

# **Documentation Technique**

## **MyCoach**

CARCENAC Mathis

## **TABLE DES MATIÈRES**

### **1. Installation**

### **2. Langages utilisés**

### **3. Déstructuration des besoins (Fonctionnalités)**

### **4. Architecture**

### **5. Base de données**

- Installation de la base de données (Déploiement)
- Exportation de la base de données

### **6. Commandes Git**

- Envoyer sur un repository à distance
- Retirer depuis un repository à distance (Déploiement)

### **7. Sécurités & Collecte des données**

- Comment les données sont collectées
- Chiffrement des mots de passes
- Protection face aux attaques par injection SQL et XSS

### **8. Différents scripts utiles**

## Installation

Le site web “MyCoach” est hébergé sur un serveur local grâce à la plateforme de développement Web de type WAMP nommé WampServer, elle permet de faire fonctionner localement les scripts PHP et est aussi utilisé pour la base de données avec PHPMyAdmin qui est intégré à WampServer. Et l’IDE utilisée est Visual Studio Code. Pour le côté versionning et sauvegarde j’ai utilisé l’outil Github, et enfin pour le backlog, j’ai utilisé l’outil Trello



## Langages utilisés

Les langages utilisés sur le site MyCoach sont HTML pour le placement du contenu, le CSS pour m'occuper de tout le design du site web et enfin j'utilise PHP pour tous les scripts du site web tel que la connexion à la base de données. Et un petit peu de JavaScript pour filtrer des informations.

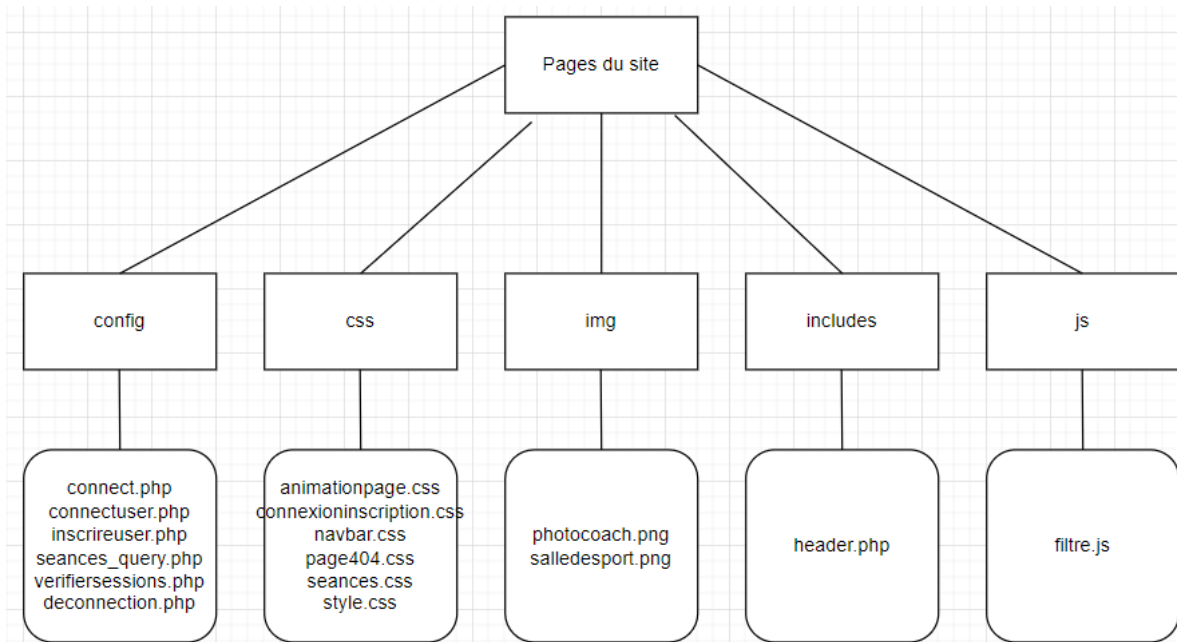


# Architecture

- Sprint 1 - Présentation d'une page vitrine du coach
  - Barre de navigation
  - Faire page de présentation de MyCoach avec un partie A propos et Activités
  - Gérer le code couleur de la page
  
- Sprint 2 - Page de détail des séances
  - Faire un schéma de la table SQL
  - Mise en place du tableau contenant les informations des séances
  - Faire une récupération des tables depuis le code de la page
  
- Sprint 3 - Gérer les utilisateurs connecté
  - Tous les utilisateurs peuvent voir la page d'accueil
  - Seulement les utilisateurs connectés peuvent accéder au séance
  
- Sprint 4 - Sécuriser et Organiser son code
  - Gestion des erreurs potentiels à la base de données
  - Proposer un code propre
  - Sauvegarde du projet (Github)
  
- Sprint 5 - Fonctionnalités Optionnelles
  - Page d'inscription au site
  - Éviter les attaques XML & XSS
  - Hasher les mots de passes dans la BDD

# Architecture

Concernant l'architecture du site, il est fait de sorte à ce qu'il y ait une catégorie pour chaque parties et utilisations pour le site, tout en haut on a les pages du site, et en dessous tous les documents utiles au site avec le fichier config qui va contenir tous les scripts, le document css qui va contenir tous les fichiers css, les images, les includes qui contient la barre de navigation à inclure pour chaque pages et enfin un fichier javascript qui va contenir le script des filtres.



# Base de données

La base de données comprend 4 tables, la table lieu, sport, seance et utilisateurs

```
CREATE TABLE IF NOT EXISTS `lieu` (  
  `numero_salle` int NOT NULL AUTO_INCREMENT,  
  `adresse` varchar(255) NOT NULL,  
  `ville` varchar(255) NOT NULL,  
  `code_postal` varchar(10) NOT NULL,  
  `nom_salle` varchar(50) DEFAULT NULL,  
  PRIMARY KEY (`numero_salle`)  
) ENGINE=InnoDB;
```

```
--  
-- Structure de la table `seance`  
--
```

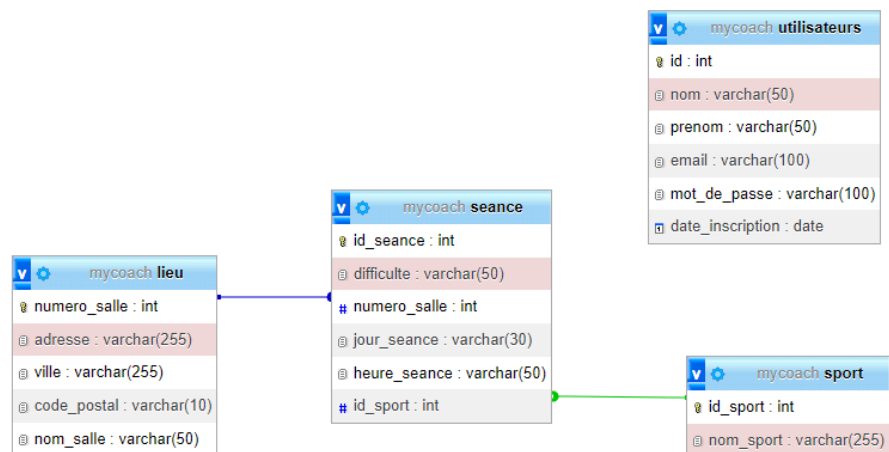
```
CREATE TABLE IF NOT EXISTS `seance` (  
  `id_seance` int NOT NULL AUTO_INCREMENT,  
  `difficulte` varchar(50) NOT NULL,  
  `numero_salle` int NOT NULL,  
  `jour_seance` varchar(30) NOT NULL,  
  `heure_seance` varchar(50) NOT NULL,  
  `id_sport` int NOT NULL,  
  PRIMARY KEY (`id_seance`),  
  KEY `id_sport` (`id_sport`),  
  KEY `numero_salle` (`numero_salle`)  
) ENGINE=InnoDB;
```

```
--  
-- Structure de la table `sport`  
--
```

```
CREATE TABLE IF NOT EXISTS `sport` (  
  `id_sport` int NOT NULL AUTO_INCREMENT,  
  `nom_sport` varchar(255) NOT NULL,  
  PRIMARY KEY (`id_sport`)  
) ENGINE=InnoDB;
```

```
--
-- Structure de la table `utilisateurs`
--

DROP TABLE IF EXISTS `utilisateurs`;
CREATE TABLE IF NOT EXISTS `utilisateurs` (
  `id` int NOT NULL AUTO_INCREMENT,
  `nom` varchar(50) NOT NULL,
  `prenom` varchar(50) NOT NULL,
  `email` varchar(100) NOT NULL,
  `mot_de_passe` varchar(100) NOT NULL,
  `date_inscription` date NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM;
```

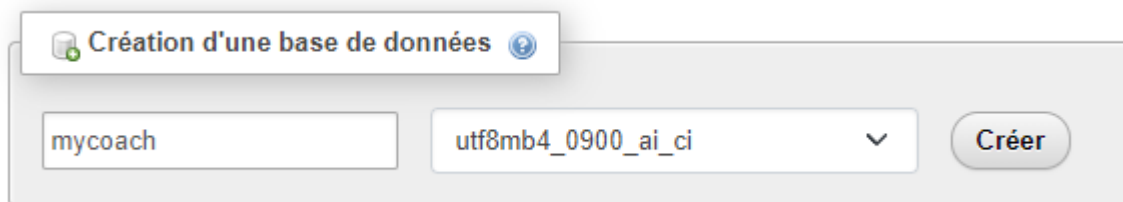


## Installation Base de données

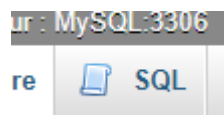
Pour installer la base de données vous aurez besoin de phpMyAdmin ou son équivalent, mais c'est cette interface que j'utilise via WampServer.

Pour que MyCoach soit connecté à la BDD l'identifiant doit être "root" et ne pas contenir de mot de passe car c'est écrit comme cela dans le code . Le nom de la bdd est "mycoach"

On va donc créer une base de données

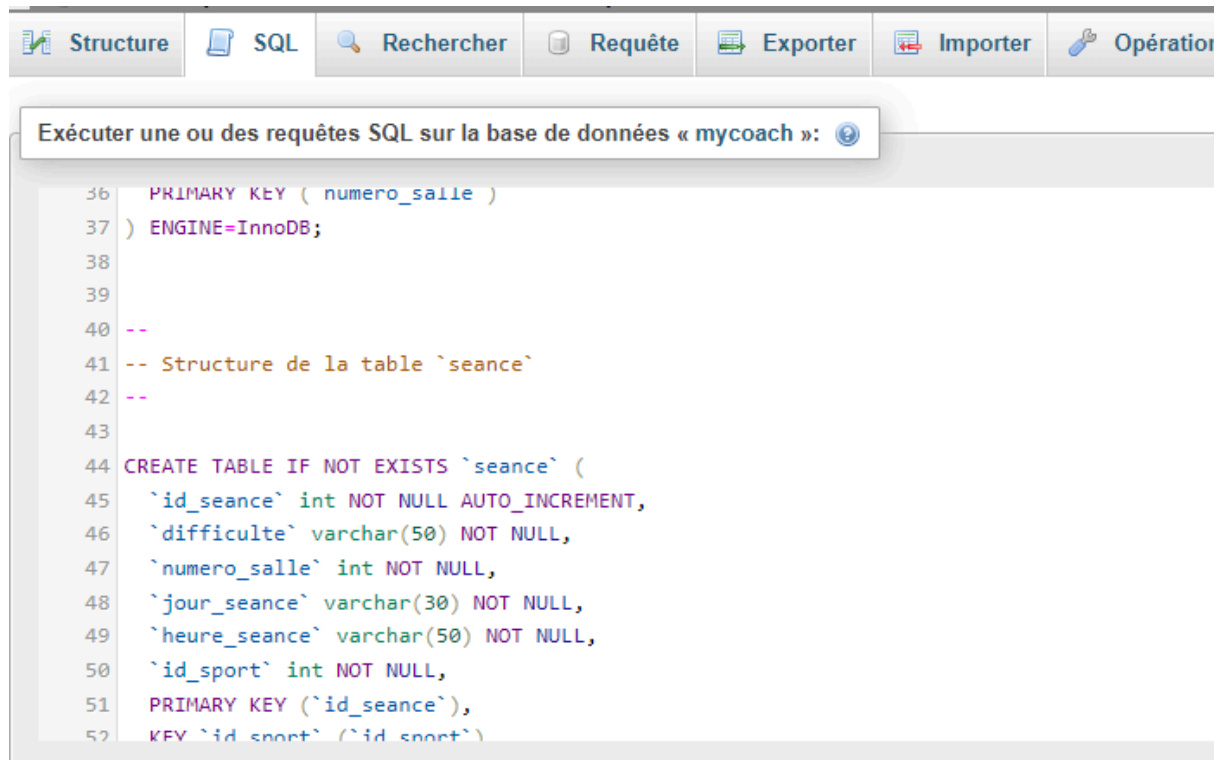


Une fois cela fait on va aller dans la catégorie "SQL" de phpMyAdmin

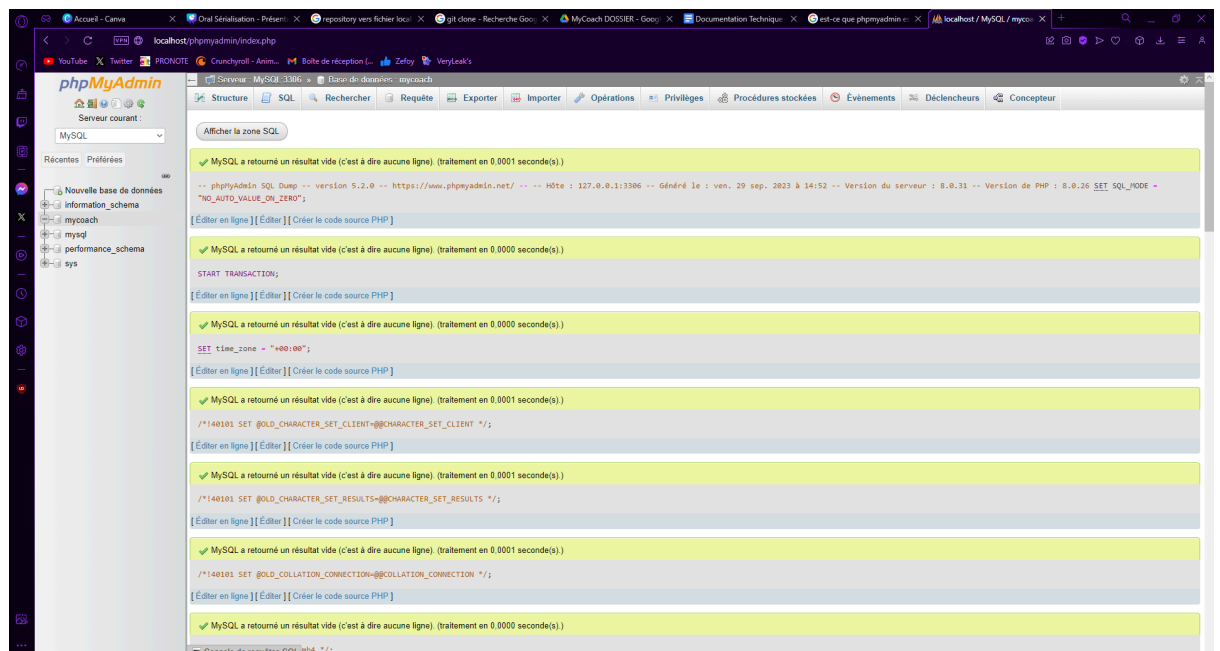




Et on Copie/Colle la requête SQL pour insérer les données et créer les tables qui se situent dans le fichier MyCoach :



Et après avoir copié/collé vous devez obtenir ce résultat :

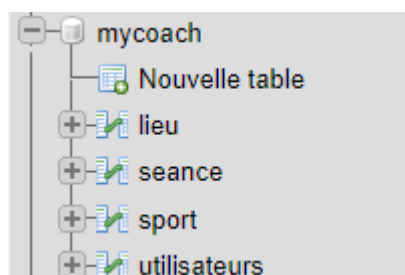


Et voici votre base de données remplis :

Table	Action	Lignes	Type	Interclassement	Taille	Perte
lieu	★ Parcourir Structure Rechercher Insérer Vider Supprimer	5	InnoDB	utf8mb4_0900_ai_ci	16,0 kio	-
seance	★ Parcourir Structure Rechercher Insérer Vider Supprimer	15	InnoDB	utf8mb4_0900_ai_ci	48,0 kio	-
sport	★ Parcourir Structure Rechercher Insérer Vider Supprimer	6	InnoDB	utf8mb4_0900_ai_ci	16,0 kio	-
utilisateurs	★ Parcourir Structure Rechercher Insérer Vider Supprimer	10	MyISAM	utf8mb4_0900_ai_ci	3,1 kio	-

## Exportation Base de données

Pour exporter la base de données depuis phpMyAdmin on se place



sur la base de données “MyCoach”

et on clique sur Exporter



Après avoir fait cela on clique sur le bouton “Exporter” en bas

Exportation des tables depuis la base de données « mycoach »

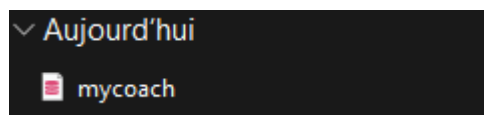
Méthode d'exportation :
☒ Rapide, n'afficher qu'un minimum d'options
☐ Personnalisée, afficher toutes les options possibles

Format :

SQL

Exporter

Et on obtient, en téléchargement le fichier mycoach.sql



Ce fichier qu'on va par la suite placer dans le dossier MyCoach pour l'envoyer sur Github par la suite.

# Commandes GIT

## Envoyer sur un Repository à distance

Voici les commandes qui m'ont permis d'insérer les fichiers depuis un fichier local sur un repository Github

Initialiser le Repository local

```
D-OZENNE+CARCENM@458-08-2 MINGW64 /c:/wamp64/www/MyCoach
$ git init
Initialized empty Git repository in C:/wamp64/www/MyCoach/.git/
```

On va créer la branch main sur Github

```
D-OZENNE+CARCENM@458-08-2 MINGW64 /c:/wamp64/www/MyCoach (master)
$ git branch -M main
```

On va ajouter tous les éléments du dossier ../www/MyCoach (le point sert à mettre tous les dossiers)

```
D-OZENNE+CARCENM@458-08-2 MINGW64 /c:/wamp64/www/MyCoach (main)
$ git add .
```

On va faire un commit avec un message "Commit 29-09-2023" qui sera affiché sur Github

```
D-OZENNE+CARCENM@458-08-2 MINGW64 /c:/wamp64/www/MyCoach (main)
$ git commit -m "Commit 29-09-2023"
[main (root-commit) eb38cbb] Commit 29-09-2023
14 files changed, 1020 insertions(+)
create mode 100644 .htaccess
create mode 100644 config/connect.php
create mode 100644 config/seances_query.php
create mode 100644 connection.php
create mode 100644 css/connexion.css
create mode 100644 css/navbar.css
create mode 100644 css/seances.css
create mode 100644 css/style.css
create mode 100644 erreur404.php
create mode 100644 img/photocoach.png
create mode 100644 includes/header.php
create mode 100644 index.php
create mode 100644 mycoach.sql
create mode 100644 seances.php
```

On va ajouter les documents locaux sur Github

```

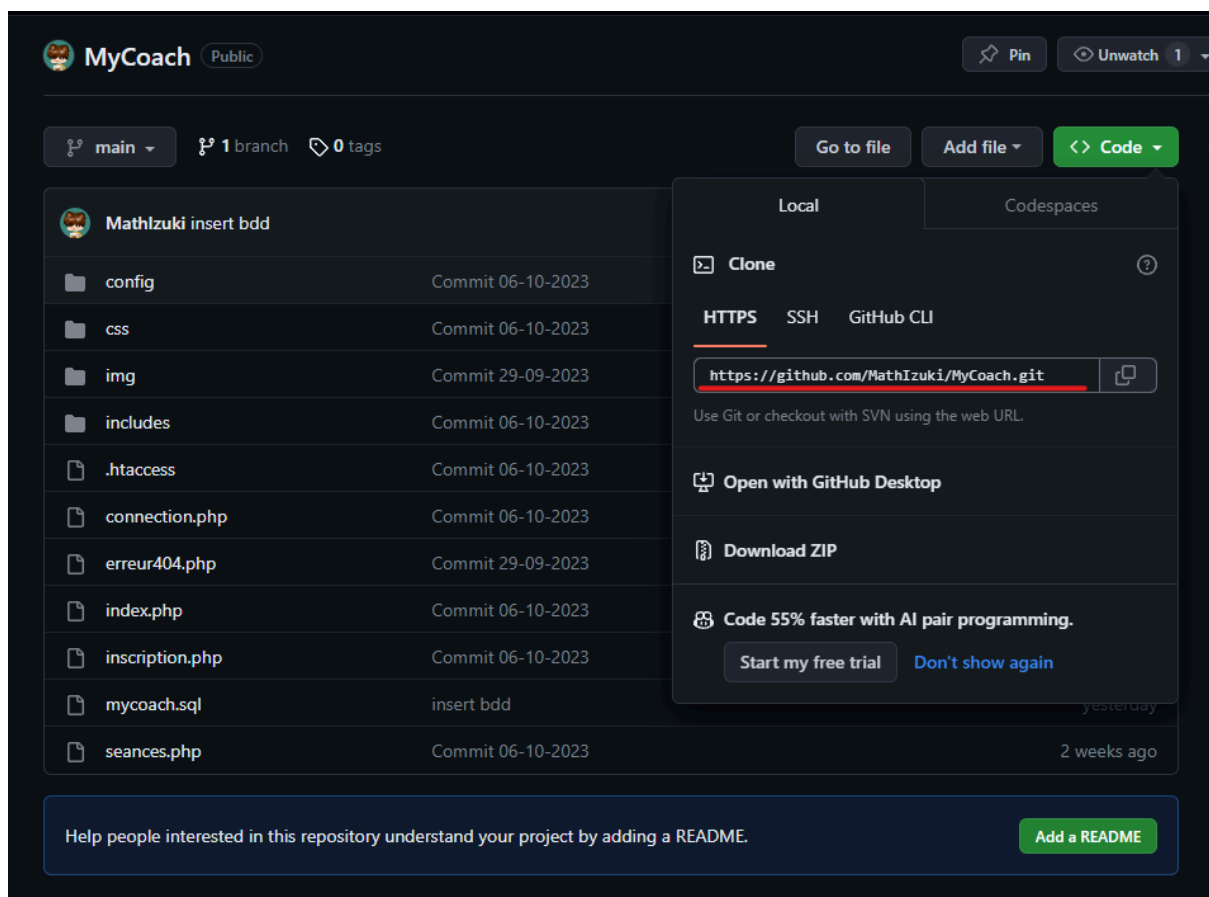
D-OZENNE+CARCENM@458-08-2 MINGW64 /c/wamp64/www/MyCoach (main)
$ git push origin main
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 4 threads
Compressing objects: 100% (17/17), done.
Writing objects: 100% (20/20), 1.03 MiB | 1.59 MiB/s, done.
Total 20 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MathIzuki/MyCoach.git
* [new branch]      main -> main

```

## Retirer depuis un repository à distance

Commandes pour cloner le repository Github vers un repository local pour pouvoir l'obtenir sur son ordinateur.

Sur Github on va aller dans le repository ou se situe son projet et on va Copier/Coller le lien du Repository.



Réinitialiser au cas où le Repository local

```
carce@Mathis MINGW64 /c/wamp64/www/MyCoach (main)
$ git init
Reinitialized existing Git repository in C:/wamp64/www/MyCoach/.git/
```

Par la suite on va ouvrir Git Bash dans le fichier voulu et on fait cette commande qui va importer les données vers le repo local

```
carce@Mathis MINGW64 /c/wamp64/www
$ git clone "https://github.com/MathIzuki/MyCoach.git"
Cloning into 'MyCoach'...
remote: Enumerating objects: 40, done.
remote: Counting objects: 100% (40/40), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 40 (delta 11), reused 36 (delta 7), pack-reused 0
Receiving objects: 100% (40/40), 1.04 MiB | 7.27 MiB/s, done.
Resolving deltas: 100% (11/11), done.
```

Une fois cela fait, tous les documents sont présents dans le fichier choisi pour le repository local, et il ne manque plus qu'à les ouvrir avec Visual Studio Code

# Sécurités & Collecte des données

## Comment les données sont collectées

Les données sont collectées lorsque l'utilisateur s'inscrit sur le site web et sont stockés dans la base de données, on va alors récupérer son nom, son prénom, son email, son mot de passe en dur qui va être chiffré.

### Chiffrement des mots de passes

Les mots de passe des utilisateurs sont chiffrés dans la base de données ce qui garantit la sécurité des mots de passe des utilisateurs.

Lors de l'inscription des utilisateurs, lorsqu'il saisira son mot de passe on utilisera une fonction php nommée "password\_hash" qui va hasher le mot de passe.

```
// Hasher le mot de passe (meilleures pratiques)
$hashedPassword = password_hash($password, PASSWORD_DEFAULT);

// Préparez la requête préparée avec des espaces réservés
$query = "INSERT INTO utilisateurs (nom, prenom, email, mot_de_passe, date_inscription) VALUES (?, ?, ?, ?, NOW())";
$stmt = $pdo->prepare($query);

// Liez les valeurs aux espaces réservés
$stmt->bindParam(1, $nom);
$stmt->bindParam(2, $prenom);
$stmt->bindParam(3, $email);
$stmt->bindParam(4, $hashedPassword);
```

Dans la base de données les mots de passes sont affichés de cette façon

				id	nom	prenom	email	mot_de_passe	date_inscripti
<input type="checkbox"/>	Éditer	Copier	Supprimer	1	Dupont	Jean	jean.dupont@example.com	\$2y\$12\$rQvAoH6i7SY0ijHTLqts2uEFvz0mczx3tkiQZI3qS.V...	2023-09-29
<input type="checkbox"/>	Éditer	Copier	Supprimer	2	Martin	Marie	marie.martin@example.com	\$2y\$12\$P9eDIFQcr3vKcFnR0lqMeeEwKQsAMIfk7QOpF5Fw3Us...	2023-09-30
<input type="checkbox"/>	Éditer	Copier	Supprimer	3	Lefebvre	Pierre	pierre.lefebvre@example.com	\$2y\$12\$rQvAoH6i7SY0ijHTLqts2uEFvz0mczx3tkiQZI3qS.V...	2023-10-01
<input type="checkbox"/>	Éditer	Copier	Supprimer	4	Dubois	Sophie	sophie.dubois@example.com	\$2y\$12\$P9eDIFQcr3vKcFnR0lqMeeEwKQsAMIfk7QOpF5Fw3Us...	2023-10-02
<input type="checkbox"/>	Éditer	Copier	Supprimer	5	Roux	Claude	clauderoux@example.com	\$2y\$12\$rQvAoH6i7SY0ijHTLqts2uEFvz0mczx3tkiQZI3qS.V...	2023-10-03
<input type="checkbox"/>	Éditer	Copier	Supprimer	6	Garcia	Isabelle	isabelle.garcia@example.com	\$2y\$12\$P9eDIFQcr3vKcFnR0lqMeeEwKQsAMIfk7QOpF5Fw3Us...	2023-10-04
<input type="checkbox"/>	Éditer	Copier	Supprimer	7	Fournier	Thomas	thomas.fournier@example.com	\$2y\$12\$rQvAoH6i7SY0ijHTLqts2uEFvz0mczx3tkiQZI3qS.V...	2023-10-05
<input type="checkbox"/>	Éditer	Copier	Supprimer	8	Moreau	Elodie	elodie.moreau@example.com	\$2y\$12\$P9eDIFQcr3vKcFnR0lqMeeEwKQsAMIfk7QOpF5Fw3Us...	2023-10-06
<input type="checkbox"/>	Éditer	Copier	Supprimer	9	Girard	Antoine	antoine.girard@example.com	\$2y\$12\$rQvAoH6i7SY0ijHTLqts2uEFvz0mczx3tkiQZI3qS.V...	2023-10-07
<input type="checkbox"/>	Éditer	Copier	Supprimer	10	Petit	Charlotte	charlotte.petit@example.com	\$2y\$12\$P9eDIFQcr3vKcFnR0lqMeeEwKQsAMIfk7QOpF5Fw3Us...	2023-10-08
<input type="checkbox"/>	Éditer	Copier	Supprimer	27	CARCENAC	Mathis	carcenacmathis81@gmail.com	\$2y\$10\$MPXqFbSgQD4sCQiNfH6FserHPowQNU69COu7JvlnQmE...	2023-10-17

Et enfin, lors de la connection des utilisateurs, on déchiffre le mot de passe hashé avec une autre fonction php “password\_verify”

```
// L'utilisateur existe, vérifiez le mot de passe
$row = $stmt->fetch(PDO::FETCH_ASSOC);
if (password_verify($MotDePasse, $row['mot_de_passe'])) {
    // Mot de passe correct, connectez l'utilisateur
    session_start();
    $_SESSION['MotDePasse'] = $MotDePasse;
    // Rediriger vers la page de bienvenue (ou toute autre page appropriée)
    header("Location: ../MyCoach/index.php");
    exit();
}
```

Grâce à ce chiffage et à ce déchiffage les données contenues dans la BDD sont chiffrées et sont donc sécurisées.

## Protection face aux attaques par injection SQL et XSS

Il y a aussi une sécurité qui prévient les attaques par injection SQL.

Lors de l'inscription :

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
    // Récupérez les données soumises par le formulaire  
    $nom = htmlentities($_POST['nom'], ENT_QUOTES, "UTF-8");  
    $prenom = htmlentities($_POST['prenom'], ENT_QUOTES, "UTF-8");  
    $email = htmlentities($_POST['email'], ENT_QUOTES, "UTF-8");  
    $password = $_POST['password'];
```

On utilise htmlentities qui va prévenir les attaques par injection SQL ou XSS, car elle va à chaque fois transformer des caractères spéciaux en la remplaçant par l'entité HTML correspondant (ex : "<" devient "&lt;")

(On applique pas ça à la variable \$password car le mot de passe est chiffré)

De même lors de la connexion de l'utilisateur :

```
// Les champs email & mdp sont bien postés et pas vides, on sécurise les données entrées par l'utilisateur  
$Email = htmlentities($_POST['email'], ENT_QUOTES, "UTF-8");  
$MotDePasse = $_POST['password'];
```



## Différents scripts utiles

Script permettant de vérifier que l'utilisateur est bien connecté (pour la page séance) :

```
<?php
    if (session_status() === PHP_SESSION_NONE) {
        session_start();
    }
    // Vérifier si la session de l'utilisateur n'est pas définie
    if (!isset($_SESSION['MotDePasse'])) {
        // L'utilisateur n'est pas connecté, rediriger vers la page de
        connexion
        header("Location: connection.php"); // Remplacez "login.php"
        par le chemin de votre page de connexion
        exit();
    }
?>
```

Script permettant de faire la requête pour recueillir les infos du tableau en prenant en compte le filtre de la difficulté et de la ville:

```
<?php
$difficulte_filter = isset($_POST['difficulte']) ? $_POST['difficulte']
: 'Toutes';
$ville_filter = isset($_POST['ville']) ? $_POST['ville'] : 'Toutes';

$query = "SELECT seance.difficulte, seance.jour_seance,
seance.heure_seance,
                lieu.nom_salle, lieu.adresse, sport.nom_sport,
lieu.ville, lieu.code_postal
FROM seance
INNER JOIN lieu ON seance.numero_salle = lieu.numero_salle
INNER JOIN sport ON seance.id_sport = sport.id_sport";

// Si une difficulté est sélectionnée, ajouter la clause WHERE
if ($difficulte_filter !== 'Toutes') {
    $query .= " WHERE seance.difficulte = :difficulte";
}

// Si une ville est sélectionnée et qu'il y avait déjà une clause WHERE
(à cause du filtre de difficulté), utilisez AND
```

```
if ($ville_filter !== 'Toutes') {
    $query .= ($difficulte_filter !== 'Toutes') ? " AND lieu.ville = :ville" : " WHERE lieu.ville = :ville";
}

$query .= " ORDER BY FIELD(seance.jour_seance, 'Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi', 'Dimanche')";

$stmt = $pdo->prepare($query);

// Si une difficulté est sélectionnée, bind la valeur
if ($difficulte_filter !== 'Toutes') {
    $stmt->bindParam(':difficulte', $difficulte_filter);
}

// Si une ville est sélectionnée, bind la valeur
if ($ville_filter !== 'Toutes') {
    $stmt->bindParam(':ville', $ville_filter);
}

$stmt->execute();

?>
```

Script permettant à l'utilisateur de s'inscrire de manière sécurisée :

```
<?php
include_once('connect.php');

$emailError = "";

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Récupérez les données soumises par le formulaire
    $nom = htmlentities($_POST['nom'], ENT_QUOTES, "UTF-8");
    $prenom = htmlentities($_POST['prenom'], ENT_QUOTES, "UTF-8");
    $email = htmlentities($_POST['email'], ENT_QUOTES, "UTF-8");
    $password = $_POST['password'];

    // Vérifiez que tous les champs sont remplis
    if (empty($nom) || empty($prenom) || empty($email) ||
empty($password)) {
        echo "Tous les champs sont obligatoires. Veuillez remplir tous
les champs.";
    } else {
        try {
            // Vérifiez si l'e-mail existe déjà dans la base de données
            $query = "SELECT COUNT(*) FROM utilisateurs WHERE email =
?";

            $stmt = $pdo->prepare($query);
            $stmt->bindParam(1, $email);
            $stmt->execute();
            $count = $stmt->fetchColumn();

            if ($count > 0) {
                // L'e-mail existe déjà, affichez un message d'erreur
                $emailError = "Cette adresse e-mail est déjà
enregistrée.";
            } else {
                // Hasher le mot de passe (meilleures pratiques)
                $hashedPassword = password_hash($password,
PASSWORD_DEFAULT);

                // Préparez la requête préparée avec des espaces
réservés

                $query = "INSERT INTO utilisateurs (nom, prenom, email,
mot_de_passe, date_inscription) VALUES (?, ?, ?, ?, NOW())";
                $stmt = $pdo->prepare($query);
```

```

        // Liez les valeurs aux espaces réservés
        $stmt->bindParam(1, $nom);
        $stmt->bindParam(2, $prenom);
        $stmt->bindParam(3, $email);
        $stmt->bindParam(4, $hashedPassword);

        // Exécutez la requête préparée
        if ($stmt->execute()) {
            // L'inscription a réussi
            session_start();
            $_SESSION['MotDePasse'] = $password;
            header("Location: ../MyCoach/index.php");
        } else {
            // Une erreur s'est produite lors de l'inscription
            echo "Erreur lors de l'inscription : " .
$stmt->errorInfo()[2];
        }
    }
} catch (PDOException $e) {
    // Gérez les erreurs de connexion PDO
    echo "Erreur de connexion à la base de données : " .
    $e->getMessage();
}
}
}
?>

```

Script permettant à l'utilisateur de se connecter de manière sécurisée : <?php

```
session_start();

// Initialisez les messages d'erreur à vide
$emailError = "";
$passwordError = "";

// Si le bouton "Submit" est cliqué
if (isset($_POST['submit'])) {
    // On vérifie que le champ "Email" n'est pas vide
    if (empty($_POST['email'])) {
        $emailError = "Le champ Email est vide.";
    } else {
        // On vérifie maintenant si le champ "Mot de passe" n'est pas vide"
        if (empty($_POST['password'])) {
            $passwordError = "Le champ Mot de passe est vide.";
        } else {
            // Les champs email & mdp sont bien postés et pas vides, on sécurise les données entrées par l'utilisateur
            $Email = htmlentities($_POST['email'], ENT_QUOTES, "UTF-8");

            $MotDePasse = $_POST['password'];

            // Inclure le fichier de connexion à la base de données
            include('connect.php');

            // Requête SQL pour rechercher l'utilisateur dans la base de données en utilisant l'email
            $query = "SELECT * FROM utilisateurs WHERE email = :Email";

            // Préparation de la requête
            $stmt = $pdo->prepare($query);

            // Liaison du paramètre
            $stmt->bindParam(':Email', $Email, PDO::PARAM_STR);

            // Exécution de la requête
            $stmt->execute();
        }
    }
}
```

```

        // Vérifier si l'utilisateur existe dans la base de données
        if ($stmt->rowCount() == 1) {
            // L'utilisateur existe, vérifiez le mot de passe
            $row = $stmt->fetch(PDO::FETCH_ASSOC);
            if (password_verify($MotDePasse, $row['mot_de_passe']))
{
                // Mot de passe correct, connectez l'utilisateur
                session_start();
                $_SESSION['MotDePasse'] = $MotDePasse;
                // Rediriger vers la page de bienvenue (ou toute
autre page appropriée)
                header("Location: ../MyCoach/index.php");
                exit();
            } else {
                // Mot de passe incorrect
                $passwordError = "Mot de passe incorrect. ";
            }
        } else {
            // L'utilisateur n'existe pas
            $emailError = "Utilisateur non trouvé.";
        }
    }
}

?>

```

## Script de connexion à la base de données : <?php

```
$host = 'localhost';
$dbname = 'mycoach';
$username = 'root';
$password = '';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username,
$password);
    // Activer les erreurs PDO
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    $errorMessage = "Erreur de connexion à la base de données : " .
    $e->getMessage();
    // Rediriger vers la page d'accueil avec le message d'erreur en
paramètre
    header("Location: index.php");
    exit; // Arrêter l'exécution du script
}
?>
```

## Script de déconnexions :

```
<?php
session_start();
// Détruire la session
session_destroy();

// Rediriger l'utilisateur vers la page précédente (ou une autre page
de votre choix)
header("Location: ".$_SERVER['HTTP_REFERER']);
exit();
?>
```

Script qui permet de faire un filtre qui trie en fonction de la difficulté et de la ville :

```
// filter.js
function filterSeances() {
    var selectDifficulty = document.getElementById("difficulte");
    var selectedDifficulty =
selectDifficulty.options[selectDifficulty.selectedIndex].value;

    var selectVille = document.getElementById("ville");
    var selectedVille =
selectVille.options[selectVille.selectedIndex].value;

    var rows = document.querySelectorAll(".table-fill tbody tr");
    for (var i = 0; i < rows.length; i++) {
        var row = rows[i];
        var difficultyCell = row.querySelector("td:nth-child(3)"); //
La cellule contenant la difficulté
        var villeCell = row.querySelector("td:nth-child(7)"); // La
cellule contenant la ville

        // Vérifiez si la ligne correspond aux filtres de difficulté et
de ville
        if ((selectedDifficulty === "Toutes" || selectedDifficulty ===
difficultyCell.textContent) &&
            (selectedVille === "Toutes" || selectedVille ===
villeCell.textContent)) {
            row.style.display = ""; // Afficher la ligne
        } else {
            row.style.display = "none"; // Masquer la ligne
        }
    }
}
```