Protocole HTTP

Claire Medrala et Fabien Coelho

MESR, Mines Paris – PSL

Février 2025

Menu

Couches ISO - Protocoles

Modèle ISO	Protocoles	Modèle TCP/IP
7 Application	FTP HTTP HTTPS DNS SMTP POP IMAP SSH LDAP	Application
6 Présentation		
5 Session		
4 Transport	TCP UDP ICMP	Transport
3 Réseau	IPv4 IPv6	Réseau
2 Liaison	Ethernet, Wifi	Lien
1 Physique	Fibre, câble, air	Physique

Part de HTTP/HTTPS dans le traffic Internet ?

https://home.cern/science/computing/birth-web/short-history-web

1989 CERN – proposition de Tim Berners-Lee
développement serveur et navigateur sur NeXT

1991 diffusion : SLAC (Californie), NCSA (Illinois)... `HTTPd` `Mosaic`

1994 création du World Wide Web Consortium (W3C) *standardisation*

- INRIA prend le relai du CERN
- 10 000 serveurs, 10 millions d'utilisateurs

Traffic

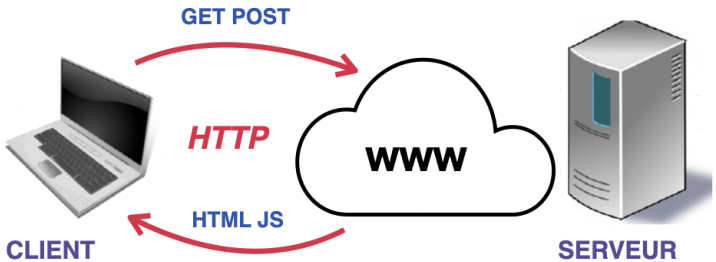
2020 augmentation du traffic lors de la crise du Covid-19

2022 traffic France : Netflix = 20%, GAFAM = 30%, $\times 20$ en 10 ans

2024 $+1$ milliard de sites, $+12$ millions de serveurs, $+5$ milliards de clients, $\frac{2}{3}$ vidéo

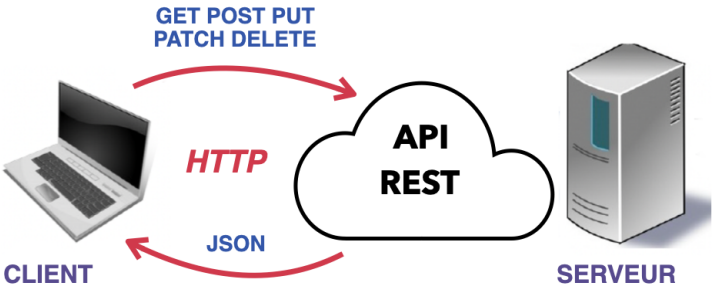
Web classiques — navigateur

URL	lien vers la page demandée
HTTP	protocole d'échanges de documents, GET POST
HTML CSS	formats des documents, formulaires, liens...
JS	scripting éventuel côté client...



Single Page Application mobile, navigateur

API	Application Programming Interface
REST	REpresentational State Transfer
HTTP	protocole d'échanges de données GET POST PUT PATCH DELETE
XML JSON	formats des données



Protocole HTTP

Définition

HT	HyperText	**navigation** entre documents
T	Tranfer	de données **typées**
P	Protocol	modèle **client-serveur**

Caractéristiques

identification	par URL, arborescence	
client	requêtes via navigateur web ou programme	
serveur	réponse via un logiciel (Nginx, Apache, Microsoft IIS)	
sans état	requêtes indépendantes	*vs* `imap smtp ftp`
documents	typés via MIME, interprétables par le client	

HTTP formats requête et réponse

Requête

`http://popote/index.html`

- méthode chemin version
- en-têtes sur plusieurs lignes...
- une ligne vide
- un contenu éventuel

```
GET /index.html HTTP/1.1
Host: popote
User-Agent: Mozilla/5.0
```

Réponse

- version code status
- en-têtes sur plusieurs lignes...
- une ligne vide
- le contenu de la réponse

```
HTTP/1.1 200 OK
Server: Apache
Content-Length: 79
Content-Type: text/html

<html><head><title>Popote</title></head>
<body><h1>Popote !</h1></body></html>
```

Versions HTTP

HTTP/1.0 — RFC 1945 *(1996)*

- une requête = une connexion TCP/IP
- pas de `Host` : IP exclusive

```
GET / HTTP/1.0
```

HTTP/1.1 — RFC 2068/2616-7/7230-5 *(1997)*

- une connexion = plusieurs requêtes
  synchro avec `Content-Length`
- `Host` obligatoire : IP partagées

```
GET / HTTP/1.1
Host: popote.fr
```

HTTP/2 — RFC 7540 *(2015)*

- protocole binaire : parallélisme, push...

HTTP/3 — RFC 9114 *(2022)*

- couche transport QUIC : Quick UDP Internet Connection

URI

Uniform Resource Identifier

Chaîne de caractères qui désigne une ressource

- URN = Uniform Ressource Name : identité d'une ressource unique

`mailto:chef@popote.fr`	mél
`urn:isbn:978-2-501-06213-8`	référence livre
`urn:oasis:names:specification:docbook:dtd:xml:4.1.2`	XML

- URL = Uniform Resource Location : localisation unique

```
ftp://ftp.popote.fr/recettes/4quart.txt
file:///home/chef/recette.txt
http://www.minesparis.psl.eu/index.html
https://www.cri.ensmp.fr/intranet/
nntp://news.popote.fr/ingredients.qualite/5432
```

URL hiérarchiques

Format

<protocole>://<autorite>/<chemin>/<ressource>#<tag>

<protocole>	http https ftp sftp file gopher nntp	
<autorite>	<login>:<mdp>@<host>:<port> port par défaut : http 80, https 443, ftp 21...	moe@comics.net
<chemin>	hiérarchie	/ /ingredient/farine/
<ressource>	souvent nom d'un fichier	index.html
<tag>	ancre (position) dans le document	

<protocole>://<autorite>/<chemin>?par=1&am=2&etre=3

? séparateur annonçant des paramètres param=val

& séparateur entre deux couples paramètre/valeur

URL

Caractères

Caractères réservés — séparateurs interprétés

- / path
- ? query
- & params
- # tag

Caractères interdits

- contrôle (US-ASCII 00-1F, 7F)
- caractères accentuées, UTF-8...
- espace, tabulation
- " < > utilisés pour délimiter une URI
- ^
- {}|\^[] potentiellement transformés

Encodage URL — % *exa exa*

- espace %20
- tab %09
- # %23
- & %26

`https://www.google.com/search?q=quatre%20quart&lang=fr`

Requête HTTP

5 méthodes

GET	consulter des données, sans les modifier
POST	ajouter des données (formulaire)
PATCH	modifier des données
PUT	remplacer des données
DELETE	supprimer des données

```
GET /image/casserole.png HTTP/1.1
Host: popote.fr
```

```
DELETE /ingredient/epinard HTTP/1.1
Host: popote.fr
```

4 méthodes

TRACE	demande la requête reçue (mirroir, pour debug)
HEAD	demande les entêtes seules, sans le document
OPTIONS	demande les méthodes disponibles
CONNECT	demande un changement de protocole (tunnel dans HTTP)

```
TRACE /ingredient/beurre HTTP/1.1
Host: popote.fr
User-Agent: Claire
```

```
HTTP/1.1 200 OK
Content-Type: message/http
Content-Length: 74

TRACE /ingredient/beurre HTTP/1.1
Host: popote.fr
User-Agent: Claire
```

HTTP Requête

exemples

```
HEAD / HTTP/1.1
Host: popote.fr
```

```
HTTP/1.1 200 OK
Server: Apache
Last-Modified: Tue, 06 Sep 2022 13:32:46 GMT
Content-Type: text/html
Content-Length: 79
```

```
OPTIONS / HTTP/1.1
Host: popote.fr
```

```
HTTP/1.1 200 OK
Allow: POST,OPTIONS,HEAD,GET,TRACE
```

```
CONNECT ssh.popote.fr:22 HTTP/1.1
Host: proxy.popote.fr

<protocole ssh ...>
```

```
HTTP/1.1 200 Connection established

<protocole ssh ...>
```

2 types de paramètres

HTTP paramètres intégrés au protocole

- **URL simple** nom = valeur texte, directement dans URL `GET`
- **corps simple** nom = valeur texte, dans le corps `POST...`
  `Content-Type: application/x-www-form-urlencoded`
- **corps upload** nom param, type mime, nom fichier et contenu
  `Content-Type: multipart/form-data`

JSON/XML dans le document transporté

- structure de données riches : liste, dico, entiers...
  `Content-Type: application/json`
  `Content-Type: application/xml`
  `Content-Type: text/xml`

méthode GET

- passage des options dans l'URL après ?, taille limitée
- nom = valeur séparés par des &, encodage URL

```
GET /recette/vegetarien?ingredient=carotte&ustensile=four HTTP/1.1
Host: popote.fr
```

méthodes POST PUT PATCH DELETE...

- idem, *mais* passage dans le corps de la requête

```
POST /recette/vegetarien HTTP/1.1
Host: popote.fr
Content-Type: application/x-www-form-urlencoded
Content-Length: 32

ingredient=salade&ustensile=four
```

méthode POST

- `Content-Type:` `multipart/form-data` + option `boundary` (aléatoire)
- `Content-Length:` ... attention fins de lignes `cr lf`
- `Content-Disposition:` `form-data` + options `name` `filename`

```
POST /upload HTTP/1.1
Host: popote.fr
Content-Length: 224
Content-Type: multipart/form-data; boundary=POPOTE

--POPOTE
Content-Disposition: form-data; name="recette"; filename="4quarts.txt"
Content-Type: text/plain

Mélanger le sucre, les œufs, le beurre et la farine.
Cuire au four à 180°C pendant une heure.

--POPOTE--
```

```
PATCH /recette/4quart HTTP/1.1
Host: popote.fr
Content-Type: application/json
Content-Length: 20

{"temps": "50min"}
```

```
PUT /recette/4quart HTTP/1.1
Host: popote.fr
Content-Type: text/xml
Content-Length: 22

<name="temps">50min</>
```

Réponse HTTP

Codes de retour HTTP

5 familles

1xx	Information	peu utilisé
2xx	Succès	OK, création, modif. . .
3xx	Redirection	nouvelle URL fournie
4xx	Erreur client	oups
5xx	Erreur serveur	OUPS

```
GET /ingredient/tea HTTP/1.1
Host: popote.fr
```

```
HTTP/1.1 418 I'm a Teapot
Date: Tue, 1 Apr 1998 00:00:01 GMT
Server: Teapot/1.0 (RFC 2324)
```

Codes de retour HTTP

2xx

10+ codes succès, dont extensions

200	OK	GET
201	Created	POST
202	Accepted	
203	Non-Authoritative Information	
204	No Content	DELETE PATCH PUT
205	Reset Content	
206	Partial Content	

```
HTTP/1.1 201 Created
Server: Apache
```

Codes de retour HTTP

10 codes de redirections

300	Multiple Choices	choix à faire
301/308	Moved Permanently / Permanent Redirect	
302/307	Found / Temporary Redirect	
303, 304, 305	See Other, Not Modified, Use Proxy	

- destination avec entête `Location: url`

```
GET / HTTP/1.1
Host: www.mines-paristech.fr
```

```
HTTP/1.1 301 Moved Permanently
Server: Apache/2.4.37 (centos) OpenSSL/1.1.1k
Location: https://www.minesparis.psl.eu/
```

Codes de retour HTTP

4xx

30+ codes d'erreurs utilisateur, dont extensions

400	Bad Request	
401/407	Unauthorized	authentification serveur/proxy
402	Payment Required	
403	Forbidden	pb de droits accès
404	Not Found	n'existe pas
405	Method Not Allowed	pour REST
409	Conflict	pour REST
410	Gone	n'existe plus
411-417	erreurs diverses	
451	Unavailable for Legal Reasons	

Codes de retour HTTP

4xx


```
HTTP/1.1 400 BAD REQUEST
Content-Length: 77
Content-Type: text/plain

type error on parameter "quantity" (could not convert string to float: 'two')
```

```
HTTP/1.1 404 NOT FOUND
Server: Apache
Content-Length: 145
Content-Type: text/html; charset=utf-8

<!doctype html>
<html lang=en>
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server.</p></html>
```

Codes de retour HTTP

5xx

20 codes erreurs serveur, dont extensions

- 500 Internal Server Error
- 501 Not Implemented
- 502 Proxy Error

```
                    HTTP/1.1 500 Internal Server Error
                    Server: Microsoft-IIS/10.0

COOK /recettes/pizza HTTP/1.1
Host: popote.fr
                    HTTP/1.1 501 Not Implemented
                    Allow: OPTIONS, HEAD, GET, POST, TRACE
                    Content-Length: 110
                    Content-Type: text/html

                    <html><head><title>501 Not Implemented</titel></head>
                    <body><p>COOK not supported for current URL</body></html>
```

Entêtes HTTP

4 contextes

Requête	uniquement dans les requêtes	`Host User-Agent Referer`
Réponse	uniquement dans les réponses	`Server Location`
Contenu	description des données transportées	`Content-*`
Général	informations restantes...	`Connection`

```
GET /recettes HTTP/1.1
Host: popote.fr
User-Agent: CocotteMinute/1.0
Referer: https://cyrillignac.com/liens.html
```

```
HTTP/1.1 200 OK
Server: Apache
Content-Type: application/json
Content-Length: 29
Connection: close

{"nb":17233,"categories":10}
```

Entêtes

groupes

7 groupes

Connexion	syntaxe, coordination	`Connection Host`
Contenu	données transportées	`Content-* Last-Modified`
Négociation	type, langue, charset, encoding	`Accept-*`
Cache	cachabilité	`Vary ETag`
Condition	vérifications de caches	`If-*`
Authentification	pour permissions	`WWW-Authenticate Authorization`
Divers	...	`From Referer`

```
GET /recettes HTTP/1.1
Host: popote.fr
If-Modified-Since: 2020-07-29
```

```
HTTP/1.1 304 Not Modified
Server: Apache
```

Content-*

Content-Type	type MIME RFC 6838, 4289
	`text/html text/plain image/png image/jpg`
	`audio/aac audio/wav video/mpeg font/ttf`
	`application/msword application/pdf application/json`

Content-Length	taille des données si disponible
Content-Encoding	algorithme de compression `identity gzip deflate`
Content-Language	RFC 5646, ISO 639-1 `fr en ko ja zh de it es pt ru uk he ar`
Content-Location	URL d'un document retourné (négociation)
Content-Range	morceau retourné (206 Partial Content)

Découpage par morceaux `chunked`

`Transfert-Encoding` taille des morceaux en hexa
fin annoncée avec 0

```
HTTP/1.1 200 OK
Server: Apache
Transfer-Encoding: chunked
Content-Type: text/html

28
<html>
<head><title>Popote</title></head>
f55
<body><h1>Popote</h1>
...
</html>
0
```

Approche proactive

`Accept`	types de données acceptées (MIME)	req
`Accept-Encoding`	compression proposée pour `Content-Encoding`	req
`Accept-Language`	langue + variante + poids	req
`Vary`	entêtes qui peuvent changer le résultat (cachabilité)	rép

```
GET /index HTTP/1.1
Host: popote.fr
Accept-Language: es, fr-FR, en-US;q=0.9
Accept: text/html, text/plain
```

```
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 19
Content-Language: en
Content-Location: /index.en.txt
Vary: Accept, Accept-Language

Index of popote.fr
```

HTTP State Management Mechanism — RFC 6265 (2965, 2109)

- HTTP *Stateless* vs identification de sessions
  panier d'achat, auth formulaire, préférences...
- **une** réponse propose un/des cookie(s) `Set-Cookie: nom=valeur`
- **les** requêtes suivantes le/les renvoie(nt) `Cookie: ...`
- client HTTP : accepte/ignore, sauvegarde, efface...
  vol de cookie = vol de session !

```
                    HTTP/1.1 200 OK
                    Set-Cookie: biscuit=64dc1be0c2e0c9e5
                    Set-Cookie: lang=fr
                    Content-Type: ...
GET / HTTP/1.1
Host: popote.fr
Cookie: biscuit=64dc1be0c2e0c9e5; lang=fr
```

Mécanisme de gestion d'état

Cookie

Attributs de `Set-Cookie`

Expires	date d'expiration max	`Expires=date`
Max-Age	durée de rétention max	`Max-Age=durée`
Domain	domaine concerné	`Domain=psl.eu`
Path	chemin concerné	`Path=/api/`
Secure	uniquement TLS, e.g. authentification	

```
HTTP/1.0 200 OK
Set-Cookie: gourmet=1e56c8af8dcd376;
  Expires=Tue, 19 Jan 2038 03:14:08 GMT;
  Domain=popote.fr;
  Path=/;
  Secure
```

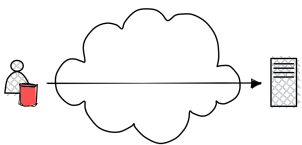
Cache et Proxy HTTP

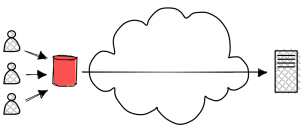
3 types de caches

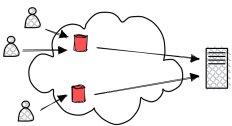
performance moins de rechargements, de latence

moyen garder les réponses précédentes (images, vidéos)

types local au navigateur, partagé, CDN (cache chez fournisseur d'accès)
CDN : Akamai, Cloudflare, Amazon CloudFront, Azure CDN, Fastly...







Cache HTTP

principes

Entêtes

critères cachabilité	`Cache-Control Vary ...`	req/rép
meta information	`Content-Length Last-Modified ETag ...`	rép
requête conditionnelle	`If-Modified-Since If-*Match ...`	req

Entête `Cache-Control`

théorie	aider le client à éviter des requêtes	
pratique	s'assurer que le client refait les requêtes !	req = €(pub)
directives	`private no-cache no-store must-revalidate...`	rép

2 Mécanismes

expiration	document obsolète	`Age Expires Last-Modified`
validation	vérification conditionnelle	`If-*`

Cache HTTP

exemples

https://www.google.com/

```
HTTP/2 200
server: gws
cache-control: private
date: Mon, 26 Feb 2024 14:15:12 GMT
expires: Mon, 26 Feb 2024 14:15:12 GMT
```

https://www.minesparis.psl.eu/

```
HTTP/1.1 200 OK
Server: Apache/2.4.37 (CentOS Stream) OpenSSL/1.1.1k
X-Powered-By: PHP/7.2.24
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
```

https://www.cri.minesparis.psl.eu/

```
HTTP/2 200
server: Apache
last-modified: Thu, 18 Jan 2024 16:41:00 GMT
etag: "4b45-60f3b07495f00"
cache-control: max-age=86400
expires: Tue, 27 Feb 2024 14:30:15 GMT
vary: Accept-Encoding
```

Proxy HTTP

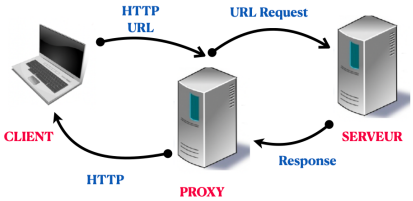
définition

Intermédiaire HTTP entre client et serveur

Squid

explicite HTTP avec URL complète — site
Protocoles : HTTP vers HTTP FTP...

implicite interception ou destinataire des requêtes — CDN, rev



Fonctions d'un proxy HTTP

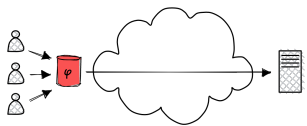
filtrage sécurité, règles, authentification (qui), traces (quoi)

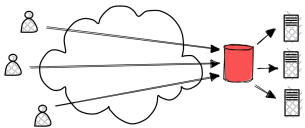
cache conservation de la réponse ?

2 types de proxy

client filtrage des accès, contrôles

serveur reverse proxy
partage de charge entre nœuds
simplification du protocole `https` → `http`





Exemple de requête proxy explicite

- entêtes ajoutées : `Via` `X-Cache`

```
GET https://www.popote.fr/ HTTP/1.1
Host: proxy.minesparis.psl.eu
```

```
HTTP/1.1 200 OK
Server: Apache
Cache-Control: private, max-age=0
Vary: Accept-Encoding
X-Cache: MISS from proxy.minesparis.psl.eu
X-Cache-Lookup: MISS from proxy.minesparis.psl.eu:3128
Via: 1.1 proxy.minesparis.psl.eu (squid/3.5.20)
Content-Type: text/html
Content-Length: 2007
Connection: keep-alive

<html><title>Popote</title><body>...</body></html>
```

Authentification

5 modes

HTTP Basic RFC 2617 puis 7617
identifiant et mdp codé dans les entêtes, uniquement dans TLS !

HTTP Digest RFC 2617
un peu de crypto (hash), **MAIS** très peu utilisé

HTTP Bearer *token* obtenu par un autre moyen (eg Basic)
typiquement pour des *applications* sur API REST

Applicatif écran de login d'une application (eg formulaire HTML)
puis mémorisation dans l'état : cookie, token, session

Certificat client via TLS
très rare, complexité de gestion des certificats...

Authentification HTTP — protocole challenge – response

Basic – Digest – Bearer — 2 échanges

1. requête initiale...
2. réponse 401 *unauthorized* — *challenge(s)*
   - identification du *realm* d'authentification
   - `WWW-Authenticate: Basic realm="Popote World"`
   - `WWW-Authenticate: Digest realm="Popote World", nonce="...", ...`
   - `WWW-Authenticate: Bearer realm="Popote World"`
3. requête... avec entête d'authentification — *response*
   - `Authorization: Basic Y2hlZjpMZXR0dWN1IQ==`
   - `Authorization: Digest ...`
   - `Authorization: Bearer token...` — *obtention ?*
4. réponse 2xx si validé, puis renvoie avec les requêtes suivantes

Authentification Basic *base64(login:mdp)*

- encodage base64 : 3 octets = 4 caractères `a-z A-Z 0-9 + / =`
- gestion du mot de passe par le client, répété à chaque requête

```
GET /recette/crepes/ingredient HTTP/1.1
Host: popote.fr
```

```
HTTP/1.1 401 Unauthorize
WWW-Authenticate: Basic realm="Popote!"
```

```
GET /recette/crepes/ingredient HTTP/1.1
Host: popote.fr
Authorization: Basic Y3Vpc3RvdDpUMG1AdGVzIQ==
```

```
HTTP/1.1 200 OK
Content-Length: 40
Content-Type: application/json

{"oeuf":6,"farine":"500g","lait":"1l"}
```

- **sécurité faible** simple encodage de caractères !
  - `Y2hlZjpMZXR0dWNlIQ==` → `chef:Lettuce!`
  - ok si dans TLS
- intégré dans le protocole HTTP : fenêtre popup gérée par le client

  Web **peu utilisé** car géré par le navigateur...

  API ok pour obtention d'un token
- version avec un *proxy* HTTP
  - réponse 407 *proxy authentication required*
  - entêtes `Proxy-Authenticate` `Proxy-Authorization`
- attention **coût** de vérification des mdp stockés (eg bcrypt) cache ?

```
htpasswd -nbB -C 12 "calvin" "H0bbes!"
# calvin:$2y$12$D14TnHqlqzTL8jZL0d8h.uzs.eb0G85Ja3pvQ7hZXrC2r0oULb1yS
```

170 ms

Authentification HTTP

Digest

- authentification de la requête (URI, contenu...) *h(req, login, mdp)*
  hashages selon contenu, paramètres, login et mdp
- divers inconvénients, peu/pas utilisé **Basic ou Param + TLS**
  pas de sel, compliqué, MITM attack, inutile...

```
GET / HTTP/1.1
Host: popote.fr
Authorization: Digest username="chef",
  real="Popote!", uri="/", qop=auth,
  nonce="cc3e3f4c393bfa5fb9dbf01c1687d341",
  opaque="4a905c98b24b874acb259fd5ba27e284",
  nc=00000001, cnonce="0fd514ae",
  response="f8b1a41a4add6600fe4d302af7219d4c"
```

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="Popote!",
  nonce="cc3e3f4c393bfa5fb9dbf01c1687d341",
  opaque="4a905c98b24b874acb259fd5ba27e284",
  domain="popote.fr",
  qop=auth
```

Authentification HTTP

Bearer

- un token authentifie un utilisateur jusqu'à expiration
  - *realm, user, expiration, signature*
- format custom ou standard JWT
- récupération avec un login/mdp, puis envoi à chaque requête
  - porté par une entête, un cookie, un paramètre...
  - vérification beaucoup moins coûteuse qu'un login/mdp

TLS

```
GET /recette/4quart/ingredient HTTP/1.1
Host: popote.fr
Authorization: Bearer cuisine:cuistot:20380119031407:cc3e3f4c393bfa5f
```

```
HTTP/1.1 200 OK
Content-Length: 59
Content-Type: application/json

{"oeuf":4,"farine":"250g","beurre":"250g","sucre":"250g"}
```

JSON Web Token – RFC 7519 https://jwt.io/

format	entête (json) "." données (json) "." signature (eg hash)
entête	type de token, algorithme de signature...
données	audience (realm), nom (login), expiration
encodage	*base 64 url* : `a-z A-Z 0-9 - _` RFC 4648-5

token

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJhdWQiOiJjdWlzaW5lIiwibmFtZSI6ImN1aXNOb3QiLCJpYXQiOjIxNDc0ODM2NDh9.
UN7TBk421d2SZ2m6XVQ686oobBSklg6gMJY49tnbxqc
```

décodage

```
{ "alg": "HS256", "typ": "JWT" }                                    header
{ "aud": "cuisine", "name": "cuistot", "iat": 2147483647 }          data
0x50ded3064e3695dd926769ba5d543af3aa286c14a4960ea0309638f6d9dbc6a7  signature
```

signature `HMAC-SHA256(header + "." + data, secret)`

- formulaire HTML
- identité associée à la session (eg cookie sécurisé)

```
<form method="POST" action="/login">
  <input type="text" name="login"/>
  <input type="password" name="pass"/>
  <input type="submit" value="Ok"/>
</form>
```

```
POST /login HTTP/1.1
Host: popote.fr
Cookie: sessionid=4a905c98b24b874acb259fd5
Content-Length: 30
Content-Type: application/x-www-form-urlencoded

login=chef&pass=Lettuce!
```

TLS

Transport Layer Security

Part HTTPS vs HTTP, en volume ?

Transport Security Layer

Objectifs — CIA

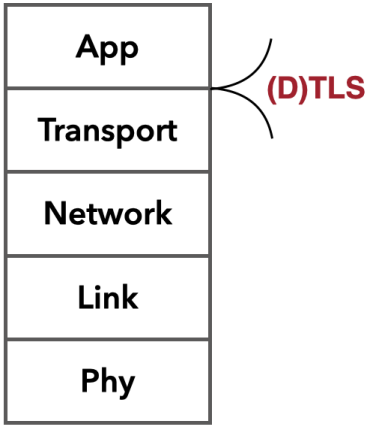
Confidentialité	chiffrement AES
Intégrité	HMAC
Authentification	certificats, clefs publique/privée

Historique

SSL	TCP (1994 - 1996)
TLS	TCP (1999 - 2006) - 2008 - 2018 (1.3)
DTLS	UDP (peu utilisé ? voir QUIC)

Applications — direct/opportuniste

- `https` 443
  `imaps` 993
- `STARTTLS`
  `smtp` `imap`



Protocole

handshake négociation d'une session sécurisée

- **complet** certificat, échange de clefs... coûteux
- **reprise** réutilisation d'une session rapide

record échange de contenu CIA

Certificat

- association clef publique – identité, durée limitée
  signé par une autorité de certification 50-200 €/an
- secret partagé chiffré $X = E(K_A, x)$, déchiffré $x = D(k_A, X)$
- autorités **très** nombreuses + indirectes !
  GeoTrust, DigiCert, Thawte, Staat de Nederland, Hongkong Post...
  Let's Encrypt gratuit, 90 jours

Conclusion

HTTP Extension

WebDAV *Web-based Distributed Authoring and Versioning*

application	partage de dossier *WebDrive...*
méthodes	`PROPFIND PROPPATCH MKCOL COPY MOVE LOCK UNLOCK...`
entêtes	nouveaux et réutilisés `DAV Depth Destination If Lock-Token Overwrite Timeout`
paramètres	passés en XML
status	4 ajoutés *207 Multi-Status, 422 Unprocessable Entity,* *423 Locked, 507 Insufficient Storage*

```
MKCOL /recettes/bretagne HTTP/1.1
Host: dav.popote.fr
```

Accès à des informations via HTTP

JavaScript applications web ou mobiles (react) `fetch axios`

Python scripts... (`urllib`) `requests`

```javascript
// requête HTTP en JS avec axios
const axios = require('axios');
const res = await axios.get('https://api.popote.fr/recettes/quiche');
quiche = res.data.json
console.log(`Quiche : ${quiche.temps}`);
```

```python
# requête HTTP en Python avec requests
import requests
res = requests.get("https://api.popote.fr/recettes/pizza")
pizza = res.json()
print(f"Pizza : {pizza['temps']}")
```

Génération de réponses dynamiques

- interface *bas niveau* entre serveur HTTP et language de programmation
  exemples : CGI (tous langages), WSGI et ASGI (Python)

```python
# fonction WSGI appelée par le serveur
# - dict environ: entêtes HTTP, corps...
# - fonction start: génération de la réponse
def hello_world(environ, start):
    start("200 OK", [("Content-Type", "text/plain")])
    yield b"Hello World!\n"
```

Surcouches *haut niveau* — *frameworks web, micro-frameworks, ORM...*

- C Ulfius
- C++ Oat++
- C# ASP.NET
- Go Gin
- Java Spring
- JS Node.js
- Perl Dancer2
- PHP Lumen
- Python Flask
- Ruby Rails
- Rust Rocket
- Scala Play

Statistiques 2025

Serveurs — https://news.netcraft.com/

nombres	+1.1 G sites, +270 M domaines, +13 M serveurs actifs
logiciel	NGINX 19%, Apache 18%, Cloudflare 16%, Google 9%
version	support HTTP/2 36%, HTTP/3 29% (ou 8% ???) (pour *top sites*)

Clients — https://gs.statcounter.com/

navigateur	chrome 67%, safari 17%, edge 5%, firefox 3%
search	Google 90%, Bing 4%, Yandex 3% ...
type	mobile 63%, desktop 35%, tablet 2%
os	Android 46%, Windows 25%, iOS 18%, OS X 5%, Linux 1%

netcat — connexion TCP/IP

```
netcat -C www.popote.fr 80
netcat -C www.popote.fr 80 < requête.req  # évite le timeout...
```

openssl — connexion TLS

```
openssl s_client -quiet -crlf -connect www.popote.fr:443
```

curl — requête HTTP ou HTTPS

```
curl -i -X GET http://api.popote.fr/version

curl -i -X PATCH -u "cheffe:G0urm@ande" -d prix="3€" \
    https://api.popote.fr/ingredient/celeri

curl -i -X POST -u "marmiton:Gl0ut0n!" -F recette=@./cake.md \
    https://api.popote.fr/recette/upload
```

w3m — navigateur en mode texte

```
w3m https://www.popote.fr/
```