Http Request Response .

[**1 Http Request 1**](#_2hteuhc1zr4r)

[Components d'una HTTP request 1](#_did7fh99s37w)

[1.1 Exemples de HTTP request 2](#_j6dkqb5rx134)

[Com funciona un HTTP request en un entorn real? 2](#_phr5wi7pl7j)

[1.2 clients HTTP 2](#_etao4wmui0pn)

[1.3 Mime Types 3](#_xycikdkcd1h7)

[**2 Http Response 5**](#_wbe1qbart7ja)

[2.1 Línia d'estat (Status Line) 5](#_8d30t4rs42u3)

[2.2 Headers 5](#_yyu0xxcm24az)

[2.3 Body (cos de la resposta) 6](#_yoijhqhxtw05)

[2.4 Codis d’estat de Response Http 6](#_31bju2sjqc51)

[1. Codi d'estat 1xx - Informatius 6](#_u0hv4dstq9jq)

[2. Codi d'estat 2xx - OK 7](#_aw4xqdriqnt5)

[3. Codi d'estat 3xx - Redirecció 7](#_2gzg3dpojm0y)

[4. Codi d'estat 4xx - Errors del client 7](#_4ia4l1nju16x)

[5. Codi d'estat 5xx - Errors del servidor 8](#_4rm9jjhhberc)

# 1 Http Request

Un **HTTP request** és una sol·licitud que un client (normalment un navegador web o una aplicació) envia a un servidor utilitzant el protocol **HTTP (HyperText Transfer Protocol)**. Aquesta sol·licitud permet al client demanar recursos o enviar dades al servidor, com ara pàgines web, imatges o informació per processar.

### **Components d'una HTTP request**

1. **Mètode HTTP**: Indica el tipus d'acció que el client vol fer. Els mètodes més comuns són:
   * **GET**: Demanar dades o recursos (com una pàgina web).
   * **POST**: Enviar dades al servidor (per exemple, formularis).
   * **PUT**: Actualitzar recursos existents al servidor.
   * **DELETE**: Eliminar un recurs al servidor.
2. **URL (Uniform Resource Locator)**: Especifica el recurs que es vol accedir o modificar.
3. **Headers**: Informació addicional sobre la sol·licitud, com ara el tipus de dades acceptades, l'autenticació, o la configuració de codificació.
4. **Body** (opcional): En cas de mètodes com **POST** o **PUT**, conté les dades que s'envien al servidor (per exemple, dades de formularis o informació en format JSON).

## 1.1 Exemples de HTTP request

GET request:

GET /index.html HTTP/1.1

Host: www.example.com

POST request:

POST /api/login HTTP/1.1

Host: www.example.com

Content-Type: application/json

Content-Length: 45

{

"username": "usuari",

"password": "1234"

}

### **Com funciona un HTTP request en un entorn real?**

1. **El client** (navegador o aplicació) inicia una sol·licitud HTTP enviant-la a un servidor.
2. **El servidor** processa la sol·licitud i respon amb un HTTP response, que pot incloure dades (com una pàgina HTML) o un estat de confirmació (com 200 OK o 404 Not Found).

## 1.2 clients HTTP

**curl** (acrònim de Client URL) és una eina de línia de comandes utilitzada per transferir dades des d'un client fins a un servidor, o viceversa, a través de diferents protocols de xarxa, com ara HTTP, HTTPS, FTP, SFTP, SMTP, POP3, i molts més.

Instal·lar curl: <https://scoop.sh/#/apps?q=curl>

**IMPORTANT** : heu d’utilitzar la ruta del curl que heu instal·lat amb scoop. A PowerShell, curl és un alias de Invoke-WebRequest i dona error

Executar una petició GET:

PS C:\Users\amallad2> C:\Users\amallad2\scoop\apps\curl\current\bin\

curl https://proven.cat

PS> C:\Users\amallad2\scoop\apps\curl\current\bin\curl https://api.chucknorris.io/jokes/categories

PS> C:\Users\amallad2\scoop\apps\curl\current\bin\curl h[ttps://api.chucknorris.io/jokes/random?category=](https://api.chucknorris.io/jokes/random?category=%7Bcategory%7D)dev

Desar la resposta en un fitxer:

PS C:\Users\amallad2> C:\Users\amallad2\scoop\apps\curl\current\bin\curl -o fitxer.html h[ttps://api.chucknorris.io/jokes/random?category=](https://api.chucknorris.io/jokes/random?category=%7Bcategory%7D)dev

Executar petició POST: <https://www.postman.com/postman/published-postman-templates/documentation/ae2ja6x/postman-echo?ctx=documentation>

PS> C:\Users\amallad2\scoop\apps\curl\current\bin\curl -X POST -H "Content-Type: application/json" -d "{'test': 'value'}" https://postman-echo.com/post

## 1.3 Mime Types

Els **MIME types** (Multipurpose Internet Mail Extensions) són estàndards utilitzats per identificar el tipus de contingut d'un fitxer o dades en comunicacions a través d'internet. Originalment, es van crear per correu electrònic, però avui dia s'utilitzen àmpliament en protocols com **HTTP** per indicar el tipus de contingut d'una resposta o sol·licitud.

**1. multipart/form-data**

multipart/form-data és un tipus de MIME utilitzat principalment per enviar formularis HTML que contenen fitxers o dades binàries (per exemple, imatges, documents, etc.). Aquest tipus de codificació permet enviar diversos tipus de dades (com text i fitxers) en una sola petició HTTP. És el més utilitzat en formularis que permeten la càrrega de fitxers.

Petició Http POST

<form action="https://example.com/upload" method="POST" enctype="multipart/form-data">

<label for="name">Nom:</label>

<input type="text" id="name" name="name" value="Joan">

<br>

<label for="file">Selecciona una imatge:</label>

<input type="file" id="file" name="file">

<br>

<input type="submit" value="Enviar">

</form>

es genera una petició POST

POST /upload HTTP/1.1

Host: example.com

Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

Content-Length: 5000

----WebKitFormBoundary7MA4YWxkTrZu0gW

Content-Disposition: form-data; name="name"

Joan

----WebKitFormBoundary7MA4YWxkTrZu0gW

Content-Disposition: form-data; name="file"; filename="image.jpg"

Content-Type: image/jpeg

(binary data of the image)

----WebKitFormBoundary7MA4YWxkTrZu0gW--

**2. application/x-www-form-urlencoded**

application/x-www-form-urlencoded és el tipus de codificació per defecte utilitzat quan s’envien dades a través d’un formulari HTML utilitzant el mètode POST (o GET en alguns casos). Les dades s’envien com a una cadena de text amb parelles clau-valor, on les claus i els valors estan separats per = i cada parella es separa per &.

Exemple petició Http POST:

POST /submit HTTP/1.1

Host: example.com

Content-Type: application/x-www-form-urlencoded

Content-Length: 27

username=John+Doe&age=30&city=New+York

**3. application/json**

application/json és el tipus MIME utilitzat per enviar dades en format JSON. JSON (JavaScript Object Notation) és un format lleuger per emmagatzemar i transportar dades, àmpliament utilitzat en API RESTful i serveis web moderns. Aquest tipus s’utilitza per a la comunicació entre clients i servidors que requereixen enviar dades estructurades de forma més complexa, com objects o arrays.

POST /submit HTTP/1.1

Host: example.com

Content-Type: application/json

Content-Length: 42

{

"username": "John Doe",

"age": 30,

"city": "New York"

}

## 1.4 Python

$ pip install requests

APIS per fer peticions <https://github.com/public-apis/public-apis>

### Exemple GET request

import requests

url = "https://api.chucknorris.io/jokes/categories"

response = requests.get(url)

if response.status\_code == 200:

print("✅ Èxit!")

print(response.text)

else:

print(f"❌ Error {response.status\_code}")

## GET request with parameter

import requests

url = "https://api.chucknorris.io/jokes/random?category=food"

response = requests.get(url)

if response.status\_code == 200:

print("✅ Èxit!")

print(response.text)

else:

print(f"❌ Error {response.status\_code}")

import requests

url = "https://api.chucknorris.io/jokes/random"

params = {"category": "food"}

response = requests.get(url, params=params)

print(response.url) # mostra la URL final amb els paràmetres

print(response.json())

# 2 Http Response

Una HTTP Response (Resposta HTTP) és la resposta que el servidor envia al client després d'una petició HTTP (com ara una petició GET o POST). La resposta inclou informació sobre si la petició va ser procesada correctament o si va ocórrer un error.

Una resposta HTTP general es compon de tres parts principals:

* Línia d'estat (Status Line)
* Headers
* Body

## 2.1 Línia d'estat (Status Line)

* El **protocol** de la resposta: Normalment HTTP/1.1 o HTTP/2.
* El **codi d'estat** HTTP: Un número que indica l'estat de la petició (com 200, 404, 500, etc.).
* La **descripció** de l'estat: Una breu explicació del codi d'estat (com "OK" o "Not Found").

HTTP/1.1 200 OK

## 2.2 Headers

Els headers són metadades que proporcionen informació addicional sobre la resposta, com el tipus de contingut, la longitud del contingut, les cookies, i altres detalls de la comunicació.

Exemples de headers comuns:

* **Content-Type**: Indica el tipus de contingut de la resposta (per exemple, text/html, application/json, image/jpeg).
* **Content-Length**: Indica la mida del contingut en bytes.
* **Date**: La data i hora en què es va generar la resposta.
* **Server**: La informació sobre el servidor que ha generat la resposta.
* **Set-Cookie**: Si s'han establert cookies al client.

Content-Type: application/json

Content-Length: 123

Date: Mon, 22 Jan 2025 12:00:00 GMT

Server: Apache/2.4.41 (Ubuntu)

### 

### 

## 2.3 Body (cos de la resposta)

El body de la resposta conté les dades que el servidor vol enviar al client. El contingut del body depèn del tipus de petició i la resposta, i pot ser una pàgina web HTML, un document JSON, un fitxer d'imatge, etc.

* **HTML**: En el cas de pàgines web, el cos serà el codi HTML.
* **JSON**: Quan s'està treballant amb API RESTful, el cos sovint serà un objecte o array JSON.
* **binari**: Quan es carrega un fitxer, el cos contindrà les dades binàries d'aquest fitxer (per exemple, una imatge o un document).

{

"status": "success",

"message": "Data received successfully",

"data": {

"id": 123,

"name": "Joan"

}

}

## 2.4 Codis d’estat de Response Http

Els **codis d'estat HTTP** són respostes numèriques que el servidor envia al client per indicar l'estat de la petició. Aquests codis es poden agrupar en cinc categories, cada una amb una finalitat específica:

### 1. Codi d'estat 1xx - Informatius

Els codis d'estat de la categoria **1xx** indiquen que el servidor ha rebut la petició i que s'està processant. Són codis informatius, que normalment no s'utilitzen en la comunicació entre el navegador i el servidor.

* **100 Continue**: El servidor ha rebut la primera part de la petició i el client pot continuar enviant la resta.
* **101 Switching Protocols**: El servidor accepta canviar el protocol seguint la petició del client (per exemple, de HTTP/1.1 a HTTP/2).

### 2. Codi d'estat 2xx - OK

Els codis de la categoria **2xx** indiquen que la petició s'ha processat correctament i que el servidor ha retornat una resposta satisfactòria.

* **200 OK**: La petició ha estat exitosa. El cos de la resposta conté els resultats sol·licitats.
* **201 Created**: La petició ha estat exitosa i ha creat un nou recurs (com en una petició POST per afegir un element).
* **202 Accepted**: La petició ha estat acceptada, però encara no s'ha processat.
* **204 No Content**: La petició ha estat processada correctament, però no hi ha contingut per retornar (per exemple, després d'una petició DELETE).

### 3. Codi d'estat 3xx - Redirecció

Els codis de la categoria **3xx** indiquen que el client ha de realitzar una altra acció per completar la petició. En general, aquests codis impliquen que el client ha de ser redirigit a una altra URL.

* **301 Moved Permanently**: El recurs sol·licitat ha estat mogut de manera permanent a una nova URL.
* **302 Found** (anteriorment anomenat "Moved Temporarily"): El recurs sol·licitat es troba temporalment en una URL diferent.
* **303 See Other**: El client ha de realitzar una petició **GET** a una altra URL per obtenir el recurs.
* **304 Not Modified**: El recurs no s'ha modificat des de la darrera sol·licitud. Això pot ser útil per a la caché del navegador.
* **307 Temporary Redirect**: El recurs ha estat mogut temporalment a una nova URL, però el mètode de la petició ha de ser mantingut.
* **308 Permanent Redirect**: El recurs ha estat mogut permanentment a una nova URL, i el mètode de la petició ha de ser mantingut.

### 4. Codi d'estat 4xx - Errors del client

Els codis de la categoria **4xx** indiquen que hi ha un problema amb la petició del client. Potser falta informació o hi ha errors en la sol·licitud.

* **400 Bad Request**: La petició és mal formada o conté dades incorrectes.
* **401 Unauthorized**: El client no ha proporcionat les credencials correctes per accedir a un recurs protegit.
* **403 Forbidden**: El servidor entén la petició, però es nega a autoritzar-la. Potser el client no té permisos.
* **404 Not Found**: El recurs sol·licitat no es troba al servidor.
* **405 Method Not Allowed**: El mètode de la petició (com GET, POST, DELETE, etc.) no és permès per al recurs sol·licitat.
* **408 Request Timeout**: El servidor ha superat el temps límit per processar la petició.
* **409 Conflict**: La petició no es pot completar a causa d'un conflicte amb l'estat actual del recurs.
* **413 Payload Too Large**: La petició és massa gran per ser processada pel servidor.
* **414 URI Too Long**: La URL sol·licitada és massa llarga.
* **415 Unsupported Media Type**: El servidor no admet el tipus de contingut de la petició.
* **429 Too Many Requests**: El client ha fet massa peticions en un temps curt.

### 5. Codi d'estat 5xx - Errors del servidor

Els codis de la categoria **5xx** indiquen que hi ha un problema amb el servidor que impedeix processar la petició, fins i tot si la petició era vàlida.

* **500 Internal Server Error**: Error general del servidor. El servidor no ha pogut completar la petició per una raó desconeguda.
* **501 Not Implemented**: El servidor no suporta la funció necessària per processar la petició.
* **502 Bad Gateway**: El servidor, actuant com a passarel·la o proxy, ha rebut una resposta no vàlida d'un servidor ascendent.
* **503 Service Unavailable**: El servidor no està disponible temporalment, possiblement per sobrecàrrega o manteniment.
* **504 Gateway Timeout**: El servidor, actuant com a passarel·la o proxy, no ha rebut una resposta a temps d'un servidor ascendent.
* **505 HTTP Version Not Supported**: El servidor no admet la versió del protocol HTTP utilitzada en la petició.