


# Projet Gestion des arbitres

*Le responsable de la ligue de Basket-ball de Lorraine désire informatiser la gestion des arbitrages des matchs des différents week-ends. Ce site sera compatible avec tous les navigateurs et adaptable en fonction de la taille de l'écran grâce à Bootstrap.*



## Présentation de l'équipe et de la mission

 Nous sommes l'équipe n°9, composée de 2 élèves très motivés ! Anthony Rodrigues et Sandra Rousselot.

Notre mission est la n°1 répondant à 2 objectifs.

### ► **T1.1 Gestion des championnats :**


- Permettre la création d'un championnat
- La modification et la suppression d'un championnat

### ► **T3.8 Modification d'une affectation :**

- Saisie d'un match
- Permettre la modification des arbitres désignés pour prendre en main un match
- BONUS : Permettre la suppression d'un match

---

## La documentation technique

 La documentation technique présentera les logiciels permettant la réalisation de notre projet.

## Prérequis

► **Nous allons présenter les prérequis nécessaires à la création de notre sites en différentes sous parties :**

- MAMPP : Serveur Local
- Sublime Text
- Module de connexion a la BDD avec PDO
- Notre BDD : PHP MY ADMIN
- Le Modèle-vue-contrôleur

## MAMPP : Serveur Local

MAMP un ensemble de logiciels MACOS permettant de mettre en place un serveur Web local, un serveur FTP et un serveur de messagerie électronique.



## Sublime Text

Sublime Text est un éditeur de texte générique codé en C++ et Python, disponible sur Windows, Mac et Linux.



## MODULE DE CONNEXION A LA BDD : PDO

Nous avons créé une fonction « connexionPDO » pour avoir accès à la base de données. Nous indiquons dans les propriétés de PDO les paramètres de la base de données.

Nous avons ensuite créé le module de connexion :

```
function connexionPDO() {  
    $login = "root";  
    $mdp = "root";  
    $bd = "arbitres";  
    $serveur = "localhost";  
  
    try {  
        $conn = new PDO("mysql:host=$serveur;dbname=$bd", $login, $mdp,  
array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES \'UTF8\''));  
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
    }  
}
```

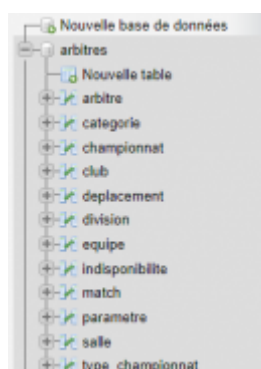
```
        return $conn;
    } catch (PDOException $e) {
        print "Erreur de connexion PDO ";
        die();
    }
}
```

Nous utilisons l'instruction *try catch* qui exécute notre programme et définit une réponse si l'une de ces instructions provoque une exception.

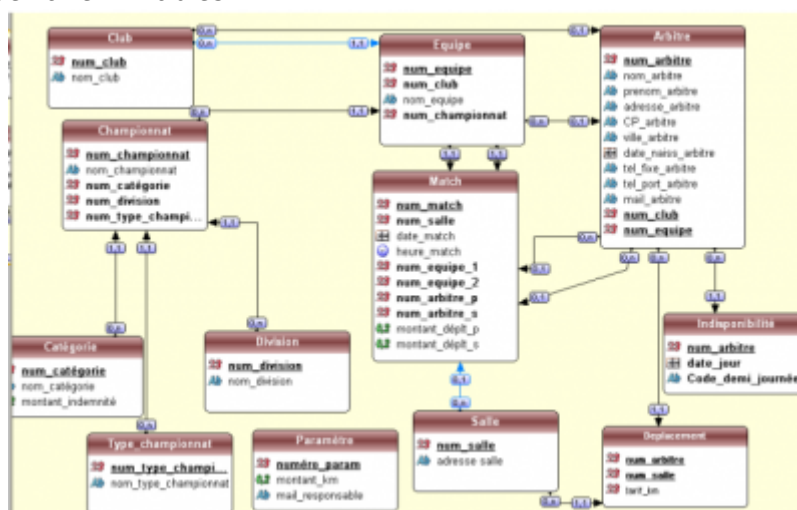


## Notre BDD : PHP MY ADMIN

Nous avons également besoin d'une base de données afin de répertorier les matchs, les arbitres, les championnats etc...



Nous avons donc répertorié 12 tables.



Notre base de données est stockée en local grâce à PhpMyAdmin.



## Le Modèle-vue-contrôleur

Pour une structure propre et simple, nous avons utilisé le modèle MVC.

L'architecture MVC est l'une des architectures logicielles les plus utilisées pour les applications Web, elle se compose de 3 modules :



- **Modèle** : noyau de l'application qui gère les données, permet de récupérer les informations dans la base de données, de les organiser pour qu'elles puissent ensuite être traitées par le contrôleur.
- **Vue** : composant graphique de l'interface qui permet de présenter les données du modèle à l'utilisateur au sein du code HTML.
- **Contrôleur** : partie principale de l'architecture, car c'est lui qui va faire le lien entre l'utilisateur, le modèle et la vue. Quand l'utilisateur souhaite afficher une page, c'est le contrôleur qui va recevoir cette requête et se charger ensuite de récupérer les données via le modèle, pour les envoyer à la vue qui sera affichée à l'utilisateur. Il est l'intermédiaire entre le modèle et la vue.

## Objectif T1.1

### ► Rappel de l'objectif :

- Permettre la création d'un championnat
- La modification et la suppression d'un championnat



### PERMETTRE LA CRÉATION D'UN CHAMPIONNAT

Pour répondre à ce besoin nous avons décidé de répertorier l'ensemble des championnats pour ensuite permettre le choix d'en ajouter une sans faire de doublon.

Visuellement nous avons mis en place un tableau, ainsi que 3 boutons : suppression, modification et ajout.



**La page actuelle : VueGestionChampionnat.php[Design]**

Gestion des championnats					
#	Nom Championnat	Catégorie	Division	Type	
1	WeekEndClash	Minime	Masculin	Eccellenze Régionale	 

Passons dès maintenant au développement.

### **La page actuelle : VueGestionChampionnat.php**

Pour afficher les informations nous avons créé une requête dans le modèle :

```
$getchampionnat= $conn->prepare("SELECT * FROM championnat INNER JOIN
categorie ON championnat.num_categorie = categorie.num_categorie INNER JOIN
division ON division.num_division = championnat.num_division INNER JOIN
type_championnat ON type_championnat.num_type_championnat =
championnat.num_type_championnat");
```

On le stock dans une variable dans le contrôleur, puis on l'affiche comme ci dessous :

```
<?php foreach($lesResultats as $leResultat){ ?>
    <th scope="row">
        <?php echo $leResultat->num_championnat; ?></th>
```

Maintenant crée le formulaire permettant l'ajout d'un championnat :

### **La page actuelle : VueAjoutChampionnat.php[Design]**

Ajout d'un championnat

Pour ajouter un championnat veuillez simplement compléter les champs ci-dessous !

Nom du championnat

Catégorie

Division

Type championnat

Envoyer

Passons dès maintenant au développement.

### **La page actuelle : VueAjoutChampionnat.php**

Dans ce formulaire nous devons sélectionner les catégories, les divisions et le type de championnat à l'aide de plusieurs requêtes dans le modèle.

```
SELECT * FROM categorie
```

Puis nous stockons le résultat dans une variable de type *array* dans le contrôleur. Puis dans la vue nous faisons une boucle *for each* afin d'afficher les résultats.

```
<h2 class="card-title">Catégorie</h2>
<div class="form-group">
  <select name="num_categorie" class="form-control">
    <?php foreach($lesResultatsCategorie as $leResultat){ // on affiche
les categories ?>
      <option value="<?php echo $leResultat->num_categorie; ?>">
        <?php echo $leResultat->nom_categorie; ?></option>
      <?php } ?>
    </select>
  </div>
```

Une fois le formulaire rempli, nous utilisons la méthode POST :

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

Et nous rechargeons la page lors de l'envoi c'est à ce moment là que le contrôleur va permettre l'insertion dans la base de donnée, en vérifiant si le formulaire a bien été envoyé.

```
if(isset($_POST["button"])){
    AjouterChampionnat($_POST["nom_championnat"], $_POST["num_categorie"],
$_POST["num_division"],
                        $_POST["num_type_championnat"]);

    header('Location:./?action=GestionChampionnat');
}
```

La première ligne va nous indiquer si l'utilisateur a bien cliqué sur le bouton. Et si c'est le cas, le contrôleur appelle la fonction *AjouterChampionnat*, prenant en compte différents paramètres dans le modèle.

```
function AjouterChampionnat($nom_championnat, $num_categorie, $num_division,
$num_type_championnat){

    try {

        $conn = connexionPDO();
        $getchampionnat= $conn->prepare("INSERT into championnat
values(null,?,?,?,?)");

        $getchampionnat->execute(array($nom_championnat, $num_categorie,
$num_division, $num_type_championnat));

    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage();
        die();
    }
}
```

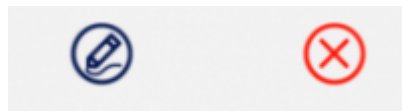
```
}
}
```

## PERMETTRE LA MODIFICATION ET LA SUPPRESSION D'UN CHAMPIONNAT

Pour répondre à ce besoin nous allons utiliser le tableau qui répertorie l'ensemble des championnats créés précédemment pour ensuite permettre la modification d'un de ces championnats.



**La page actuelle : VueGestionChampionnat.php[Design]**



Passons dès maintenant au développement.



**La page actuelle : VueGestionChampionnat.php**

```
<td>
    <form action="?action=ModificationChampionnat" method="POST">
        <input name="num_championnat" type="hidden" value="<?php echo
$leResultat->num_championnat; ?>"></input>
        <input width="40" height="40" type="image"
src="images/edit.png">
    </form>
</td>
<td>
    <form action="" method="POST">
        <input name="action" type="hidden" value="delete"></input>
        <input name="num_championnat" type="hidden" value="<?php echo
$leResultat->num_championnat; ?>"></input>
        <input width="40" height="40" type="image"
src="images/delete.png"></input>
    </form>
</td>
```

Quand l'utilisateur clique sur modification ou suppression nous transmettons 2 données : Le numéro de la championnat sélectionné : *num\_championnat* L'action que l'utilisateur désire : delete (suppression) avec POST

Pour la suppression nous allons repasser par le contrôleur :

```
if (isset($_POST["action"])){
    if ($_POST["action"] == "delete"){
        SupprChampionnat($_POST["num_championnat"]);
        header('Location: ../?action=GestionChampionnat');
    }
}
```

```
}
```

Comme précédemment nous faisons appel à une requête PHP stoker sur la page du modèle via une fonction :

```
$getchampionnat= $conn->prepare("DELETE FROM championnat WHERE  
num_championnat = ?");  
  
$getchampionnat->execute(array($championnat));
```

Pour la modification nous allons être rediriger par le contrôleur principal en transmettant via POST le championnat sélectionné :

```
$lesActions["ModificationChampionnat"] =  
"controleurModificationChampionnat.php";
```

Nous vérifions si la variable \$\_POST["num\_championnat"] sur *controleurModificationChampionnat.php* a bien été récupérée :

```
if (isset($_POST["num_championnat"])) {  
    $leResultatChampionnat =  
    GetChampionnatPersonnalisee($_POST["num_championnat"]);  
    $lesResultatsDivision = getDivision();  
    $lesResultatsType_championnat = getType_championnat();  
    $lesResultatsCategorie = getCategorie();  
}
```

Si c'est le cas nous affichons les caractéristiques du championnat sur la page *vueModificationChampionnat.php*



### **La page actuelle : vueModificationChampionnat.php[Design]**

Si l'utilisateur modifie les informations du championnat, nous repassons par le contrôleur de la même page qui vérifie que l'utilisateur a bien envoyé le formulaire :

```
if (isset($_POST["button"])) {  
    ModifChampionnat($_POST["nom_championnat"], $_POST["num_categorie"],
```



```
$_POST["num_division"], $_POST["num_type_championnat"],
$_POST["num_championnat"]);
    header('Location: ./?action=GestionChampionnat');
}
```

Si `$_POST["button"]` existe alors nous modifions le championnat avec la fonction stocker dans le modèle :

```
$modificationmatch= $conn->prepare("UPDATE `match` SET num_arbitre_p = ?,
num_arbitre_s = ? WHERE num_match = ?");
$modificationmatch->execute(array($num_arbitre_principal,
$num_arbitre_secondaire, $num_match));
```

Nous utilisons *try catch* pour chacune de nos requêtes mais par soucis de clarté nous ne les avons pas afficher ici 😊

### Objectif 3.8

#### ► Rappel de l'objectif :

- Saisie d'un match
- Permettre la modification des arbitres désignés pour prendre en main un match
- BONUS : Permettre la suppression d'un match

### PERMETTRE LA SAISIE D'UN MATCH

Pour répondre à ce besoin nous avons décidé de répertorier l'ensemble des matchs pour ensuite permettre le choix d'en ajouter un sans faire de doublon.

Visuellement nous avons mis en place un tableau, ainsi que 3 boutons : suppression, modification et ajout.



#### La page actuelle : VueGestionMatch.php[Design]

#	Salle	Date	Equipe 1	Equipe 2	Nom Arbitre	Suppléant
1	1	2021-04-25	les Coureurs Hamès	les Dindons	Hippolyte	Aphrodite

Passons dès maintenant au développement.



#### La page actuelle : VueGestionMatch.php

Pour afficher les informations nous avons créé une requête dans le modèle :

```
function GetMatchPersonnalisee($num_match){

    $resultat = array();
    try {
```

```
$conn = connexionPDO();
$getMath = $conn->prepare("SELECT num_match, date_match, num_equipe_1,
num_equipe_2, num_arbitre_p, num_arbitre_s, num_salle, p.nom_arbitre as
nom_arbitre_principal, s.nom_arbitre as nom_arbitre_secondaire,
e1.nom_equipe as nom_equipe_1, e2.nom_equipe as nom_equipe_2 FROM `match`
INNER JOIN arbitre p ON p.num_arbitre = match.num_arbitre_p INNER JOIN
arbitre s ON s.num_arbitre = match.num_arbitre_s INNER JOIN equipe e1 ON
e1.num_equipe = match.num_equipe_1 INNER JOIN equipe e2 ON e2.num_equipe =
match.num_equipe_2 WHERE num_match = ?");

$getMath->execute(array($num_match));

$resultat = $getMath->fetchAll(PDO::FETCH_OBJ);

} catch (PDOException $e) {
    print "Erreur !: " . $e->getMessage();
    die();
}
return $resultat;
}
```

On le stock dans une variable dans le contrôleur, puis on l'affiche comme ci dessous :



```
<?php foreach($lesResultats as $leResultat){ ?>
    <th scope="row">
        <?php echo $leResultat->num_match; ?></th>
```

Maintenant crée le formulaire permettant l'ajout d'un match:



### La page actuelle : VueAjoutMatch.php[Design]

Pour ajouter un match veuillez simplement compléter les champs ci-dessous !

<b>Dans quelle salle ?</b> <input type="text" value="Salle : 1 / Adresse : 99, rue des Dunes"/>	
<b>Quand</b> <input type="text" value="24/04/2021"/>  <input type="text" value="10:00"/> 	<b>Les équipes</b> <input type="text" value="les Martins-Pêcheurs Argentés"/> <input type="text" value="les Remarquables"/>
<b>Les montants déplacement</b> <input type="text" value="548"/> <input type="text" value="362"/>	<b>Les arbitres</b> <input type="text" value="Rousselot"/> <input type="text" value="Hippolyte"/>
<input type="button" value="Envoyer"/>	

Passons dès maintenant au développement.

### **La page actuelle : VueAjoutMatch.php**

Dans ce formulaire nous devons renseigner dans quelle salle se passe le match, quand se passera t-il, quels sont les équipes qui se combattent, ainsi que les arbitres qui superviserons le match. Pour transmettre cela nous utiliserons plusieurs requêtes dans le modèle.

Une fois le formulaire remplie, nous utilisons la méthode POST :

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

Et nous rechargeons la page lors de l'envoi c'est à ce moment là que le contrôleur va permettre l'insertion dans la base de donnée, en vérifiant si le formulaire a bien été envoyé.

### ► **Mais avant nous avons plusieurs contraintes imposées :**

- un arbitre (joueur ou entraîneur) ne peut diriger un match du championnat dans lequel il évolue
- l'arbitre primaire doit être différent de l'arbitre secondaire
- L'équipe 1 doit être différent de l'équipe 2
- L'arbitre doit être présent dans la table déplacement

Pour répondre à ces contraintes nous avons plusieurs contrôles à effectuer :

### **La page actuelle : ControleurAjoutMatch.php**

Premièrement nous vérifions que l'utilisateur a bien envoyé le formulaire :

```
if(isset($_POST["button"]))
```

Pour effectuer les tests nous allons récupérer les *num\_club* des arbitres renseignés :

```
$clubDeLarbitre1 =  
GetArbitrePersonnalisee($_POST["num_arbitre_1"])->num_club;  
$clubDeLarbitre2 =  
GetArbitrePersonnalisee($_POST["num_arbitre_2"])->num_club;  
  
$clubDeLequipe1 = GetEquipePersonnalisee($_POST["num_equipe_1"])->num_club;  
$clubDeLequipe2 = GetEquipePersonnalisee($_POST["num_equipe_2"])->num_club;
```

Pour cela nous avons créé 2 fonctions :

```
function GetArbitrePersonnalisee($num_arbitre){  
  
    try {  
  
        $conn = connexionPDO();  
        $getNomArbitrePersonnalisee = $conn->prepare("SELECT * FROM arbitre  
WHERE num_arbitre = ?");  
  
        $getNomArbitrePersonnalisee->execute(array($num_arbitre));  
  
    }  
}
```

```
$resultat = $getNomArbitrePersonnalisee->fetch(PDO::FETCH_OBJ);

} catch (PDOException $e) {
    print "Erreur !: " . $e->getMessage();
    die();
}
return $resultat;
}
```

et

```
function GetEquipePersonnalisee($num_equipe){

    try {

        $conn = connexionPDO();
        $getNomArbitrePersonnalisee = $conn->prepare("SELECT * FROM equipe WHERE
num_equipe = ?");

        $getNomArbitrePersonnalisee->execute(array($num_equipe));

        $resultat = $getNomArbitrePersonnalisee->fetch(PDO::FETCH_OBJ);

    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage();
        die();
    }
    return $resultat;
}
```

Ensuite nous allons effectué des contrôles tout en stockant les erreurs dans un tableau afin de les affichées :

```
$erreur = [];
$ajoutPossible = true; //si il y a une erreur il sera passé en false
```

Les deux premiers contrôlent consistent à vérifier que les arbitres n'appartiennent pas au même club que les équipes qui vont s'affronter.

```
if($clubDeLarbitre1 == $clubDeLequipe1 || $clubDeLarbitre1 ==
$clubDeLequipe2){
    $ajoutPossible = false;
    $erreur[] = "Impossible : L'arbitre 1 appartient au même club que
l'une des équipes! ";
}

if($clubDeLarbitre2 == $clubDeLequipe1 || $clubDeLarbitre2 ==
```

```
$clubDeLequipe2){
    $ajoutPossible = false;
    $erreur[] = "Impossible : L'arbitre 2 appartient au même club que
l'une des équipes! ";
}
```

On vérifie que l'équipe 1 et équipe 2 sont bien différentes, tout comme les deux arbitres :

```
if($_POST["num_equipe_1"] != $_POST["num_equipe_2"]){
    if($_POST["num_arbitre_1"] != $_POST["num_arbitre_2"]){
```

Puis nous vérifions la dernière contrainte qui indique que si un arbitre n'est pas inscrit dans la table *DEPLACEMENT* entre la salle et lui même, alors il ne peut pas être ajouté.

```
    if(DeplacementExiste($_POST["num_arbitre_1"], $_POST["num_salle"]) ==
0){
        $ajoutPossible = false;
        $erreur[] = "Vous devez ajouter le déplacement pour l'arbitre 1,
entre lui et la salle ! ";
    }
    if(DeplacementExiste($_POST["num_arbitre_2"], $_POST["num_salle"]) ==
0){
        $ajoutPossible = false;
        $erreur[] = "Vous devez ajouter le déplacement pour l'arbitre 2,
entre lui et la salle ! ";
    }
}
```

Pour finir nous vérifions qu'il n'y a pas d'erreur en testant la variable `$ajoutPossible`, si aucune erreur n'a été trouvée alors on insère le match :

```
if($ajoutPossible == true){
    AjouterMatch($_POST["num_salle"], $_POST["date_match"],
$_POST["heure_match"],
    $_POST["num_equipe_1"], $_POST["num_equipe_2"], $_POST["num_arbitre_1"],
$_POST["num_arbitre_2"],
    $_POST["montant_deplt_p"], $_POST["montant_deplt_s"]);

    header('Location:./?action=GestionMatch');
}
```

Pour cela on utilise une requête dans le modèle de type *INSERT INTO*.

Jeu d'essai : Si nous rencontrons des erreurs elles seront parcourues puis affichées :

Impossible : L'arbitre 1 appartient au même club que l'une des équipes!

Impossible : L'arbitre 2 appartient au même club que l'une des équipes!

Vous devez ajouter le déplacement pour l'arbitre 1, entre lui et la salle !

Vous devez ajouter le déplacement pour l'arbitre 2, entre lui et la salle !

Pour les parcourir nous parcourons le tableau comme ceci :

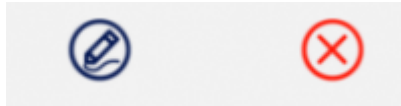
```
<?php if(isset($erreur)){  
    foreach($erreur as $erreur_valeur){ // on affiche les erreurs ?>  
        <div class="alert alert-danger" role="alert">  
            <h2><?php echo $erreur_valeur; ?></h2>  
        </div>  
    } } ?>
```

## PERMETTRE LA MODIFICATION ET LA SUPPRESSION D'UN MATCH

Pour répondre à ce besoin nous allons utiliser le tableau qui répertorie l'ensemble des matchs créés précédemment pour ensuite permettre la modification d'un de ces matchs.



**La page actuelle : VueGestionMatch.php[Design]**



Passons dès maintenant au développement.



**La page actuelle : VueGestionMatch.php**

```
<td>  
    <form action="?action=ModificationMatch" method="POST">  
        <input name="num_match" type="hidden" value="<?php echo  
$leResultat->num_match; ?>"></input>  
        <input width="40" height="40" type="image"  
src="images/edit.png">  
    </form>  
</td>  
<td>  
    <form action="" method="POST">  
        <input name="action" type="hidden" value="delete"></input>  
        <input name="num_match" type="hidden" value="<?php echo  
$leResultat->num_match; ?>"></input>  
        <input width="40" height="40" type="image"
```

```
src="images/delete.png"></input>
    </form>
</td>
```

Quand l'utilisateur clique sur modification ou suppression nous transmettons 2 données : Le numéro du match sélectionné : *num\_match* L'action que l'utilisateur désire : delete (suppression) avec POST ou la modification

Pour la suppression nous allons repasser par le contrôleur :

```
if (isset($_POST["action"])){
    if ($_POST["action"] == "delete"){
        SupprMatch($_POST["num_match"]);
        header('Location: ./?action=GestionMatch');
    }
}
```

Comme précédemment nous faisons appel à une requête PHP stoker sur la page du modèle via une fonction :

```
function SupprMatch($match){

    try {

        $conn = connexionPDO();
        $supprmatch= $conn->prepare("DELETE FROM `match` WHERE num_match = ?");

        $supprmatch->execute(array($match));

    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage();
        die();
    }
}
```

Pour la modification nous allons être rediriger par le contrôleur principal en transmettant via POST le match sélectionné :

```
$lesActions["ModificationMatch"] = "controleurModificationMatch.php";
```

Nous vérifions si la variable *\$\_POST["num\_match"]* sur *controleurModificationMatch.php* a bien été récupérée.

Si c'est le cas nous affichons les caractéristiques du championnat sur la page *vueModificationChampionnat.php*



**La page actuelle : vueModificationMatch.php[Design]**

Si l'utilisateur modifie les informations du championnat, nous repassons par le contrôleur de la même page qui vérifie que l'utilisateur a bien envoyé le formulaire, tout en respectant les mêmes contraintes que lors de l'ajout.

### ► Mais avant nous avons plusieurs contraintes imposées :

- un arbitre (joueur ou entraîneur) ne peut diriger un match du championnat dans lequel il évolue
- l'arbitre primaire doit être différent de l'arbitre secondaire
- L'équipe 1 doit être différente de l'équipe 2
- L'arbitre doit être présent dans la table déplacement

Nous ne détaillerons pas les contraintes car nous l'avons déjà fait précédemment pour l'ajout

Pour terminer nous allons mettre en place un système de connexion, pour ce faire nous allons créer une fonction vérifiant si l'utilisateur est bien connecté, nous l'utiliserons sur tout les pages dans lesquels "être connecté" est nécessaire :

```
function logged_only(){  
  
    if (session_status() == PHP_SESSION_NONE) {  
        session_start();  
    }  
  
    if(!isset($_SESSION['user'])){  
        header('Location: ./?action=Connection');  
        exit();  
    }  
}
```

Si l'utilisateur n'est pas connecté alors il sera redirigé sur la page de connexion.

Nous allons ensuite créer une fonction qui vérifie le rôle de l'utilisateur.

Il y a 2 types d'authentification : "Responsable" ou "Arbitre". Pour la mission 1, seul le responsable a la possibilité de réaliser les tâches programmées.

```
function only_reponsable(){
```



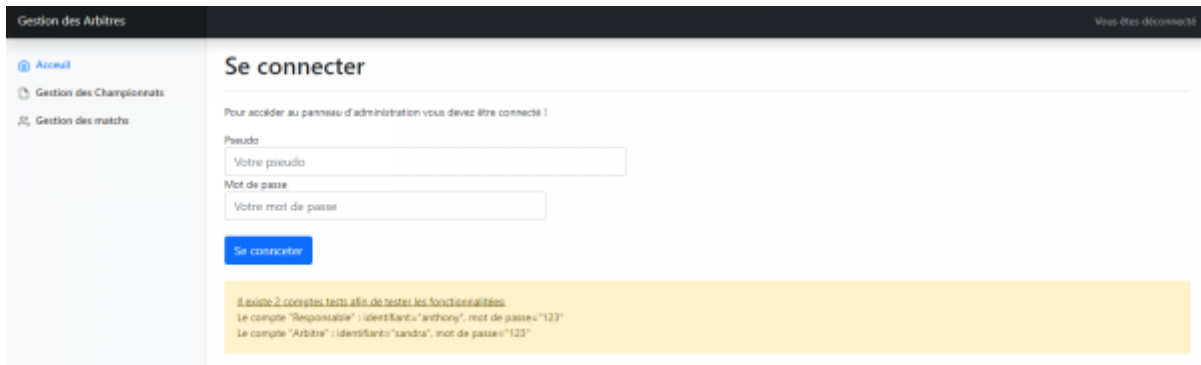
```

if($_SESSION['user']->role != "responsable"){
    header('Location: ./?action=accueil');
    exit();
}
}
}

```

Si dans la variable session le rôle stocker n'est pas responsable alors il ne pourra pas accéder à la page en question.

Pour terminée nous allons crée la page de connexion :  **La page actuelle :**  
**vueConnection.php[Design]**



Nous récupérons les informations du formulaire à l'aide de la méthode POST, puis nous les traitons sur le controleur :

```

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

    $connectionApprouvee = ConnectionUtilisateur($_POST['pseudo'],
    $_POST['mdp']);

    if($connectionApprouvee == true){
        header('Location: ./?action=default');
    }
}
}

```

Nous testons grâce à `$connectionApprouvee` que la connexion à pu se faire, en utilisant la fonction `ConnectionUtilisateur()` dans le modèle :

```

function ConnectionUtilisateur($pseudo, $mdp){

    $connectionApprouvee = false;

    $conn = connexionPDO();

    $chercherUser = $conn->prepare("SELECT * FROM user WHERE pseudo =
?");

    $chercherUser->execute(array($pseudo));

    if($chercherUser->rowCount() != 0){

```

```
$ligne = $chercherUser->fetch(PDO::FETCH_OBJ);

if($mdp == $ligne->mot_de_passe) {

    session_start();

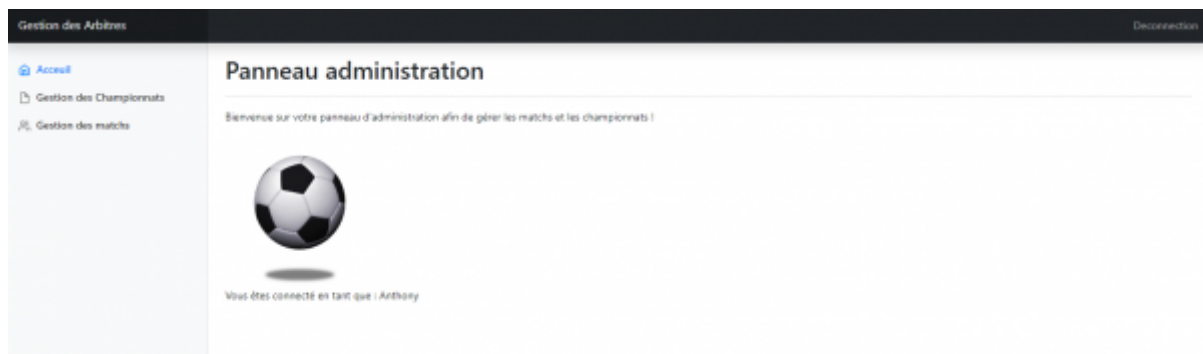
    $_SESSION['user'] = $ligne;

    $connectionApprouvee = true;

}
}
```

Nous récupérons l'enregistrement avec le pseudo, s'il n'existe pas alors *\$connectionApprouvee* reste à *false* est la connexion ne se fera pas. En revanche si le mot de passe correspond, nous stockons toutes les informations de l'utilisateur (id, nom, prenom, role), dans la variable session sous forme de tableau associatif.

Pour terminée nous avons crée une page de déconnection qui vide la SESSION avec de permettre la connection sur un autre compte :



```
<?php
logged_only();

unset($_SESSION['user']);

header('Location: ../?action=Connection');
?>
```

Tout fonctionne correctement et nous traitons les cas d'erreurs, merci de votre attention !



From:

<http://ppe.boonum.fr/> - **AP.SIO**

Permanent link:

<http://ppe.boonum.fr/doku.php?id=slam:ws:2021:sio:ap2.2:equipe9:accueil>

Last update: **2021/04/26 23:12**

