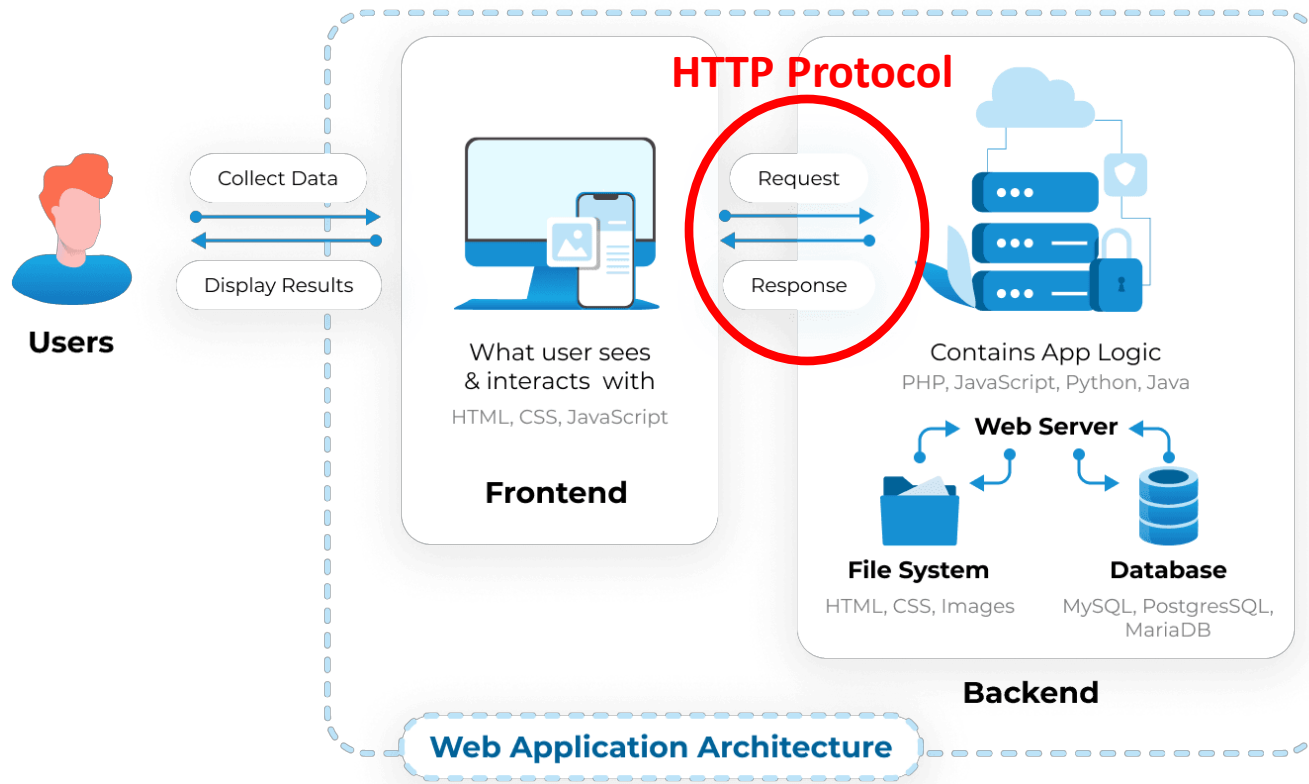


Protocollo HTTP

HTTP e Applicazioni web

Il protocollo HTTP è alla base delle applicazioni Web e rende possibili architetture client-server eterogenee.



HTTP

Il protocollo HTTP (HyperText Transfer Protocol) è uno dei protocolli applicativi più utilizzati.

Nasce per trasferire pagine HTML da un server ad un browser ma è oggi utilizzato in modo molto più ampio:

- Trasferendo anche altre informazioni associate ad una pagina HTML (css e javascript per esempio)
- Trasferendo comandi senza che sia necessaria la presenza di un browser (API)

Per questi motivi HTTP può anche essere considerato come un protocollo di trasporto costruito sopra TCP / IP.

HTTP

- HTTP risale agli anni 90 ed è stato esteso nel tempo
- Oggi è in utilizzo la versione 3 (uscita nel 2022)
- HTTP è un protocollo client server:
 - Il server si mette in attesa di una connessione rimanendo passivo fintanto che non ce e sono
 - Il client deve interpellare espressamente il server per poter comunicare
- HTTP è un protocollo request-response:
 - Il client invia al server delle richieste (di pagine HTML, di immagini, di operazioni...)
 - Il server risponde alla richiesta, soddisfacendola o negandola
 - In assenza di richieste il server rimane passivo
- Al server HTTP è tipicamente associato un nome di dominio
- Il server HTTP è tipicamente in ascolto sulla porta 80

HTTP

- Le richieste e le risposte sono in formato testuale ed hanno un formalismo ben preciso (sebbene non rigido)
- Le richieste e le risposte viaggiano in chiaro. Vengono inseriti in pacchetti TCP e in datagrammi IP senza alcuna protezione. Chiunque le intercetta può leggerle.
- Esiste una versione di HTTP, chiamata HTTPS, che aggiunge anche la cifratura delle informazioni trasmesse (la vedremo tra poco)
- HTTP utilizza TCP. È pertanto un protocollo che lavora su una connessione stabilita, affidabile e con gestione degli errori
- L'interazione è di tipo stateless, ovvero ogni coppia di richiesta/risposta è indipendente da tutte quelle avvenute prima e da tutte quelle avvenute dopo. Il server non comunica mai di sua iniziativa ma solamente in risposta ad una richiesta.

HTTP - Request e Response

Ogni interazione tra client e server include una richiesta dal client alla quale segue una risposta dal server.

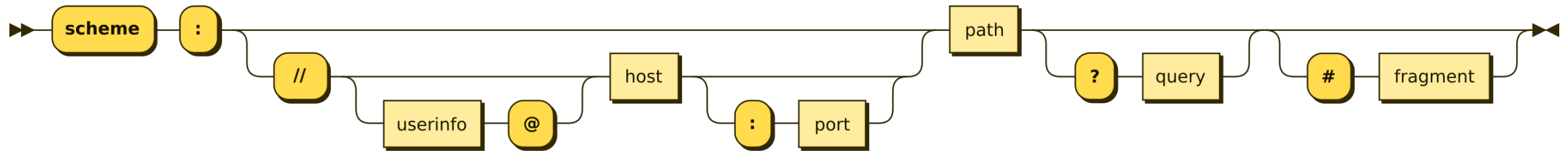
L'interazione è di tipo stateless. Ogni richiesta/risposta è indipendente da tutte quelle avvenute in precedenza.

Ogni richiesta è composta dalle seguenti informazioni principali:

- Url: Indirizzo della risorsa richiesta al server (con parametri opzionali)
- Tipologia di richiesta
- Header della richiesta
- Corpo della richiesta (opzionale)

HTTP - URL

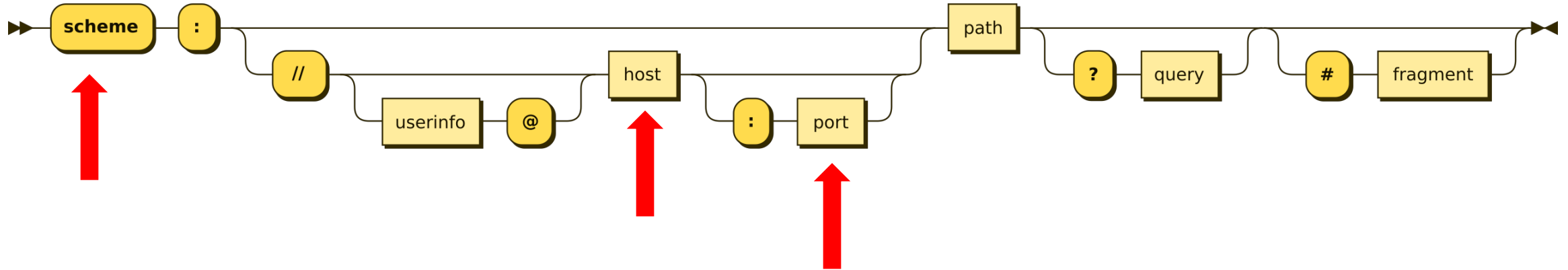
L'url di una risorsa web è nella forma che segue:



Per esempio:

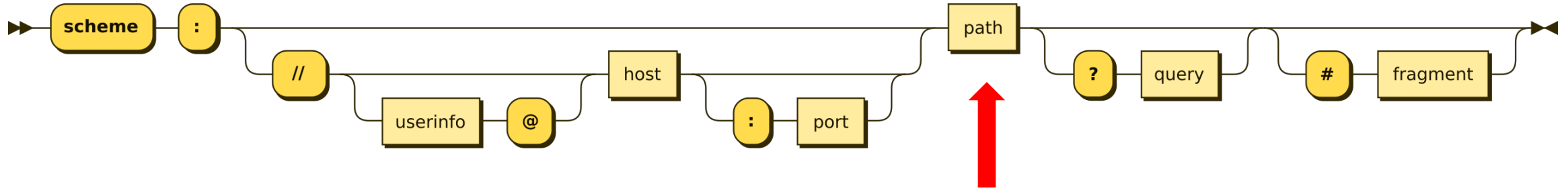
`https://www.example.com/search?keywords=python`

HTTP – URL – Schema, host e port



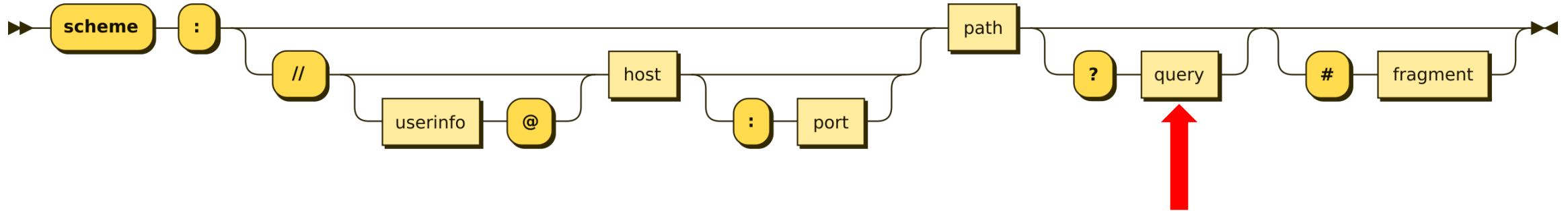
- Lo schema rappresenta il protocollo da utilizzare, principalmente http o https e determina anche la porta di default su cui il server è in ascolto: 80 per http e 443 per https
- Il nome dell'host identifica l'indirizzo IP del server. Può essere espresso direttamente, oppure espresso tramite un nome di dominio (a sua volta convertito in IP tramite il DNS)
- La porta va indicata quando il server non è in ascolto su quella di default.

HTTP – URL – Path



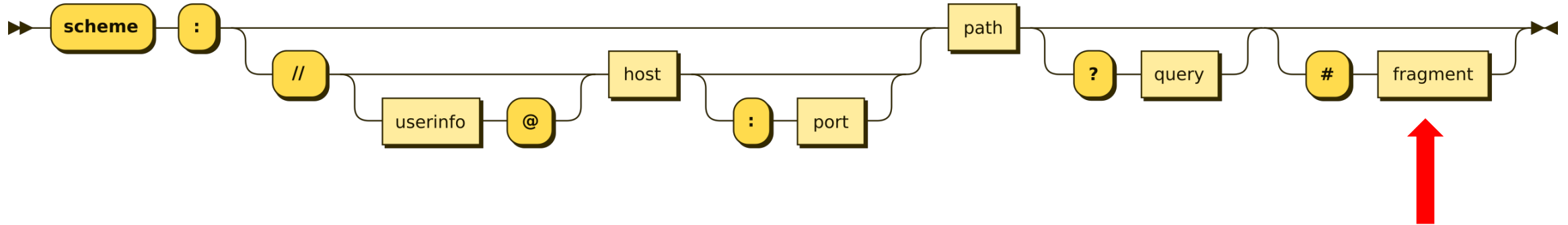
- Il path rappresenta una risorsa all'interno del server
- Può essere associato ad un file (p.e. html) o ad un metodo dell'applicazione di backend
- E' espresso analogamente ai path del file system linux, ovvero in forma strutturata utilizzando / come separatore
- All'assenza di path è associata la risorsa principale del server (tipicamente homepage)
- Esempio: /wiki/Hypertext_Transfer_Protocol

HTTP – URL – Path



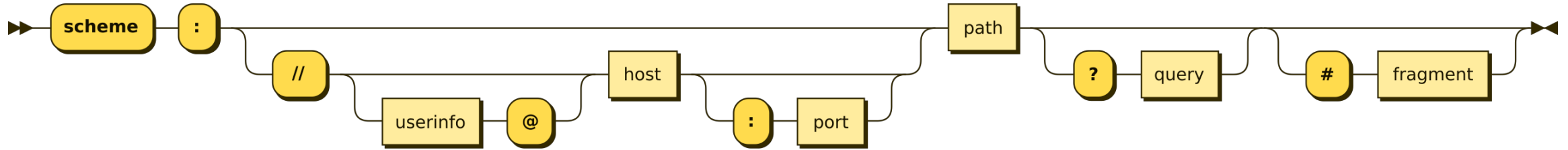
- Alla risorsa identificata dal path possono essere inviati parametri aggiuntivi.
- I parametri sono interpretati esclusivamente dal server e quindi il loro significato cambia in ogni applicazione
- Sono espressi nella forma nomeparametro=valore
- Parametri multipli devono essere separati dal carattere &. Per esempio
par1=valore1&par2=valore2

HTTP – URL – Fragment



- Il fragment rappresenta un informazione parametrica ad uso e consumo del client
- Non viene mai inviata al server. In caso di invio il server lo ignora
- Sono utilizzati principalmente per:
 - Effettuare navigazioni all'interno di una stessa pagina
 - Memorizzare informazioni

HTTP – URL – Codifica URL



- La sintassi dell'URL prevede alcuni caratteri speciali (/ ? & # ...) che pertanto non possono essere direttamente utilizzati all'interno di path, query e fragment
- Il carattere spazio è invece non ammesso, e pertanto non utilizzabile in nessuna parte dell'URL
- L'utilizzo dei caratteri speciali e dello spazio è comunque possibile utilizzando una particolare codifica detta codifica URL o codifica percentuale

HTTP – URL – Codifica URL

- Nella codifica URL ogni carattere è sostituito con il carattere % seguito dalla rappresentazione esadecimale del codice ASCII del carattere
- Possono essere codificati tutti i caratteri ma devono necessariamente esserlo quelli speciali e quelli non stampabili

- Esempi:

- & → %26

- # → %23

- / → %2F

- Spazio → %20

(in alternativa può essere usato il carattere +)

| Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char |
|---------|-----|------------------------|---------|-----|---------|---------|-----|------|---------|-----|-------|
| 0 | 0 | [NULL] | 32 | 20 | [SPACE] | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 1 | [START OF HEADING] | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 2 | [START OF TEXT] | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 3 | [END OF TEXT] | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 4 | [END OF TRANSMISSION] | 36 | 24 | \$ | 68 | 44 | D | 100 | 64 | d |
| 5 | 5 | [ENQUIRY] | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 6 | [ACKNOWLEDGE] | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 7 | [BELL] | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 8 | [BACKSPACE] | 40 | 28 | (| 72 | 48 | H | 104 | 68 | h |
| 9 | 9 | [HORIZONTAL TAB] | 41 | 29 |) | 73 | 49 | I | 105 | 69 | i |
| 10 | A | [LINE FEED] | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | B | [VERTICAL TAB] | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | C | [FORM FEED] | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | D | [CARRIAGE RETURN] | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | E | [SHIFT OUT] | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | F | [SHIFT IN] | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | [DATA LINK ESCAPE] | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | [DEVICE CONTROL 1] | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | [DEVICE CONTROL 2] | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | [DEVICE CONTROL 3] | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | [DEVICE CONTROL 4] | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | [NEGATIVE ACKNOWLEDGE] | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | [SYNCHRONOUS IDLE] | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | [ENG OF TRANS. BLOCK] | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | [CANCEL] | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | [END OF MEDIUM] | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | [SUBSTITUTE] | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | [ESCAPE] | 59 | 3B | ; | 91 | 5B | [| 123 | 7B | { |
| 28 | 1C | [FILE SEPARATOR] | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | |
| 29 | 1D | [GROUP SEPARATOR] | 61 | 3D | = | 93 | 5D |] | 125 | 7D | } |
| 30 | 1E | [RECORD SEPARATOR] | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | [UNIT SEPARATOR] | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | [DEL] |

HTTP - Verbi

Una richiesta Web può assumere quattro tipologie principali, ognuna delle quali identifica il tipo di operazione che vogliamo eseguire sulla risorsa identificata tramite URL

- GET = read
- PUT = update
- POST = create
- DELETE = delete
- HEAD = metadata (header) generati dalla richiesta GET sulla medesima risorsa
- OPTIONS = elenco delle operazioni permesse sulla risorsa

HTTP – Header

- Sono di "servizio" necessarie al server per elaborare correttamente, da un punto di vista tecnico, le richieste
- Sono espressi nella forma key:value
- Alcuni header sono obbligatori (p.e. host)
- Alcuni header sono obbligatori sono in alcune situazioni (p.e. content-type e content-length sono obbligatori in caso di presenza del corpo)

NB: In caso di connessione HTTPS gli header sono cifrati e pertanto leggibili solamente dal client e dal server.

HTTP – Corpo delle richieste

Il corpo della richiesta contiene informazioni aggiuntive ed opzionali. Per esempio:

- Parametri aggiuntivi, necessari al server per elaborare correttamente, da un punto di vista logico, le richieste (per esempio: dati da inserire nel database, credenziali di accesso...)
- Informazioni di autenticazione, necessarie al server per identificare il client e per gestirne le autorizzazioni
- Cookies

NB: In caso di connessione HTTPS il corpo della richiesta è cifrato e pertanto leggibile solamente dal client e dal server.

HTTP - Request e Response

Request

Pretty Raw Hex \n ≡

```
1 GET / HTTP/1.1
2 Host: duckduckgo.com
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:93.0) Gecko/20100101
  Firefox/93.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Upgrade-Insecure-Requests: 1
8 Sec-Fetch-Dest: document
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-Site: none
11 Sec-Fetch-User: ?1
12 Te: trailers
13 Connection: close
14
15
```

Response

Pretty Raw Hex Render \n ≡

```
1 HTTP/2 200 OK
2 Server: nginx
3 Date: Tue, 02 Nov 2021 15:43:09 GMT
4 Content-Type: text/html; charset=UTF-8
5 Vary: Accept-Encoding
6 Etag: W/"61803cf6-1682"
7 Strict-Transport-Security: max-age=31536000
8 Permissions-Policy: interest-cohort=()
9 Content-Security-Policy: default-src 'none' ; connect-src https://duckduckgo.com h
  x3sjasowoarfbgcmvfimaftt6twagswzczad.onion/ ; script-src blob: https://duckduckgo.
  ufc4m.onion/ https://duckduckgogg42xjoc72x3sjasowoarfbgcmvfimaftt6twagswzczad.onion
  ://duckduckgo.com https://*.duckduckgo.com https://3g2upl4pq6kufc4m.onion/ https://
10 X-Frame-Options: SAMEORIGIN
11 X-Xss-Protection: 1;mode=block
12 X-Content-Type-Options: nosniff
13 Referrer-Policy: origin
14 Expect-Ct: max-age=0
15 Expires: Tue, 02 Nov 2021 15:43:08 GMT
16 Cache-Control: no-cache
17
18 <!DOCTYPE html>
19 <!--[if IEMobile 7 ]> <html lang="it-IT" class="no-js iem7"> <![endif]-->
20 <!--[if lt IE 7]> <html class="ie6 lt-ie10 lt-ie9 lt-ie8 lt-ie7 no-js" lang="it-IT"
21 <!--[if IE 7]> <html class="ie7 lt-ie10 lt-ie9 lt-ie8 no-js" lang="it-IT"> <![en
22 <!--[if IE 8]> <html class="ie8 lt-ie10 lt-ie9 no-js" lang="it-IT"> <![endif]-->
23 <!--[if IE 9]> <html class="ie9 lt-ie10 no-js" lang="it-IT"> <![endif]-->
24 <!--[if (gte IE 9)|(gt IEMobile 7)|!(IEMobile)|!(IE)]><!--><html class="no-js" lang
  <!--<![endif]-->
```

HTTP - Request e Response

Le response contengono principalmente:

- Un codice di risposta (che ne identifica lo stato)
- I dati della risposta, complessivi di header (per esempio l'html della pagina richiesta).

Codici di risposta:

- 2xx: Operazione avvenuta con successo
- 3xx: Redirect verso altra risorsa
- 4xx: Risorsa non trovata o errori di autenticazione (in generale Client error)
- 5xx: Server Error

HTTP - Request e Response

Request

Pretty Raw Hex \n ≡

```
1 GET / HTTP/1.1
2 Host: www.google.it
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:94.0) Gecko/20100101
  Firefox/94.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9
10
```

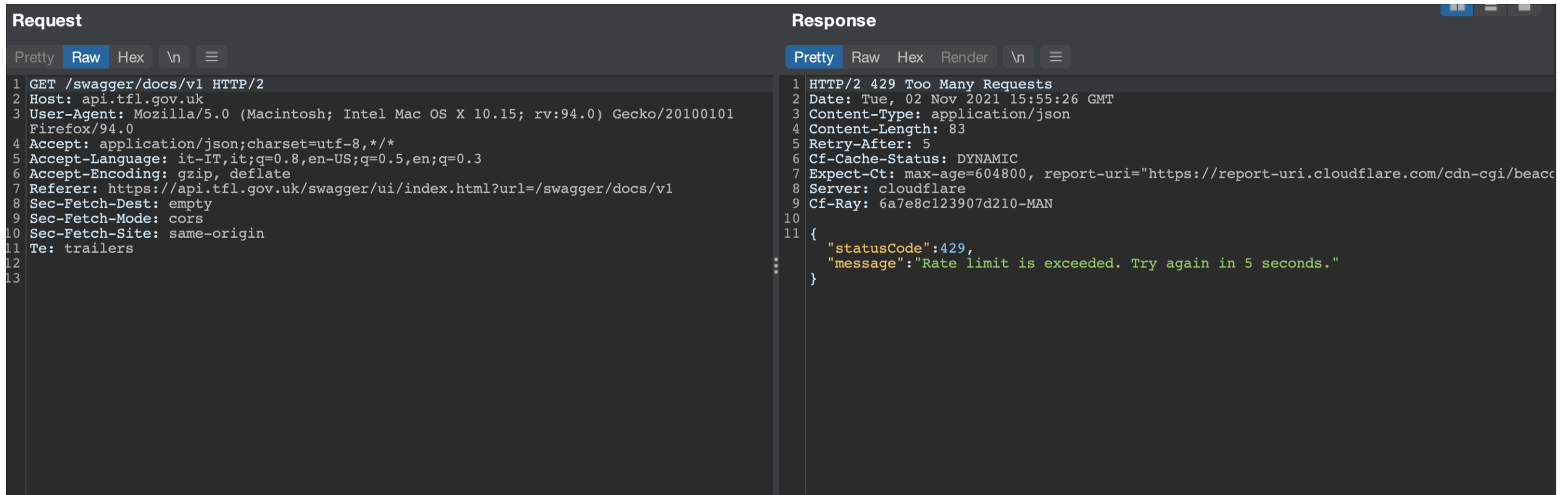
Response

Pretty Raw Hex Render \n ≡

```
1 HTTP/1.1 302 Found
2 Location: https://www.google.it/?gws_rd=ssl
3 Cache-Control: private
4 Content-Type: text/html; charset=UTF-8
5 Date: Tue, 02 Nov 2021 15:49:58 GMT
6 Server: gws
7 Content-Length: 230
8 X-XSS-Protection: 0
9 X-Frame-Options: SAMEORIGIN
10 Connection: close
11
12 <HTML>
13   <HEAD>
14     <meta http-equiv="content-type" content="text/html; charset=utf-8">
15     <TITLE>
16       302 Moved
17     </TITLE>
18   </HEAD>
19   <BODY>
20     <H1>
21       302 Moved
22     </H1>
23     The document has moved
24     <A HREF="https://www.google.it/?gws_rd=ssl">here
25   </A>
26 </BODY>
27 </HTML>
```

HTTP - Request e Response

Le response possono contenere dati di vario tipo. I più diffusi sono HTML per le pagine e json per i dati. In caso di connessioni HTTPS il corpo della risposta è cifrato.



```
Request
Pretty Raw Hex \n
1 GET /swagger/docs/v1 HTTP/2
2 Host: api.tfl.gov.uk
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:94.0) Gecko/20100101 Firefox/94.0
4 Accept: application/json; charset=utf-8, */*
5 Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Referer: https://api.tfl.gov.uk/swagger/ui/index.html?url=/swagger/docs/v1
8 Sec-Fetch-Dest: empty
9 Sec-Fetch-Mode: cors
10 Sec-Fetch-Site: same-origin
11 Te: trailers
12
13

Response
Pretty Raw Hex Render \n
1 HTTP/2 429 Too Many Requests
2 Date: Tue, 02 Nov 2021 15:55:26 GMT
3 Content-Type: application/json
4 Content-Length: 83
5 Retry-After: 5
6 Cf-Cache-Status: DYNAMIC
7 Expect-Ct: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon"
8 Server: cloudflare
9 Cf-Ray: 6a7e8c123907d210-MAN
10
11 {
12   "statusCode": 429,
13   "message": "Rate limit is exceeded. Try again in 5 seconds."
14 }
```