synthèse projet Agora BO

Contexte:

- La MJC Agora ne dispose pas d'un service informatique. Elle a fait appel à la société de services du numérique (ESN) Logma dans laquelle je suis embauché.. Je participe au développement du site d'administration des données de la MJC. Mes collègues de Logma ont déjà démarré le développement et mettent à ma disposition : La base de données MySQL Jeux . Le design général de l'application et la page d'accueil, Un exemple de gestion des genres de jeux.

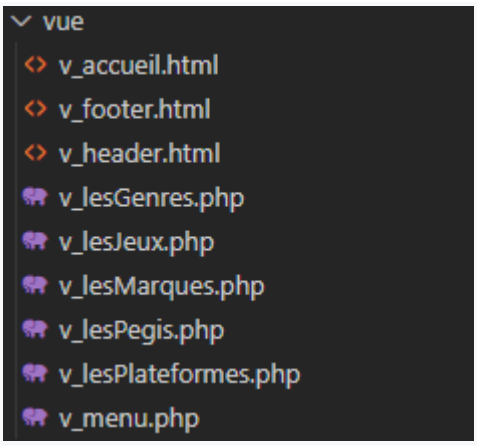
Synthèse mission 1 :

- L'architecture MVC (Modèle-Vue-Contrôleur) est un modèle de conception utilisé dans le développement logiciel, en particulier dans le développement d'applications Web. Elle divise une application en cinq composants principaux, chacun ayant un rôle spécifique :
- voici un exemple de MVC:

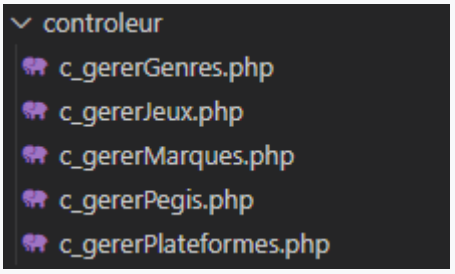

```php
public function getLesGenres(): array
{
    $requete =  'SELECT idGenre as identifiant, libGenre as libelle
                    FROM genre
                    ORDER BY libGenre';
    try {
        $resultat = PdoJeux::$monPdo->query($requete);
        $tbGenres  = $resultat->fetchAll();
        return $tbGenres;
    } catch (PDOException $e) {
        die('<div class = "erreur">Erreur dans la requête !<p>'
            . $e->getmessage() . '</p></div>');
    }
}
```

voici un exemple d'une fonction de la table les genres, on peut voir le pdo avec la requête sql, puis une fonction try et catch

On crée un fichier modèle dans la quelle on va cré un fichier class.Pdo.inc.php .Le modèle représente la logique métier de l'application, c'est-à-dire les données manipulées, les règles de traitement et les interactions avec la base de données.



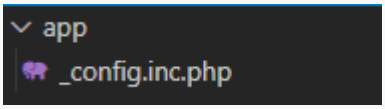
On crée un fichier vue dans laquelle on va créer un fichier v_latable.php ou html. La vue concerne tout l'interface graphique,c'est-à dire ce que l'utilisateur va voir à l'écran.



```php
if (!isset($_POST['cmdAction'])) {
    $action = 'afficherGenres';
}
else {
    // par défaut
    $action = $_POST['cmdAction'];
}

$idGenreModif = -1;     // positionné si demande de modification
$notification = 'rien'; // pour notifier la mise à jour dans la vue

// selon l'action demandée on réalise l'action
switch($action) {

    case 'ajouterNouveauGenre': {
        if (!empty($_POST['txtLibGenre'])) {
            $idGenreNotif = $db->ajouterGenre($_POST['txtLibGenre']);
            // $idGenreNotif est l'idGenre du genre ajouté
            $notification = 'Ajouté';   // sert à afficher l'ajout réalisé dans la vue
        }
      break;
    }

    case 'demanderModifierGenre': {
            $idGenreModif = $_POST['txtIdGenre']; // sert à créer un formulaire de modification pour ce genre
        break;
    }

    case 'validerModifierGenre': {
        $db->modifierGenre($_POST['txtIdGenre'], $_POST['txtLibGenre']);
        $idGenreNotif = $_POST['txtIdGenre']; // $idGenreNotif est l'idGenre du genre modifié
        $notification = 'Modifié';  // sert à afficher la modification réalisée dans la vue
        break;
    }

    case 'supprimerGenre': {
        $idGenre = $_POST['txtIdGenre'];
        $db->supprimerGenre($idGenre);

     //  à compléter, voir quelle méthode appeler dans le modèle
        break;
    }
}

// l' affichage des genres se fait dans tous les cas
$tbGenres  = $db->getLesGenres();
require 'vue/v_lesGenres.php';

?>
```

voici un exemple de controleur , on peut voir l'instruction switch avec différentes condition .

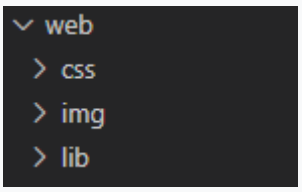
Il reçoit les demandes de l'utilisateur, traite ces demandes, interagit avec le modèle pour récupérer ou manipuler les données nécessaires, puis transmet les résultats à la vue appropriée pour l'affichage.



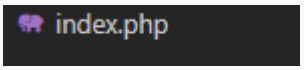
```php
<?php
/**
* paramètres de configuration de l'application AgoraBo
*
* @package default
* @author md
* @version 1.0
*/
// gestion d'erreur
ini_set('error_reporting', E_ALL); // en phase de développement
//ini_set('error_reporting', 0); // en phase de production
// constantes pour l'accès à la base de données
define('DB_SERVER', 'localhost'); // serveur MySql
define('DB_DATABASE', 'jeux'); // nom de la base de données
define('DB_USER', 'userAgoraBo'); // nom d'utilisateur
define('DB_PWD', 'SBcmeHtEAIpbB9Uk'); // mot de passe
define('DSN','mysql:dbname='.DB_DATABASE.';host='.DB_SERVER);
?>
```

voici un exemple d'application, on peut voir les différents paramètres avec le mot de passe , utilisateur, nom de base de donnée, connexion serveur .

Dossier dans laquelle on va crée un fichier paramètres de configuration de l'application Agora



La partie web concerne les images, feuille de styles de l'application.



La partie index sert à ce que les différents fichier de différents dossier peuvent communiquer.

```php
switch ($uc) {
    case 'index': {
            $menuActif = '';
            require 'vue/v_menu.php';
            require 'vue/v_accueil.html';
            break;
        }
    case 'gererGenres': {
            $menuActif = 'Jeux';    // pour garder le menu correspondant ouvert
            require 'vue/v_menu.php';
            require './controleur/c_gererGenres.php';
            break;
        }
    case 'gererPlateformes': {
            $menuActif = 'Jeux';
            require 'vue/v_menu.php';
            require './controleur/c_gererPlateformes.php';
            break;
        }
    case 'gererMarques': {
            $menuActif = 'Jeux';
            require 'vue/v_menu.php';
            require './controleur/c_gererMarques.php';
            break;
        }
    case 'gererPegis': {
            $menuActif = 'Jeux';
            require 'vue/v_menu.php';
            require './controleur/c_gererPegis.php';
            break;
        }
    case 'gererJeux': {
        $menuActif = 'Jeux';
        require 'vue/v_menu.php';
        require './controleur/c_gererJeux.php';
        break;
    }
    default:{
    break;
    }
}


// Fermeture de la connexion (C)
$db = null;

// pied de page
require("vue/v_footer.html");
```

synthèse mission 2 :

Le hachage est un processus cryptographique qui prend une chaîne de caractères en entrée et produit une chaîne de caractères de longueur fixe en sortie, appelée hash. C'est utilisée par les développeurs pour sécurisés des données sensibles ou cas ou des pirates prennent la main sur le code.

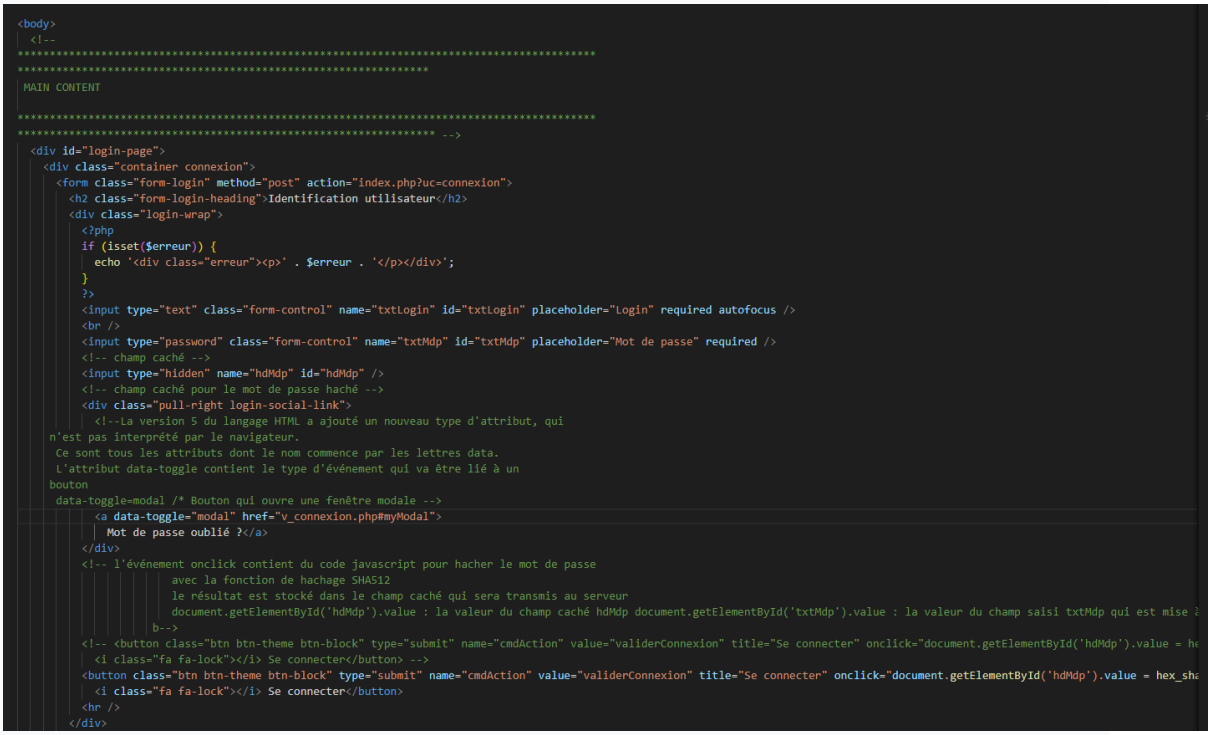
Il préférable de haché avec javascript car celui offre plus de fonctionnalité que php comme le captchat.

pour générer des sels on peut utiliser la fonction PHP random_bytes (int $length) : string

Cette fonction génère une chaîne de caractères de longueur arbitraire d'octets aléatoires cryptographiques qui convient à un usage cryptographique, comme lors de la génération de sels, de clés ou de vecteurs d'initialisation.

La partie vue du fichier connexion est divisé en 2 partie :

- Une phase d'identification :



- Une Fenêtre modale si on clique sur Mot de passe oublié

```
    <!-- 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>
          </div>
        </div>
      </div>
    </div>
    <!-- modal -->
  </form>
 </div>
</div>
<script src="./web/libAgora/sha512.js"></script>
</body>

</html>
```

Pour le hachage du mot de passe : on utilise la fonction javascript sha512 :

puis on integre dans la partie vue de connexion ce morceau :

```
Connexion" title="Se connecter" onclick="document.getElementById('hdMdp').value = hex_sha512(document.getElementById('txtMdp').value);document.getElementById('txtMdp').value =

xion" title="Se connecter" onclick="document.getElementById('hdMdp').value = hex_sha512(document.getElementById('txtMdp').value);document.getElementById('txtMdp').value = ' ';">
```

Puis on termine par ajuster les fichiers dans l'index.php.

Synthèse mission 3 :

Twig est un moteur de template développé pour le langage de programmation PHP. Son objectif principal est de simplifier la création de vues HTML en séparant la logique de présentation du code PHP.

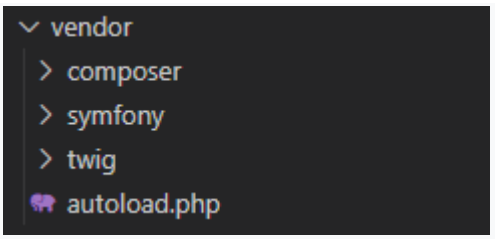
Twig est généralement utilisé dans le cadre du développement web avec PHP, notamment dans les frameworks MVC tels que Symfony.

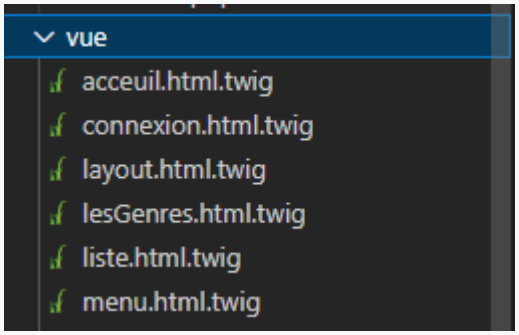
En utilisant Twig, les développeurs peuvent réduire le risque d'injection de code malveillant dans les templates. L'interprétation stricte de la syntaxe Twig garantit que seuls les éléments de template valides sont exécutés, ce qui réduit la surface d'attaque potentielle.

Cas d'utilisation pour la mission 3 :

- teléchargement de twig :
- vérification de la version PHP avec la commande php –v

- ajouter Twig à l’application avec la commande Composer require "twig/twig:^2.0.
- cré le dossier vendor contenant les bibliothèque



- Le template père est communément appelé layout dans le dossier vue
- 

- ensuite ajuster les codes en remplacant le code php par la template twig (dans le fichier index.php,c_connexion,c_gerer_genres,app.php)