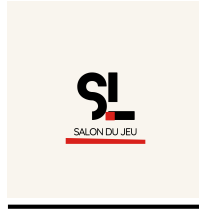


# Bienvenue dans la description de mon programme:



Les utilisateurs déjà créer en utilisant password\_hash //

User: admin / mdp:admin

User: tech / mdp:tech

User (gp déjà créer): flo / mdp:flo

Le criptage du mot est effectué sur Bcrypt:

<https://www.bcrypt.fr/>

- Page connexion basse de donnée / connexion.php:

```
<?php
$host='localhost'
$port='3306'
$dbname='20241216_sio1_moutouvirin_florian_bd'
$username='root'
$password=''

try

$pdo=newPDO 'mysql:host=localhost;dbname=20241216_sio1_moutouvirin_florian_bd;
charset=utf8' 'root' ''
    echo"Connexion réussie !"
    catch PDOException$e
        die "Erreur de connexion : ".$e->getMessage

?>
```

**Explication:** ce code, permet de se connecter à une base de données MySQL via PHPmyAdmin.

'host': adresse du serveur qui héberge la base de données. 'localhost' signifie

qu'il se situe en sur la même machine que le scripte PHP.

'port': le port utilisé par MySQL (3306 est le port par défaut).

'dbname': le nom de la BD que l'on veut se connecter.

'username' & 'password': identifiant pour la bonne connection à la DB.

'try': on crée une connexion avec l'objet 'PDO'

'new PDO()': - connecte MySQL à PHP grâce aux infos (adresse, base, utilisateur, encodage UTF-8).

- Si ça marche affiche "connexion réussie !"
- Si ça rate (catch):
  - o Capture l'erreur '(\$e)'
  - o Affiche l'erreur et stoppe le script 'die()'
- Page accueil / index.php:

```
<?php
include 'include/auth.php'
?>
<!DOCTYPEhtml>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title> </title>
  <link rel="stylesheet" href="styles/index.css">
</head>
<body>
  <div class="container">
    <div class="logo">
      
    </div>
    <div class="video-container">
      <video width="100%" autoplay loop muted>
        <source src="images/video noel.mp4" type="video/mp4">
      </video>
    </div>

    <div class="navigation">
      <?php if isset($_SESSION['user']) ?>
        <p>
<strong><?php echo htmlspecialchars($_SESSION['user'], ENT_QUOTES); ?></strong> </p>
        <a class="btn" href="sommaire.php"> </a>
      <?php else ?>
        <p> </p>
        <a class="btn" href="include/login.php"> </a>
    </div>
  </div>
</body>
</html>
```

```
        <?phpendif ?>
    </div>
</div>
</body>
</html>
```

**Explication:** 'include': ajoute la fonction PHP externe (auth.php) au script. & permet de vérifier si un utilisateur est connecté (authentification).

'<!doctype html>' & '<html lang=fr>' indique que le document est en HTML5. 'lang="fr"' indique que le contenu est en français.

En tête de la page:

'<head> </head>'

- '<meta charset="UTF-8">': c'est l'encodage pour afficher les caractères spéciaux.
- '<title>': titre de la page (s'affiche sur l'onglet du navigateur).
- '<link rel="stylesheet" href="styles/index.css">': c'est le frontend de la page. Il charge une feuille de style pour personnaliser l'apparence et l'interface.

Corps de la page:

- '<body>': contient tout ce qui est visible sur la page.
- '<div class="container">': structure principale pour regrouper les éléments.

Affichage du logo:

- '<img>': affiche une image (logo).
- '<src="images/sallon du jeu.png">': chemin vers l'image.
- 'alt="logo du salon du Jeu":' texte alternatif si l'image ne charge pas.

Vidéo intégrée:

- '<video width="100%" autoplay loop muted>': insère une vidéo, demarage automatique, redémarre quand elle finit, et pas de son.
- '<source>': chemin vers la video.

#### Navigation:

- Condition 'if' dans '<div>':
  - Si l'utilisateur est connecté '(\$\_session['user']) existe:
    - Affiche un message avec son pseudo (securisé avec htmlspecialchars).
    - Propose un lien vers la page (sommaire.php)
  - Sinon
    - Affiche un message pour les visiteurs
    - Propose un liens vers la page (login.php)
- Page d'authentification / auth.php:

```
<?php
session_start
include 'connexion.php'

// Vérification des rôles
function checkRole $allowedRoles
    if !isset $_SESSION 'role' || !in_array $_SESSION 'role'
$allowedRoles
        header "Location: login.php"
        exit

// Déconnexion
if isset $_GET 'logout'
    session_destroy
    header "Location: login.php"
    exit

?>
```

**Explication:** ce code permet, de proteger les pages qui demande un authentification ou des permissions spécifiques.

Démarrage de la session:

- 'session\_start()': active une session pour pouvoir utiliser ou modifier des session (comme \$\_session['rôle']).

Inclure le fichier DB connexion.php:

- 'include': ajoute le fichier connexion.php.

Vérification des rôles d'utilisateur:

- 'checkRole(\$allowedRoles)': fonction qui verifie si l'utilisateur à l'un des rôles autorisés.
  - o '\$\_session['role']': contient le rôle de l'utilisateur connecté.
  - o 'isset(\$\_session[role])': si aucun rôle n'est défini (utilisateur pas connecté).
  - o 'in\_array(\$\_session['role'], \$allowedRoles)': si le rôle de l'utilisateur n'est pas dans la liste des rôles autorisés.
  - o Redirige vers login.php' pour demander une connexion.

Déconnexion:

- 'isset(\$\_get['logout'])': vérifie si l'URL contient '?logout'.
- 'session\_destroy()': supprime toutes les données de session (l'utilisateur est déconnecté).
- 'header("Location: login.php")': redirige l'utilisateur vers la page de connexion.
- 'exit': arrête le script après la redirection.
- Page déconnexion / logout.php:

```
<?php
session_start      // Démarre la session

// Vérifie si une session est active
if isset $_SESSION 'user'
    // Détruit toutes les variables de session
    $_SESSION=

    // Détruit la session elle-même
    session_destroy

    // Redirige vers la page de connexion ou d'accueil
    header 'Location: ../index.php'
    exit
else
    // Si aucun utilisateur connecté, redirige simplement vers l'accueil
    header 'Location: ../index.php'
```

exit

- 'session\_start()': active le système de session pour accéder aux variable de session existantes.
- 'isset(\$\_session['user'])': vérifie si un utilisateur est connecté.

Déconnexion d'un utilisateur connecté:

- '\$session = [];': vide toutes les variables de session pour supprimer les informations de l'utilisateur.
- '\$\_session\_destroy()': termine complètement la session côté serveur.
- 'header('location: ../index.php');': redirige l'utilisateur vers la page accueil.
- 'exit();': arrête immédiatement l'exécution du script après la redirection.

Gestion du cas où personne n'est connecté:

- '}'else{': si aucune session utilisateur n'existe:
  - o Redirige directement vers accueil.

- Page connexion / login.php:

**(voir fichier dans include/login.php)**

**Explication:**

*session\_start();*

*include 'connexion.php';*

- Démarre ou reprend une session pour gérer des données comme '\$\_session'.
- Charge le fichier qui gère la connexion à la base de données.

*if (\$\_SERVER['REQUEST\_METHOD'] === 'POST') {*

- Vérifie si le formulaire à été soumis avec la méthode POST. (méthode de transmission des informations sensibles et/ou volumineuses.)

**Connexion de l'utilisateur:**

*if (isset(\$\_POST['login'])) {*

- Verifie si le bouton "se connecter" à été cliqué.

```
if (isset($_POST['username']) && isset($_POST['password'])) {
```

```
    $username = $_POST['username'];
```

```
    $password = $_POST['password'];
```

- Récupère le nom d'utilisateur et le mot de passe saisis dans le formulaire.

```
$stmt = $pdo->prepare("SELECT * FROM user WHERE username = :username");
```

```
$stmt->execute([':username' => $username]);
```

```
$user = $stmt->fetch(PDO::FETCH_ASSOC);
```

- Exécute une requête pour rechercher l'utilisateur dans la base grâce à son nom.

```
if ($user && password_verify($password, $user['password_hash'])) {
```

- Vérifie si l'utilisateur existe et si le mot de passe saisi correspond au mot de passe enregistré (haché) dans la base.

```
    $_SESSION['username'] = $user['username'];
```

```
    $_SESSION['role'] = $user['role'];
```

```
    session_regenerate_id(true);
```

```
    header("Location: sommaire.php");
```

```
    exit;
```

Si valide:

- Stocke le nom d'utilisateur et le rôle dans la session.
- Renouvelle l'ID de session pour plus de sécurité.
- Redirige vers la page "sommaire.php".

```
} else {
```

```
    $error = "Identifiants invalides.";
```

```
}
```

Si invalide:

- Définit un message d'erreur.

### **Inscription d'un nouvel utilisateur:**

```
} elseif (isset($_POST['register'])) {
```

- Vérifie si le bouton "s'inscrire" a été cliqué.

```
if (isset($_POST['username']) && isset($_POST['password'])) {
```

```
    $username = $_POST['username'];
```

```
    $password = password_hash($_POST['password'], PASSWORD_DEFAULT);
```

- Récupère le nom utilisateur et le mot de passe, puis hache le mot de passe pour sécuriser son stockage.

```
$stmt = $pdo->prepare("INSERT INTO user (username, password_hash, role)
VALUES (:username, :password, 'gp')");
```

```
$stmt->execute([':username' => $username, ':password' => $password]);
```

- Ajoute l'utilisateur dans la base de données avec le rôle "gp" (Grand Public).

```
echo "<p>Inscription réussie. Vous pouvez vous connecter.</p>";
```

- Affiche un message de succès après inscription.

```
} else {
```

```
    $error = "Veuillez remplir tous les champs.";
```

```
}
```

- Affiche un message si des champs sont vides.

### **Affiche de la page HTML:**

```
<!DOCTYPE html>
```

```
<html lang="fr">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <title>Connexion</title>
```



```
<link rel="stylesheet" href="style/login.css">
```

```
<link rel="icon" type="image/png" href="images/sallon du jeu 2.png">
```

```
</head>
```

```
<body>
```

- Structure classique pour une page HTML
- ‘<link rel="stylesheet">’: charge une page de style css pour le design (frontend)
- ‘<link rel="icon">’: affiche une icône dans l’onglet du navigateur.

### **Formulaire de connexion:**

```
<form method="post" action="">
```

```
<label for="username">Nom d'utilisateur :</label>
```

```
<input type="text" name="username" required>
```

```
<label for="password">Mot de passe :</label>
```

```
<input type="password" name="password" required>
```

```
<button type="submit" name="login">Se connecter</button>
```

```
</form>
```

- Permet à un utilisateur existant de se connecter.

### **Formulaire d'inscription:**

```
<h2>Inscription (Grand Public uniquement)</h2>
```

```
<form method="post" action="">
```

```
<label for="username">Nom d'utilisateur :</label>
```

```
<input type="text" name="username" required>
```

```
<label for="password">Mot de passe :</label>
```

```
<input type="password" name="password" required>
```

```
<button type="submit" name="register">S'inscrire</button>
```

```
</form>
```

- Permet à un nouveau utilisateur (Grand Public) de s'inscrire.

### **Affichage des erreurs:**

```
<?php if (isset($error)): ?>
```

```
<p class="error"><?= htmlspecialchars($error) ?></p>
```

```
<?php endif; ?>
```

- Affiche un message d'erreur (comme "identifiant invalides").
- Page sommaire / sommaire.php:

**(voir fichier dans include/sommaire.php)**

### **Inclusion et récupération du rôle utilisateur:**

```
<?php
```

```
include 'auth.php';
```

```
$role = $_SESSION['role'] ?? null;
```

```
?>
```

- 'include 'auth.php';': charge le fichier qui vérifie l'authentification et la session utilisateur.
- '\$\_session['role'] ?? null': récupère le rôle de l'utilisateur connecté depuis la session. Si la clé role n'existe pas, la valeur sera nul.

### **Début de la page HTML:**

```
<!DOCTYPE html>
```

```
<html lang="fr">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<title>Sommaire</title>
```

```
<link rel="stylesheet" href="style/sommaire.css">
```

```
</head>
```

```
<body>
```

- HTML:
  - '`<title>`': définit le titre affiché dans l'onglet du nav.
  - '`<link>`': charge une feuille de style pour le design.

### **Message de bienvenue:**

```
<h1>Bienvenue, <?= htmlspecialchars($_SESSION['username']) ?></h1>
```

- Affiche un message de bienvenu avec le nom de l'utilisateur connecté.  
'`htmlspecialchars()`': protège contre les attaques (injection de code dans le nom d'utilisateur).

### **Menu conditionnel basé sur le rôle:**

```
<ul>
```

```
<?php if ($role === 'admin' || $role === 'tech'): ?>
```

```
<li><a href="ajouter_exposant.php">Ajouter/Supprimer un  
exposant</a></li>
```

```
<li><a href="affecter_exposant.php">Ajouter/Supprimer une affectation à  
un stand</a></li>
```

```
<?php endif; ?>
```

- Si le rôle de l'utilisateur est admin ou tech:
  - Affiche des option pour gérer les exposants et assigner des stands.

```
<?php if ($role !== 'gp'): ?>
```

```
<?php endif; ?>
```

- Bloc conditionnel laissé vide afin de rajouter plus tard des option spécifique à d'autres rôles.

### **Option accessibles à tous les utilisateurs:**

```
<li><a href="liste_stand.php">Gérer les stands</a></li>
```

```
<li><a href="liste_exposant.php">Voir les exposants</a></li>
<li><a href="liste_locations.php">Voir les locations</a></li>
</ul>
```

- Liens vers des pages pour:
  - o Gérer les stands.
  - o Consulter les exposants.
  - o Voir les locations de stands.

### **Déconnexion:**

```
<a class="logout" href="auth.php?logout=true">Se déconnecter</a>
```

- Lien pour déconnecter l'utilisateur.
- Passe un paramètre 'logout=true' à la page 'auth.php', qui gère la déconnexion.
- Page ajouter/supprimer exposant / ajouter\_exposant.php:

**(voir document dans include)**

### **Inclusion & vérification des rôles:**

```
<?php
include 'auth.php';
checkRole(['admin', 'tech']);
```

- 'include 'auth.php';

### **Gestion des actions "POST":**

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (isset($_POST['add'])) {
```

- Verifie si une requête "POST" a été envoyée (soumission d'un formulaire).
- Si le bouton 'ajouter' à été cliqué (name="add").

### **Ajout d'un exposant:**

```
$nom = trim($_POST['NomExp']);
```

```
$rue = trim($_POST['AdrRueExp']) ?: null;
```

```
$cp = trim($_POST['AdrCPExp']) ?: null;
```

```
$ville = trim($_POST['AdrVilleExp']) ?: null;
```

```
$type = $_POST['CodeTypeExp'];
```

- Recupère et nettoie les données du formulaire:

- 'trim()': supprime les espaces inutiles.
- Si une donnée est vide, elle est remplacée par 'nul'.

```
if (empty($nom) || empty($type)) {
```

```
    $message = "Le nom et le type d'exposant sont obligatoires.";
```

```
    $messageType = "error";
```

```
} else {
```

```
    $stmt = $pdo->prepare("INSERT INTO exposant (NomExp, AdrRueExp, AdrCPExp, AdrVilleExp, CodeTypeExp) VALUES (:nom, :rue, :cp, :ville, :type)");
```

```
    $stmt->execute([':nom' => $nom, ':rue' => $rue, ':cp' => $cp, ':ville' => $ville, ':type' => $type]);
```

```
    $message = "Exposant ajouté avec succès.";
```

```
    $messageType = "success";
```

```
}
```

- Si les champs obligatoire (NomExp, CodeTypeExp) sont vides:
  - Affiche un message d'erreur.
- Sinon:
  - Prépare et exécute une requête SQL pour insérer un nouvel exposant dans la base de données.
  - Affiche un message de confirmation.

### **Suppression d'un exposant:**

```
} elseif (isset($_POST['delete'])) {  
    $id = $_POST['id'];  
  
    $stmt = $pdo->prepare("DELETE FROM exposant WHERE NumeroExp =  
:id");  
  
    $stmt->execute([':id' => $id]);  
  
    $message = "Exposant supprimé avec succès.";  
    $messageType = "success";  
}  
}
```

- Si le bouton 'supprimer' a été cliqué (name="delete"), récupère l'ID de l'exposant sélectionné.
- Exécute une requête SQL pour supprimer cet exposant.

### **Structure de la page HTML:**

#### **Entête:**

```
<!DOCTYPE html>  
<html lang="fr">  
<head>  
    <meta charset="UTF-8">  
    <title>Gestion des Exposants</title>  
    <link rel="stylesheet" href="style/styleaexp.css">  
</head>  
<body>
```

- Document HTML en français.
- Charge une feuille de style CSS pour le design.

### **Message de feedback:**

```
<?php if (!empty($message)): ?>
```

```
    <p class="message <?= htmlspecialchars($messageType) ?>">
```

```
        <?= htmlspecialchars($message) ?>
```

```
    </p>
```

```
<?php endif; ?>
```

- Un message (succès/erreur) (exposant ajouter avec succès) est défini, il s'affiche dans une balise <p>:
  - o La classe CSS est basée sur le type de message (success ou error).
  - o 'htmlspecialchars()' protège le contenu du message.

### **Formulaire pour ajouter un exposant:**

```
<h2>Ajouter un nouvel exposant</h2>
```

```
<form method="post" action="">
```

```
    <label for="NomExp">Nom Exposant :</label>
```

```
    <input type="text" name="NomExp" required>
```

```
    <label for="AdrRueExp">Rue :</label>
```

```
    <input type="text" name="AdrRueExp">
```

```
    <label for="AdrCPExp">Code Postal :</label>
```

```
    <input type="text" name="AdrCPExp">
```

```
    <label for="AdrVilleExp">Ville :</label>
```

```
    <input type="text" name="AdrVilleExp">
```

```
    <label for="CodeTypeExp">Type Exposant :</label>
```

```
    <select name="CodeTypeExp" required>
```

```
        <option value="">-- Sélectionner un type --</option>
```

```
        <option value="ASS">Association sans but lucratif</option>
```

```
        <option value="ENT">Entreprise</option>
```

```

        <option value="COM">Commerçant</option>
    </select>
    <button type="submit" name="add">Ajouter</button>
</form>

```

- Champs du formulaire:
  - Nom, adresse, code postal, ville (optionnels sauf nom)
  - Liste déroulante pour choisir un type d'exposant récupérer dans la table de donnée CodeTypeExp.
- Le bouton 'ajouter' envoie une requête "POST".

### **Formulaire pour supprimer un exposant:**

```

<h2>Supprimer un exposant</h2>
<form method="post" action="">
    <label for="id">Sélectionner un Exposant :</label>
    <select name="id" required>
        <option value="">-- Sélectionner un exposant --</option>
        <?php
            $stmt = $pdo->query("SELECT NumeroExp, NomExp FROM exposant");
            while ($exposant = $stmt->fetch(PDO::FETCH_ASSOC)) {
                echo "<option value='{ $exposant['NumeroExp']}'>" .
                    htmlspecialchars($exposant['NomExp']) . "</option>";
            }
        ?>
    </select>
    <button type="submit" name="delete">Supprimer</button>
</form>

```

- Liste déroulante en utilisant la méthode "QUERY":



- Rempli dynamiquement avec les exposants récupérés depuis la base de données.
- Le bouton 'supprimer' envoie une requête "POST" avec l'ID sélectionné.

***Lien pour revenir au sommaire:***

`<a href="sommaire.php">Retour au sommaire</a>`

- Lien simple pour revenir à la page sommaire.

- Page voir les exposant / liste\_exposant.php:
- Page ajouter/supprimer une affectation à un stand / affecter\_exposant.php:
- Page gérer les stands / liste\_stand.php:
- Page voir les locations / liste\_locations.php:

## Explication de la base de donnée SQL (voir fichier de mon dossier):

**(voir document dans le dossier "sql" de mon projet)**

Les informations incluent :

- Le nom de la base : 20241216\_sio1\_moutouvirin\_florian\_bd.
- Le serveur utilisé : 127.0.0.1:3306.
- La version de MySQL : 5.7.23.

**Les Tables et Leurs Rôles:**

### **Table exposant**

Contient les informations sur les exposants (comme les entreprises ou associations).

Colonnes principales :

- NumeroExp : ID unique de l'exposant.
- NomExp : Nom de l'exposant.
- AdrRueExp, AdrCPExp, AdrVilleExp : Adresse.
- CodeTypeExp : Type d'exposant (référence à typeexposant).

Données : Les exposants incluent des entreprises, des associations, et des commerçants.

### **Table location**

Gère les locations des stands par les exposants.

Colonnes :

- NumeroLocation : ID unique de la location.
- NumeroStand et NumeroExp : Références aux stands loués et aux exposants.
- DateDebut, DateFin : Période de location.
- PrixTotal : Coût total. (cela pour une futur fonctionnalité)

### **Table louer**

Définit les tarifs de location selon le type d'exposant et le type de stand.

Colonnes :

- CodeTypeExp : Type d'exposant.
- CodeTypeStd : Type de stand.
- Montant : Tarif.

### **Table stand**

Gère les informations des stands.

Colonnes :

- NumeroStand : ID du stand.
- NumeroAllee, PlaceAllee : Localisation dans l'allée.
- CodeTypeStd : Type de stand.
- NumeroExp : Référence à l'exposant (si déjà réservé).

### **Table typeexposant**

Liste les types d'exposants.

Colonnes :

- CodeTypeExp : Code du type. (AN)
- LibTypeExp : Description (ex. : entreprise, association).

### **Table typestand**

Décrit les types de stands.

Colonnes :

- CodeTypeStd : Code du type.

- LibTypeStd : Description (ex. : mobilier inclus).

#### **Table user**

Gère les utilisateurs pour accéder à la base.

Colonnes :

- id : ID unique.
- username, password\_hash : Identifiants de connexion.
- role : Rôle de l'utilisateur (admin, tech, utilisateur général).

#### **Points Techniques:**

- **Clés Primaires** : Chaque table a des colonnes uniques pour identifier les données (ex. : NumeroExp dans exposant).
- **Relations** :
  - CodeTypeExp lie exposant et typeexposant.
  - NumeroExp et NumeroStand relient plusieurs tables.
- **Données d'exemple** :
  - Des exposants comme "Temple des dés" (Lyon) ou "Cartes et dragons" (Marseille).
  - Tarifs : Un commerçant paye 2400€ pour un stand de type 1.

Explication frontend (CSS) / pages: