SÉANCE 1

Introduction au Développement Backend

Architecture Client / Serveur

Comment fonctionne le web ?

Le Client

C'est l'initiateur (navigateur web, application mobile). Il envoie une **requête**

pour demander une ressource. Il ne peut rien faire tant qu'il n'a pas reçu de réponse.

Le Serveur

Machine puissante connectée 24/7. Elle écoute, traite la demande (calculs, base de données) et renvoie une **réponse** .

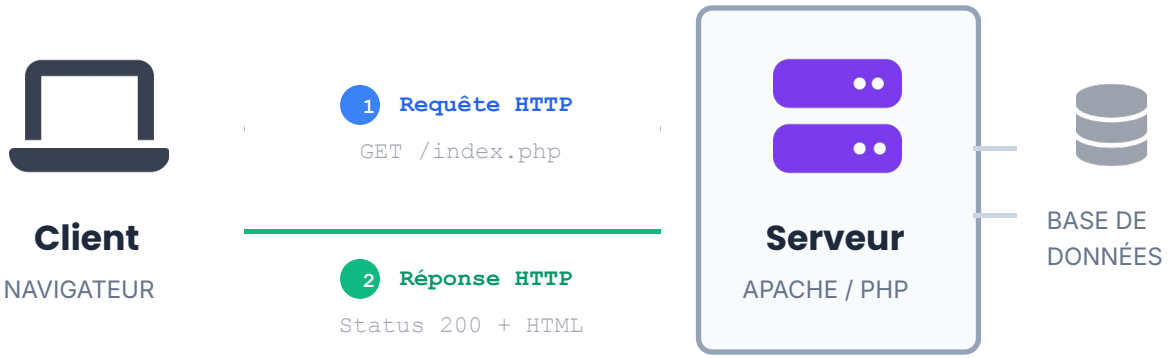
Cycle de vie

Le web est basé sur des échanges atomiques :

1 Requête ➔ Traitement ➔ 1 Réponse.

C'est un protocole *sans état* (Stateless).

FLUX DE DONNÉES



Analogie du Restaurant

Le Client est le client à table. Le Serveur est le serveur/cuisinier. La Requête est la commande. La Réponse est le plat servi.

Protocole HTTP / HTTPS

HyperText Transfer Protocol

Anatomie d'une requête

URL : L'adresse de la ressource.

Headers : Métadonnées (Navigateur, Langue, Cookie).

Body : Données envoyées (facultatif, pour POST).

Codes de Statut (Réponse)

Le serveur indique si ça s'est bien passé :

200 OK: Succès

404: Non trouvé

301: Redirection

500: Erreur Serveur

HTTPS & Sécurité

S pour Secure. Utilise TLS/SSL pour chiffrer la communication. Sans ça, tout passe en clair sur le réseau (mots de passe, CB...).

STRUCTURE DES PAQUETS

↑ REQUÊTE (CLIENT) — HTTP/1.1

En-têtes

```
GET /cours/backend.html HTTP/1.1
Host: www.monsite.com
User-Agent: Mozilla/5.0...
Accept: text/html
... (ligne vide) ...
<pas de corps pour un GET>
```

↓

↓ RÉPONSE (SERVEUR) — Chiffré TLS

Headers

Body (HTML)

```
HTTP/1.1 200 OK
Content-Type: text/html
Server: Apache
... (ligne vide) ...
<html>
<h1>Bienvenue</h1>
</html>
```

Outils pour voir ça : DevTools (F12) > Onglet Réseau (Network)

Méthodes HTTP : GET vs POST

GET

Lecture seule : Conçu pour récupérer des données sans modifier l'état du serveur.

Visibilité : Les données sont visibles dans l'URL (query parameters).

Historique : Peut être mis en cache, ajouté aux favoris et reste dans l'historique.

Limité : Longueur d'URL limitée (env. 2000 caractères).

APPARENCE URL

```
GET example.com/search.php ?q=velo&page=2
```

VS

POST

Action / Écriture : Conçu pour envoyer, créer ou modifier des données.

Discrétion : Les données sont dans le corps (body) de la requête, invisibles dans l'URL.

Volatile : Ne peut pas être mis en cache ou ajouté aux favoris facilement.

Illimité : Permet d'envoyer de gros volumes (textes longs, fichiers).

REQUEST PAYLOAD

```
POST /login.php HTTP/1.1
...headers...

username=admin&password=1234
```

Sites Statiques vs Dynamiques

Statique

CLIENT SIDE

Pré-existant : Les fichiers HTML/CSS/JS sont stockés tels quels sur le disque du serveur.

Vitesse : Chargement très rapide (pas de traitement backend nécessaire).

Sécurité : Surface d'attaque réduite (pas de base de données à pirater directement).

Usage : Blogs simples, documentation, portfolios.

STRUCTURE SERVEUR (FICHIERS)

```
/var/www/html
  index.html   (Contenu fixe)
  about.html
  style.css
```

VS

Dynamique

SERVER SIDE

À la volée : Le serveur "construit" le code HTML à chaque requête (PHP, Node, Python).

Personnalisable : Contenu unique par utilisateur (ex: panier, profil, tableau de bord).

Connecté : Lit et écrit des données dans une Base de Données (MySQL, PGSQL).

Usage : E-commerce, Réseaux sociaux, Applications métier.

LOGIQUE BACKEND (PHP)

```php
<?php
if($user->is_logged_in ) {
echo"<h1>Bienvenue, ".$user->name ."</h1>";
}else{
echo"<a href='/login'>Connectez-vous</a>";
}
?>
```

PHP Introduction

Historique & Usage

Créé en 1994, PHP (Hypertext Preprocessor) propulse encore près de 77% du web (WordPress, Laravel, Symfony).

C'est un langage conçu spécifiquement pour être intégré facilement dans du HTML.

Dynamisme

Contrairement au HTML statique, PHP est exécuté sur le serveur avant que la page ne soit envoyée au navigateur. Le client ne voit jamais le code PHP, seulement le HTML résultant.

index.php

```php
<!DOCTYPE html>
<html>
<body>
<h1>Mon Site</h1>

<?php
$nom="Étudiant";
$heure=date("H");

if($heure<18){
echo"<p>Bonjour, $nom !</p>";
}else{
echo"<p>Bonsoir, $nom !</p>";
}
?>

</body>
</html>
```

Exécuté sur le serveur, invisible pour le client

SORTIE HTML (CLIENT) `<p>Bonjour, Étudiant !</p>`