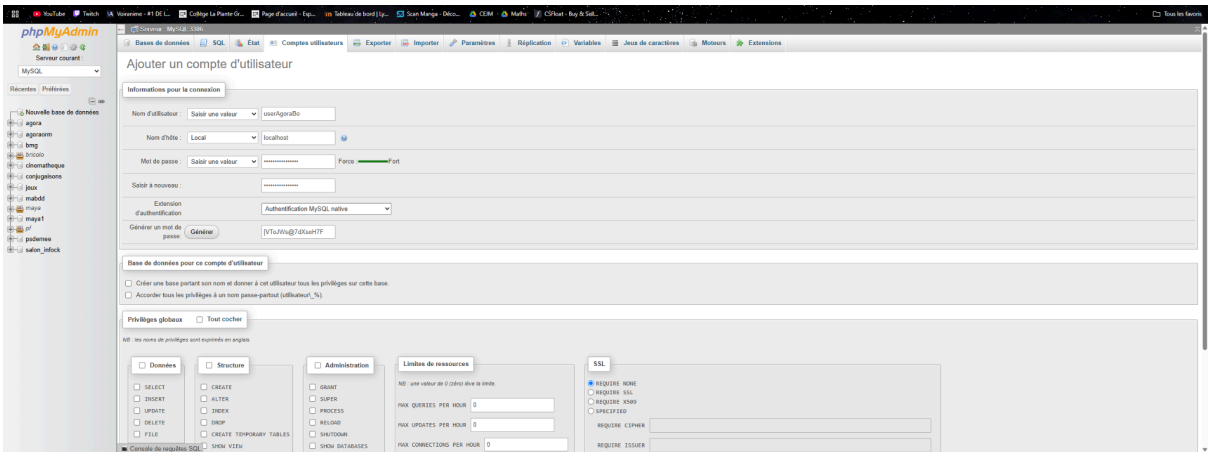
Compte-Rendu AGORA Mission 2

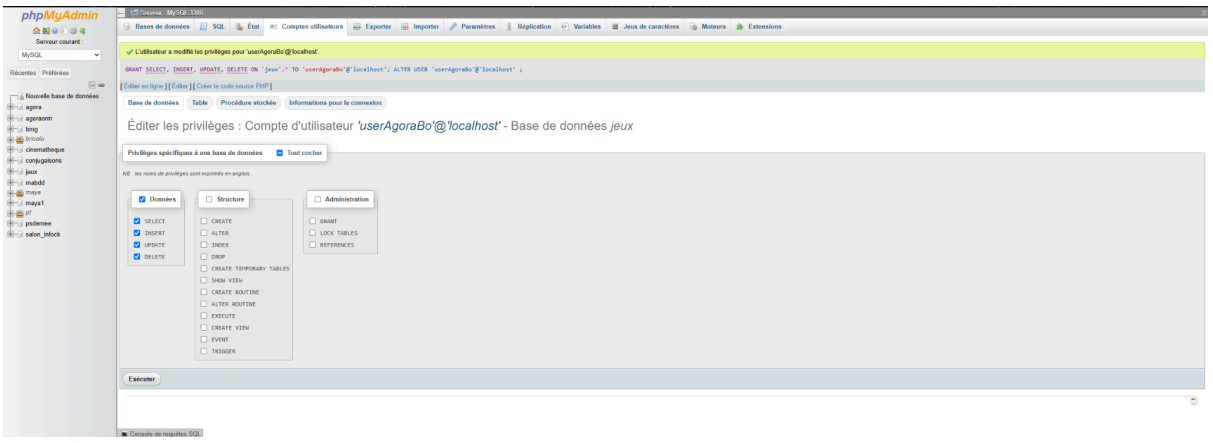
Sommaire :

Création d'un Nouvel Utilisateur :

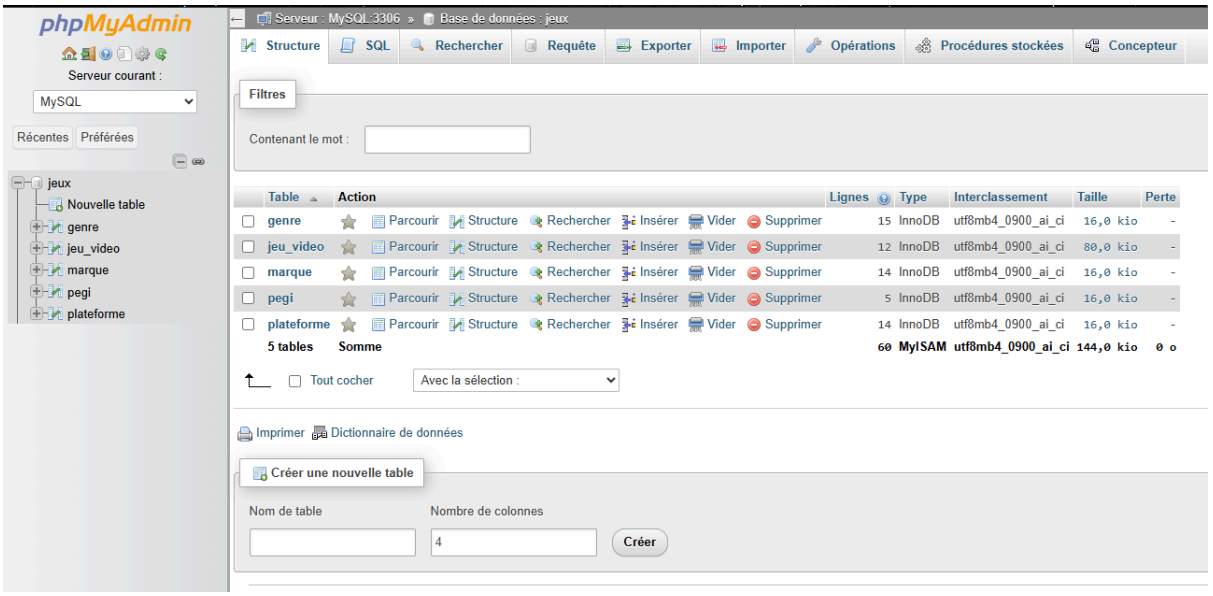
Tout d'abord, il faut venir créer un nouvel utilisateur qui va posséder des droits restreints contrairement à root :



Une fois l'utilisateur créé, il faut lui attribuer des droits sur la table jeux, car c'est celle qui nous intéresse présentement :



Maintenant il suffit de nous reconnecter au serveur de BDD avec notre nouvel utilisateur pour vérifier qu'il n'a bien accès qu'à la BDD jeux.



Nous disposons donc d'un nouvel utilisateur qui ne possède que des droits vis-à-vis des données, et ceux, uniquement sur la BDD jeux.

Il faut également penser à aller modifier nos identifiants de connexion à la BDD dans notre App, car pour le moment il se connecte en root, et nous on ne veut plus ça.

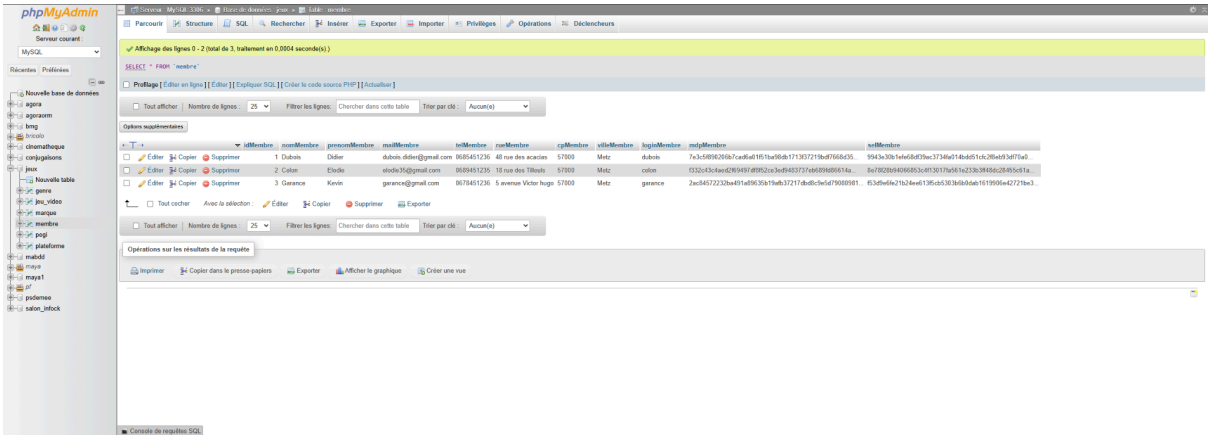
_config.inc.php

app > _config.inc.php > ...

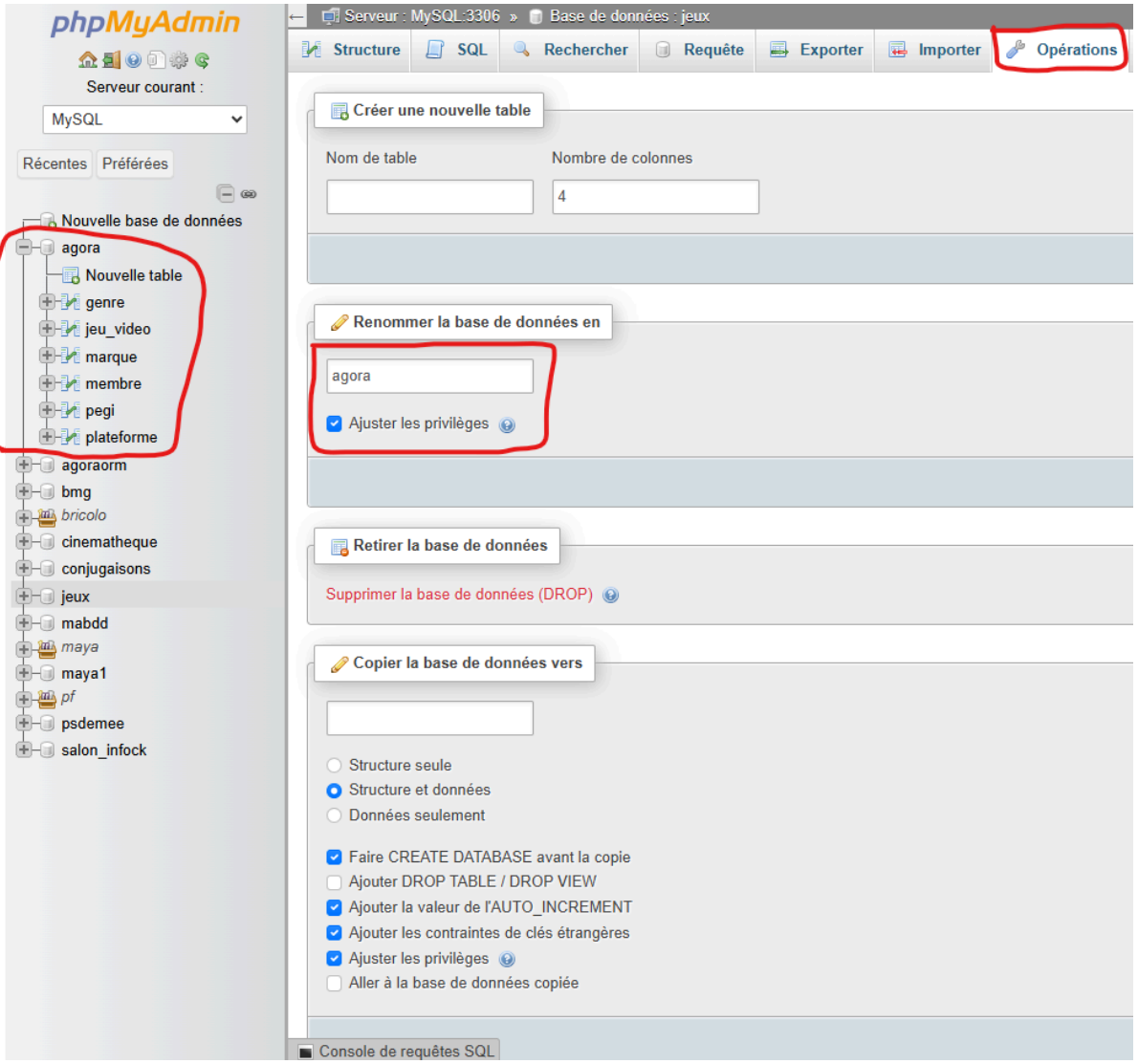
```php
<?php
/**
 * paramètres de configuration de l'application AgoraBo
 *
 * @package default
 * @author md
 * @version    1.0
 */

// gestion d'erreur
ini_set(option: 'error_reporting', value: E_ALL);   // en phase de développement
//ini_set('error_reporting', 0);        // en phase de production

// constantes pour l'accès à la base de données
define(constant_name: 'DB_SERVER', value: 'localhost'); // serveur MySql
define(constant_name: 'DB_DATABASE', value: 'jeux');        // nom de la base de données
define(constant_name: 'DB_USER', value: 'userAgoraBo');         // nom d'utilisateur
define(constant_name: 'DB_PWD', value: 'A6_iHTH9D*vtJg*c');         // mot de passe
define(constant_name: 'DSN',value: 'mysql:dbname='.DB_DATABASE.';host='.DB_SERVER);

?>
```

Adaptation de la BDD :

Une fois notre utilisateur créé et ses droits accordés, il faut maintenant venir adapter notre BDD.
Pour ce faire on va lui ajouter une nouvelle table : “membre” à l’aide d’un script fourni. Cette table ressemble à ceci :



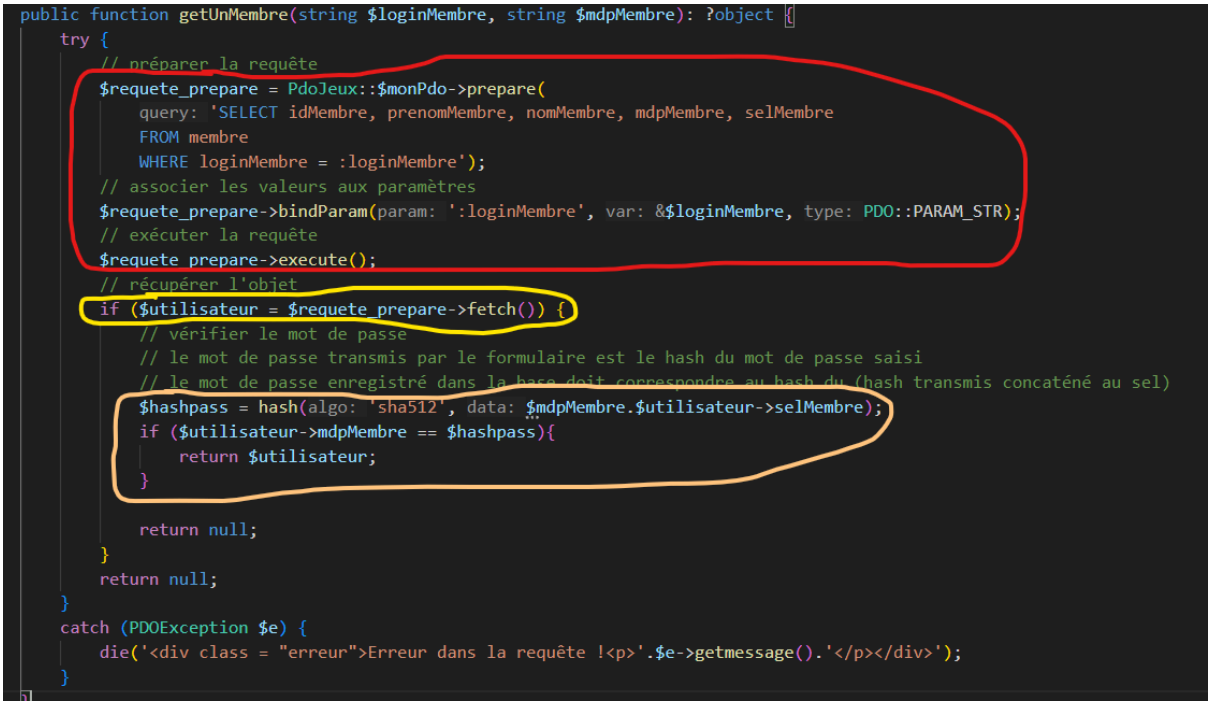
Nouveau “problème”, cette table ne représente en rien des jeux, il faut donc venir modifier le nom de notre BDD pour faire passer “jeux” en “agora” :



La partie modification de notre SGBD est maintenant terminée.

<u>Développez le modèle (MVC) :</u>

Il faut maintenant adapter notre modèle pour pouvoir récupérer un de nos membres, et pour ce faire il faut créer une nouvelle fonction :



```php
public function getUnMembre(string $loginMembre, string $mdpMembre): ?object {
    try {
        // préparer la requête
        $requete_prepare = PdoJeux::$monPdo->prepare(
            query: 'SELECT idMembre, prenomMembre, nomMembre, mdpMembre, selMembre
            FROM membre
            WHERE loginMembre = :loginMembre');
        // associer les valeurs aux paramètres
        $requete_prepare->bindParam(param: ':loginMembre', var: &$loginMembre, type: PDO::PARAM_STR);
        // exécuter la requête
        $requete_prepare->execute();
        // récupérer l'objet
        if ($utilisateur = $requete_prepare->fetch()) {
            // vérifier le mot de passe
            // le mot de passe transmis par le formulaire est le hash du mot de passe saisi
            // le mot de passe enregistré dans la base doit correspondre au hash du (hash transmis concaténé au sel)
            $hashpass = hash(algo: 'sha512', data: $mdpMembre.$utilisateur->selMembre);
            if ($utilisateur->mdpMembre == $hashpass){
                return $utilisateur;
            }

            return null;
        }
        return null;
    }
    catch (PDOException $e) {
        die('<div class = "erreur">Erreur dans la requête !<p>'.$e->getmessage().'</p></div>');
    }
```

La partie rouge de notre fonction est une simple requête préparée qui va venir récupérer l'ensemble des attributs de nos membres pour le ":loginMembre" qu'on va lui attribuer lors de l'appel de notre fonction.
Dans la partie jaune, le test qui est effectué est simplement pour tester si notre requête s'est bien effectué et qu'on arrive à attribuer les résultats dans une variable.
Dans la partie Beige, on attribue dans une variable nommée "hashpass", la concaténation du mot passe de l'utilisateur (fourni lors de l'appel de la fonction) et du selMembre (qui se trouve dans la BDD). Le tout avec la fonction hash("sha512") qui va permettre de hash l'ensemble.
Ensuite on vient vérifier que le hachage que l'on vient d'effectuer dans $hashpass est bien identique au mot de passe de l'utilisateur (qui est censé avoir le même hachage). Si ça fonctionne on retourne les informations contenu dans $utilisateurs, sinon on ne retourne rien.

Développez la vue : le formulaire d'authentification :

Notre modèle est maintenant fonctionnel mais le souci étant que pour que notre fonction soit appelée, on doit lui soumettre un login et un mot de passe. Actuellement, on ne le fait à aucun moment. C'est donc pour cela que l'on va devoir mettre en place un formulaire d'authentification :

```
vue > v_connexion.php > div#login-page
<!--
*************************************************************************************
*************************************************************
MAIN CONTENT

*************************************************************************************
************************************************************ -->
    <div id="login-page">
        <div class="container connexion">
            <form class="form-login" method="post" action="index.php?uc=connexion">
                <h2 class="form-login-heading">Identification utilisateur</h2>
                <div class="login-wrap">
                    <input type="text" class="form-control" name="txtLogin" id="txtLogin" placeholder="Login" required autofocus />
                    <br>
                    <input type="password" class="form-control" name="txtMdp" id="txtMdp" placeholder="Mot de passe" required />
                    <div class="pull-right login-social-link">

<!--La version 5 du langage HTML a ajouté un nouveau type d'attribut, qui
n'est pas interprété par le navigateur.
Ce sont tous les attributs dont le nom commence par les lettres data.
L'attribut data-toggle contient le type d'événement qui va être lié à un
bouton
data-toggle=modal /* Bouton qui ouvre une fenêtre modale -->

                        <a data-toggle="modal" href="v_connexion.php#myModal"> Moudupass oublié ?</a>
                    </div>

                    <button class="btn btn-theme btn-block" type="submit" name="cmdAction" value="validerConnexion" title="Se connecter" >
                        <i class="fa fa-lock"></i> Se connecter</button>
                    <hr>
                </div>

<!-- la fenêtre modale -->
                <div aria-hidden="true" aria-labelledby="myModalLabel" role="dialog" tabindex="-1" id="myModal" class="modal fade">
                    <div class="modal-dialog">
                        <div class="modal-content">
                            <div class="modal-header">
                                <button type="button" class="close" data-dismiss="modal" ariahidden="true">&times;</button> <!-- data-dismiss ferme les fenêtres modales-->
                                <h4 class="modal-title">Mot de passe oublié ?</h4>
                            </div>
                            <div class="modal-body">
                                <p>Entrez votre adresse mail pour réinitialiser votre mot de passe.</p>
                                <input type="text" name="txtEmail" id="txtEmail" placeholder="Email" autocomplete="off" class="form-control placeholder-no-fix" />
                            </div>
                            <div class="modal-footer">
                                <button data-dismiss="modal" class="btn btn-default" type="button">Cancel</button>
                                <button class="btn btn-theme" type="button">Submit</button>
                            </div>
                        </div>
                    </div>
                </div>
<!-- modal -->
            </form>
        </div>
</div>
```

(encore une fois, je ne souhaite pas expliquer ce formulaire, c'est du html à coup de div dans tous les sens 😅)

Hachez le mot de passe :

Pour ce faire, il suffit de créer dans le dossier Web un nouveau dossier "libAgora" dans lequel on va venir stocker un fichier contenant une fonction javascript de hachage.
Il faut donc naturellement faire référence à ce script dans le footer :

```
    <!-- pour la connexion -->
    <script src="web/libAgora/sha512.js"></script>
</body>
```

Mais également ajouter notre script au formulaire de connexion, pour que le script se lance, et donc vient hacher le mot de passe au moment où l'utilisateur clique sur le bouton "Se Connecter" :

```
        <button class="btn btn-theme btn-block" type="submit" name="cmdAction" value="validerConnexion" title="Se connecter"
            onclick="document.getElementById('hdMdp').value = hex_sha512(document.getElementById('txtMdp').value);document.getElementById('txtMdp').value = ' ';">
            <i class="fa fa-lock"></i> Se connecter</button>
            <!-- champ caché pour le mot de passe haché -->
            <input type="hidden" name="hdMdp" id="hdMdp" />
        <hr>
    </div>
```

Développez le contrôleur :

Notre vue et notre modèle sont maintenant bien codé, mais il faut maintenant adapter nos contrôleurs pour que tout fonctionne bien.

```php
<?php
/**
 * Page d'accueil de l'application AgoraBo

 * Point d'entrée unique de l'application
 * @author MD
 * @package default
 */

// démarrer la session !!!!!!!! A FAIRE AVANT TOUT CODE HTML !!!!!!!!
session_start();

require 'vue/v_header.html';    // entête des pages HTML

// inclure les bibliothèques de fonctions
require_once 'app/_config.inc.php';
require_once 'modele/class.PdoJeux.inc.php';

// Connexion au serveur et à la base (A)
$db = PdoJeux::getPdoJeux();

// Si aucun utilisateur connecté, on considère que la page demandée est la page deconnexion
// $_SESSION['idUtilisateur'] est crée lorsqu'un utilisateur autorisé se connecte (dans c_connexion.php)
if (!isset($_SESSION['idUtilisateur'])){
 require 'controleur/c_connexion.php';
} else {
```

La partie Rouge vient simplement démarrer la session.
La partie jaune vient d'abord tester si le contenu de notre $_SESSION[“idUtilisateur”] n'est pas vide, si elle est vide l'utilisateur est rediriger sur la page de connexion, sinon le code est exécuté.

Étant donné qu'on à démarré une session, il faut pouvoir gérer le cas de la déconnexion :

```php
case 'deconnexion' : {
    require 'controleur/c_deconnexion.php';
    break;
}
```

Créez un contrôleur secondaire :

Il faut maintenant créer un contrôleur qui va se charger de la connexion :

controleur > c_connexion.php > ...

```php
<?php
if (!isset($_POST['cmdAction'])){
    $action = 'demanderConnexion';
}

else {
    // par défaut
    $action = $_POST['cmdAction'];
}

switch ($action) {
    case 'demanderConnexion': {
        require 'vue/v_connexion.php';
        break;
    }

    case 'validerConnexion': {
        // vérifier si l'utilisateur existe avec ce mot de passe
        $utilisateur = $db->getUnMembre($_POST['txtLogin'],$_POST["hdMdp"]);
        // si l'utilisateur n'existe pas
        if ($utilisateur == null){
            // positionner le message d'erreur $erreur
            $erreur = "C'est null, dommage";
            // inclure la vue correspondant au formulaire d'authentification
            require "vue/v_connexion.php";

        } else {
            // créer trois variables de session pour id utilisateur, nom et prénom
            $_SESSION['idUtilisateur'] = $utilisateur->idMembre;
            $_SESSION['nomUtilisateur'] = $utilisateur->nomMembre;
            $_SESSION['prenomUtilisateur'] = $utilisateur->prenomMembre;

            // redirection du navigateur vers la page d'accueil
            header(header: 'Location: index.php');
            exit;
        }
        break;
    }
}
```

La partie rouge elle attribue à la variable $utilisateur, l'objet de la BDD qui correspond au login avec son mot de passe (à l'aide de la fonction getUnMembre). Puis elle vient vérifier si cela s'est bien passé, sinon elle renvoie un message d'erreur et renvoie à la page de connexion. Si oui, elle vient créer 3 variables de session pour l'id le nom et le prénom de notre utilisateur, et va le rediriger vers la page d'accueil.

Il faut ensuite rapidement modifier la vue afin de pouvoir gérer le message d'erreur :

```
<!--
******************************************************************************
************************************************************
 MAIN CONTENT

******************************************************************************
************************************************************ -->
    <div id="login-page">
        <div class="container connexion">

            <form class="form-login" method="post" action="index.php?uc=connexion">
                <h2 class="form-login-heading">Identification utilisateur</h2>
                <div class="login-wrap">
                    <?php
                    if (isset($erreur)) {
                        echo '<div class = "erreurCnx"><p>'.$erreur.'</p></div>';
                    }
                    ?>
                    <input type="text" class="form-control" name="txtLogin" id="txtLogin" placeholder="Login" required autofocus />
                    <br>
                    <input type="password" class="form-control" name="txtMdp" id="txtMdp" placeholder="Mot de passe" required />
                    <div class="pull-right login-social-link">
```

Elle utilise un style précis "erreurCnx", qu'il faut donc allez ajouter à notre feuille de style :

```
.erreurCnx {
 display:block;
 color: red;
 font-weight:bold;
}
```

Adaptez les vues v_header et v_accueil :

Dans le fichier "v_header.html" on va avoir besoin de mettre du code php, donc il faut le renommer en "v_header.php" et donc lui ajouter le code nécessaire :

```
<!--header start-->
<header class="header black-bg">
  <div class="sidebar-toggle-box">
    <div class="fa fa-bars tooltips" data-placement="right" data-original-title="Toggle Navigation"></div>
  </div>
  <!--logo start-->
  <a href="index.php" class="logo"><b>Ago<span>rA</span></b></a>
  <!--logo end-->
  <div class="top-menu">
    <ul class="nav pull-right top-menu">
      <?php
      // Si aucun utilisateur connecté, on affiche la demande de connexion
      if (!isset($_SESSION['idUtilisateur'])){
        echo '<li><a class="logout" href="index.php?uc=connexion&action=demanderConnexion">Se connecter</a></li>';
      }
      else {
        echo '<a href="" class="userAgo">'.$_SESSION["prenomUtilisateur"].' '.$_SESSION["nomUtilisateur"].'</a>';
        echo '<li><a class="logout" href="index.php?uc=deconnexion&action=demanderDeconnexion">Se déconnecter</a></li>';
      }
      ?>
    </ul>
  </div>
</header>
<!--header end-->
```

Pour “v_accueil.html” pareil, il faut le renommer et lui ajouter le code nécessaire :

```
<!-- ********************************************************************************************************
  MAIN CONTENT
  ********************************************************************************************************
<div class="container">
    <div id="blocImage">

        <!--<img src="vue/img/greece-1162816_640.jpg" alt="Agora en Grèce" class="responsive">-->
           <img src="web/img/greece-3348294_1280.jpg" alt="" class="responsive">

        <div id="blocImageTexte">
           Bonjour <?php echo $_SESSION["prenomUtilisateur"].' '.$_SESSION["nomUtilisateur"] ; ?>
        </div>
    </div>
</div>
```

Cela permet de pouvoir dire Bonjour à l’utilisateur avec les bonnes informations le concernant, et pour ce faire on utilise notre session.

Il faut également penser à allez dans “index.php” et modifier les endroits où on appelait notre “v_accueil.html” pour le mettre en “.php”.

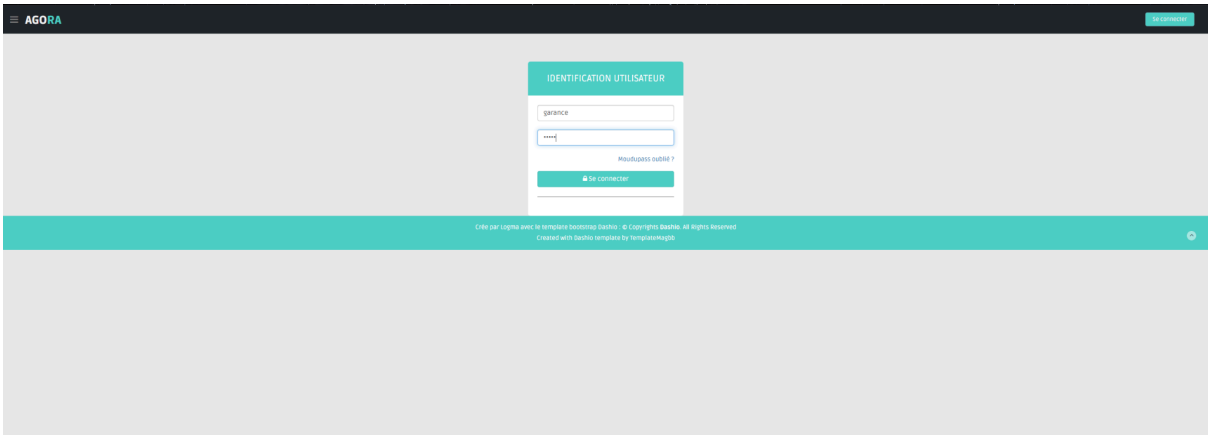
Déconnexion :

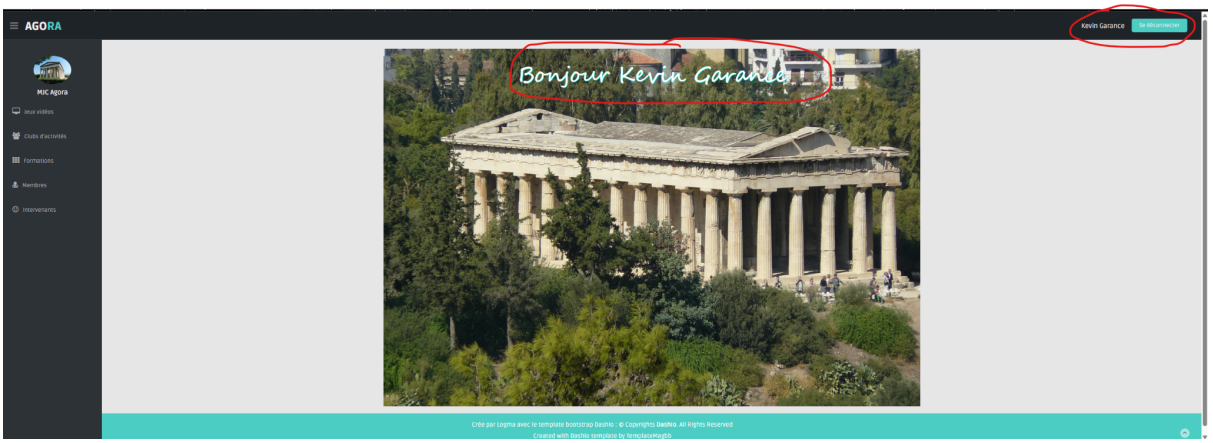
La dernière étape consiste à gérer la déconnexion, afin que l’utilisateur soit déconnecté et redirigé vers la page de connexion :

controleur > c_deconnexion.php

```
1   <?php
2   // supprimer la session
3   session_destroy();
4   // redirection vers le controleur principal !!!!! pas de HTML avant !!!!!
5   header(header: 'Location: index.php');
6   ?>
```

C’est terminé !


AGORA
Se connecter
IDENTIFICATION UTILISATEUR
garance
Maudupass oublié ?
Se connecter
Crée par Logma avec le template bootstrap Dashio : © Copyrights Dashio. All Rights Reserved
Created with Dashio template by TemplateMagbb


AGORA
Kevin Garance
Se déconnecter
MJC Agora
Jeux vidéos
Clubs d'activités
Formations
Membres
Intervenants
Bonjour Kevin Garance
Crée par Logma avec le template bootstrap Dashio : © Copyrights Dashio. All Rights Reserved
Created with Dashio template by TemplateMagbb