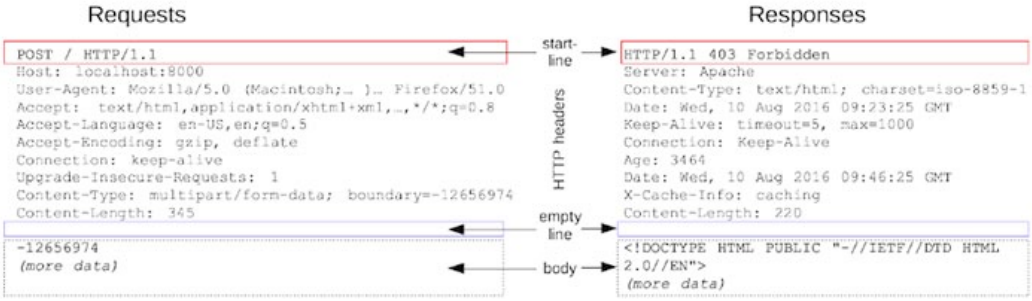
Interactions entre l'homme et la machine sur le WEB

Synthèse : communication client/serveur

Les échanges client-serveur : comment ça fonctionne ?

1. **Le rôle du client :**
    - Le client est un ordinateur ou un appareil (comme ton smartphone ou ton navigateur web) qui fait une **demande**. Par exemple, quand tu tapes une URL, ton navigateur (le client) demande une page web.
    -
2. **Le rôle du serveur :**
    - Le serveur est une machine puissante ou un programme qui stocke des ressources (pages web, images, vidéos, etc.). Il répond aux demandes du client en envoyant les données demandées.
    -
3. **Le protocole de communication :**
    - Les échanges utilisent un langage commun appelé **protocole HTTP/HTTPS**.
      Exemple :
        - Le client dit : "Salut, ça va ? "
        - Le serveur répond :"oui et toi ?"
        - Le client dit :" oui ! tu disponible pour m'envoyer une ressource ?"
        - le serveur répond : "oui, demande-moi !"
        - Le client dit : *"Je veux la page d'accueil de ce site"*.
        - Le serveur répond : *"Voici le contenu demandé !"Exemple de requête et de réponse :*



Exemple d'en-tête d'un requête client :

- **GET** : méthode utilisée pour demander une ressource.
- **/index.html** : le fichier ou la page demandée.
- **Host** : le serveur où se trouve la ressource
- **User-Agent** : informations sur le client (par exemple, un navigateur).
- **Accept** : type de contenu attendu par le client (ici du texte HTML).

Exemple d'une en-tête réponse serveur :

- **200 OK** : indique que tout s'est bien passé.
- **Content-Type** : type de contenu (ici, une page HTML).
- **Content-Length** : taille de la réponse (en octets).

Faire la différence entre la méthode GET et la méthode POST pour envoyer les données d'un formulaire.

Il existe deux méthodes pour transmettre les informations saisies dans un formulaire par un utilisateur.

La méthode GET est la méthode la plus utilisée, elle passe les réponses de l'utilisateur par l'URL.
Cette méthode est donc limitée par la **taille limite d'une URL** et les données sensibles sont **accessibles à tous**, ce qui n'est pas sécurisé.

À la soumission du formulaire, on est redirigé vers la page suivante, et le mot de passe est visible dans l'URL, ce qui ne permet pas d'avoir un mot de passe confidentiel.

La méthode POST possède des caractéristiques différentes : les paramètres, c'est-à-dire les données saisies par l'utilisateur, sont passés non pas dans l'URL, mais dans la requête elle-même.

On précise **method="post"** dans la balise HTML pour l'utiliser.
Voici un exemple de formulaire réalisé avec la méthode POST, où l'utilisateur doit rentrer son nom et son prénom avant de cliquer sur « Envoyer ».

« Envoyer ».

<form method="post" action="mapage.fr">	Création d'un formulaire utilisant la méthode POST. Les données seront envoyée à mapage.fr.
<input type="text" name="nom">	Création d'un champ de saisie de texte, appelé « nom ».

Comme les données du formulaire sont envoyées dans la requête elle-même, il n'y a **pas de limite de taille** et les données ne sont **pas visibles** par les autres. Ces données ne sont cependant pas protégées (elles peuvent être interceptées et lues par une tierce personne) si le protocole utilisé n'est pas du type HTTPS.