OBJECTIFS : On se propose dans ce Tp :

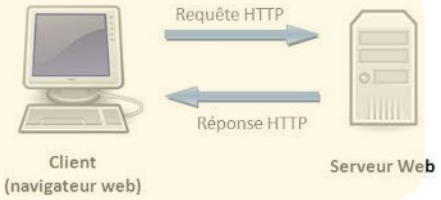
- De constituer et d'utiliser une base de données contenant des informations sur le championnat de ligue 1 de football.

- De créer une page web qui demande de saisir le numéro d'une journée de championnat et le nom d'une équipe et qui affiche en retour le résultat du match correspondant.

Cet objectif sera atteint en 4 parties :

- *Partie 1* : Constitution de la bdd en écrivant les requêtes Sql dans *phpMyAdmin* :
  - création des tables,
  - insertion « manuelle » d'un petit nombre d'enregistrements,
  - tests de l'efficacité du modèle relationnel utilisé.

- ***Partie 2* : On recommence le travail de la partie 1 en utilisant à présent une communication http locale entre le navigateur web et le serveur sur lequel se trouve la bdd et les fichiers Php.**



- *Partie 3* : On constitue le fichier Php qui permettra d'afficher la page web présentée ci-dessus. On teste l'ensemble en local (le serveur Web virtuel Apache MySQL Php crée avec le logiciel UwAmp se trouve sur votre ordinateur).
- *Partie 4* : On télécharge son travail sur le serveur Apache d'un site clone de *nsibranly.fr* afin de tester l'ensemble sur un serveur distant. On en profite pour analyser les requêtes http échangées entre le navigateur sur votre ordinateur et le serveur distant (serveur Ovh sur la région Lilloise) .

1. Découverte du php et mise en route :

⇨ Télécharger le dossier partie2.zip à partir de https://nsibranly.fr , le décompresser et copier les 4 fichiers qu'il comprend, **directement dans le dossier UwAmp/UwAmp/www**. Ces fichiers sont :

- *index.php* que l'on manipulera tout au long de ce tp. On y écrira les requêtes Sql qui permettront d'interroger la Bdd.

- *sql_manager.php* est un fichier contenant des fonctions utilisées dans *index.php* qui permettent de se connecter à la Bdd locale et d'afficher les données retournées par cette Bdd sur votre navigateur.

- *style.css* contient le css pour l'affichage des données.

- *ligue1.sql* contient toutes les commandes sql qui permettent de créer les 5 tables vues dans la partie précédente et surtout de les remplir. Les tables *equipes*, *joueurs*, *dates* et *matchs* seront complètes. La table *buts* ne contiendra que les résultats des 8 premières journées du championnat.

⇨ Ouvrir avec Visual Studio Code le fichier *index.php* :

```
<!DOCTYPE html>
<html lang="fr-FR">
    <head>
        <title>sql-requetes</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="style.css">
    </head>

    <body>
        <?php
            include "sql_manager.php" ; // fichier contenant les fonctions connectDb() et affiche_tableau()
            $connexion = connectDb("ligue1"); // connexion à la bdd
            try {
                $ligne_sql =
                    "
                    ";
                $requete = $connexion->prepare($ligne_sql);
                $requete->execute();

                try {
                    $resultat = $requete->fetchAll(PDO::FETCH_ASSOC);
                    //affiche_tableau($resultat);
                    affiche_tableau2($resultat);
                }catch (Exception $erreur) {
                    echo ' ';
                }

            }catch(PDOException $e){
                die('Erreur : ' . $e->getMessage());
            }
        ?>
    </body>
</html>
```

Le début de la zone php est identifiée par **< ?php**

Nos requêtes Sql, on les écrira ici

PARTIE contenant du code php . En dehors de cette zone c'est de l'html

La fin de la zone php est identifiée par **?>**

On commence par découvrir quelques fondamentaux du langage *php*.

⇨ Copier ce fichier *index.php* sous un autre nom, par exemple *tests.php* toujours dans le dossier UwAmp/UwAmp/www .

⇨ Sur ce fichier *tests.php*, supprimer le contenu de la partie php existante en laissant uniquement les balises < ?php et ?> .

⇨ Ouvrir votre navigateur et exécuter dans l'Url : http://localhost/tests.php afin de lire ce fichier. Aucun affichage étant réalisé dans celui-ci, le navigateur vous retourne une page blanche.

⇨ Ecrire dans la partie php la ligne ci-dessous et actualiser la page dans votre navigateur.

```
<?php
    echo "Y a quelqu'un ici";
?>
```

Les instructions *php* se terminent obligatoirement par un **;**

⇨ Faire un clic droit sur la page du navigateur afin d'afficher le « *code source de la page* ».

On constate que votre navigateur n'a reçu **aucun** code php.

En effet lorsque vous rentrez l'Url http://localhost/tests.php , votre navigateur va chercher sur le réseau le serveur qui héberge cette page. Celui-ci interprète les instructions *html* et *php* qui sont dans le fichier, **mais ne renvoie que** de l'*html* vers le navigateur.

```html
<!DOCTYPE html>
<html lang="fr-FR">
    <head>
        <title>sql-requetes</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="style.css">
    </head>

    <body>
        Y a quelqu'un ici     </body>
</html>
```

L'instruction php `echo "Y a quelqu'un ici";` permet de renvoyer en html : `Y a quelqu'un ici`

⇨ Ecrire à présent :

```php
<?php
    echo "<style>";
        echo ".coul {background-color:yellow;}";
    echo "</style>";
    echo "<div class = 'coul'>";
        echo "On peut écrire n'importe quelle ligne html, suffit
                de mettre echo devant. <br>
                Pas beau tout ça ?
                <br> Mais attention de NE PAS OUBLIER le ;
            ";
    echo "</div>";
?>
```

Après la commande php « ***echo*** » on met une chaine de caractère

.. et regarder ce qui est renvoyé par votre navigateur. Affichez aussi le code source de la page. Y a-t-il du php ?

⇨ Ecrire à présent :

```php
<?php
    $ma_premiere_variable = "TOUTE VARIABLE commence par $, t'as compris ?";
    echo "<style>";
        echo ".coul {background-color:yellow;}";
    echo "</style>";
    for ($i = 0 ; $i < 25 ; $i++){
        echo "<div class = 'coul'>";
            echo " avertissement ".$i."<br>" ;
            echo $ma_premiere_variable;
        echo "</div>";
    }

?>
```

En python : for i in range (n) : est équivalent en php à : for ($i=0 ; $i<n ; $i++){..}

Pour concaténer 2 chaine de caractères, on utilise . en php, alors qu'en python, c'est le + qui est utilisé

et regarder ce qui est renvoyé par votre navigateur. Affichez aussi le code source de la page.

⇨ Ecrire à présent :

En *python* et *php*, les listes sont définies presque de la même manière

```php
<?php
    $ma_premiere_liste = ["green","yellow","red","blue","purple","white"];
    $nb_elt = count($ma_premiere_liste) ;
    $i = 0;
    foreach ($ma_premiere_liste as $valeur){
        $i = $i + 1;
        echo    "
            <div  style = 'background-color : $valeur'>
                <p>Cette liste comprend $nb_elt éléments<p>
                <p>L'élément $i a comme valeur : $valeur<p>
                <p>Cette structure est l'équivalent en python de
                    for valeur in liste :<p>
            </div>
            ";
    }
?>
```

En python : for elt in liste :
est équivalent en php à :
foreach ($liste as $elt){..}

Après la commande php « ***echo*** » on met **une chaine de caractère**

On met du code html dans une « grande » chaine de caractère.

Pour insérer une variable dans une chaine de caractère, en php, on n'est pas obligé d'utiliser la concaténation.

et regarder ce qui est renvoyé par votre navigateur. Affichez aussi le code source de la page.

⇨ Ecrire à présent :

$mon_dictionnaire1 est un tableau associatif. Les clés sont 'green' ou 'yellow' …, les valeurs sont 'Les syntaxes', …..
La structure équivalente en *python*, ce sont les ***dictionnaires***.

$liste est un tableau en *php*. La structure équivalente en python, c'est la *liste indexée*

```php
<?php
    $mon_dictionnaire1["green"] = "Les syntaxes ";
    $mon_dictionnaire1["yellow"] = "des commandes ";
    $mon_dictionnaire1["red"] = "pythons sont ";
    $mon_dictionnaire1["blue"] = "proches du  ";
    $mon_dictionnaire1["white"] = "php  ";
    $mon_dictionnaire2["green"] = "Le php ";
    $mon_dictionnaire2["yellow"] = "permet de générer ";
    $mon_dictionnaire2["red"] = "de l'html ";
    $mon_dictionnaire2["blue"] = "de manière  ";
    $mon_dictionnaire2["white"] = "automatique  ";

    $liste = [$mon_dictionnaire1 , $mon_dictionnaire2];
    echo count($liste);

    for ($i=0 ; $i < count($liste) ;$i++){
        foreach ($liste[$i] as $cle=>$valeur){
            echo "
                <div  style = 'background-color : $cle'>
                    <p>
                        On affiche le dictionnaire qui est stocké
                        dans l'élément $i de la liste
                    <p>
                    <p>
                        L'élément du dictionnaire repéré par la clé $cle
                        a comme valeur : $valeur
                    <p>
                </div>
            " ;
        }
    }
```

On met du code html dans une « grande » chaine de caractère.

*ATTENTION : les variables insérées dans la chaine de caractère qui suit **echo** ne peuvent pas être des listes.*

⇨ Fermer le fichier *tests.php* qui nous a permis de nous familiariser avec ce nouveau langage. Ouvrir à nouveau le fichier *index.php* dont on détaille rapidement les commandes principales :

```php
<!DOCTYPE html>
<html lang="fr-FR">
    <head>
        <title>sql-requetes</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="style.css">
    </head>

    <body>
        <?php
            include "sql_manager.php" ; // fichier contenant les fonctions connectDb() et affiche_tableau()
            $connexion = connectDb("ligue1"); // connexion à la bdd
            try {
                $ligne_sql =
                "
                    SQL
                ";
                $requete = $connexion->prepare($ligne_sql);
                $requete->execute();

                try {
                    $resultat = $requete->fetchAll(PDO::FETCH_ASSOC);
                    //affiche_tableau($resultat);
                    affiche_tableau2($resultat);
                }catch (Exception $erreur) {
                    echo ' ';
                }

            }catch(PDOException $e){
                die('Erreur : ' . $e->getMessage());
            }
        ?>
    </body>
</html>
```

Le début de la zone php est identifiée par **< ?php**

On exécute la fonction ***connectDb()*** qui permet de se connecter à la base 'ligue1' et retourne un objet nommé ***$connexion***

Nos requêtes Sql, on les écrira ici

On exécute la fonction ***prepare()*** qui prend en argument la requête Sql. Cette fonction retourne un objet nommé ***$requete.***
On exécute ensuite la fonction ***execute()*** sur cet objet . Comme son nom l'indique, elle exécute la requête Sql.

On affiche les données éventuelles retournées par la Bdd (retour du SELECT …). Structure try, car il n'y a pas toujours un retour

La fin de la zone php est identifiée par **?>**

2. Utilisation de ce fichier pour communiquer avec la Bdd et pour créer et remplir les différentes tables.

⇨ On commence par supprimer la base de données nommée ligue1 et qui a été créée dans la 1ère partie. A cet effet, exécuter le fichier *index.php* et ayant écrit la commande Sql :

```php
$ligne_sql =
    "
        DROP DATABASE ligue1;

    ";
```

⇨ On continue en recréant une nouvelle base *ligue1*, en utilisant à nouveau phpMyAdmin. Mais ce sera la seule fois.

```sql
CREATE DATABASE ligue1 CHARACTER SET utf8;
USE ligue1;
```

⇨ Ouvrir avec *Visual Studio Code* le fichier *ligue1.sql* . Il contient toutes les instructions *Sql* qui permettent de créer chacune des 5 tables et de les remplir. Insérer par un copié-collé toutes ces instructions dans la chaine de caractère $*ligne_sql* et exécuter le fichier *index.php* dans le navigateur.

Vérifier dans *phpMyAdmin* après actualisation, que tout s'est bien passé.

3. Utilisation de *index.php* pour communiquer avec la Bdd afin d'exploiter les données qu'elle contient.

Les requêtes Sql seront à présent toutes écrites dans le fichier ***index.php*** . On n'utilise plus du tout *phpMyAdmin* que l'on peut fermer définitivement.

⇨ Compléter le tableau suivant en donnant soit la commande SQL , soit sa signification en français. Si la ligne est déjà complète, exécutez tout simplement la commande pour la comprendre. Dans tous les cas exécutez les requêtes via le fichier *index.php* :

Commande SQL	***Signification***
`SELECT * FROM equipes ;`	Sélectionne toutes les lignes de la table ***equipes***
`SELECT * FROM joueurs ;`	
`SELECT * FROM dates ;`	
	Sélectionne toutes les lignes de la table ***matchs***
	Sélectionne toutes les lignes de la table ***buts***
	Sélectionne le nombre de lignes de la table ***matchs***
`SELECT COUNT(*) AS nbButs FROM buts ;`	
	Sélectionne le nom et la taille des joueurs qui ont une taille supérieure à 195 cm

`SELECT j.nom AS nom , e.nom AS club , j.taille FROM joueurs AS j`
`JOIN equipes AS e ON j.id_equipes = e.id`
`WHERE taille > 195;`	

`SELECT nom , pays FROM joueurs` `ORDER BY pays;`	Sélectionne le nom et la nationalité des joueurs de la table ***joueurs*** en les triant par pays
`SELECT pays FROM joueurs` `GROUP BY pays;`	

`SELECT pays , AVG(taille) FROM joueurs` `GROUP BY pays;`	Sélectionne la taille moyenne des joueurs regroupés par nationalité

`SELECT COUNT(*) , ville FROM equipes AS e`
`JOIN joueurs AS j ON e.id = j.id_equipes`
`GROUP BY ville`	

	Donne l'âge moyen des joueurs pour chacune des équipes

`SELECT COUNT(*) , ville FROM equipes AS e`
`JOIN joueurs AS j ON e.id = j.id_equipes`
`JOIN buts AS b ON b.id_joueurs = j.id`
`GROUP BY ville`	

	Donne le nombre de buts marqués en 2ième mi-temps, pour chacune des équipes

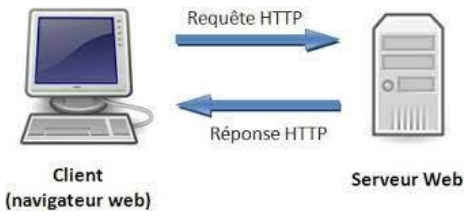
`SELECT COUNT(*) , ville FROM equipes AS e`
`JOIN joueurs AS j ON e.id = j.id_equipes`
`JOIN buts AS b ON b.id_joueurs =j.id`
`WHERE csc = 1`
`GROUP BY ville`	

`SELECT COUNT(*) , ville FROM equipes AS e`
`JOIN joueurs AS j ON e.id = j.id_equipes`
`JOIN buts AS b ON b.id_joueurs =j.id`
`JOIN matchs AS m ON b.id_matchs = m.id`
`WHERE num_journee = 1`
`GROUP BY ville`	

	Donne le nom des joueurs et leur club qui ont marqués un but lors de la première journée de championnat

4. On insère dans la requête Sql des variables

On reprend la dernière requête Sql écrite précédemment. Un utilisateur sur son navigateur peut être à des milliers de km du serveur sur lequel est stocké et exécuté le fichier *index.php* . Il souhaite avoir la liste des joueurs qui ont marqués sur une certaine journée du championnat qu'il aura choisi. Cet utilisateur n'est pas le propriétaire du site et n'a aucun accès en écriture sur le fichier *index.php* .



On voit ici comment lui permettre de transmettre au serveur le numéro de la journée du championnat qui l'intéresse …..

⇨ Rajouter dans le fichier *index.php*, la partie entourée en bleu ci-dessous et insérer la variable $num dans la requête Sql :

```php
<?php
    include "sql_manager.php" ; // fichier contenant les fonctions connectDb() et affiche_tableau()
    $connexion = connectDb("ligue1"); // connexion à la bdd

    if (isset($_GET["num"])){
        $num = $_GET["num"];
    }else{
        $num = 1;
    }
    echo $num ;

    try {
        $ligne_sql =
            "
                SELECT j.nom , ville FROM equipes AS e
                JOIN joueurs AS j ON e.id = j.id_equipes
                JOIN buts AS b ON b.id_joueurs =j.id
                JOIN matchs AS m ON b.id_matchs = m.id
                WHERE num_journee = $num
               ORDER BY ville

            ";
        $requete = $connexion->prepare($ligne_sql);
        $requete->execute();

        try {
            $resultat = $requete->fetchAll(PDO::FETCH_ASSOC);
            //affiche_tableau($resultat);
            affiche_tableau2($resultat);
        }catch (Exception $erreur) {
            echo ' ';
        }

    }catch(PDOException $e){
        die('Erreur : ' . $e->getMessage());
    }
?>
```

isset() est une fonction php qui retourne TRUE si la variable qui est en argument existe. Cette fonction retourne FALSE sinon

$_GET[] est un tableau associatif (équivalent en python d'un dictionnaire) . ***$_GET["num"]*** stocke une chaine de caractère qui peut être transmise par l'utilisateur **via l'url**.

Pour cela il doit rajouter à l'url de la page : **?num=5** s'il veut donner la valeur 5 à $_GEL['num'] , ce qui donnerait comme url :
localhost/ ?num=5

Le numéro de journée est celui stocké dans la variable *$num*

On dit que l'utilisateur transmet une donnée au serveur par la méthode GET .

⇨ Tester ce procédé pour afficher le nom des buteurs pour différentes journées du championnat.

⇨ **Application** : modifier à présent le fichier *index.php* afin qu'il renvoie la liste des buteurs de la journée numéro ***num*** de l'équipe de la ville de ***ville***. Pour choisir la 2ième journée et la ville de Marseille, l'url sera :

http://localhost/?num=2&ville=Marseille

La requête Sql dans le fichier sera :

```
SELECT j.nom , ville FROM equipes AS e
JOIN joueurs AS j ON e.id = j.id_equipes
JOIN buts AS b ON b.id_joueurs =j.id
JOIN matchs AS m ON b.id_matchs = m.id
WHERE num_journee = $num AND ville = '$ville'
```

On met ici des ' ' autour de *$ville*, car la requête Sql exige des ' ' autour d'un attribut qui est une chaine de caractère

On ne met pas de ' ' autour de *Marseille* dans l'Url

⇨ Tester le bon fonctionnement.

Cette seconde partie est terminée. Ses objectifs étaient :

- de toujours pratiquer le Sql en tapant au clavier des lignes de commande d'insertion et de sélection,
- d'interroger la base de donnée via un fichier php,
- d'instaurer un dialogue entre l'utilisateur (Client) et le serveur en utilisant une méthode GET qui permet de transmettre des données de l'utilisateur vers le serveur, via l'Url
- enfin d'avancer dans l'objectif principal que l'on s'est fixé au départ et qui est de créer une page web qui affiche le résultat d'un match quelconque du championnat.

On continue à présent sur la troisième partie de ce travail. Son objectif sera de créer le fichier php de la page web ci-contre et de le faire fonctionner sur le serveur local.