

# **Devoir – Partie Théorique : Protocole HTTP**

# 1. Introduction au protocole HTTP

Le protocole HTTP (HyperText Transfer Protocol) est le langage utilisé pour la communication entre un client (comme un navigateur web) et un serveur sur Internet. C'est grâce à lui que l'on peut consulter des pages web, envoyer des formulaires ou encore télécharger des fichiers.

Ce protocole a été inventé au début des années 1990 par Tim Berners-Lee, en même temps que le HTML et le premier navigateur web. Depuis, plusieurs versions ont vu le jour :

Il a donc créé la version HTTP/0.9 : qui était très simple, ne servait qu'à transférer du texte.

En 1996, la version HTTP/1.0 vu le jour, avec l'ajout des headers (métadonnées), du code de statut.

En 1999, la version HTTP/1.1 prit sa place qui est devenu la version la plus utilisée pendant plusieurs années.

En 2015 la version HTTP/2 est arrivée. Cette version est plus rapide avec une meilleure gestion des connexions.

La version HTTP/3 est la plus récente, cette version est encore plus rapide et plus sécurisée, basée sur un protocole appelé QUIC (utilise UDP au lieu de TCP).

HTTP est un protocole sans état (stateless), ce qui signifie que le serveur ne garde pas en mémoire les requêtes précédentes. Chaque requête est indépendante, le serveur ne se souvient pas de l'utilisateur entre deux requêtes. Pour garder une trace d'un utilisateur d'une page à l'autre, on utilise des cookies.

Enfin, la version HTTPS (le "S" signifie secure) ajoute une couche de sécurité grâce au chiffrement TLS/SSL.

## 2. Interactions entre client et serveur

HTTP fonctionne selon un modèle client-serveur.

Le rôle du client (navigateur, application...) son action principale est l'envoi d'une requête http par exemple charger une page, ou l'envoi d'un formulaire, il utilise un navigateur pour le faire.

Le rôle du serveur, lui reçoit cette requête, la traite et renvoie une réponse HTTP.

Chaque action sur un site web (cliquer sur un lien, soumettre un formulaire, charger une image) déclenche en réalité une nouvelle requête HTTP.

Une fois la réponse reçue, la connexion est généralement fermée, et une nouvelle sera ouverte pour la prochaine ressource demandée.

### 3. Structure d'une requête HTTP

Une requête HTTP se compose de trois parties principales :

#### La ligne de requête

Contient la méthode HTTP, le chemin de la ressource et la version du protocole.

#### Les headers (en-têtes)

Ce sont des lignes d'informations supplémentaires envoyées au serveur avec la requête, qui servent à donner des détails au serveur.

#### Le corps (body)

C'est la partie facultative de la requête qui contient les données envoyées (par exemple via un formulaire).

Les **paramètres d'URL** (ou *query parameters*) sont parfois ajoutés directement à l'adresse :

GET /search?q=cyber&page=2

### 4. Structure d'une URL

Une **URL** (Uniform Resource Locator) est l'adresse complète d'une ressource sur Internet.

Voici une structure détaillée d'une URL

Élément	Description
https	protocole utilisé
www.example.com	nom de domaine du serveur
:8080	port
/articles/page.html	chemin de la ressource
?lang=fr	paramètres envoyés

Une URL, c'est un peu comme une adresse postale : elle indique où aller et quoi chercher.

## 5. Fonctionnement d'un formulaire HTML

Un formulaire HTML permet à l'utilisateur d'envoyer des données au serveur. Il est défini avec la balise `<form>` et contient des champs tels que `<input>`, `<textarea>`, `<select>`, etc..

Voici un exemple de structure de formulaire:

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

L'attribut **action** indique l'URL où les données seront envoyées.

L'attribut **method** indique la méthode HTTP utilisée :

Avec la méthode GET, les données du formulaire sont visibles dans l'URL.

L'avantage est que c'est plus simple et plus rapide mais l'inconvénient, c'est que les données sont visible dans l'URL et donc ne sont pas sécurisé.

Cette méthode est utilisé pour une recherche google.

Avec la méthode POST, les données sont envoyées dans le corps de la requête http, et ne sont pas visible dans l'URL.

L'avantage, c'est que c'est plus sécurisé et ça permet d'envoyer plus de données mais l'inconvénient est que les données ne sont pas visible dans la barre d'adresse.

Cette méthode est souvent utilisé pour les connexions, les inscriptions, etc ...

Quand on clique sur "Envoyer", le navigateur récupère les données du formulaire, il crée la requête HTTP et l'envoie au serveur, qui lui traite les données, puis qui renvoie ensuite une réponse http (page de confirmation, message d'erreur, etc.).

## 6. Structure d'une réponse HTTP

La réponse http, c'est ce que le serveur renvoie après avoir traité la requête.

Elle contient :

Une **ligne de statut** : indique le résultat

Des **headers** : précisent le type, la taille, etc.

Un **body** : contient le contenu de la page ou la donnée demandée.

Voici un exemple d'une structure d'une réponse:

HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 1256

<html>

<body>Bienvenue sur mon site !</body>

</html>

**HTTP/1.1 200 OK**, c'est la ligne de statut

HTTP/1.1, c'est la version du protocole

200 OK , c'est le code de statut qui veut dire succès

**Headers** : donnent des informations supplémentaires

Content-Type: text/html, c'est le type du contenu renvoyé

Content-Length: 1256, c'est la taille du contenu

Server: Apache, c'est le type de serveur utilisé

**Body** : c'est le contenu réel de la page à afficher (HTML pour l'exemple)

## 7. Les headers les plus importants

Header	Description
<b>Content-Type</b>	Type de contenu envoyé
<b>Content-Length</b>	Taille du corps
<b>Set-Cookie</b>	Envoie un cookie au client
<b>Cookie</b>	Renvoie les cookies du client
<b>Authorization</b>	Informations d'authentification
<b>WWW-Authenticate</b>	Demande d'identification
<b>Cache-Control</b>	Gestion du cache
<b>Location</b>	Redirection vers une autre URL
<b>Host</b>	Nom du site demandé
<b>User-Agent</b>	Type de navigateur utilisé
<b>Accept</b>	Formats acceptés par le client
<b>Strict-Transport-Security</b>	Forcer HTTPS
<b>X-Content-Type-Options</b>	Protection contre le "sniffing"
<b>X-Frame-Options</b>	Empêche l'intégration en iframe
<b>Allow</b>	Méthodes autorisées

## 8. Les statuts codes les plus importants

Chaque réponse http commence par une ligne de statut qui contient http/version code message comme par exemple

HTTP/1.1 200 OK

HTTP/1.1 404 Not Found

Voici les cinq familles de code que l'on peut avoir :

Famille	Type de réponse
1xx	Information
2xx	Succès
3xx	Redirection
4xx	Erreur du client
5xx	Erreur du serveur

Voici la liste des codes les plus important et leurs descriptions :

Code	Signification	Description
200	OK	Requête réussie
201	Created	Ressource créée
204	No Content	Succès sans contenu
301	Moved Permanently	Redirection permanente
302	Found	Redirection temporaire
400	Bad Request	Mauvaise requête
401	Unauthorized	Authentification requise
403	Forbidden	Accès refusé
404	Not Found	Page inexistante
405	Method Not Allowed	Méthode non autorisée
413	Payload Too Large	Corps trop volumineux
500	Internal Server Error	Erreur interne du serveur
503	Service Unavailable	Serveur indisponible

## 9. Les verbes HTTP les plus importants

Les verbes http précisent ce que le client veut faire avec la ressource sur le serveur.

Verbe	Rôle
<b>GET</b>	Lire une ressource
<b>POST</b>	Envoyer des données
<b>PUT</b>	Remplacer une ressource existante
<b>PATCH</b>	Modifier partiellement une ressource
<b>DELETE</b>	Supprimer une ressource
<b>OPTIONS</b>	Lister les méthodes autorisées
<b>HEAD</b>	Obtenir les en-têtes sans le contenu



## Conclusion

Le protocole HTTP est la base de toutes les communications sur le web.

Il permet aux clients (navigateurs) et aux serveurs d'échanger des informations selon un format clair et universel.

Comprendre la structure des requêtes, des réponses, des codes et des headers est essentiel, surtout en cybersécurité, car c'est à ce niveau que l'on peut détecter, analyser ou même prévenir certaines attaques (comme l'injection, la manipulation de cookies, ou les redirections non sécurisées).