

1.1. Exercice d'application 20 : test de la connexion à la base de données sur plusieurs pages

Mettre en place un **module complet de gestion des étudiants**, première brique fonctionnelle de la **plateforme de gestion d'étudiants et de QCM**.

Vous allez :

1. Vérifier que la **connexion à MySQL avec PDO fonctionne** sur deux pages.
2. Implémenter les **fonctionnalités de création, modification, suppression et affichage** des étudiants.
3. Organiser votre projet de manière claire, avec un **fichier de fonctions**, et utiliser **Bootstrap** pour une interface propre.

Étape 1 : test de la connexion MySQL

Dans cette première étape, vous allez **vérifier que le script de connexion PDO fonctionne correctement**. Cette vérification se fera dans deux pages différentes qui serviront de base au projet.

- Créez un dossier 20-application dans le dossier Web de votre serveur Apache :

C:/xampp/htdocs/Web/20-application

Dans ce dossier, vous devez :

1. **Créer un fichier db.php** contenant la connexion PDO à une base de données.
2. **Créer les fichiers index.php et etudiants.php**.
3. Dans **index.php** et **etudiants.php**, vous devez :
 - inclure le fichier db.php ;
 - afficher un message simple prouvant que la connexion est bien établie.

- Créer un dossier pages dans lequel vous allez mettre la page d'inscription et de connexion : inscription.php et connexion.php.

Vous devez inclure le fichier etudiants.php dans index.php, car cette page deviendra **la page d'accueil du projet** et sera utilisée pour rediriger vers le module de gestion des étudiants.

Étape 2 : mise en place de la gestion des étudiants

Base de données à utiliser

Créez une base de données nommée plateforme_qcm, avec la table suivante :

```
CREATE TABLE etudiants (
    id INT AUTO_INCREMENT PRIMARY KEY,
    prenom VARCHAR(100) NOT NULL,
    nom VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    VARCHAR(255) NOT NULL,
    date_inscription TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

Structure des fichiers à créer

Fichier	Fonction
db.php	Connexion PDO centralisée
etudiants.php	Contiendra toutes les fonctions (ajout, modification, suppression, affichage)

index.php	Page d'accueil qui inclut etudiants.php et contenir une barre de navigation et des liens vers la page d'inscription et de connexion
inscription.php	Doit inclure etudiant.php et permettre de renseigner un formulaire d'inscription et d'être enregistrer dans la base de données
connexion.php	Doit inclure etudiant.php et un formulaire de connexion qui permet de vérifier si les informations saisies sont correctes

Pour cette version, **vous devez inclure toutes les fonctions directement dans etudiants.php.**

```
function ajouterEtudiant(PDO $conn, string prenom, string $nom, string $email, string $motdepasse)

function modifierEtudiant(PDO $conn, int $id, string prenom, string $nom, string $email, string $motdepasse)

function supprimerEtudiant(PDO $conn, int $id)

function recupererEtudiants(PDO $conn)

function recupererEtudiantParId(PDO $conn, int $id)
```

Ces fonctions doivent :

- utiliser des requêtes préparées avec `prepare()` et `execute()` ;
- retourner un booléen ou un tableau ;
- afficher des messages propres et clairs si besoin.

Fonctionnalités à tester :

Action	Résultat attendu
Inscription	Message inscription réussi
Email déjà existant	Message d'erreur
Champs vides	Message d'erreur
Connexion	Message connexion réussi
Email incorrect	Message d'erreur
Mot de passe incorrect	Message d'erreur

Astuce

Vous pouvez regrouper toutes les **alertes Bootstrap** dans une fonction `afficherMessage()` pour centraliser le rendu des messages.

Centraliser la connexion dans un fichier unique vous permet :

- D'éviter la duplication du code de connexion.
- De faciliter la maintenance : si les paramètres changent, vous n'avez qu'un seul fichier à modifier.
- De construire un projet structuré, prêt à évoluer vers une architecture MVC dans les supports suivants.

Mise en garde

Ne modifiez jamais directement la base sans validation. Utilisez toujours des **requêtes préparées**. Pour supprimer un étudiant, **affichez une confirmation ou passez par une action spécifique avec `$_GET['supprimer']`**.

Exercice : Directives de réalisation du projet fil rouge

Ce projet est la suite de l'application 20. Vous pouvez le copier dans un dossier C:/xampp/htdocs/Web/plateforme_qcm.

Vous allez construire, étape par étape, une **plateforme web de gestion d'étudiants et de QCM en ligne**. Cette application servira de **projet fil rouge** pour tout le reste de la formation, et évoluera vers une version orientée objet dans le support suivant.

L'objectif est de manipuler **tous les concepts introduits dans le support 03**, notamment :

- La connexion à une base de données avec PDO.
- Les requêtes SQL sécurisées.
- La gestion des utilisateurs, des rôles, des sessions.
- L'interface utilisateur avec Bootstrap.
- L'organisation du projet et l'écriture de fonctions réutilisables.

Étapes obligatoires à implémenter

1. Connexion PDO centralisée

- Fichier unique contenant la connexion.
- Gestion propre des erreurs (bloc try/catch).

2. Gestion des utilisateurs (étudiants et admin)

- Inscription avec validation des champs.
- Vérification de l'email en base.
- Stockage sécurisé du mot de passe (password_hash()).

- Connexion avec vérification (password_verify()).
- Sessions pour maintenir l'état de connexion.

3. Page de profil

- Accessible uniquement si connecté.
- Affiche les informations utilisateur.
- Permet la personnalisation de l'interface (thème clair/sombre via cookies).

4. Gestion des rôles (admin vs étudiant)

- Un utilisateur peut avoir le rôle admin ou etudiant.
- L'admin peut :
 - Consulter tous les utilisateurs.
 - Modifier ou supprimer un utilisateur.
 - Ajouter des étudiants manuellement.
- L'étudiant peut :
 - Voir ses informations.
 - Accéder à l'espace QCM plus tard.

5. Modélisation relationnelle

Entité	Attributs	Clés
utilisateurs	id (PK), numero_etudiant (unique), nom, prenom, email, mot_de_passe, role, date_inscription	PK : id
qcms	id (PK), titre, description	PK : id

questions	id (PK), qcm_id (FK), texte_question	PK : id FK : qcm_id → qcms(id)
reponses	id (PK), question_id (FK), texte_reponse, est_correcte (booléen)	PK : id FK : question_id → questions(id)
resultats	id (PK), utilisateur_id (FK), qcm_id (FK), score, date_passe	PK : id FK : utilisateur_id → utilisateurs(id), qcm_id → qcms(id)
notes	id (PK), utilisateur_id (FK), matiere, valeur, date_note	PK : id FK : utilisateur_id → utilisateurs(id)

Vous devez :

1. **Créer toutes ces tables** dans votre base plateforme_qcm.
2. **Insérer quelques données initiales**, notamment un compte admin.
3. Implémenter un **système de connexion complet**.
4. Afficher un dashboard (Tableau de bord).
5. Permettre d'afficher son profil pour voir **ses informations**.
6. Pour l'administrateur :
 1. **Afficher** la liste des étudiants
 2. **Voir les détails d'un étudiant**
 3. **Ajouter** un étudiant
 4. **Modifier** un étudiant
 5. **Supprimer** un étudiant

Suggestions de modules à réaliser pour aller plus loin

- Système de **pagination** sur la liste des utilisateurs.
- Fonction “**Se souvenir de moi**” avec cookies.
- Import et export de la liste des étudiants en **fichier CSV**.
- Interface plus poussée en utilisant les composants Bootstrap (modales, alertes, etc.).