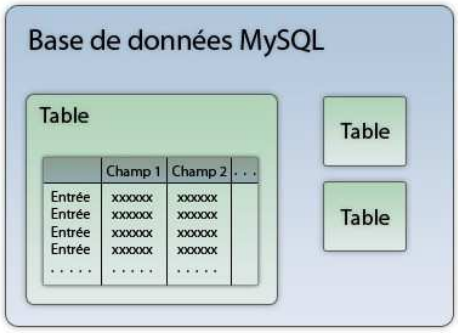
Mini-projet n°3

Vous devez réaliser un site web qui affiche des messages postés par des membres. Les attendus sont les suivants :

- Vous devez travailler en collaboration au sein de groupes de 2 ou 3.
- Vous devez concevoir les éléments principaux de la solution avant de passer à la programmation (structure, algorithme …).
- La page principale doit afficher les dix derniers messages du plus récent au plus ancien et permettre d'en rentrer un.
- Le membre sera identifié par un pseudo qu'il choisira.
- Vous devez produire un document présentant la structure du site et les scripts des différentes pages web.
- Les langages utilisés seront HTML/CSS et PHP.
- Vous utiliserez le système de gestion de bases de données MYSQL.



Critères d'évaluation

L'évaluation est *individuelle*. Elle consiste en une lettre qui traduit le niveau de l'exposé. Il y a 4 critères évalués : l'évaluation globale est une moyenne de l'évaluation de chacun de ces critères.

PRODUCTION		
Critères de réussite	**Niveau d'exigence**	
Collaboration au sein de l'équipe dans le cadre du projet.	**D**	*Insuffisant : Il n'y a pas d'implication dans l'équipe et chacun travaille pour soi.*
	C	*Acceptable : des efforts réels de travail collaboratif sont visibles.*
	B	*Satisfaisant : les rôles sont répartis et le travail collaboratif se fait en interaction avec le professeur.*
	A	*Excellent : les rôles sont bien définis, l'interaction avec le professeur est constructive.*
Description et explication de la situation et du programme	**D**	*Insuffisant : la situation n'est pas présentée et les solutions choisies ne sont pas expliqués.*
	C	*Acceptable : la situation et présentée, mais les choix ne sont pas argumentés.*
	B	*Satisfaisant : les choix sont présentés et les instructions sont globalement expliquées.*
	A	*Excellent : les instructions importantes sont expliquées rigoureusement. les choix sont argumentés clairement.*
Conception et réalisation de la solution informatique en réponse au problème	**D**	*Insuffisant : le code source n'est pas commenté.*
	C	*Acceptable : les instructions sont globalement comprises. Le vocabulaire est globalement correct mais certains termes sont imprécis. Il y a peu d'étapes inutiles.*
	B	*Satisfaisant : le vocabulaire utilisé dans les commentaires est précis et maîtrisé. Les noms des variables sont judicieusement choisis et les instructions sont comprises.*
	A	*Excellent : le vocabulaire utilisé est riche et rigoureux. Les noms des variables sont clairs. Les étapes sont pertinentes et précisément commentées.*

Document 1 : principe d'une base de données

Une base de données est une construction qui permet d'enregistrer et surtout de classer des informations. Un système de gestion de base de données, comme MySQL par exemple, permet d'enregistrer et de sélectionner les informations contenues dans une base de données. La base de données est donc fondamentalement un fichier stocké sur un disque dur, et le système de gestion de base de données est le programme qui permet de gérer les informations de ce fichier.

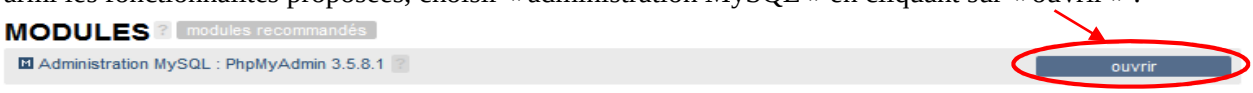
Une base de données est comme une armoire de rangement. Dans cette armoire, la *base* elle-même, il y a des blocs appelés *tables*. Chaque bloc contient des colonnes, appelées *champs*, et des *étagères*, appelées entrée, de sorte que le bloc est divisé en cases. Une information élémentaire dans une base de données correspond ainsi à une case, contenue dans une table, et localisée par une entrée et un champ.

Une base de données gère donc les informations sous forme de tableaux : un tableau est une table, une entrée est une ligne de ce tableau et un champ est une colonne. Il y a généralement de nombreuses tables dans une seule base.

Document 2 : administration d'une base de données avec phpMyAdmin

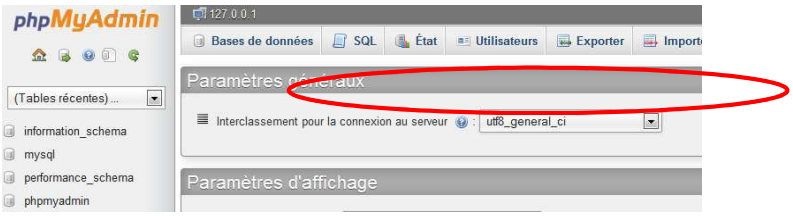
Pour visualiser et faire des opérations manuellement sur une base de données, on peut utiliser un logiciel appelé phpMyAdmin, qui procure une interface graphique facile à utiliser. En particulier, ce logiciel permet de créer une base de données et les tables qu'elle contient avec leurs champs.

Pour accéder à phpMyAdmin, il faut d'abord lancer le logiciel EASYPHP et vérifier que les services sont actifs.
Il faut ensuite ouvrir un navigateur comme FIREFOX et entrer comme URL :
127.0.0.1/home ou localhost /home
Parmi les fonctionnalités proposées, choisir « administration MySQL » en cliquant sur « ouvrir » :



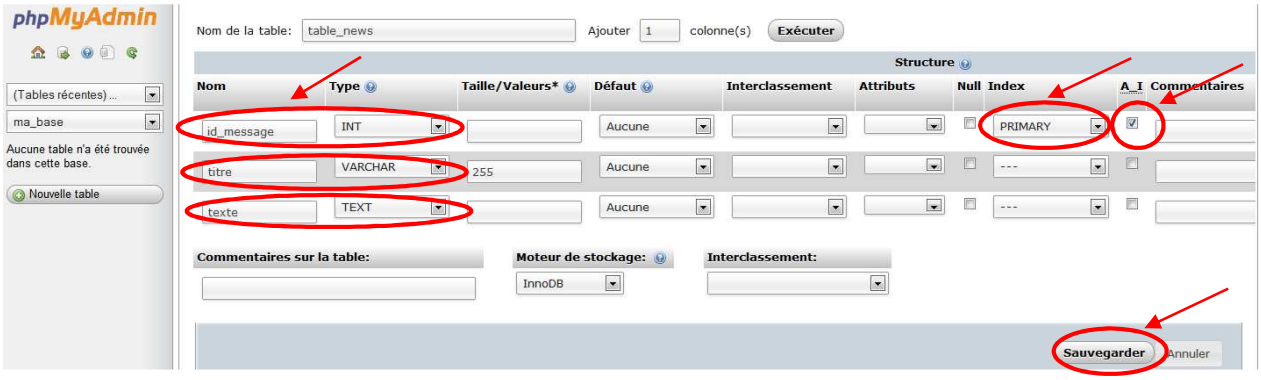
La page d'accueil de phpMyAdmin permet de créer une nouvelle base de données en cliquant sur « Bases de données » puis en rentrant le nom de la base, par exemple « ma_base », et en choisissant comme interclassement « utf8_general_ci »

Il faut ensuite rentrer au moins une table dans la base, par exemple « table_news » pour gérer des brèves sur un site, en précisant le nombre de champs de la table, ici « 3 ». La table est créée en cliquant sur « Exécuter ».

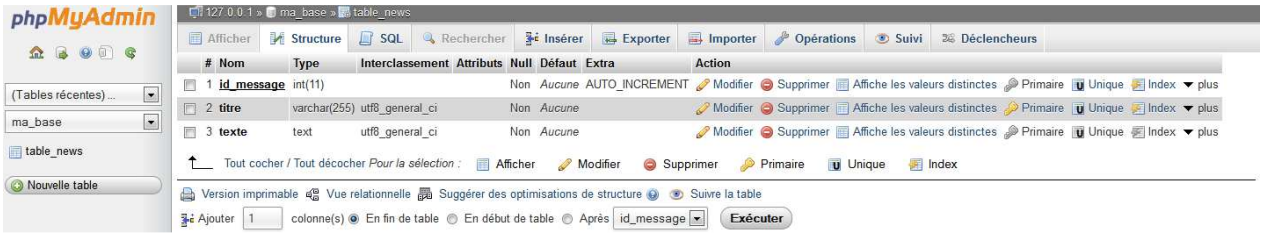


L'étape suivante consiste à préciser les noms des champs et le type de variable qu'ils contiendront. L'option d'incrémentation automatique (*Auto Increment*) permet d'augementer la valeur champ choisi de une unité à chaque nouvelle entrée dans la table. On choisit souvent un champ appelé « id », ici « id_message », pour numéroter les entrées et les identifier facilement.

Il faut enfin définir une « clé primaire » pour le classement des données, afin d'assurer qu'une entrée est unique et peut être identifiée de vaçon univoque. Cette étape est très importante, elle n'est pas obligatoire mais son oubli peut rendre la table inutilisable. Généralement, c'est le champ « id » qui joue ce rôle. Ce choix se fait en sélectionnant « PRIMARY » dans l'index du champ choisi comme clé primaire.
Les champs sont en définitive créés en cliquant sur « sauvegarder ».



A ce stade, il existe donc dans la base « ma_base » une seule table « table_news » qui est divisée en trois champs : « id_message », « titre » et « texte ». Il n'y a pour l'instant aucune entrée dans cette table, comme on peut le vérifier en cliquant sur l'onglet « structure ».



La table peut être entièrement gérée avec phpMyAdmin. Cependant, cela n'a pas grand intérêt : l'idée est d'automatiser la gestion de la table en utilisant des instructions PHP qui permettent de faire des opérations sur la base de données via le langage SQL. Le langage PHP permet donc de manipuler le langage SQL, ce qui permet de programmer des modifications de la base de données.

Document 3 : dialoguer avec une base de données en utilisant le langage PHP.

Le module « pdo » du langage PHP permet de gérer une base de données, et ce pour n'importe quel système de gestion de base données (SGBD) : MySQL, SQlite, Oracle…
Il est actuellement recommandé d'utiliser ce module, plutôt que les fonctions de base de MySQL, qui tendent à devenir obsolètes.

Le principe est de créer un objet qui représente la connexion avec la base de données stockée sur un serveur dédié.

```php
<?php
$bdd = new PDO('mysql:host=localhost;dbname=ma_base', 'root', '');
?>
```

La variable **$bdd** est un objet instancié à partir de la classe PDO et réalise la connexion avec la base de données.
Les attributs sont :

- **mysql:host** qui est le chemin d'accès à la base de données, ici « localhost (127.0.0.1) » dans le cas d'un serveur local simulé par le logiciel EASYPHP par exemple.
- **dbname** doit avoir plusieurs valeur : le nom de la base de données, ici « ma_base », l'identifiant de connexion à la base, ici « root » dans le cas d'un serveur local, et le mot de passe de connexion (ici il n'y en pas, d'où les guillemets vides).

Il est ensuite possible de récupérer des informations contenues dans la base de données. Cela se fait par l'intermédiaire d'une requête réalisée par l'instruction **`query()`** qui permet d'exécuter une commande en langage SQL. La requête la plus simple consiste à récupérer dans un objet tout le contenu d'une table selon la syntaxe suivante :

```php
<?php
$bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '');
$reponse = $bdd->query('SELECT * FROM table_news');
?>
```

l'instruction **SELECT** récupère les champs spécifiés de la table indiquée par **FROM.** Ici, l'argument de l'instruction **SELECT** est « * », ce qui signifie que tout le contenu de la table est récupéré.
L'objet « **$reponse** » contient alors l'ensemble des champs et des entrées de la table « table_news ».

L'étape suivante consiste à récupérer le contenu de chaque champ, entrée par entrée, dans une liste associative : cette liste contient les éléments d'une entrée, chaque élément correspondant au contenu d'un champ de la table.
L'instruction **fetch()** permet de récupérer les entrées contenues dans l'objet **$reponse** : à chaque fois qu'elle est exécutée, cette instruction passe à l'entrée suivante.
Par exemple, le script ci-dessous affiche les brèves de la table « table_news » une à une :

```php
<?php
//récupération et affichage de chaque brève.
//connexion à la base de données :
$bdd = new PDO('mysql:host=localhost;dbname=ma_base', 'root', '');

//récupération de l'ensemble des brèves :
$reponse = $bdd->query('SELECT * FROM table_news');

//affichage des brèves une à une, tant qu'il y en a :
while ($entree = $reponse->fetch())
    {
        echo '<p> voici la brève n° ' . $entree['id_message'];
        echo ' qui a pour titre : ' . $entree['titre'] . ', ';
        echo 'et contenant cette information : ' . $entree['texte'] . '</p>';
    }

//clôture de la connexion :
$reponse->closeCursor();
?>
```

Le test de la boucle **while** est particulier : il contient en même temps une affectation et un test. Cela est possible par le fait que l'instruction **fetch()** renvoie le booléen « false » lorsque la fin de la sélection (ici, toute la table) est atteinte.

Il faut enfin fermer les résultats de recherche avec l'instruction **closeCursor()** après avoir traité chaque requête : cela permet de replacer le curseur de sélection à la première entrée de la table.

Pur récupérer seulement quelques éléments de la table, il existe de nombreux critères de sélection. Les trois principaux sont **WHERE, ORDER BY** et **LIMIT**.
Le critère **WHERE** est particulièrement utile :

```php
//récupération de la brève n°5:
$reponse = $bdd->query('SELECT * FROM table_news WHERE id_message=5');
```

Les critères de sélections peuvent être combinés pour affiner le tri :

```php
//récupération des 5 premiers titres des brèves :
$reponse = $bdd->query('SELECT titre FROM table_news ORDER BY id_message LIMIT 0,5');
```

Lorsque le contenu d'un champ de type chaîne de caractère est utilisé dans une requête, il faut échapper les guillemets indiquant la chaîne de caractère par des antislashs, comme ici par exemple :

```php
$reponse = $bdd->query('SELECT texte FROM table_news WHERE titre=\'quoi de neuf ?\' ORDER BY id_message DESC LIMIT 100,50');
```

Cette requête demande les textes des brèves dont le titre est « quoi de neuf ? » et les classe par numéro décroissant, à partir de la 100e en partant de la fin et seulement pour les 50 suivantes.

Pour insérer des entrées dans une table, on utilise la commande SQL « **INSERT INTO** ». Par exemple, la requête suivante demande d'insérer la chaîne de caractères **'météo demain'** dans le champ « **titre** » et **'il fera beau et froid demain, brr!'** dans le champ « **texte** » de la table « **table_news** » :

```
INSERT INTO table_news(titre, texte) VALUES('météo demain','il fera beau et froid demain, brr!')
```

Le champ « id_message » n'est ici pas nécessaire car il est en incrémentation automatique : sa valeur sera automatiquement augmentée de une unité et insérée en entrée.
Avec le langage PHP, la fonction adaptée est la fonction **exec()** pour insérer, à la place de **query()** précédemment pour récupérer des données. Il faut ici veiller à échapper les guillemets qui entourent les chaînes de caractère grâce à l'antislash :

```php
<?php
$bdd->exec('INSERT INTO table_news(titre, texte) VALUES(\'météo demain\',\'il fera beau et froid demain, brr!\')');
?>
```

Lorsque les données à insérer sont apportés par des variables, envoyées en paramètres dans l'URL ou venant de formulaires, le moyen le plus sûr est d'utiliser une requête préparée.
Le script suivant prépare une requête attendant des valeurs identifiées de façon nominative (il est possible d'utiliser simplement un point d'interrogation), puis l'exécute en fournissant les valeurs à l'aide d'un tableau associatif :

```php
<?php
//prépare l'insertion de variables identifiées nomminativement :
$requete = $bdd->exec('INSERT INTO table_news(titre, texte) VALUES(:titre,:texte)');
//exécute la requête préparée avec pour valeur des données issues de formulaires :
$requete->execute(array('titre' => $_GET['titre_news'],'texte' => $_GET['texte_news']));
?>
```

Il faut être vigilant ici sur les noms des variables :

- La requête préparée attend des variables appelées ici « **titre** » et « **texte** » et spécifiées par le mot-clé **VALUES** avec « : » précédent leur nom. Les différents champs sont séparés par des virgules. Les noms de ces variables n'ont aucune importance, ils peuvent être identiques aux champs ou différents.
- la commande « **execute** » utilise un tableau associatif créé ici directement dans la commande, c'est ce qui est fait le plus souvent. Les clés du tableau associatif doivent être identiques aux noms choisis pour les variables dans la requête préparée. Les valeurs sont ici des éléments du tableau **$_GET** , valeurs entrées par l'utilisateur via un formulaire, ou passées en paramètres dans l'URL, sous les noms **titre_news** et **texte_news.**

La sécurité est assurée par cette méthode pour les commandes SQL : un utilisateur malveillant ne pourra pas envoyer de requête SQL lorsque la requête a été préparée. Cependant, il peut toujours écrire n'importe quoi dans l'URL ou le formulaire. Cela n'a pas été pris en compte ici, mais en toute rigueur, il faut utiliser la fonction **strip_slashes()** ou **htmlspecialchars()** pour éliminer ce risque.

JOKER 1

Page web de nom « acces_reserve.php » protégée par mot de passe :

```php
<?php

if isset($_POST['motdepasse']))
    {
    if $_POST['motdepasse'] == "sorbetcitron"
        {
?>
            <p> Bienvenu sur le site de l'ordre du phenix ! </p>
<?php
        }
    else
        {
?>

<div>
    <p> Veuillez taper votre mot de passe </p>

    <form action="acces_reserve.php" method="post">
    <input type="password" name="motdepasse"/>
    <input type="submit" name="VALIDER"/>
    </form>

</div>
```

JOKER 2

Il faut deux fichiers :

- minichat.php : contient le formulaire permettant d'ajouter un message et liste les 10 derniers messages.
- minichat_post.php : insère le message reçu avec $_POST dans la base de données puis redirige vers minichat.php.

Il aurait été possible de tout faire sur une seule page PHP, mais pour bien séparer le code il est préférable d'utiliser 2 fichiers comme ici.

Pour la redirection, il faut utiliser l'instruction « header » :

```php
<?php
// Effectuer ici la requête qui insère le message
// Puis rediriger vers minichat.php comme ceci :
header('Location: minichat.php');
?>
```

D'après *www.openclassroom.com*

JOKER 3

Page minichat_post de traitement du formulaire :

```php
  <?php
// Connexion à la base de données
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root','');
// Insertion du message à l'aide d'une requête préparée
    $req = $bdd->prepare('INSERT INTO minichat (pseudo, message)
  VALUES(?, ?)');
    $req->execute(array($_POST['pseudo'], $_POST['message']));

// Redirection du visiteur vers la page du minichat
     header('Location: minichat.php');
  ?>
```

D'après *www.openclassroom.com*

JOKER 4

La page principale

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Mini-chat</title>
        <meta http-equiv="Content-Type" content="text/html;  charset=iso-
8859-1" />
    </head>

    <body>

    <form action="minichat_post.php" method="post">
        <p>
        <label for="pseudo">Pseudo</label> : <input type="text"
name="pseudo" id="pseudo" /><br />
        <label for="message">Message</label> :  <input
type="text"  name="message" id="message" /><br />
        <input type="submit" value="Envoyer" />
    </form>

<?php
// Connexion à la base de données
    $bdd = new PDO('mysql:host=localhost;dbname=test', 'root', '');
    // Récupération des 10 derniers messages
    $reponse = $bdd->query('SELECT pseudo, message FROM minichat
ORDER BY ID DESC LIMIT 0, 10');

    // Affichage de chaque message (toutes les données sont
protégées par htmlspecialchars)
    while ($donnees = $reponse->fetch())
    {
        echo '<p><strong>' . htmlspecialchars($donnees['pseudo']) .
'</strong> : ' . htmlspecialchars($donnees['message']) . '</p>';
    }

    $reponse->closeCursor();
}
?>
    </body>
</html>
```