PHP : Bases de données (exercices)

Projet guidé : Créer une application PHP pour gérer une liste d'utilisateurs (CRUD)

🎯 Objectif pédagogique

À travers cette activité, tu vas apprendre à :

- Organiser un projet PHP en plusieurs fichiers
- Lire, ajouter, modifier et supprimer des données dans une base MySQL
- Protéger tes données contre les erreurs et les attaques
- Écrire du code clair, structuré et réutilisable

🧱 Étape 0 : Préparer ton environnement

✔ À faire avant de commencer :

1. Lance ton serveur local (XAMPP, MAMP, WAMP)
2. Ouvre **phpMyAdmin**
3. Crée une base de données nommée **gestion_utilisateurs**
4. Exécute la requête suivante pour créer une table `utilisateurs` :

```sql
CREATE TABLE utilisateurs (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nom VARCHAR(100),
  email VARCHAR(100)
);
```

📂 Structure du projet

Crée un dossier nommé `projet_crud`. Il contiendra les fichiers suivants :

```
/projet_crud
│
├── config/
│   └── db.php          ← Fichier de connexion centralisé
│
├── index.php           ← Liste des utilisateurs
├── ajouter.php         ← Formulaire + traitement d'ajout
├── modifier.php        ← Formulaire + traitement de modification
├── supprimer.php       ← Suppression
```

🧩 Étape 1 : Connexion à la base (`config/db.php`)

📌 Consigne

Crée un fichier `db.php` dans un dossier `config/` et écris ce code :

```php
<?php
$host = 'localhost';
$dbname = 'gestion_utilisateurs';
$username = 'root';
$password = '';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Erreur de connexion : " . $e->getMessage());
}
?>
```

🧠 Explication

- On utilise **PDO** pour se connecter à la base de données. C'est une méthode moderne, flexible et sécurisée.
- Le bloc `try/catch` permet de **gérer les erreurs de connexion**. Si la connexion échoue, on affiche un message clair.
- On place ce code dans un fichier à part pour **le réutiliser dans toutes les pages**.

🧩 Étape 2 : Afficher les utilisateurs (`index.php`)

📌 Consigne

Crée un fichier `index.php` et ajoute le code suivant :

```php
<?php
require_once 'config/db.php';

$sql = "SELECT * FROM utilisateurs";
$stmt = $pdo->query($sql);
$utilisateurs = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>
```

```
<h1>Liste des utilisateurs</h1>

<a href="ajouter.php">➕ Ajouter un utilisateur</a>

<ul>
```

```php
<?php foreach ($utilisateurs as $u): ?>
    <li>
        <?= htmlspecialchars($u['nom']) ?> (<?= htmlspecialchars($u['email']) ?>)
        - <a href="modifier.php?id=<?= $u['id'] ?>">🖍️</a>
        - <a href="supprimer.php?id=<?= $u['id'] ?>" onclick="return confirm('Confirmer la sup
    </li>
<?php endforeach; ?>
</ul>
```

🧠 Explication

- `require_once` inclut le fichier de connexion.
- `$pdo->query()` exécute la requête SQL.
- `fetchAll(PDO::FETCH_ASSOC)` récupère les résultats sous forme de tableau associatif.
- La **boucle `foreach`** parcourt chaque utilisateur et l'affiche.
- `htmlspecialchars()` évite les attaques XSS (injections de code HTML/JS).

🧩 Étape 3 : Ajouter un utilisateur (`ajouter.php`)

📌 Consigne

Crée un fichier `ajouter.php`. Ce fichier contient le **formulaire** et le **traitement des données**.

```php
<?php
require_once 'config/db.php';

if (!empty($_POST['nom']) && !empty($_POST['email'])) {
    $stmt = $pdo->prepare("INSERT INTO utilisateurs (nom, email) VALUES (:nom, :email)");
    $stmt->execute([
        ':nom' => $_POST['nom'],
        ':email' => $_POST['email']
    ]);
    header("Location: index.php"); // Redirection vers la page d'accueil
    exit;
}
?>

<h1>Ajouter un utilisateur</h1>

<form method="post">
    <input type="text" name="nom" placeholder="Nom" required>
    <input type="email" name="email" placeholder="Email" required>
    <button type="submit">Ajouter</button>
</form>
```

🧠 Explication

- On teste si les champs sont remplis avec `!empty()`.
- On utilise `prepare()` et `execute()` pour **éviter les injections SQL**.

- `header("Location: index.php")` redirige l'utilisateur après l'ajout.

🧩 Étape 4 : Modifier un utilisateur (`modifier.php`)

📌 Consigne

Crée un fichier `modifier.php`. Ce fichier permet de **pré-remplir le formulaire** et de le **mettre à jour**.

```php
<?php
require_once 'config/db.php';

// 1. On récupère l'utilisateur à modifier
if (!empty($_GET['id'])) {
    $stmt = $pdo->prepare("SELECT * FROM utilisateurs WHERE id = :id");
    $stmt->execute([':id' => $_GET['id']]);
    $utilisateur = $stmt->fetch(PDO::FETCH_ASSOC);

    if (!$utilisateur) {
        die("Utilisateur introuvable.");
    }
}

// 2. On traite la mise à jour
if (!empty($_POST['nom']) && !empty($_POST['email'])) {
    $stmt = $pdo->prepare("UPDATE utilisateurs SET nom = :nom, email = :email WHERE id = :id")
    $stmt->execute([
        ':nom' => $_POST['nom'],
        ':email' => $_POST['email'],
        ':id' => $_POST['id']
    ]);
    header("Location: index.php");
    exit;
}
?>

<h1>Modifier un utilisateur</h1>

<form method="post">
    <input type="hidden" name="id" value="<?= $utilisateur['id'] ?>">
    <input type="text" name="nom" value="<?= htmlspecialchars($utilisateur['nom']) ?>" require
    <input type="email" name="email" value="<?= htmlspecialchars($utilisateur['email']) ?>" re
    <button type="submit">Enregistrer les modifications</button>
</form>
```

🧠 Explication

- On récupère l'utilisateur via l'URL (`modifier.php?id=3`).
- On pré-remplit le formulaire avec les valeurs actuelles.
- Le champ `hidden` permet de **conserver l'id** de l'utilisateur à modifier.
- Après validation, la base est mise à jour et l'utilisateur est redirigé.

🧩 Étape 5 : Supprimer un utilisateur (`supprimer.php`)

📌 Consigne

Crée un fichier `supprimer.php` :

```php
<?php
require_once 'config/db.php';

if (!empty($_GET['id'])) {
    $stmt = $pdo->prepare("DELETE FROM utilisateurs WHERE id = :id");
    $stmt->execute([':id' => $_GET['id']]);
}

header("Location: index.php");
exit;
```

🧠 Explication

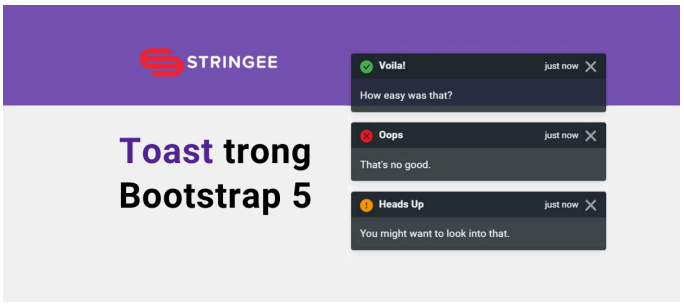
- On récupère l'`id` dans l'URL (ex: `supprimer.php?id=4`)
- On supprime la ligne correspondante
- On redirige vers la page principale

✅ Récapitulatif

Action	Fichier	Fonction
Créer	`ajouter.php`	Formulaire + ajout
Lire	`index.php`	Liste les utilisateurs
Mettre à jour	`modifier.php`	Formulaire pré-rempli + update
Supprimer	`supprimer.php`	Supprime un utilisateur

Voici l'étape supplémentaire à ajouter à ton exercice CRUD pour afficher un message de confirmation après une modification (ou une autre action) à l'aide de **messages flash** :

🧩 Étape 6 : Afficher un message de confirmation (messages flash)



📌 Consigne

1. Crée un fichier `flash.php` avec ce petit morceau de code :

```php
<?php
session_start();

function set_flash($message) {
    $_SESSION['flash'] = $message;
}

function get_flash() {
    if (!empty($_SESSION['flash'])) {
        $msg = $_SESSION['flash'];
        unset($_SESSION['flash']); // Supprimé après affichage
        return $msg;
    }
    return null;
}
?>
```

2. Dans le haut de chaque page (par exemple `index.php`), ajoute ceci **juste après** `require_once` :

```php
require_once 'flash.php';

if ($msg = get_flash()) {
    echo "<p style='background: #d4edda; color: #155724; padding: 10px;'>$msg</p>";
}
```

3. Dans les scripts `ajouter.php`, `modifier.php`, et `supprimer.php`, **juste avant chaque redirection**, ajoute :

```php
set_flash("Action réussie !");
```

Exemples :

```php
set_flash("Utilisateur modifié !");
header("Location: index.php");
exit;
```

🧠 Explication

- Un **message flash** est un message **temporaire** qui s'affiche une seule fois, après une action.
- On utilise `$_SESSION` pour le stocker **entre deux pages**.
- Il est automatiquement **effacé après affichage**, ce qui évite qu'il ne s'affiche plusieurs fois.

Très bien ! Voici comment intégrer un **message flash avec un composant Toast Bootstrap 5**, qui s'affiche automatiquement et disparaît au bout de quelques secondes.

🧩 Étape 6 (suite) : Messages flash avec un Toast Bootstrap 5

📌 Étapes à suivre

1. Inclure Bootstrap (si ce n'est pas encore fait)

Dans le `<head>` de ton fichier `index.php` (et éventuellement dans les autres), ajoute :

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="styl
```

Avant la balise `</body>` :

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></s
```

2. Modifier le code du message flash

Dans le haut de `index.php` (ou autre page), juste après `require_once 'flash.php'`, ajoute ceci :

```
<?php if ($msg = get_flash()): ?>
<div class="toast-container position-fixed top-0 end-0 p-3" style="z-index: 1055;">
  <div class="toast align-items-center text-bg-success border-0 show" role="alert" aria-live="
    <div class="d-flex">
      <div class="toast-body">
        <?= htmlspecialchars($msg) ?>
      </div>
      <button type="button" class="btn-close btn-close-white me-2 m-auto" data-bs-dismiss="toa
    </div>
  </div>
</div>
<script>
    const toastLive = document.getElementById('toastFlash');
    if (toastLive) {
      const toastBootstrap = bootstrap.Toast.getOrCreateInstance(toastLive);
      toastBootstrap.show();
    }
</script>
<?php endif; ?>
```

3. Ne change rien dans `flash.php` :

```
<?php
session_start();

function set_flash($message) {
```

```php
    $_SESSION['flash'] = $message;
}

function get_flash() {
    if (!empty($_SESSION['flash'])) {
        $msg = $_SESSION['flash'];
        unset($_SESSION['flash']);
        return $msg;
    }
    return null;
}
```

4. Dans les fichiers `ajouter.php`, `modifier.php`, `supprimer.php` :

Juste **avant le** `header(...)`, ajoute un message :

```php
set_flash("Utilisateur ajouté !");
```

ou

```php
set_flash("Utilisateur supprimé !");
```

🧠 Ce que tu fais et pourquoi

- Le **toast Bootstrap** est une boîte qui apparaît en haut à droite.
- Il **s'affiche automatiquement** et **disparaît au bout de 3 secondes** (`data-bs-delay="3000"`).
- Le script JavaScript déclenche l'affichage automatiquement au chargement de la page.
- Grâce aux sessions, le message est **persistant entre deux pages** sans rester affiché plus longtemps que nécessaire.