Créer un système d'inscription & connexion en PHP / MySQL

Version SÉCURISÉE

1. Objectif

Créer un système sécurisé :

- Mot de passe hashé
- Protection contre l'injection SQL
- Vérification correcte de l'email
- Aucune fuite d'information
- Sessions propres

Structure du projet

```
/projet-auth/
    index.php
    register.php
    login.php
    logout.php
    config.php
```

---

2. Base de données

> Même structure que la version non sécurisée — pas besoin de changer (ni de créer une nouvelle table) :

```sql
CREATE TABLE register (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nom VARCHAR(50),
    prenom VARCHAR(50),
    mail VARCHAR(100) UNIQUE,
    mdp VARCHAR(255)
);
```

Note : champ `mdp` assez long pour stocker un hash.

---

3. config.php

```php
<?php
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
$connection = mysqli_connect("localhost", "root", "", "cours_auth");
$connection->set_charset("utf8mb4");

if (!$connection) {
    die("Erreur de connexion à la base de données.");
}
?>
```

---

4. Page d'INSCRIPTION sécurisée — register.php

Points améliorés :

- Préparation des requêtes (`prepare`)
- Hash du mot de passe (`password_hash`)
- Vérification uniquement de l'email
- Aucune concaténation SQL
- Messages neutres

Code sécurisé

```php
<?php
require('config.php');
session_start();

$message = "";

if (isset($_POST['submit'])) {

    $nom    = trim($_POST["nom"]);
    $prenom = trim($_POST["prenom"]);
    $mail   = trim($_POST["mail"]);
    $mdp    = $_POST["mdp"];

    // Vérifier si l'email existe déjà
    $stmt = $connection->prepare("SELECT id FROM register WHERE mail =
?");
    $stmt->bind_param("s", $mail);
    $stmt->execute();
    $stmt->store_result();

    if ($stmt->num_rows > 0) {
        $message = "Ce compte existe déjà.";
    } else {

        // Hash sécurisé
```

```
        $mdpHash = password_hash($mdp, PASSWORD_DEFAULT);

        // Insertion sécurisée
        $stmt = $connection->prepare("
            INSERT INTO register (nom, prenom, mail, mdp)
            VALUES (?, ?, ?, ?)
        ");
        $stmt->bind_param("ssss", $nom, $prenom, $mail, $mdpHash);
        $stmt->execute();

        header("Location: login.php");
        exit;
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Inscription sécurisée</title>
</head>
<body>

<h1>Créer un compte</h1>

<?php if ($message != "") echo "<p style='color:red'>$message</p>"; ?>

<form action="" method="post">
    <input type="text" name="nom" placeholder="Nom" required><br><br>
    <input type="text" name="prenom" placeholder="Prénom" required><br>
<br>
    <input type="email" name="mail" placeholder="Email" required><br><br>
    <input type="password" name="mdp" placeholder="Mot de passe" required>
<br><br>

    <input type="submit" name="submit" value="Valider">
</form>

<p>Déjà un compte ? <a href="login.php">Connexion</a></p>

</body>
</html>
```

---

5. Page de CONNEXION sécurisée — login.php

Points améliorés :

- Sélection *uniquement* par email
- Vérification du hash avec `password_verify()`
- Erreur unique (évite fuite : email existe ou pas)

- Session sécurisée

Code sécurisé

```php
<?php
require('config.php');
session_start();

$message = "";

if (isset($_POST['submit'])) {

    $mail = trim($_POST["mail"]);
    $mdp  = $_POST["mdp"];

    // On récupère l'utilisateur seulement par email
    $stmt = $connection->prepare("SELECT id, prenom, mdp FROM register
WHERE mail = ?");
    $stmt->bind_param("s", $mail);
    $stmt->execute();

    $result = $stmt->get_result();

    if ($result->num_rows === 1) {

        $user = $result->fetch_assoc();

        // Vérification du mot de passe hashé
        if (password_verify($mdp, $user['mdp'])) {

            // On initialise la session
            $_SESSION['user'] = $user['prenom'];

            // Sécurisation : régénération de session
            session_regenerate_id(true);

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

    // Message neutre (ne dit pas si email existe)
    $message = "Identifiants incorrects.";
}
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Connexion sécurisée</title>
```

```
</head>
<body>

<h1>Connexion</h1>

<?php if ($message != "") echo "<p style='color:red'>$message</p>"; ?>

<form action="" method="post">
    <input type="email" name="mail" placeholder="Email" required><br><br>
    <input type="password" name="mdp" placeholder="Mot de passe" required>
<br><br>

    <input type="submit" name="submit" value="Se connecter">
</form>

<p>Pas encore de compte ? <a href="register.php">Inscription</a></p>

</body>
</html>
```

---

6. Page d'accueil sécurisée — index.php

Même principe que la version non sécurisée.

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Accueil</title>
</head>
<body>

<h1>Accueil</h1>

<?php if (!isset($_SESSION['user'])): ?>

    <p>Vous n'êtes pas connecté.</p>
    <a href="login.php">Connexion</a> |
    <a href="register.php">Inscription</a>

<?php else: ?>

    <p>Bienvenue, <strong><?php echo htmlspecialchars($_SESSION['user']);
?></strong> !</p>
    <a href="logout.php">Se déconnecter</a>

<?php endif; ?>
```

```
</body>
</html>
```

---

7. Déconnexion — logout.php

```php
<?php
session_start();
$_SESSION = [];
session_destroy();
setcookie(session_name(), '', time() - 3600); // Supprime le cookie de session
header("Location: index.php");
exit;
?>
```

---

8. Schéma de la version sécurisée

```
[Formulaire HTML]
       ↓
[PHP → nettoyage / trim]
       ↓
[Prepared Statement → SELECT email]
       ↓
[password_verify()]
       ↓
[Connexion OK → session_regenerate_id()]
       ↓
[Redirection]
```

---

9. Conclusion

La version sécurisée n'est pas juste une « mise à jour » du code vulnérable — c'est **un changement de posture** : on passe de « faire fonctionner » à « faire fonctionner correctement et en sécurité ».

Le vrai gain : la confiance

Lorsqu'on protège les données utilisateurs, on ne sécurise pas seulement une base SQL :
on protège **la confiance** que les utilisateurs placent dans notre application.

> La sécurité n'est pas une option technique. C'est un contrat moral entre notre code et celui qui l'utilise.

Exercice pratique — Construire votre mini système d'authentification sécurisé

Objectif

Reproduire le fonctionnement complet d'un système d'inscription et de connexion **sécurisé** en PHP/MySQL en respectant toutes les bonnes pratiques étudiées.

L'objectif n'est pas d'avoir un "beau site", mais un code **sûr, propre, et compréhensible**.

Énoncé

Vous allez coder un projet minimal d'authentification avec :

1. Une page d'inscription (`register.php`)
2. Une page de connexion (`login.php`)
3. Une page d'accueil sécurisée (`index.php`)
4. Une page de déconnexion (`logout.php`)
5. Un fichier de configuration (`config.php`)

Contraintes techniques à respecter

Catégorie	Exigence
Base de données	Même structure que celle du cours (`register`).
Sécurité SQL	Obligatoirement des requêtes préparées `prepare() + bind_param()`.
Mots de passe	Obligatoirement hashés avec `password_hash()`.
Vérification	Authentification uniquement basée sur l'email, puis vérification via `password_verify()`.
Sessions	Régénération obligatoire après connexion : `session_regenerate_id(true)`.
Messages d'erreurs	Messages neutres : ne jamais préciser "email incorrect" ou "mot de passe faux".
HTML	Tous les affichages de données utilisateurs doivent passer par `htmlspecialchars()`.

Étapes

1. **Créer la base `cours_auth`** si elle n'existe pas encore.
2. **Coder `config.php`** avec connexion MySQL sécurisée et `mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT)`.
3. **Faire le formulaire d'inscription.**
   - Récupérer `$_POST`.
   - Vérifier que les champs ne sont pas vides.
   - Vérifier si l'email existe déjà.

    - Hasher le mot de passe.
    - Enregistrer via requête préparée.
4. **Faire le formulaire de connexion.**
    - Vérifier l'email.
    - Comparer le mot de passe avec `password_verify()`.
    - Si ok → créer session + redirection vers `index.php`.
5. **Afficher le prénom de l'utilisateur connecté** sur `index.php`.
6. **Créer la déconnexion.**
    - `session_start()`
    - `session_destroy()`
    - Redirection vers index.
7. **Tester volontairement** des attaques simples sur le formulaire de connexion pour vérifier que votre système résiste.

Critères d'évaluation

- Le code s'exécute sans erreur.
- Aucune injection SQL possible.
- Les mots de passe ne sont jamais stockés en clair.
- Les erreurs ne révèlent aucune donnée sensible.
- Le code est lisible et commenté.
- Le comportement est identique après régénération de session.

À retenir

> La sécurité n'est jamais un ajout tardif.
> C'est une **discipline quotidienne** dans la manière d'écrire chaque ligne de code.
