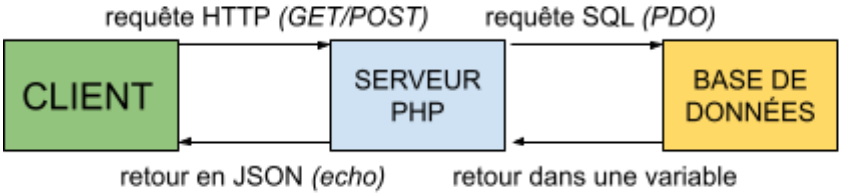
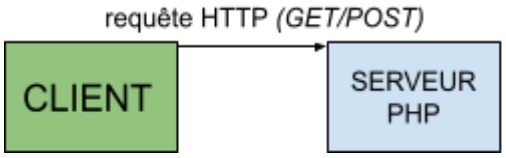
Protocole Client-Serveur Onion Wizz

Module Amélioration Participative, PATINIER Quentin et MOALIC Baptiste

Dans le cadre de notre projet PACT Onion Wizz, nous mettons en place un système d'amélioration participative : un utilisateur pourra partager des Points d'intérêts, des trajets avec les autres utilisateurs depuis l'application.
Cela implique la création d'un protocole de communication client-serveur.
Voici un schéma représentant les outils de communications que nous emploierons :



Précisons chacune des interactions :

- 

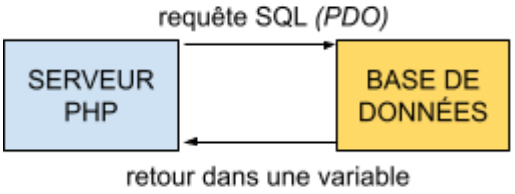
Pour récupérer les données entrées par le client, nous passerons ces valeurs en paramètre d'une URL de la façon suivante pour la méthode GET par exemple :

```
https://<URL_de_base>/fichier.php?attribut1=valeur1&attribut2=
valeur2&...
```

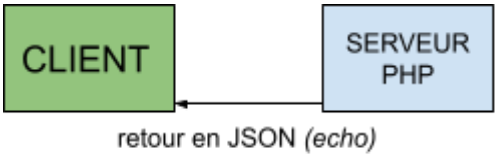
Le serveur stockera ainsi ces données dans des tableaux $_GET['attribut1'] etc…

D'autres méthodes HTTP existent, dont le fonctionnement diffère. La méthode POST notamment, que nous utiliserons car elle permet une modification de l'état interne du serveur, ce que nous cherchons dans notre cas : l'écriture dans la base de données. Son fonctionnement diffère par le fait que les éléments de la requête se situent dans le corps de la requête, ce qui permet une transmission des données plus sécurisée, et permet de transmettre des données plus volumineuses.
Nous utiliserons également la méthode DELETE.

Notons que nous devrons vérifier les données qui seront envoyées par ces requêtes HTTP, afin d'éviter les injections SQL ayant pour but d'accéder à nos données à l'aide de requêtes fabriquées

- 

Supposons que nous ayons reçu les éléments à chercher dans la base de données depuis l'URL. Il faut ensuite faire une requête à la base de données en SQL pour recevoir les résultats de la requête. Les requêtes SQL fréquemment utilisées sont notamment SELECT … FROM, WHERE, GROUP BY, HAVING, LIMIT…
Pour pouvoir adresser une requête SQL et manipuler la réponse dans le code PHP, nous utilisons l'extension PDO proposée par PHP. Cette dernière nous permettra d'accéder à la base de données, passer les requêtes, et les récupérer de façon ordonnée dans notre code.

- 

La requête HTTP ayant été faite par le client attend une réponse. On l'enverra avec la structure echo de PHP. La réponse peut contenir plusieurs éléments, il est donc important de l'ordonner. Nous construisons ainsi une réponse sous format JSON. On utilise la fonction json_encode() de PHP, avec comme paramètre par exemple l'objet PHP que l'on veut renvoyer. Par exemple

```
$result->param1=result1;
$result->param2=result2;
// etc…
echo json_encode($result);
```

Ce qui renverra en JSON :
`{"param1"="result1","param2"="result2", … }`

Sachant que l'on peut également retourner des tableaux sous format JSON etc… c'est adaptable aux données que l'on veut envoyer/récupérer.
La requête HTTP peut d'ailleurs également se faire avec POST sous format JSON.

Sources utiles :

- Cours d'OpenClassrooms sur la création d'un site web en PHP et MySQL
- Documentation de l'API Tomtom