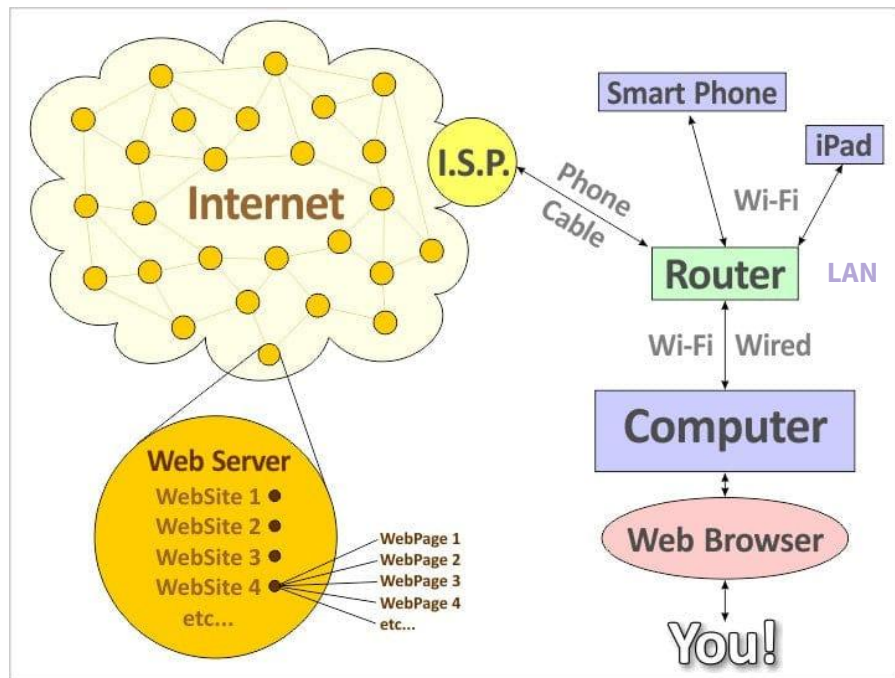
Servizio online che aiuta a trovare informazioni sul web

(es. Google, Bing, DuckDuckGo)

IN SOSTANZA:

Il **browser** è lo strumento di navigazione, il **motore di ricerca** è il servizio che aiuta a trovare i contenuti.

Come riusciamo a connetterci a internet



ISP, Internet Service Provider:

- permette di accedere a internet
- di 3 tipi, a seconda del tipo di rete (internazionale, nazionale, locale) es. Iliad, Vodafone, etc.

LAN, Local Area Network:

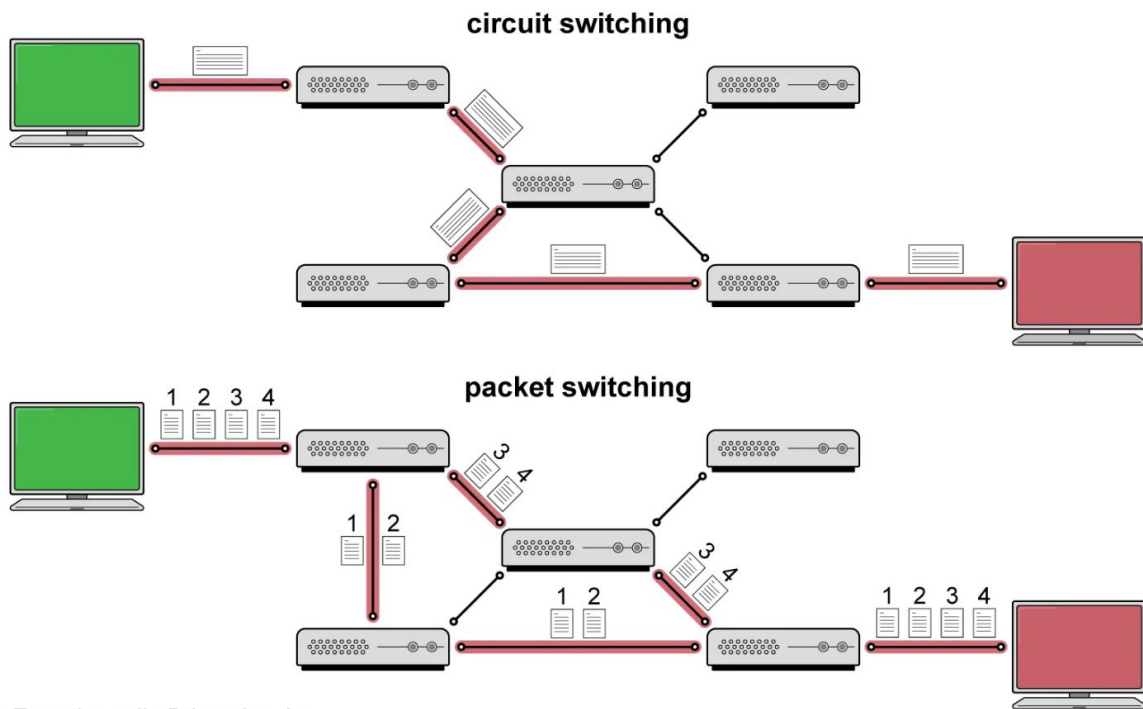
gruppo di dispositivi connessi tra di loro in uno spazio ristretto (es. stampante connessa al telefono tramite Wi-Fi)

Router:

separa la LAN da internet

Come viaggiano le informazioni su internet

Switching networks



commutazione a circuito:

tutte le risorse vengono usate per l'invio di un unico messaggio.

commutazione a pacchetto:

il messaggio viene suddiviso in più pacchetti numerati.

> maggior sfruttamento delle risorse

> possibilità di invio di più messaggi simultaneamente

Diversi tipi di protocolli

- **TCP/IP**: standard per la comunicazione su internet
- **FTP** (File Transfer Protocol): trasferimento file
- **SMTP**: gestione email
- **HTTPS/HTTP**: usati per la navigazione web

Step 2:

HTTP

HTTP vs HTTPS

Entrambe sono protocolli di trasferimento di pagine web.

HTTP -> HyperText Transfer Protocol

Per cosa sta la S?

HTTP Secure

Versione sicura che usa una crittografia SSL/TLS

- **protegge i dati** da intercettazioni e manomissioni
- **indispensabile per la sicurezza online** soprattutto nei siti che gestiscono dati sensibili

Anatomia di una richiesta HTTP

metodo uri ver protocollo

GET /api/page HTTP/1.1

Request Line

```
Accept: application/json, text/plain, */*
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-GB,en;q=0.5
Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImVFV3pvZE1PamhBQk1SYnZqV0Z4ciJ
Connection: keep-alive
Host: www.site.com
Origin: https://www.site.com
Referer: https://www.site.com
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Sec-GPC: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36
sec-ch-ua: "Brave";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?0
```

Header

Body (opzionale)

Anatomia di una **risposta** HTTP:

- request line -> response line
- si aggiunge lo stato della risposta alla response line (es. 200 OK)

Scomponiamolo: Request Line

A horizontal black bar representing an HTTP request line. It is divided into three colored boxes: a red box containing 'GET', a green box containing '/api/page', and a cyan box containing 'HTTP/1.1'.

```
GET /api/page HTTP/1.1
```

Metodo, specifica cosa vogliamo fare con la risorsa.

> esistono diversi tipi: **GET, POST, PUT, DELETE**

URI, indica al server dove si trova la risorsa.

> può essere un percorso assoluto (<https://www.sito.com/api/page>)

> o relativo (/api/page)

Versione, indica la versione del protocollo in uso.

Scomponiamolo: Response Line



The diagram shows an HTTP response line 'HTTP/1.1 200 OK' on a dark background. Three colored boxes highlight the components: a cyan box around 'HTTP/1.', a magenta box around '.1', and an orange box around '200'. The text 'OK' is not enclosed in a box.

Component	Value
Version	HTTP/1.1
Code	200
Message	OK

Versione, indica la versione del protocollo in uso.

Codice, numero a tre cifre che indica il risultato della richiesta.

Messaggio, stringa che specifica il motivo dell'errore.

Codici e messaggi di errore

esistono 5 categorie di **status code**:

> 1**, risposta informativa

> 2**, successo

> 3**, cambio locazione (redirect)

> 4**, errore del client

> 5**, errore del server

alcuni codici e messaggi “famosi”:

- **200 Ok**, la richiesta è andata a buon fine
- **404 Not Found**, la risorsa richiesta non è stata trovata
- **500 Internal Server error**, il server ha riscontrato un errore e non ha potuto completare la richiesta
- **418 I'm a teapot**, aggiunto come pesce d'aprile nel 1998

Scomponiamolo: Headers

Sono utilizzati per mandare delle informazioni aggiuntive al server

```
Accept: application/json, text/plain, */*
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-GB,en;q=0.5
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImVFV3pvZE1PamhBQk1SYnZqV0Z4ciJ
Connection: keep-alive
Host: www.site.com
Origin: https://www.site.com
Referer: https://www.site.com
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Sec-GPC: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36
sec-ch-ua: "Brave";v="131", "Chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?0
```

Headers obbligatori:

- **Host.** specifica l'indirizzo
- **Content-Encoding / Content-Length** (nel caso ci sia il body) specifica la codifica e la lunghezza del body

Come è fatto un URL

<https://www.esempio.com/pagina?id=1>

Schema, indica il protocollo

Dominio, indica il nome del sito

Percorso, specifica la risorsa all'interno del sito

Parametri, opzionali, usati solitamente per filtrare informazioni

Scomponiamolo: Body

Contiene tutti i dati da inviare nella richiesta, i quali possono essere di qualsiasi tipo (JSON, Text, File, etc)

Il contenuto dipende dall'applicazione Web, alcuni esempi:

- campi di una form di login (username e password)
- dati anagrafici (es. SPID)
- file delle tasse universitarie da scaricare
- immagine da pubblicare sui social

Cookies

File di testo che il sito web usa per memorizzare alcune informazioni.

Possibili utilizzi:

- Gestione della sessione (mantenimento login)
- Memorizzare preferenze di personalizzazione
- Mantenimento del carrello negli e-commerce

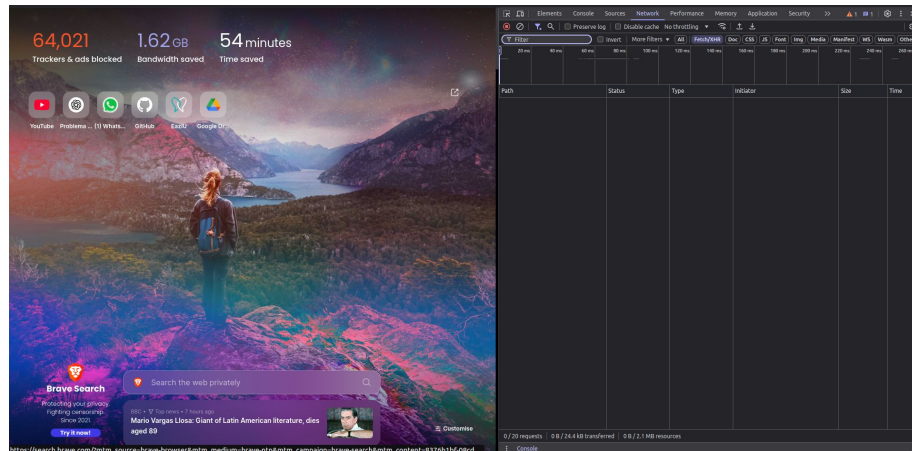
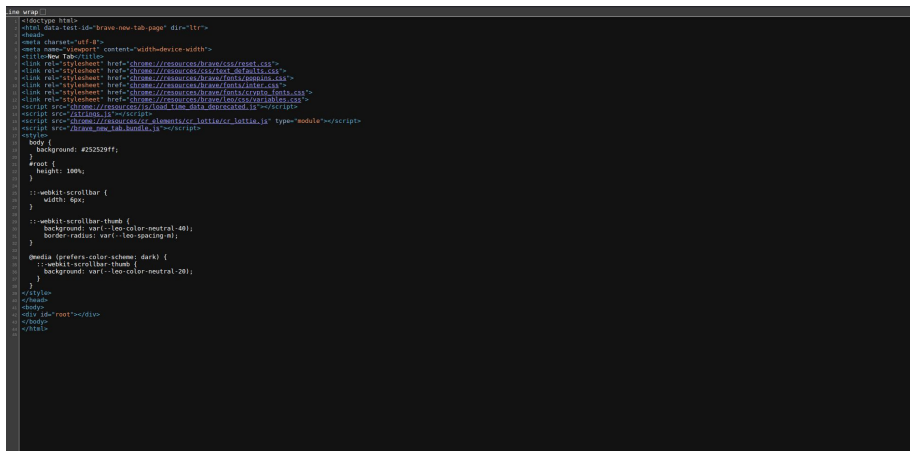
Step 3:

Assets

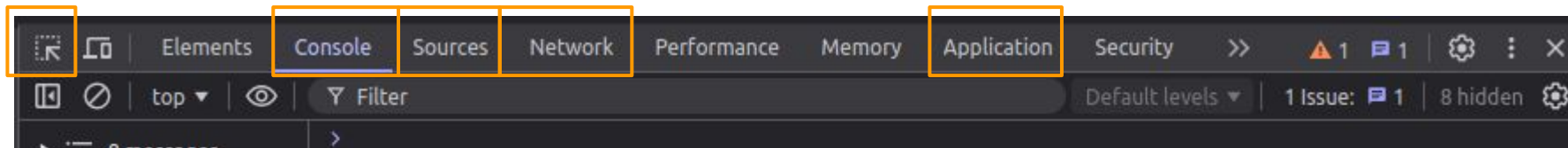
Web developer assets

Client side testing:

- > Accesso al **sorgente della pagina**: click destro -> “visualizza sorgente pagina”
- > **Dev tools**: F12 oppure ctrl + shift + I



Dev tools



- **Inspector**, permette di individuare elementi nel codice html della pagina
- **Console**, permette di visualizzare i messaggi di log e di lanciare comandi
- **Sources**, da accesso al codice sorgente della pagina e permette di effettuare il debug delle istruzioni
- **Network**, mostra il traffico HTTP generato dal browser
- **Application**, mostra i dati memorizzati inclusi i cookies

Qualcosa in più: CURL

Permette di fare richieste HTTP da terminale

`curl [options / URLs]`

Opzioni utili:

- **-H** , header
- **-X** , metodo (default GET)
- **-v** , output verboso
- **man curl** , per vedere tutte le opzioni possibili

```
a@p: $ curl -v www.google.com
* Trying 216.58.204.228:80...
*   Trying 2a00:1450:4002:415::2004:80...
* Immediate connect fail for 2a00:1450:4002:415::2004: Network is unreachable
* Connected to www.google.com (216.58.204.228) port 80 (#0)
> GET / HTTP/1.1
> Host: www.google.com
> User-Agent: curl/7.81.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Mon, 14 Apr 2025 10:24:19 GMT
< Expires: -1
< Cache-Control: private, max-age=0
< Content-Type: text/html; charset=ISO-8859-1
< Content-Security-Policy-Report-Only: object-src 'none';base-uri 'self';script-src 'nonce-mefbdE9bAhUE4v7x2Dw6wg' 'strict-dynamic' 'report-sample' 'unsafe-eval' 'unsafe-inline' https: http:;report-uri https://csp.withgoogle.com/csp/gws/other-hp
< Server: gws
< X-XSS-Protection: 0
< X-Frame-Options: SAMEORIGIN
< Set-Cookie: AEC=AVcja2c3np4bbmTbC18QLtQ0aONVT1dgEBDP9xPLOUZDLGON3SvldGt-3FY; expires=Sat, 11-Oct-2025 10:24:19 GMT; path=/; domain=.google.com; Secure; HttpOnly; SameSite=Lax
```

GRAZIE !