**BACKEND – 1ère partie**

**10/07/2023 – CV du lundi**

Une image contenant texte, ordinateur, capture d’écran, conception

Description générée automatiquement

* Début du travail **backend** : présentation :
  + Le **protocole HTTP** (requête HTTP, traitement de la requête, réponse HTTP) ;
  + Historique fonctionnement des réseaux télécoms ;
  + Fonctionnement du réseau internet :
    - présentation du datagram : paquets de données numériques, envoyés par des chemins qui peuvent être différents, et plusieurs paquets pouvant être envoyés en même temps
    - adresse IP *(IPv4, IPv6)*
    - protocole TCP/IP
    - nom de domaine et serveurs DNS *(Domain Name Server – sorte d’annuaire)* + fonctionnement : client -> box -> serveur DNS -> box -> client -> serveur contenant le site web
    - et quand client connecté au serveur contenant le site internet : **requête** HTTP faite par le client / **réponse** HTTP faite par le serveur :
      * le serveur (« serveur web ») a besoin d’un programme qui gère le protocole http (nous, on va travailler avec un programme open-source : APACHE, qui va gérer le protocole HTTP)
* *(Essai sur example.com :*
  + *Essai sur le port 80, qui est le port par défaut de tous les sites internet (taper* [*https://www.example.com*](https://www.example.com)*, c’est la même chose que de taper* [*https://www.example.com:80*](https://www.example.com:80) *)*
* Présentation et installation du package **XAMPP** : permet d’avoir le **serveur APACHE** + un plugin qui permet d’interpréter du PHP + un serveur de base de données.
* Tous les ordinateurs ont leur propre IP, qui s’appelle localhost ou qui a l’IP 127.0.0.1 :
  + C’est l’adresse de notre serveur local à nous. Quand on tape ça, c’est comme si on allait sur un serveur à l’IP 127.0.0.1 – mais en l’occurrence, ce serveur se trouve sur notre PC.
  + (-> donc on peut faire en même temps du frontend et du backend sur notre machine : le frontend, c’est le navigateur et le backend, c’est un serveur (qui se trouve sur notre machine, mais c’est comme si elle se trouvait à distance). *[Note : quand on va développer, la partie backend ne sera pas sur un serveur distant, mais sur notre PC.]*
  + [Fin de l’enregistrement : il manque environ 1 h de cours, au cours de laquelle on a vu la présentation d’APACHE et du PHP (c’est APACHE avec le module PHP qui va interpréter le PHP ; sinon, sans cette interprétation, le PHP ne fait que renvoyer du texte].
* Utilisation de APACHE :
  + Installer XAMPP (installer APACHE et MySQL, mais pas Mercury, ni Tomcat).
  + Dans le répertoire où a été installé XAMPP, chercher le répertoire htdocs et y créer un répertoire où on mettra les applications que l’on va créer (par exemple, un dossier myapps).
  + Dans le XAMPP Control Panel, démarrer APACHE et MySQL.
  + Dans la barre d’adresse du navigateur, taper localhost/myapps/monfichier.php (ou 127.0.0.1/myapps/monfichier.php).  *(Mais ne pas ouvrir le fichier en double cliquant dessus, car dans ce cas, il s’ouvrirait dans le navigateur et on serait côté client – ce qu’on ne veut pas ici, on veut être côté serveur. Côté client, le PHP ne serait pas interprété : on aurait juste du texte).*
* Vu en PHP :
  + exo01.php : comparaison PHP / Javascript : se ressemble beaucoup :
    - possibilité de passer des **balises** HTML
    - déclaration de variables : $a = 5 ;
    - structure :
      * **< ?php ……………….. ?>**
      * **$** avant les variables
      * **point-virgule** en fin d’instruction
    - opérations : idem
    - comparaison : idem
    - peuvent être utilisés : AND ou && ; OR ou ||
    - afficher un résultat :
      * **print\_r** : fonction PHP qui affiche la valeur du contenu des variables, des tableaux, des objets
      * **echo**
      * **var\_dump**
    - boucles for
    - if, if else, switch
    - **concaténer** : avec le point (.)
    - **tableau**: cf. tablo.php :
      * création d’un tableau : $tablo = **array**(« texte 1 », 2, 3, 4) ;
      * longueur d’un tableau : $taille = **count**($tablo)
      * rechercher dans un tableau
      * foreach
  + dans le dossier « inclusions » :
    - un des intérêts du PHP est de permettre de **découper des pages**. On peut faire un petit programme pour l’en-tête, un autre pour le pied de page, etc. 🡪 permet de gagner du temps quand il y a beaucoup de pages, plus facile et plus rapide à maintenir ou à corriger.
    - pour ça, utilisation de require (ou mieux : require\_once). Exemple : < ?php require ‘entete.php’ ; ?>
    - conseil de JAM : se créer un répertoire avec ses propres fonctions, auxquelles on fera appel
* Exo à faire : laredoute.php :
  + à partir du tableau donné par JAM, faire la 1ère page qui n’affiche que le menu ( ?)