1. Internet : Historique

1.1 Militaire

- Né aux États-Unis, L'origine d'Internetse situe dans le domaine militaire (USA<>URSS).
- Le projet ARPA (Advanced research project agency). Son objectif est l'application des technologies de pointe à la défense. Les activités initiales de l'ARPA portaient sur l'espace, les missiles balistiques et le nucléaire.
- Une partie de la recherche technologique de la défense américaine a était réalisée par des sous-traitant auprès d'institutions de haut niveau. La possibilité de communications entre la base opérationnelle et les sous-traitants à travers des liens directs entre les divers ordinateurs.
- En 1962, l'ARPA lança un programme de recherche: l'Advanced research project agency network (ARPAnet).
- En 1972, le réseau devient une réalité faisant fonctionner une quarantaine de sites. ARPAnet est présenté pour la première fois au public par Robert E. Kahn au cours de la première ICCC (International Conference on Computer Communications) à Washington.
- En 1972, le premier programme de courrier électronique permettant de diffuser et recevoir des messages sur le réseau.
- De 1972-1983, l'évolution des protocoles de communication (la famille de protocoles TCP/IP (Transmission Control Protocol/Internet Protocol)).

1.2 Scientifique et Universitaire

- En 1984 l'introduction du système d'adressage par domaines (DNS).
- En 1984, le gouvernement du Royaume Uni annonce la construction de JANET (Joinacademic network) pour desservir les universités britanniques.
- En 1985, la National science foundation (NSF) décide la construction de NSFnet (basé sur le protocole ARPAnet), un réseau de communication, pour permettre à l'ensemble de la communauté universitaire américaine d'accéder au réseau.
- En 1987, Uunet : la première entreprise commerciale de l'Internet.
- De 1987-1990 : début de l'exploitation commerciale d'Internet.

1.3 Secteur privé et grand publique

- Entre 1990 et 1993, la création du **World Wide Web** : grâce aux chercheur du Centre européen de recherche nucléaire (CERN) situé à Genève et NCSA (National Center for Computing Application) situé aux Etats-Unis.
- Le développement des 2 standards Hypertext markup langage (html) et l'Hypertext tranfer protocol (http) au CERN par Tim Berners-Lee.
- La création du premier navigateur multimédia « Mosaic » au NCSA par Marc Andreessen et Éric Bina.

2. Généralités sur le World Wide Web(WWW)

- Créé en 1989 au CERN par Tim Berners-Lee
- Mettre en ligne de la documentation (initialement technique pour physiciens)
- Principes du Web :
  - Hypertexte: HTML
  - Client/serveur: HTTP
  - Schéma de désignation: URL

A. Hypertexte (HTML)

Texte "normal"

- Organisation linéaire
- Éventuellement renvois sous forme de sommaires, index, notes de bas de page

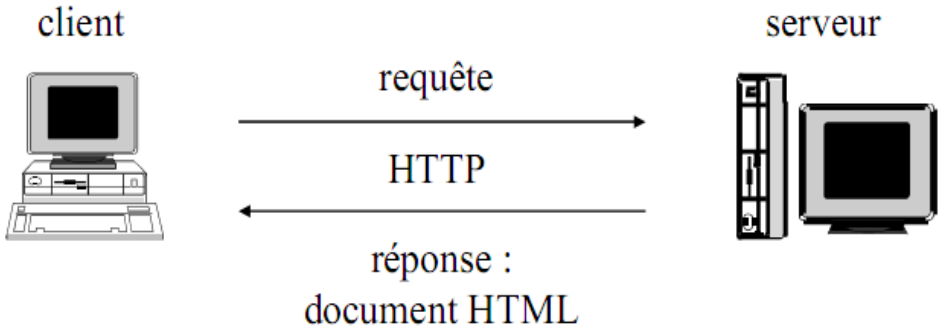
Hypertexte

- Organisation pas forcément linéaire
- Texte enrichi de liens :

  ✓ Renvoi vers une partie d'un autre document
  ✓ Renvoi vers une partie du même document
  ✓ Renvoi vers un document

B. Client/serveur (protocole http)

- Client : le navigateur (Google Chrome, Mozilla Firefox, …)
- Serveur : le serveur Web (Apache, Microsoft IIS, …)
- Le client émet une requête
- Le serveur répond en fournissant le document demandéou un message d'erreur si le document n'existe pas.



C. Schéma de désignation

- Uniform Resource Locator (URL) : permet de désigner une page Web
- Chaque page a un nom unique=>pas d'ambiguïté possible
- Forme générale : *protocole*://*serveur*/*page*
  Par exemplehttp://www.univ-constantine2.dz/indexfr.html
- Organisation hiérarchique possible pour les pages (= hiérarchie fichier disque) :*protocole*://*serveur*/*répertoire*/…/*page* **par exemple** http://www.univ-constantine2.dz/news/index.html

Remarque :

URL, URN et URI : quelle est la différence ?
URI=URL+URN
Nous allons voir + de détails plus tard ! (Style REST)

3. Les applications Web

Une application Web est un ensemble de pages qui interagissent avec les utilisateurs, lesunes avec les autres, ainsi qu'avec les différentes ressources d'un serveur Web, notamment lesbases de données.

3.1. Définition d'une application Web

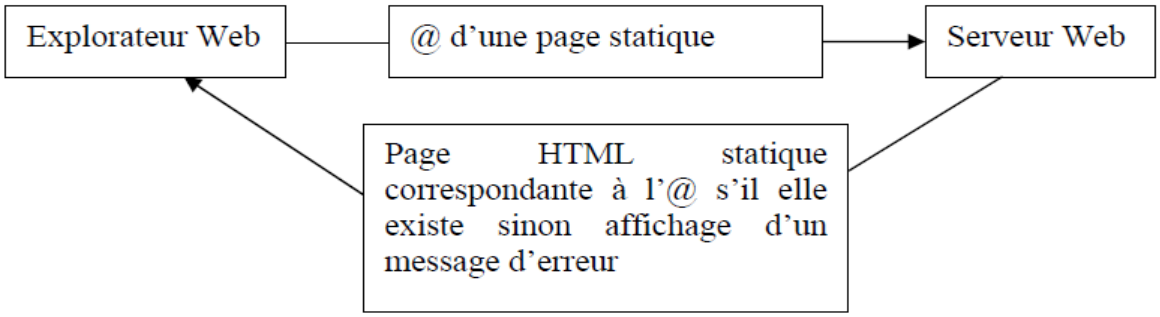
Une application Web est un site Web dont le contenu des pages est partiellement ou entièrement indéterminé. Le contenu final d'une page est déterminé uniquement lorsque l'utilisateur requiert une page depuis le serveur Web. Le contenu final d'une page varie d'une requête à une autre en fonction des actions de l'utilisateur, ce type de page est appelé **page dynamique** parce qu'elle est modifiée par le serveur avant d'être transmise au navigateur qui la sollicite.
A l'inverse, une page statique n'est pas modifiée lorsqu'un visiteur la consulte : le serveur Web transmet la page au navigateur qui la sollicite sans la modifier.

3.2. Type de page

Dans une page Web, il y a deux types de page : statique et dynamique.

3.2.1. Page statique

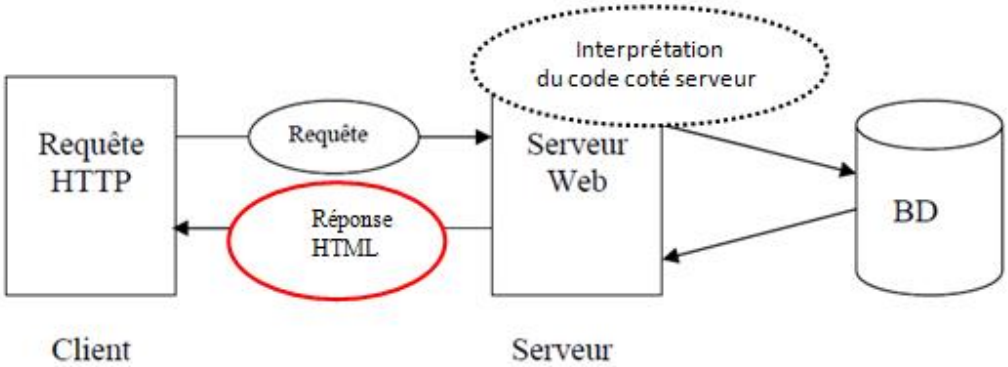
Ce sont des documents HTML préparé avant la demande d'un client. Donc une page statique n'est pas modifiée lorsqu'un visiteur la consulte : le serveur web transmet la page au navigateur qui la sollicite sans la modifier (Dans certains cas, le changement de la méthode http peut engendrer quelques modifications !).
L'exploitation de ces pages se fait comme le montre l'illustration suivante :



Réponse d'un serveur à une demande de page web statique

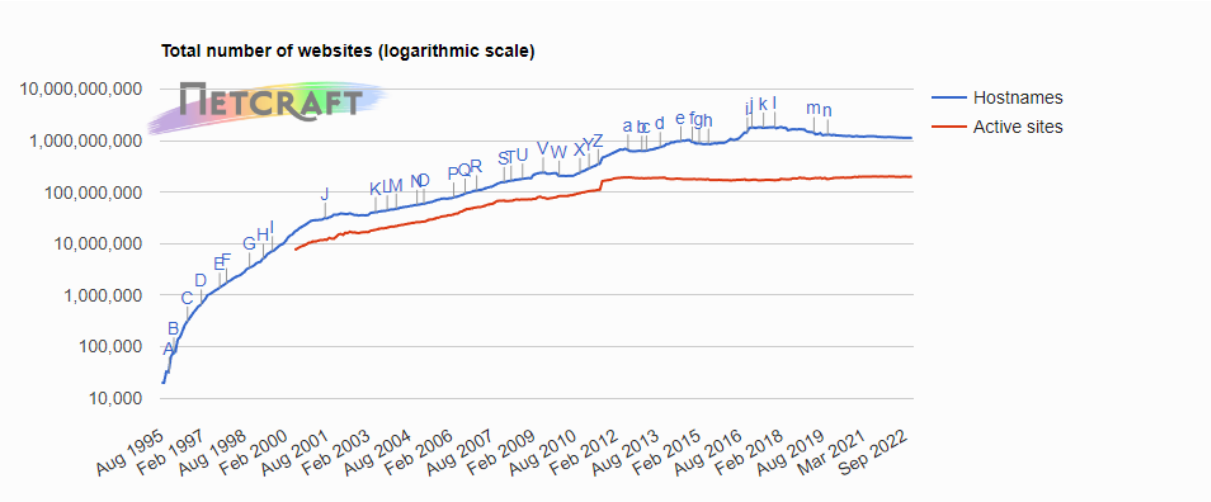
3.2.2. Page dynamique

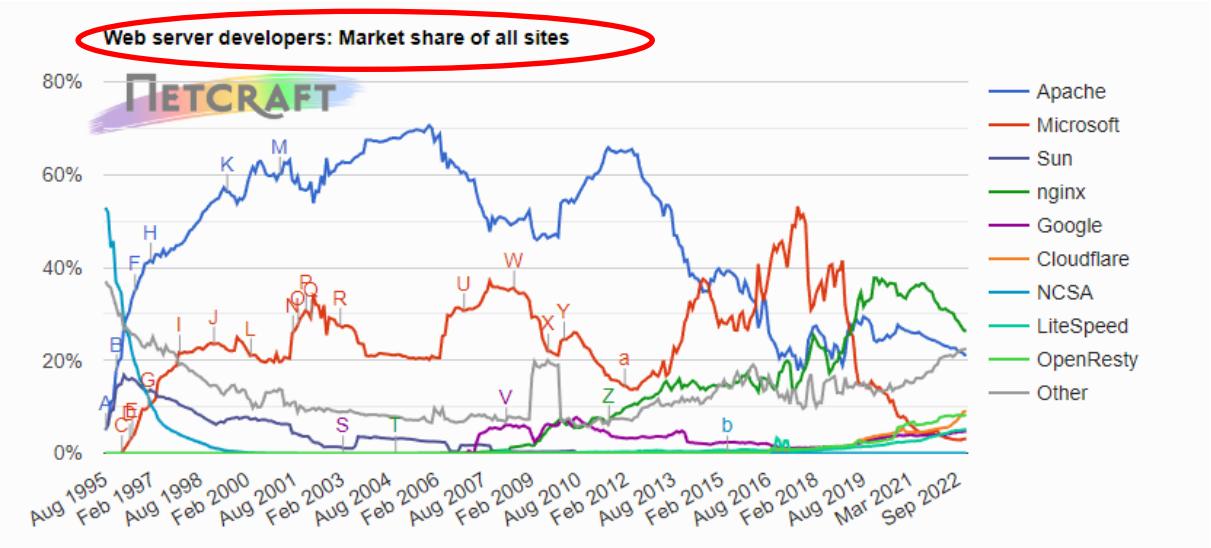
Le contenu final d'une page varie d'une requête à une autre en fonction des actions de l'utilisateur, donc elle sera modifiée par le serveur avant d'être transmise au navigateur qui la sollicite.
De ce fait, s'il est nécessaire de mettre en place une interactivité avec l'utilisateur ou si des demandes individuelles doivent être réalisées pour permettre l'exploitation d'information contenue dans une base de données, la programmation HTML n'est donc plus suffisante.



Reponse d'un serveur à une demande de page Web dynamique

3.3. Serveur Web : Définition, classement et sondage
Un serveur web est un logiciel qui renvoie des pages aux requêtes de navigateur web. Une requête de page est générée lorsqu'un utilisateur clique sur un lien d'une page, ou saisit URL dans le champ @ du navigateur. Les serveurs web les plus courants sont :
Apache, Microsoft Internet Information Server (IIS), NGINX, Google servers…..
La figure suivante présente le classement des serveurs web sur le marché selon le sondage de Netcraft réalisé en Décembre2022/Janvier
2023 (https://news.netcraft.com/archives/category/web-server-survey/):

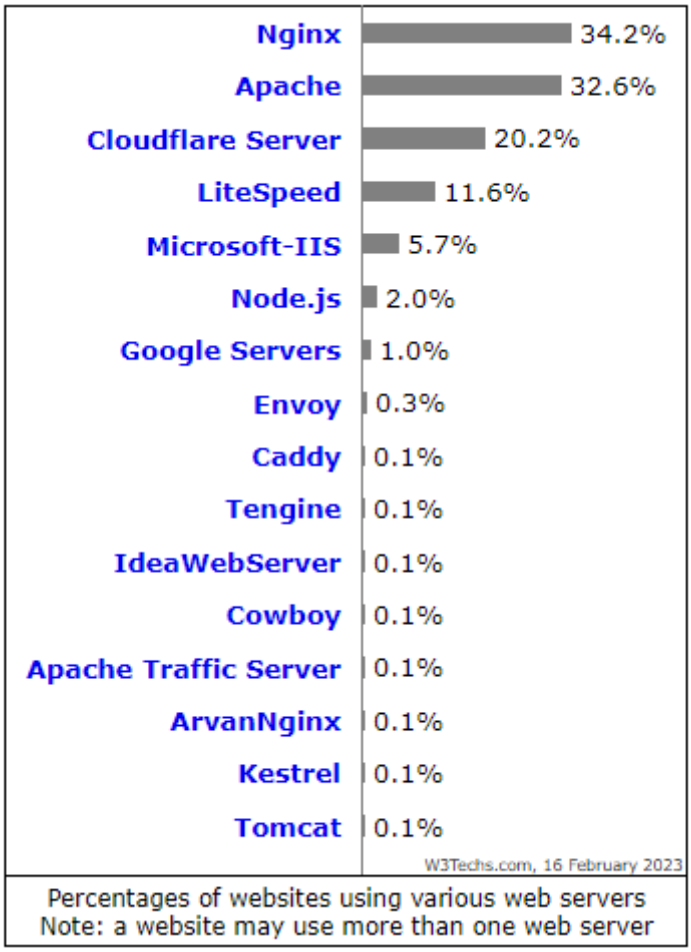




Le tableau suivant résume les résultats de sondage de tous les sites web jusqu'à décembre 2022.

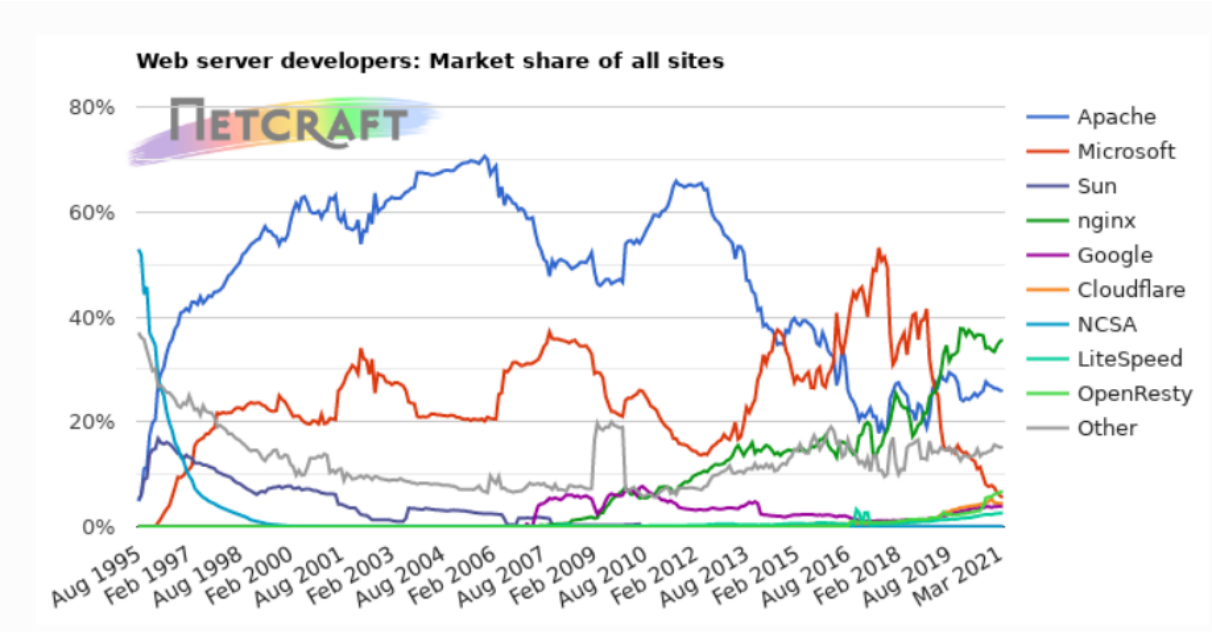
Serveur Web	**Décembre 2022**	**Pourcentage**
nginx	295,366,783	26.25%
Apache	235,541,408	20.93%
Cloudflare	102,829,746	9.14%
OpenResty	92,711,293	8.24%

Dans le même contexte, les statistiques de W3Techs (https://w3techs.com/technologies/overview/web_server) font apparaître une progression du serveur web NGINX pour tous les sites.



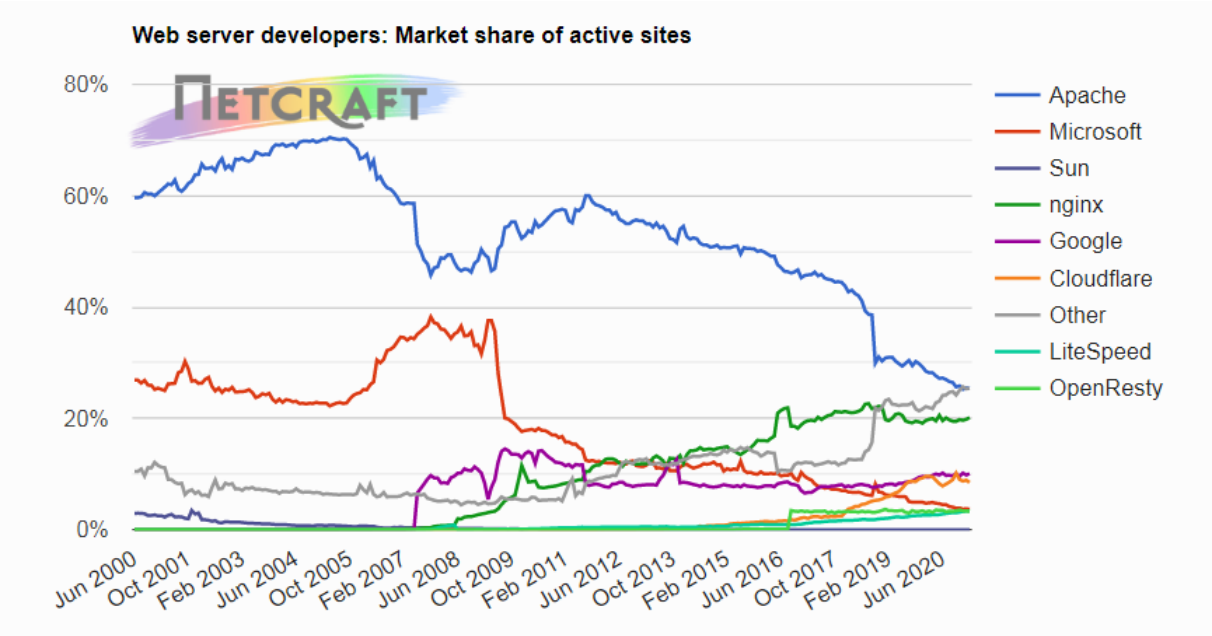
D'autres ancienne statistiques

La figure et le tableau suivants présentent le classement des serveurs web (pour tous les sites) sur le marché selon le sondage de Netcraft publié le 30 Avril 2021 (selon les réponses de **1,212,139,815** sites enregistrés)



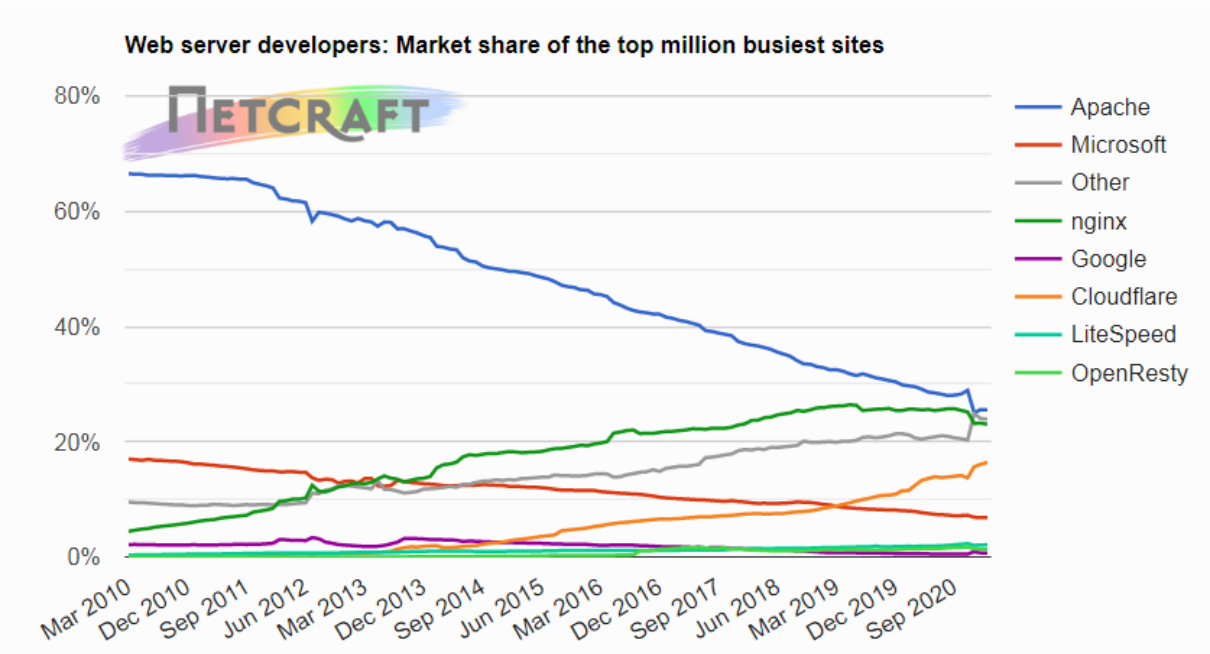
Développeur	Avril 2021	Pourcentage
nginx	432,167,302	35.65%
Apache	313,948,741	25.90%
OpenResty	81,935,391	6.76%
Microsoft	67,182,740	5.54%

Pour les sites actifs, la figure et le tableau suivants présentent le classement des serveurs web sur le marché selon le sondage de Netcraft publié en Mars 2021



Développeur	Mars 2021	Pourcentage
Apache	50,506,614	25.38%
nginx	40,111,872	20.15%
Google	20,086,396	10.09%
Cloudflare	16,932,600	8.51%

Pour le top million sites web, les résultats sont présentés dans la figue et le tableau ci-dessous :



Développeur	Mars 2021	Pourcentage
Apache	255,542	25.55%
nginx	230,565	23.06%
Cloudflare	164,235	16.42%
Microsoft	68,493	6.85%

4. Protocole HTTP (pour la version 1.1)

4.1. Définition

L'acronyme **HTTP** signifie *Hypertext Transfer Protocol* (protocole de transfert hypertexte). Ce protocole définie la communication entre un client (exemple: navigateur) et un serveur sur le World Wide Web (WWW).

Ce protocole inventé par ***Tim-Berner Lee*** au début des années 1990, fonctionne sur le principe "requête-réponse".C'est le protocole le plus utilisé sur Internet. La version 0.9 était uniquement destinée à transférer des données sur Internet (en particulier des pages Web écrites en HTML). La version 1.0 du protocole permet désormais de transférer des messages avec des en-têtes décrivant le contenu du message en utilisant un codage de type MIME*.

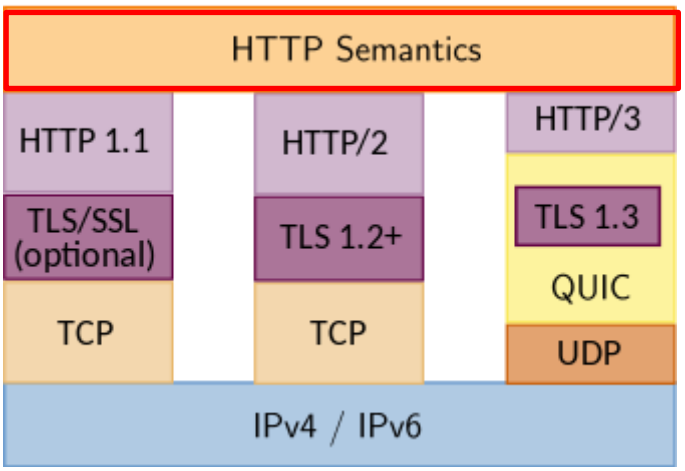
Le but du protocole HTTP est de permettre un transfert de fichiers (essentiellement au format HTML) localisés grâce à une URL entre un navigateur (le client) et un serveur Web.

Dans le modèle OSI, le protocole HTTP est un protocole de la couche application.

> ** Le type MIME (Multipurpose Internet Mail Extensions) est un standard qui a été proposé par les laboratoires Bell Communications en 1991 afin d'étendre les possibilités du courrier électronique (mail), c'est-à-dire de permettre d'insérer des documents (images, sons, texte, ...) dans un courrier.*
> *Depuis, le type MIME est utilisé d'une part pour typer les documents attachés à un courrier mais aussi pour typer les documents transférés par le protocole HTTP. Ainsi lors d'une transaction entre un serveur web et un navigateur internet, le serveur web envoie en premier lieu le type MIME du fichier envoyé au navigateur, afin que ce dernier puisse savoir de quelle manière afficher le document.*
> *Une image GIF a par exemple le type MIME suivant :*
> *Content-type: image/gif*

4.2 Chronologie

- 1991 : naissance de HTTP/0.9
- Mai 1996 : RFC 1945 définissant HTTP/1.0 par T. Berners-Lee, **R. Fielding** et H. Frystyk***(RFC : Request For comments)***
- Janvier 1997 : \href{http://www.ietf.org/rfc/rfc2068.txt}**{RFC2068}** définissant HTTP/1.1
- Protocole transitionnel avant HTTP/2.0 : SPDY proposé par Google (sous-couche pour HTTP)
- Mai 2015 : HTTP/2.0 proposé par la RFC 7540
- 2018/2019 : http 3.0 (évolution du protocole Quic de Google)
- Le 6 juin 2022, l'IETF (Internet Engineering Task Force) a publié HTTP/3 comme Standard Proposé dans la **RFC 9114 (https://datatracker.ietf.org/doc/rfc9114/)**



4.3 Fonctionnement : sémantique HTTP (à partir de HTTP 1.1)

4.3.1 Requête HTTP

Dès que le client est connecté au serveur, il envoie sa requête. Une requête HTTP est un ensemble de lignes envoyé au serveur par le navigateur. Elle comprend :

- **Une ligne de requête**: c'est une ligne précisant le type de document demandé, la méthode qui doit être appliquée, et la version du protocole utilisée. La ligne comprend trois éléments devant être séparés par un espace : {Méthode}{Espace}{Resource}{Espac}{Version}
    - ✓ La méthode (GET, POST,….)
    - ✓ L'URL
    - ✓ La version du protocole utilisé par le client (*HTTP/1.0, HTTP/1.1*)
- **Les champs d'en-tête de la requête**: il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la requête et/ou le client (Navigateur, système d'exploitation, ...). Chacune de ces lignes est composée d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête
- **Le corps de la requête**: c'est un ensemble de lignes optionnelles devant être séparées des lignes précédentes par une ligne vide et permettant par exemple un envoi de données par une commande POST lors de l'envoi de données au serveur par un formulaire.

Une requête HTTP a donc la syntaxe suivante :

méthode ressource HTTP/version (1.0 ou 1.1 par exemple)
option1 : valeur1
…
optionN : valeurN
(ligne vide → fin de l'entête) (<crlf> : retour chariot)
… données éventuelles à transmettre … (Corps de la requête)

a) Méthodes HTTP

Nous allons maintenant voir les méthodes proposées par la version 1.1 du protocole HTTP.

- **GET :** Cette méthode est la plus employée, il s'agit normalement d'une simple requête pour récupérer une ressource (téléchargement d'un document, affichage d'une page web,…). Deux requêtes GET portant sur le même document devraient retourner des réponses sémantiquement identiques (certains en-têtes pouvant influer sur le comportement du serveur, les réponses peuvent ne pas être totalement identiques).
- **HEAD :** cette méthode est utilisée pour obtenir les en-têtes d'une ressource, pas le corps.Elle est similaire et presque équivalente à la méthode GET sauf que le corps de la réponse HTTP n'est pas transmis. Cela permet souvent d'économiser beaucoup de bande passante.
- **POST :** c'est la méthode utilisée pour envoyer des données vers une ressource sur le serveur.Elle est utilisée pour demander un traitement d'informations au serveur (par exemple, envoie d'un formulaire). Les requêtes POST peuvent provoquer des communications entre le serveur et d'autres modules (voire d'autres serveurs), pour effectuer le traitement des données. Les données à traiter sont spécifiées dans le corps de la requête.Le document désigné par la requête via la page est la ressource qui doit traiter les données et générer la réponse (un script PHP par exemple). De ce fait, il est tout à fait probable que deux requêtes POST identiques reçoivent des réponses différentes ou même sémantiquement opposées.

- **PUT et DELETE :** Ces méthodes permettent d'enregistrer et supprimer une ressource sur le serveur.Donc ces méthodespeuvent provoquer des remplacements ou suppressions de fichiers, ce qui incite des failles de sécurités sur un serveur. De ce fait, la plupart des serveurs Web requièrent une configuration spéciale indiquant une ressource ou un document chargé de traiter ces requêtes.

- **OPTIONS :** La méthode OPTIONS permet au client de demander au serveur ces capacités de prise en charge de HTTP. Si la requête précise une ressource, alors la demande concerne celle-ci spécifiquement. Si au contraire la ressource indiquée est *, alors la requête concerne les capacités globales du serveur (Tous les serveurs ne les implémentent pas forcément).

- **TRACE :**La méthode TRACE demande au serveur de renvoyer dans le corps de sa réponse, les en-têtes qu'il a reçu dans sa requête. Le corps de la réponse à cette requête doit être le contenu des en-têtes de la requête elle-même. Une requête TRACE ne comporte pas de corps. La méthode TRACE permet donc véritablement de tracer les intermédiaires entre les 2 bouts du réseau (le client et le serveur final). **NB.**
  1.*Des proxies, des passerelles ou encore des firewalls peuvent se loger entre le client et le serveur (le bout à bout). Ainsi, chaque intermédiaire est susceptible de modifier la requête qu'il reçoit, souvent en ajoutant son identité à l'intérieur.*

  2.*TRACE peut introduire des problèmes de sécurité* !

- **CONNECT :**Cette méthode est censée être utilisée pour demander une utilisation du serveur en tant que proxy.De ce fait, cette méthode ne fonctionne que sur les proxies, et permet de demander à celui-ci de relayer une requête sur certains ports. Si un client utilisant le proxy a besoin de se connecter via HTTPS, il va alors demander au proxy via la méthode CONNECT, de router ses requêtes vers le serveur de destination.

b) En-tête HTTP d'une requête

- **Host :** indique le nom d'hôte du serveur, par exemple www.site.dz. Cet en-tête est obligatoire pour les requêtes HTTP/1.1, c'est elle qui permet d'héberger plusieurs sites Web sur un même serveur.
- **Date :** date de la requête.
- **Cookie :** Ceten-tête permet au client de fournir un cookie au serveur. Sa valeur est simplement le nom et la valeur du cookie, séparés par un égal.Pour en envoyer plusieurs, ils sont séparés par un point-virgule.
- **User-Agent :** permet d'indiquer la signature du programme effectuant la requête. C'est une chaîne donnant des informations sur le client, comme le nom et la version du navigateur, du système d'exploitation (En général, il s'agit du nom complet du programme et de sa version).

D'autres en-têtes sont présentés dans le tableau suivant :

Nom de l'en-tête	Description
Accept	Type de contenu accepté par le navigateur (par exemple *text/html*).
Accept-Charset	Jeu de caractères attendu par le navigateur
Accept-Encoding	Codage de données accepté par le browser
Accept-Language	Langage attendu par le browser (anglais par défaut)
Authorization	Identification du browser auprès du serveur
Content-Encoding	Type de codage du corps de la requête
Content-Language	Type de langage du corps de la requête
Content-Length	Longueur du corps de la requête
Content-Type	Type de contenu du corps de la requête (par exemple *text/html*).
Date	Date de début de transfert des données
Forwarded	Utilisé par les machines intermédiaires entre le browser et le serveur
From	Permet de spécifier l'adresse e-mail du client
From	Permet de spécifier que le document doit être envoyé s'il a été modifié depuis une certaine date
Link	Relation entre deux URL
Orig-URL	URL d'origine de la requête
Referer	URL du lien à partir duquel la requête a été effectuée

c) Le corps de la requête

Le corps de la requête doit (ou peut) être vide pour les requêtes GET, HEAD, DELETE, CONNECT, TRACE et OPTIONS (dans le dernier cas, il est laissé éventuellement rempli pour de futures versions du protocole HTTP).
Dans le cas des requêtes POST et PUT, il correspond aux données à traiter. Il n'y a pas de caractères réservés ou interdits dans le corps de la requête au niveau du protocole HTTP. Cependant, les données doivent être conformes au format attendu par la ressource responsable de leur traitement.

Exemples des requêtes HTTP

Méthode GET

```
GET /en/html/dummy.php?page=dvp&article=http HTTP/1.1
Host: www.developper.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.9.0.8)
Gecko/2009032609 Firefox/3.0.8 (.NET CLR 3.5.30729) FirePHP/0.2.4
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: rememberme=false;VAR=-120

{éventuel contenu de la requête ici}
```

Méthode POST

```
POST /cgi-bin/login.cgi HTTP/1.1
Host: localhost:80
Content-Type: application/x-www-form-urlencoded
Content-Length: 28
Connection: close
(ligne vide → fin de l'entête)
login=myname&password=mypass (données POST)
```

4.3.2 Réponse HTTP

Quand le serveur a fini de recevoir la requête, il la traite et renvoie la réponse appropriée (éventuellement une erreur).Les réponses sont bâties sur un modèle similaire à celui des requêtes. Une réponse HTTP est un ensemble de lignes envoyées au navigateur par le serveur. Elle comprend :

- **Une ligne de statut**: c'est une ligne précisant la version du protocole utilisé et l'état du traitement de la requête à l'aide d'un code et d'un texte explicatif. La ligne comprend trois éléments devant être séparés par un espace :
  {Version}{Espace}{Code Status}{Espace}{Phrase Status}
  - ✓ La version du protocole utilisé

- ✓ Le code de statut
- ✓ La signification du code

- **Les champs d'en-tête de la réponse**: il s'agit d'un ensemble de lignes facultatives permettant de donner des informations supplémentaires sur la réponse et/ou le serveur. Chacune de ces lignes est composée d'un nom qualifiant le type d'en-tête, suivi de deux points (:) et de la valeur de l'en-tête
- **Le corps de la réponse**: il contient le document demandé

Une réponse HTTP a donc la syntaxe suivante:

```
HTTP/1.X statut description (1.0 ou 1.1)
option1 : valeur1
…
optionN : valeurN
 (ligne vide → fin de l'entete)<crlf>
… donnees à transmettre …
```

a) Code de status

Les Codes Status sont regroupés en familles, qui sont au nombre de cinq pour les versions 1.0 et 1.1 de HTTP. La famille est indiquée par le premier chiffre du code (celui des centaines), il va de 1 à 5.Tout code inconnu doit être traité par le client comme le code de base de la famille (*X*00) et être présenté à l'utilisateur.Si la famille est inconnue, le code doit être traité comme un code Erreur Serveur (famille 5*xx*) et être indiqué à l'utilisateur.

- **1xx :** information, réponse intermédiaire du serveur (exemple : 100 continue, 101 switchingprotocols)
- **2xx :** succès de la requête (exemple : 200 OK)
- **3xx :** redirection de ressource
- **4xx** : erreur du client HTTP (exemples : 400 bad request, 403 forbidden, 404 not found)
- **5xx :** erreur au niveau du serveur empêchant la satisfaction de la requête (exemples : 500 internal server error, 501 not implemented)

Le tableau suivant résume l'ensemble des codes possibles :

Code	Message	Description
10x	***Message d'information***	***Ces codes ne sont pas utilisés dans la version 1.0 du protocole. Mais pour la version 1.1, voir le RFC 2616 (Fielding, et al.)***
20x	***Réussite***	***Ces codes indiquent le bon déroulement de la transaction***
200	OK	La requête a été accomplie correctement

201	CREATED	Elle suit une commande POST, elle indique la réussite, le corps du reste du document est sensé indiquer l'URL à laquelle le document nouvellement créé devrait se trouver.
202	ACCEPTED	La requête a été acceptée, mais la procédure qui suit n'a pas été accomplie
203	PARTIAL INFORMATION	Lorsque ce code est reçu en réponse à une commande GET, cela indique que la réponse n'est pas complète.
204	NO RESPONSE	Le serveur a reçu la requête mais il n'y a pas d'information à renvoyer
205	RESET CONTENT	Le serveur indique au navigateur de supprimer le contenu des champs d'un formulaire
206	PARTIAL CONTENT	Il s'agit d'une réponse à une requête comportant l'en-tête *range*. Le serveur doit indiquer l'en-tête *content-Range*
30x	***Redirection***	***Ces codes indiquent que la ressource n'est plus à l'emplacement indiqué***
301	MOVED	Les données demandées ont été transférées à une nouvelle adresse
302	FOUND	Les données demandées sont à une nouvelle URL, mais ont cependant peut-être été déplacées depuis...
303	METHOD	Cela implique que le client doit essayer une nouvelle adresse, en essayant de préférence une autre méthode que GET
304	NOT MODIFIED	Si le client a effectué une commande GET conditionnelle (en demandant si le document a été modifié depuis la dernière fois) et que le document n'a pas été modifié il renvoie ce code.
40x	***Erreur due au client***	***Ces codes indiquent que la requête est incorrecte***
400	BAD REQUEST	La syntaxe de la requête est mal formulée ou est impossible à satisfaire

401	UNAUTHORIZED	Le paramètre du message donne les spécifications des formes d'autorisation acceptables. Le client doit reformuler sa requête avec les bonnes données d'autorisation
402	PAYMENT REQUIRED	Le client doit reformuler sa demande avec les bonnes données de paiement
403	FORBIDDEN	L'accès à la ressource est tout simplement interdit
404	NOT FOUND	Le serveur n'a rien trouvé à l'adresse spécifiée.
50x	***Erreur due au serveur***	***Ces codes indiquent qu'il y a eu une erreur interne du serveur***
500	INTERNAL ERROR	Le serveur a rencontré une condition inattendue qui l'a empêché de donner suite à la demande
501	NOT IMPLEMENTED	Le serveur ne supporte pas le service demandé
502	BAD GATEWAY	Le serveur a reçu une réponse invalide de la part du serveur auquel il essayait d'accéder en agissant comme une passerelle ou un proxy
503	SERVICE UNAVAILABLE	Le serveur ne peut pas vous répondre à l'instant présent, car le trafic est trop dense.
504	GATEWAY TIMEOUT	La réponse du serveur a été trop longue vis-à-vis du temps pendant lequel la passerelle était préparée à l'attendre

b) En-tête de la réponse

Voici les principaux en-têtes utilisés lors des réponses.

- **Content-type et Content-length :** Ces deux en-têtes sont censées indiquer le type MIME et la taille en octets du corps de la réponse. De plus, l'en-tête Content-type peut indiquer le charset utilisé dans le corps de la réponse (dans le cadre d'un type texte). Il est alors indiqué par la mention "charset=" suivie du nom du charset. Il suit le type MIME, en est séparé par un point-virgule.

- **Location :** Redirection vers une nouvelle URL associée au document. A sa réception, le client est généralement censé renvoyer une requête sur l'adresse indiquée. Ce comportement dépend du code status renvoyé avec la réponse.

- **Date:** Date de génération de la réponse.
- **Server:** Spécifie le modèle du serveur http (par exemple Apache/2.2.3)

- **Set-Cookie :** cet en-tête permet d'indiquer au client des cookies à stocker.

- **Connection:** gestion de la connexion (close, keep-alive, . . .). Introduit par HTTP 1.0, suggère un mode de connexion à la partie opposée. Depuis HTTP 1.1, la valeur par défaut est Keep-Alive, qui signifie d'utiliser une connexion persistante.

Le tableau suivant contient les principaux en-têtes utilisés (une description exhaustive se trouve dans la RFC 2616).

Nom de l'en-tête	**Description**
Content-Encoding	Type de codage du corps de la réponse
Content-Language	Type de langage du corps de la réponse
Content-Length	Longueur du corps de la réponse
Content-Type	Type de contenu du corps de la réponse (par exemple *text/html*).
Expires	Date limite de consommation des données
Forwarded	Utilisé par les machines intermédiaires entre le browser et le serveur

Voici quelques exemples des réponses HTTP :

```
HTTP/1.1 200 OK
X-Powered-By: PHP/5.2.6
Set-Cookie: bbsessionhash=5dc22176091fde3ea648f61564e566dd; path=/;
HttpOnly
Cache-Control: private
Pragma: private
Content-Type: text/html; charset=ISO-8859-1
Content-Encoding: gzip
```

```
Content-Length: 58840
Date: Fri, 24 Apr 2009 15:05:08 GMT
Server: Apache
```

```
HTTP/1.0 200 OK
Date : Sat, 15 Jan 2000 14:37:12 GMT Server : Microsoft-IIS/2.0
Content-Type : text/HTML
Content-Length : 1245
Last-Modified : Fri, 14 Jan 2000 08:25:13 GMT
```

Remarque

Les en-têtes HTTP peuvent être regroupés en fonction de leurs contextes:

- En-tête général: En-têtes s'appliquant à la fois aux requêtes et aux réponses, mais sans relation avec les données finalement transmises dans le corps.
- En-tête de requête: En-têtes contenant plus d'informations sur la ressource à extraire ou sur le client lui-même.
- En-tête de réponse: En-têtes avec des informations supplémentaires sur la réponse, telles que son emplacement ou sur le serveur lui-même (nom et version, etc.).
- En-tête d'entité: en-têtes contenant plus d'informations sur le corps de l'entité, comme sa longueur de contenu ou son type MIME.

Les en-têtes peuvent également être regroupés en fonction de la façon dont les serveurs mandataires (serveurs proxy généralement) les gèrent:

- En-têtes de bout en bout : Ces en-têtes doivent être transmis au destinataire final du message (le serveur pour une requête ou le client pour une réponse). Les serveurs intermédiaires doivent retransmettre les en-têtes de bout en bout sans modification et les caches doivent les stocker.
- En-têtes sauts par sauts: Ces en-têtes ne sont significatifs que pour une seule connexion au niveau du transport et ne doivent pas être retransmis par des mandataires ni mis en cache. Ces en-têtes sont les suivants: Connetion, Keep-Alive, Proxy-Authenticate, Proxy-Authorization, TE, Trailer, Transfer-Encoding et Upgrade.

Pour plus de détails:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

IANA (*Internet AssignedNumbersAuthority*) tient également un registre des nouveaux en-têtes de message HTTP proposés.

https://www.iana.org/assignments/message-headers/message-headers.xhtml

5. Protocoles, outils et sites pour tester et connaître le statut des en-têtes HTTP

Ci-dessous des outils utiles pour connaître l'état de l'en-tête HTTP d'une page web (notamment pour connaître le statut http d'une page). Aussi, des outils qui permettent le contrôle de trafic HTTP.

- **telnet :** Telnet est un protocole permettant d'exécuter des commandes saisies au clavier sur une machine distante. L'outil Telnet est une implémentation du protocole Telnet. Il fonctionne dans un environnement client/serveur, c'est-à-dire que la machine distante est configurée en serveur et par conséquent attend qu'une machine lui demande un service.
- **Live HTTP Header** :extension Firefox très pratique permettant d'obtenir différentes informations sur l'en-tête HTTP de la page en cours.
- **Web-Sniffer :**http://web-sniffer.net/

Autres outils pour analyser le trafic entrant et sortant des sites Web etdes extranets, ainsi pour mieux analyser et gérer les différentes requêtes HTTP :

- Fiddler
- Burp
- HTTPRequester (pour Firefox)
- Postman
- …

Références bibliographiques

https://news.netcraft.com/archives/category/web-server-survey/

https://www.**developpez.com**

https://**openclassrooms**.com/

www.**w3schools**.com

RFC 2616 : https://www.w3.org/Protocols/rfc2616/rfc2616.html