

# Course: Php

## Un saut dans le Web Dev

---

Ali ZAINOUL <ali.zainoul.az@gmail.com>

Crystal Clear Code  
May 28, 2025



# Mon premier CRUD

- 1 Séance 1 – Cadrage et lancement du projet
  - Objectifs de la formation
  - Présentation du projet fil rouge
  - Installation des outils nécessaires
  - Variables d'environnement et configuration
  - Organisation du projet et arborescence initiale
  - Mini cahier des charges à rédiger en groupe

# Php et BDD: une longue histoire d'amour

## 2 Séance 2 – Modélisation base de données et structure PHP

- Objectif de la séance
- Modélisation conceptuelle et physique
- Premiers pas MySQL
  - Via Terminal
  - Via PHPMyAdmin
- Script SQL de création
- Connexion PHP à la base de données (PDO)
- Structure initiale des fichiers PHP
- Création des fichiers partagés

# Poser les bases

## 3 Séance 3 – Formulaire et traitement POST

- Objectif de la séance
- Formulaire HTML simple
- Traitement des données reçues
- Points de sécurité à respecter

# Mon premier CRUD

- 4 Séance 4 – Affichage dynamique depuis la base
  - Objectif de la séance
  - Connexion à la base et requête SELECT
  - Boucle d’affichage HTML avec PHP
  - Structuration HTML minimale et accessibilité

# Mon premier CRUD - Suite

- 5 Séance 5 – Mise à jour, suppression et sécurité
  - Objectif de la séance
  - Formulaire pré-rempli pour modification
  - Traitement de la mise à jour
  - Suppression d'un formateur
  - Protection et sécurité

# Sessions, sécurité et conclusion

## 6 Séance 6 – Connexion utilisateur et finalisation

- Objectif de la séance
- Création de la table utilisateurs
- Insertion d'un utilisateur de test
- Formulaire de connexion
- Vérification des identifiants et session
- Affichage conditionnel dans le header

# Les bases de MySQL

## 7 Séance 7 – Soutenance et retour d'expérience

- Objectif de la séance
- Contenu de la soutenance
- Livrables attendus



## **\*\* Introduction \*\***

# Objectifs pédagogiques

Cette formation a pour objectif de permettre aux apprenants de :

- Comprendre les bases du langage **PHP procédural** ;
- Mettre en place une **base de données MySQL** locale ;
- Développer une application web en **PHP pur**, sans frameworks ;
- Utiliser des formulaires HTML avec la méthode POST ;
- Réaliser les opérations **CRUD** (Créer, Lire, Mettre à jour, Supprimer) ;
- Implémenter un système d'**authentification sécurisé** via `$_SESSION` ;
- Éviter les failles courantes : XSS, injections SQL, etc.

# Projet fil rouge : Annuaire PHP des Formateurs

Le projet fil rouge consiste à développer un annuaire de formateurs permettant :

- L'ajout d'un formateur (nom, prénom, email, domaine) ;
- La visualisation dynamique de tous les formateurs ;
- La modification et la suppression des données existantes ;
- La sécurisation des actions par un système de connexion ;
- L'organisation du code selon une structure claire de fichiers PHP.

# Installation de PHP et MySQL

Avant de commencer, il est nécessaire d'installer les outils suivants en local :

- **PHP** : moteur d'exécution des scripts PHP;
- **MySQL** : système de gestion de base de données relationnelle ;
- **phpMyAdmin** : interface graphique de gestion des bases MySQL ;
- **VS Code** : éditeur de code avec coloration syntaxique et extensions utiles.

Option 1 : Utiliser un environnement complet comme XAMPP, Laragon ou MAMP.

Option 2 : Installer chaque composant individuellement (avancé).

# Variables d'environnement PHP

Dans certains cas (notamment avec l'installation manuelle), il peut être nécessaire :

- D'ajouter le chemin de PHP à la variable `PATH` du système ;
- De vérifier que PHP est bien accessible via la commande :

```
1 php -v
```

- De configurer le fichier `php.ini` pour activer certaines extensions (`pdo_mysql`, `mbstring`, etc.).

# Structure de projet minimale

Dès la première séance, l'organisation du projet doit être mise en place :

- Un dossier racine contenant :

```
(base) █ ~ % tree annuaire-formateurs
```

```
annuaire-formateurs
```

```
├── README.md
├── add.php
├── assets
│   └── style.css
├── delete.php
├── includes
│   ├── db.php
│   ├── footer.php
│   └── header.php
├── index.php
├── login.php
└── update.php
```

```
3 directories, 10 files
```

- L'utilisation d'un outil de gestion comme Trello ou Google Drive est encouragée.

# Mini cahier des charges

Les apprenants doivent rédiger ensemble un mini cahier des charges incluant :

- Le nom du projet ;
- Les fonctionnalités à développer pour une V1 (MVP) ;
- Le découpage des tâches ;
- Les rôles des membres du groupe ;
- Un planning indicatif par séance.

## **\*\* Squelette du projet \*\***



# Ce que vous allez apprendre aujourd'hui

- Concevoir une base de données relationnelle simple ;
- Créer des tables via un script SQL clair et structuré ;
- Comprendre le lien entre base de données et traitement PHP ;
- Mettre en place les fichiers principaux du projet PHP ;
- Organiser les fichiers de manière modulaire et maintenable.

# Modèle conceptuel et physique

On souhaite stocker les informations suivantes pour chaque formateur :

- **Nom, Prénom ;**
- **Email (unique) ;**
- **Domaine de formation ;**
- **Date d'ajout (facultatif mais recommandé).**

**Clé primaire :** un identifiant unique auto-incrémenté.

**Nom de la table :** `trainers`

# Connexion à MySQL en tant que root

Avant toute chose, se connecter à MySQL en tant qu'administrateur :

```
1 mysql -u root -p
```

Entrez ensuite votre mot de passe.

# Création de la base de données annuaire\_php

Créer une base dédiée au projet avec encodage UTF-8 :

```
1 CREATE DATABASE trainers_db CHARACTER SET utf8mb4 COLLATE  
   utf8mb4_general_ci;
```

# Sélection de la base de données

Se positionner sur la base créée :

```
1 USE trainers_db;
```

# Création de la table trainers

Créer une table pour stocker les informations des formateurs :

```
1 CREATE TABLE trainers (  
2   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
3   first_name VARCHAR(100) NOT NULL,  
4   last_name VARCHAR(100) NOT NULL,  
5   email VARCHAR(150) NOT NULL UNIQUE,  
6   domain VARCHAR(100) NOT NULL,  
7   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
8 );
```

# Création d'un utilisateur SQL restreint

Créer un utilisateur pour sécuriser l'accès au projet :

```
1 CREATE USER 'trainers_db_user'@'localhost' IDENTIFIED BY  
  'phpsecure';
```

# Attribution des privilèges minimaux

Accorder uniquement les droits nécessaires à l'utilisateur :

```
1 GRANT SELECT, INSERT, UPDATE, DELETE ON trainers_db.* TO  
  'trainers_db_user'@'localhost';
```

Actualiser les droits :

```
1 FLUSH PRIVILEGES;
```



# Test de connexion avec l'utilisateur restreint

Tester la connexion depuis le terminal :

```
1 mysql -u trainers_db_user -p
```

Puis dans MySQL :

```
1 USE trainers_db;  
2 SHOW TABLES;
```

# Test d'insertion de données avec l'utilisateur restreint

Tester l'insertion depuis MySQL :

```
1 INSERT INTO trainers (first_name, last_name, email, domain)
2 VALUES
3 ('WOLF', 'Woof', 'woof@wolf.com', 'Maths and IT'),
4 ('BIG', 'Mac', 'Big@Mc.do', 'Python and Topology');
```

# Fichier PHP de connexion à la base de données

Dans `includes/db.php`, ajouter le code suivant :

```
1 <?php
2 $host = 'localhost';
3 $dbname = 'trainers_db';
4 $user = 'trainers_db_user';
5 $pass = 'phpsecure';
6 try {
7     $pdo = new
8         PDO("mysql:host=$host;dbname=$dbname;charset=utf8mb4",
9             $user, $pass);
10     $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
11 } catch (PDOException $e) {
12     die('Erreur de connexion : ' . $e->getMessage());
13 }
```

# Test local avec un serveur PHP

Lancer le serveur depuis la racine du projet :

```
1 php -S localhost:8000
```

Créer un fichier temporaire test.php :

```
1 <?php require 'includes/db.php'; echo 'Connexion réussie.'; ?>
```

Afficher la page via le navigateur : <http://localhost:8000/test.php>

# Accès à phpMyAdmin sur macOS (via Homebrew)

- phpMyAdmin est une interface graphique pour gérer MySQL en local.
- Sous macOS, on utilise Homebrew pour l'installer.

## Étapes :

### 1. Installer phpMyAdmin :

```
1 brew install phpmyadmin
```

### 2. Lancer le serveur local depuis le répertoire partagé :

```
1 cd /opt/homebrew/share/phpmyadmin  
2 php -S localhost:8080
```

### 3. Accéder à l'interface dans le navigateur : <http://localhost:8080>

# Accès à phpMyAdmin sur Unix/Linux (Debian, Ubuntu)

- Sur Linux, phpMyAdmin est accessible via les paquets APT.

## Étapes :

1. Installer Apache, PHP et phpMyAdmin :

```
1 sudo apt update
2 sudo apt install apache2 php phpmyadmin
```

2. Redémarrer Apache :

```
1 sudo systemctl restart apache2
```

3. Accéder à phpMyAdmin : <http://localhost/phpmyadmin>

**Remarque :** Pendant l'installation, choisir le serveur apache2 lorsqu'on vous le demande.

# Accès à phpMyAdmin sur Windows (méthode recommandée)

**Recommandé : utiliser XAMPP**

**Étapes :**

1. Télécharger XAMPP :

<https://www.apachefriends.org/fr/index.html>

2. Installer et ouvrir le **XAMPP Control Panel**

3. Démarrer les modules :

- Apache
- MySQL

4. Accéder à phpMyAdmin dans le navigateur :

<http://localhost/phpmyadmin>

**Connexion :**

■ Utilisateur : root

■ Mot de passe : (laisser vide par défaut)

# phpMyAdmin sur Windows (installation manuelle avancée)

## Méthode sans XAMPP (pour utilisateurs expérimentés)

1. Installer PHP manuellement dans C:\php
2. Installer MySQL séparément depuis <https://dev.mysql.com>
3. Télécharger phpMyAdmin : <https://www.phpmyadmin.net>
4. Décompresser dans C:\phpmyadmin
5. Lancer un serveur PHP :

```
1 cd C:\phpmyadmin  
2 php -S localhost:8080
```

6. Ouvrir : <http://localhost:8080>

## Remarques :

- Nécessite que le répertoire C:\php soit dans le PATH
- Configuration à faire dans config.inc.php



# Création de la table MySQL

Voici un exemple de script SQL à exécuter via phpMyAdmin :

```
1 CREATE DATABASE IF NOT EXISTS trainers_db CHARACTER SET utf8mb4;
2
3 USE trainers_db;
4
5 CREATE TABLE trainers (
6     id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
7     first_name VARCHAR(100) NOT NULL,
8     last_name VARCHAR(100) NOT NULL,
9     email VARCHAR(150) NOT NULL UNIQUE,
10    domain VARCHAR(100) NOT NULL,
11    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
12 );
```

# Connexion PDO à la base de données

Créer un fichier `includes/db.php` pour centraliser la connexion :

```
1 <?php
2 $host = 'localhost';
3 $dbname = 'trainers_db';
4 $user = 'trainers_db_user';
5 $pass = 'phpsecure';
6 try {
7     $pdo = new
8         PDO("mysql:host=$host;dbname=$dbname;charset=utf8mb4",
9             $user, $pass);
10     $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
11 } catch (PDOException $e) {
12     die('Database connection failed: ' . $e->getMessage());
13 }
```

# Fichiers PHP de base à créer

Organisation recommandée : Même structure que précédemment.

Objectifs :

- `index.php` : page d'accueil, liste des formateurs ;
- `add.php` : formulaire d'ajout ;
- `update.php`, `delete.php` : à venir ;
- `includes/` : code réutilisable (connexion, en-têtes, etc.).

# Header et Footer communs

Fichier includes/header.php :

```
1 <?php
2 // header.php
3 ?>
4 <!DOCTYPE html>
5 <html lang="fr">
6 <head>
7     <meta charset="UTF-8">
8     <title>Annuaire des formateurs</title>
9     <link rel="stylesheet" href="assets/style.css">
10 </head>
```

# Header et Footer communs

Fichier includes/header.php :

```
1
2 <body>
3   <h1>Annuaire des formateurs</h1>
4   <nav>
5     <a href="index.php">Accueil</a> |
6     <a href="add.php">Ajouter un formateur</a>
7   </nav>
8   <hr>
```

# Fichier de pied de page

Fichier includes/footer.php :

```
1 <?php
2 // footer.php
3 ?>
4     <hr>
5     <footer>
6         <p>&copy; <?php echo date('Y'); ?> Annuaire PHP</p>
7     </footer>
8 </body>
9 </html>
```

**\*\* Premiers pas \*\***

# Ce que vous allez apprendre aujourd'hui

- Créer un formulaire HTML pour saisir un formateur ;
- Traiter les données envoyées en méthode POST ;
- Nettoyer et valider les champs reçus (sécurité) ;
- Insérer les données dans la base MySQL ;
- Comprendre la logique du couple formulaire + traitement.



# Fichier add.php – Formulaire

Le formulaire utilise la méthode POST et pointe vers lui-même :

```
1 <?php include 'includes/header.php'; ?>
2
3 <h2>Ajouter un formateur</h2>
4 <form action="add.php" method="post">
5     <label>Prénom :</label><br>
6     <input type="text" name="first_name" required><br>
7     <label>Nom :</label><br>
8     <input type="text" name="last_name" required><br>
9     <label>Email :</label><br>
10    <input type="email" name="email" required><br>
11    <label>Domaine :</label><br>
12    <input type="text" name="domain" required><br>
13    <br>
14    <button type="submit">Enregistrer</button>
15 </form>
```

# Traitement POST en bas de add.php

On ajoute le code de traitement tout en bas du fichier :

```
1 <?php
2 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
3     require 'includes/db.php';
4
5     $first_name = trim(htmlspecialchars($_POST['first_name']));
6     $last_name  = trim(htmlspecialchars($_POST['last_name']));
7     $email      = trim(htmlspecialchars($_POST['email']));
8     $domain     = trim(htmlspecialchars($_POST['domain']));
9
10    if (!empty($first_name) && !empty($last_name) &&
11        !empty($email) && !empty($domain)) {
12        $sql = "INSERT INTO trainers (first_name, last_name,
13                                     email, domain)
14               VALUES (:first_name, :last_name, :email,
15                       :domain)";
```

# Bonnes pratiques de sécurité

- Utiliser `htmlspecialchars()` pour éviter les failles XSS ;
  - Utiliser `trim()` pour enlever les espaces superflus ;
  - Vérifier la présence des champs via `!empty()` ;
  - Toujours utiliser les **requêtes préparées** (PDO ou mysqli) ;
  - Ne jamais faire confiance aux données entrantes.
- Ces protections doivent être mises en œuvre systématiquement.

## **\*\* Lecture des données \*\***

# Ce que vous allez apprendre aujourd'hui

- Récupérer les données enregistrées dans la base MySQL ;
- Utiliser une requête `SELECT` simple avec PDO ;
- Parcourir les résultats avec une boucle `foreach` ou `while` ;
- Structurer un tableau HTML d'affichage ;
- Ajouter des liens d'action vers des pages futures : `update.php`, `delete.php`.

# Récupérer les données avec PDO

Dans le fichier `index.php`, on inclut le fichier de connexion et on exécute une requête :

```
1 <?php
2 require 'includes/header.php';
3 require 'includes/db.php';
4
5 $sql = "SELECT * FROM trainers ORDER BY last_name";
6 $stmt = $pdo->query($sql);
7 $trainers = $stmt->fetchAll();
8 ?>
```

# Afficher dynamiquement les résultats

On boucle sur les formateurs pour les afficher dans une table HTML :

```
1 <h2>Liste des formateurs</h2>
2
3 <table border="1" cellpadding="8">
4   <thead>
5     <tr>
6       <th>Nom</th>
7       <th>Prénom</th>
8       <th>Email</th>
9       <th>Domaine</th>
10      <th>Actions</th>
11    </tr>
12  </thead>
13  <tbody>
14    <?php foreach ($trainers as $trainer): ?>
15      <tr>
```

# Améliorer l'affichage et l'accessibilité

Bonnes pratiques pour l'affichage :

- Utiliser une balise `<table>` bien structurée ;
- Ajouter un en-tête clair avec `<thead>` et `<th>` ;
- Encadrer les cellules avec `border` et `padding` ;
- Échapper toutes les données utilisateur avec `htmlspecialchars()` ;
- Préparer les liens pour les actions de mise à jour et de suppression.

Ces liens ne sont pas encore fonctionnels, mais ils permettent d'anticiper les prochaines étapes.



**\*\* CRUD \*\***

# Ce que vous allez apprendre aujourd'hui

- Modifier un formateur existant avec un formulaire pré-rempli ;
- Supprimer un formateur de la base en toute sécurité ;
- Utiliser des requêtes UPDATE et DELETE préparées ;
- Protéger l'application contre les erreurs courantes et failles classiques ;
- Ajouter des messages de succès et d'erreur dans l'interface.

# Fichier update.php (affichage du formulaire)

```
1 <?php
2 require 'includes/db.php';
3
4 if (isset($_GET['id'])) {
5     $id = intval($_GET['id']);
6     $stmt = $pdo->prepare("SELECT * FROM trainers WHERE id =
7         :id");
8     $stmt->execute([':id' => $id]);
9     $trainer = $stmt->fetch();
10 }
11 ?>
12 <form method="post">
13     <input type="hidden" name="id" value="<?php echo
14         $trainer['id']; ?>">
15     <label>Prénom :</label><br>
```

# Traitement POST dans update.php

```
1 <?php
2 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
3     $stmt = $pdo->prepare("UPDATE trainers SET
4         first_name = :first_name,
5         last_name = :last_name,
6         email = :email,
7         domain = :domain
8         WHERE id = :id");
9
10    $stmt->execute([
11        ':first_name' => trim($_POST['first_name']),
12        ':last_name'   => trim($_POST['last_name']),
13        ':email'       => trim($_POST['email']),
14        ':domain'      => trim($_POST['domain']),
15        ':id'          => intval($_POST['id'])
16    ]);
```

# Fichier delete.php

```
1 <?php
2 require 'includes/db.php';
3
4 if (isset($_GET['id'])) {
5     $id = intval($_GET['id']);
6     $stmt = $pdo->prepare("DELETE FROM trainers WHERE id = :id");
7     $stmt->execute([':id' => $id]);
8
9     echo "<p>Formateur supprimé avec succès.</p>";
10 }
11 ?>
```

# Bonnes pratiques de sécurité

- Toujours utiliser des requêtes préparées (`prepare`, `bindValue`) ;
- Ne jamais insérer directement des variables dans les requêtes SQL ;
- Échapper les sorties HTML avec `htmlspecialchars()` ;
- Utiliser `intval()` ou des filtres pour les identifiants ;
- Ajouter des messages d'erreurs ou de validation clairs.

## **\*\* Gestion des sessions et sécurité \*\***

# Ce que vous allez apprendre aujourd'hui

- Implémenter une page de connexion avec vérification des identifiants ;
- Gérer les sessions utilisateurs (`$_SESSION`) ;
- Afficher des éléments conditionnellement (si l'utilisateur est connecté) ;
- Nettoyer et commenter le code ;
- Préparer la soutenance avec un fichier README.



# Table users pour l'authentification avec rôles

```
1 CREATE TABLE users (  
2   id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
3   username VARCHAR(50) NOT NULL UNIQUE,  
4   password VARCHAR(255) NOT NULL,  
5   role ENUM('admin', 'standard') NOT NULL DEFAULT 'standard'  
6 );
```

On insérera un utilisateur de test avec un mot de passe hashé (ex : via `password_hash()`) et un rôle associé (admin ou standard).

```
1 php -r "echo password_hash('admin123', PASSWORD_DEFAULT);"
```

# Ajouter un utilisateur avec un mot de passe sécurisé

Avant de tester le formulaire, insérons un utilisateur dans la table `users` avec un mot de passe hashé.

## 1. Générer un mot de passe hashé en PHP :

```
1 <?php
2 echo password_hash('admin123', PASSWORD_DEFAULT);
3 ?>
```

## 2. Copier le hash généré (exemple) :

\$2y\$10\$kpZqgOnz9J3qIkHWTN1SGONKxG3rB4eAVYeC7v2YutRIfQdk4  
/xPC

# Ajouter un utilisateur avec un mot de passe sécurisé - Suite

## 3. Insérer dans MySQL :

```
1 INSERT INTO users (username, password)
2 VALUES ('admin',
    '$2y$10$kpZqg0nz9J3qIkHWTN1SGONKxG3rB4eAVYeC7v2YutRIfQdk4/xPC');
```

Ce compte test peut être utilisé pour la connexion : admin / admin123.

# Insertion d'utilisateurs avec rôles

Pour insérer un utilisateur admin avec le rôle admin et un utilisateur user avec le rôle standard, nous allons utiliser la commande SQL suivante :

```
1 -- Insérer l'utilisateur admin avec rôle admin
2 INSERT INTO users (username, password, role)
3 VALUES ('admin',
4         '$2y$12$TfEWmesLWhy0y1oU7p6ePeG9CBQNVdDKF0.FbFpi8S2rsEZ1xmPgy',
5         'admin');
6
7 -- Insérer l'utilisateur user avec rôle standard
8 INSERT INTO users (username, password, role)
9 VALUES ('user',
10        '$2y$12$1gx9BBGA8HNRlRRTiIDDa.61HqDT3TsSiwCir0vv3lmu/rKnctgV2%',
11        'standard');
```

**Remarque :** Les mots de passe sont stockés sous forme de hash (utilisez `password hash()` pour les générer).

# Fichier login.php

```
1 <form method="post">
2   <label>Nom d'utilisateur :</label><br>
3   <input type="text" name="username" required><br>
4
5   <label>Mot de passe :</label><br>
6   <input type="password" name="password" required><br>
7
8   <button type="submit">Connexion</button>
9 </form>
```

# Connexion avec session PHP

```
1 <?php
2 session_start();
3 require 'includes/db.php';
4
5 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
6     $stmt = $pdo->prepare("SELECT * FROM users WHERE username =
7         :username");
8     $stmt->execute([':username' => $_POST['username']]);
9     $user = $stmt->fetch();
10
11     if ($user && password_verify($_POST['password'],
12         $user['password'])) {
13         $_SESSION['user'] = $user['username'];
14         echo "<p>Connexion réussie.</p>";
15     } else {
16         echo "<p>Identifiants incorrects.</p>";
17     }
18 }
```

# Afficher le nom d'utilisateur connecté

Dans le fichier `includes/header.php`, ajouter :

```
1 <?php session_start(); ?>
2 <nav>
3     <?php if (isset($_SESSION['user'])): ?>
4         <p>Bienvenue, <?php echo $_SESSION['user']; ?> | <a
           href="logout.php">Déconnexion</a></p>
5     <?php else: ?>
6         <a href="login.php">Connexion</a>
7     <?php endif; ?>
8 </nav>
```

**\*\* Rendu final \*\***



# Objectifs finaux

- Présenter le projet en groupe de manière professionnelle ;
- Valoriser les compétences acquises pendant la formation ;
- Rendre les livrables : code source, README, export base SQL ;
- Réfléchir aux améliorations possibles du projet ;
- Faire un retour d'expérience et s'autoévaluer.

# Présentation orale (10 minutes)

Chaque groupe présente :

- Le nom du projet et son objectif ;
- Une démonstration du site (ajout, affichage, modification, suppression) ;
- Les difficultés rencontrées et comment elles ont été surmontées ;
- Le rôle de chaque membre ;
- Les éléments techniques clefs (sécurité, sessions, CRUD).

# Livrables finaux à rendre

- Code source complet dans une archive ZIP ou un dépôt Git ;
- Fichier `README.md` documenté (structure du projet, base de données, instructions d'installation) ;
- Export SQL de la base (via `phpMyAdmin`) ;
- Facultatif : support PDF de la soutenance.

## **\*\* Les bases de MySQL \*\***

# Connexion à MySQL

Se connecter à MySQL avec un utilisateur existant :

```
1 mysql -u root -p
```

Puis entrer le mot de passe associé à l'utilisateur.

# Commandes de gestion des bases de données

```
1 CREATE DATABASE nom_de_la_base;  
2 SHOW DATABASES;  
3 USE nom_de_la_base;  
4 DROP DATABASE nom_de_la_base;
```

Ces commandes permettent de créer, lister, sélectionner ou supprimer des bases de données.

# Commandes de gestion des tables

```
1 CREATE TABLE nom_table (...);  
2 SHOW TABLES;  
3 DESCRIBE nom_table;  
4 DROP TABLE nom_table;
```

Utilisées pour créer, afficher, décrire ou supprimer des tables dans la base active.

# Commandes de manipulation des données (CRUD)

## Insérer des données :

```
1 INSERT INTO nom_table (col1, col2) VALUES ('val1', 'val2');
```

## Lire les données :

```
1 SELECT * FROM nom_table;  
2 SELECT col1, col2 FROM nom_table;
```

## Mettre à jour :

```
1 UPDATE nom_table SET col1 = 'valeur' WHERE id = 1;
```

## Supprimer :

```
1 DELETE FROM nom_table WHERE id = 1;
```



# Commandes liées aux utilisateurs

```
1 CREATE USER 'nom'@'localhost' IDENTIFIED BY 'motdepasse';  
2 GRANT ALL PRIVILEGES ON base.* TO 'nom'@'localhost';  
3 SHOW GRANTS FOR 'nom'@'localhost';  
4 FLUSH PRIVILEGES;
```

Ces commandes permettent de créer un utilisateur SQL et de lui attribuer des droits sur une base.

# Commandes système utiles

```
1 EXIT;
```

Quitter une session MySQL proprement.

```
1 SHOW VARIABLES LIKE 'datadir';  
2 SHOW VARIABLES LIKE 'port';
```

Afficher des informations sur la configuration du serveur MySQL.