Moulin Vincent

Compte rendu TP Agora

Table des matières

1) Cahier des charges :

- Développement Sites Web
- Respect de la norme MVC
- Gérer la connexion et la déconnexion
- Sécurise la connexion aux sites et à la base de données

2) Objectif :

L'objectif de cette mission était de mettre à jour le site du mjc ainsi que sa base de données afin de respecter les normes mvc, de pouvoir gérer la connexion et la déconnexion d'un membre et ainsi de faire marcher les pages pegis, jeux, etc.

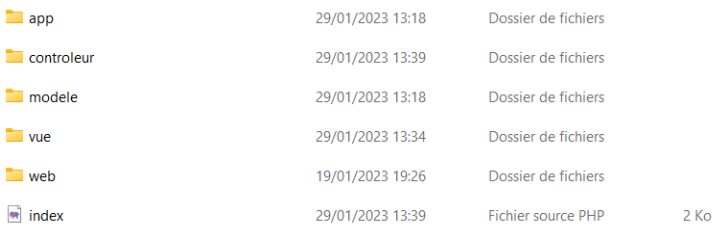
3) Mission 1 :

a) Analyse de l'environnement

Dans ce premier temps on a pu étudier l'environnement du site dans l'état dans lequel il est et l'environnement de travail, ainsi que la base de données jeux qui va nous permettre de compléter les pages du site

b) Architecture application

Dans la deuxième partie on a pu adapter le projet à l'architecture MVC qui permet une meilleure lecture, meilleur maintenance et permettant meilleur compréhension ; chaque partie du site sera séparée selon leur utilisation (exemple comme la vue, conception, modèle, …) et respectant d'autre convention (exemple le nommage exemple cammelCase, PascalCase, snake_case). Ainsi l'index permettra juste d'accéder aux pages et les lie et ne regroupe pas tous le code du projet cette architecture ressemble un peu près à ceci



c) Développement des pages

Dans cette troisième et dernière partie le principal objectif était de développer les pages du site web permettant de voir des tables de la base de données (les tables jeux, pegis, marque, genre et plateforme) seront ensuite appeler dans l'index afin de pouvoir accéder à la page et le css permettant de les voir dans l'accueil ; les fonctions sont développées dans le modèle et utiliser dans le contrôleur afin de les afficher dans un fichier approprié dans la vue

```
</div>
<div class="panel-body">
    <table class="table table-striped table-advance table-hover">
    <thead>
      <tr class="tableau-entete">
        <th><i class="fa fa-bullhorn"></i> Identifiant</th>
        <th><i class="fa fa-bookmark"></i> Nom </th>
        <th><i class="fa fa-bookmark"></i> Prix </th>
        <th><i class="fa fa-bookmark"></i> Date </th>
        <th><i class="fa fa-bookmark"></i> Genre </th>
        <th><i class="fa fa-bookmark"></i> Plateforme </th>
        <th><i class="fa fa-bookmark"></i> Pegi </th>
        <th><i class="fa fa-bookmark"></i> Marque </th>

        <th></th>
      </tr>
    </thead>
    <tbody>
```

Code permettant d'afficher les titres des colonnes.

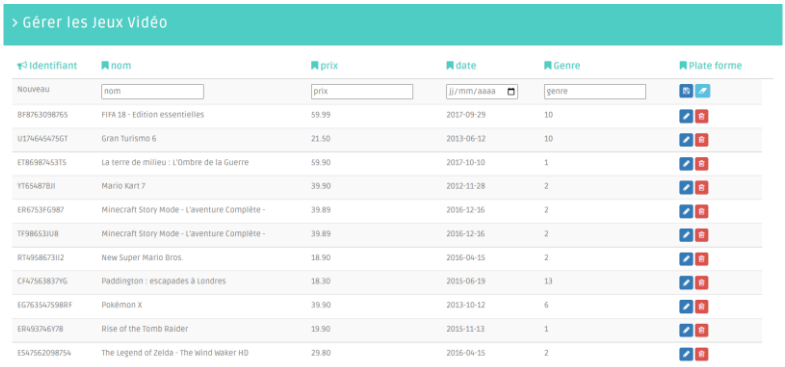
```
<form action="index.php?uc=gererJeu" method="post">
    <td>Nouveau</td>
    <td>
        <input type="text" id="txtnom" name="txtnom" size="24" required minlength="4"  maxlength="24"  placeholder="n
    </td>
    <td>
```

Code permettant de pouvoir modifier les colonnes de la base de données avec l'aide d'une fonction modifier qui se trouve dans le modèle, puis avec d'autre fonction nous avons la possibilité de supprimer de ligne ou d'en rajouter dans la base de données et les affichant aussi.

```
<?php
foreach ($tbJeu as $Jeu) {
?>
  <tr>

    <!-- formulaire pour modifier et supprimer les genres-->
    <form action="index.php?uc=gererPlateforme" method="post">
    <td><?php echo $Jeu->id; ?><input type="hidden"  name="txtrefJeu" value="<?php echo $Jeu->identifiant; ?>" />
    <td><?php
        if ($Jeu->id != $refJeuModif) {
            echo $Jeu->nom;
            ?>
            </td>
            <td>
                <?php echo $Jeu->prix ?>
            </td>
            <td>
```

Code permettant lui d'afficher toute les lignes qu'il y a dans une table de la base de donnée, en récupèrent ce qu'il y a dans l'objet de la table. Ceci nous permet d'obtenir des pages comme ceci



Nous pouvoir qu'on possède la possibilité de modifier, supprimer des ligne grâce aux fonctions dans le modèle ; cette application étant développé en PHP, nous pouvons récupérer les informations et les stocker avec l'aide d'un utilisateur admin root récupérer dans l'app avec le fichier config.

```php
// 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', 'root');          // nom d'utilisateur
define('DB_PWD', '');               // mot de passe
define('DSN','mysql:dbname='.DB_DATABASE.';host='.DB_SERVER);
```

4) Mission 2 :

a) Sécurisation de la base de donnée

Dans cette partie nous avons pu créer un utilisateur AgoraBo permettant de sécuriser et éviter d'utiliser l'utilisateur root, l'utilisateur AgoraBo ayant seulement les droits sur la base de données jeux qui sera ensuite renommée en agora afin d'une meilleure compréhension

b) Récupération des utilisateurs

Dans cette partie l'objectif était de récupérer les utilisateurs de la base de donnée et puis de hash le mot de passe afin qu'il soit sécurisé et illisible en utilisent la méthode du sel, nous pouvons voir la fonction qui permet ceci ci-dessous

```php
public function getUnMembre(string $loginMb, string $mdpMembre): ?object {
    try {
        // préparer la requête
        $requete_prepare = PdoJeux::$monPdo->prepare('SELECT idMembre, prenomMembre, nomMembre,
            mdpMembre, selMembre FROM membre WHERE loginMembre = :unLoginMembre');
        // associer les valeurs aux paramètres
        $requete_prepare->bindParam(':unLoginMembre', $loginMb, PDO::PARAM_STR);
        // exécuter la requête
        $requete_prepare->execute();
        $utilisateur = $requete_prepare->fetch();
        $mdpHash = hash('SHA512', $mdpMembre.$utilisateur->selMembre);
        // récupérer l'objet

            // vérifier le mot de passe
        if ($utilisateur->mdpMembre==$mdpHash){
            return $utilisateur;
        }

        else{
            return null;
        }

    }
    catch (PDOException $e) {
        die('<div class="error">Error in the query!<p>'. $e->getMessage(). '</p></div>');
    }
}
```

c) Authentification

Nous avons en dernier pus développer les fonctions qui nous permette de nous connecter au site avec les mots de passe et login précédemment récupérer, aussi nous avons pu développer la page nous permettant de nous connecter, ainsi que de s'occuper de hash les mots de passe des utilisateurs.

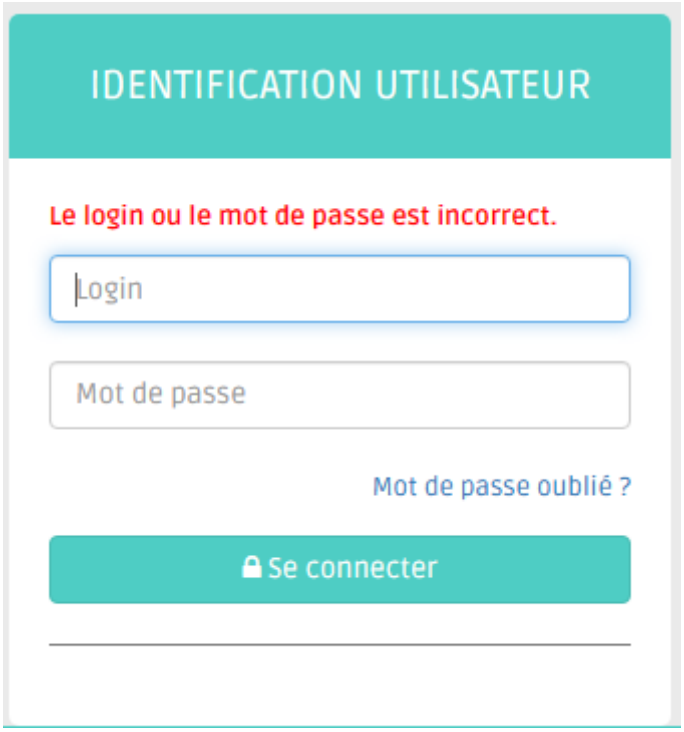
```php
ue > v_connexion.php
<div id="login-page">
    <div class="container connexion">
        <form class="form-login" method="post" action="index.php?uc=connexion">
            <h2 class="form-login-heading">Identification utilisateur</h2>
            <div class="login-wrap">
                <input 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">
                    <a data-toggle="modal" href="v_connexion.php#myModal"> Mot de passe oublié ?</a>
                </div>
                <button class="btn btn-theme btn-block" type="submit" name="cmdAction" value="validerConnexion" title="Se connecter" onclick="document.g
                    <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>

            <!-- 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" aria-hidden="true">&times;</button> <!-- data-dismiss ferme les fen
                            <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-f
                        </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>
```

Ce code nous permet d'obtenir la page de connexion.

Nous pouvons nous connecter avec les l'aide de mot de passe et login de membre stocker dans la base de données, ainsi maintenant que nous arrivons sur le site nous atterriront sur une page comme ceci



Cette page nous permet de nous connecter grâce à la fonction get un membre, fichier v_connexion et c_connexion

```php
<?php

if (!isset($_POST['cmdAction'])){
    $action = 'demanderConnexion';
} else {
    $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) {
            $erreur = 'Le login ou le mot de passe est incorrect.';
            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('Location: index.php');
            exit;
        }
        break;
    }
}

?>
```

Ce code nous permet d'utiliser la fonction get un membre et en récupérer les information (id, mot de passe et login) dans un tableau afin de tester si ceci corresponde bien à ceux présent dans la base de données et si ceci est le cas alors ceci autorisera la connexion.

Nous avons aussi utiliser les méthodes session nous permettant de récupérer les variables entre les pages et pouvoir les utiliser dans n'importe quels fichier et à la fin on doit appeler la fonction session_destroy(), nous permettant de fermer la session ouverte afin que celle-ci ne soit pas ouvert en permanence et nous permettant de nous déconnecter et de retourner sur le formulaire de déconnexion.

5) Mission 3 :

a) Objectif

L'objectif de cette partie était de mettre à jour les fichiers du projet pour utiliser Twig comme moteur de templates, en adaptant les vues existantes. L'objectif était de transformer le code PHP en structure MVC et d'utiliser les fonctionnalités de Twig pour simplifier l'affichage des données et améliorer la lisibilité du code.

b) Mise en place de Twig

Pour commencer, il était nécessaire de télécharger Composer, un gestionnaire de dépendances, afin de pouvoir utiliser Twig dans le projet. Une fois que Composer était installé, il a fallu mettre à jour tous les fichiers de vue, tels que le menu, les vues spécifiques (v_pegis, etc.), ainsi que les appels correspondants dans les contrôleurs et l'index.

L'utilisation de Twig nécessite de respecter les conventions de nommage spécifiques à Twig. Par exemple, au lieu d'utiliser la syntaxe PHP pour afficher une variable avec la balise <?php echo $var; ?>, on utilise la syntaxe Twig avec une double accolade {{ var }}. Cela permet d'afficher directement la variable et simplifie la lecture du code.

Ici nous pouvons voir un code twig :

```
{% extends 'base.html.twig' %}

{% block title %}Accueil{% endblock %}

{% block body %}
<style>
    .example-wrapper { margin: 1em auto; max-width: 800px; width: 95%; font: 18px/1.5 sans-serif; }
    .example-wrapper code { background: #F5F5F5; padding: 2px 6px; }
</style>
```

Et ici la même chose en html :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Accueil</title>
    <style>
        .example-wrapper { margin: 1em auto; max-width: 800px; width: 95%; font: 18px/1.5 sans-serif; }
        .example-wrapper code { background: #F5F5F5; padding: 2px 6px; }
    </style>
</head>
<body>
```

Nous pouvons voir que le code twig est plus simple à la compréhension et moins lourd demande moins de balise et na pas forcément besoin de toute les balise de base d'html, sont avantage à twig ce qu'il réduit le temps d'écriture du code.

La mise en place de Twig a permis d'harmoniser le code en utilisant les fonctionnalités avancées de Twig, telles que les boucles, pour manipuler les données et générer les vues de manière plus concise. Cela a également amélioré la lisibilité du code en réduisant la quantité de balise permettant de plus facilement lire le code et de voir à quoi correspond chaque ligne.

En résumé, la mise en place de Twig dans le projet a permis d'adapter les vues existantes en utilisant une syntaxe simplifiée et rend le code plus concis et facilement maintenable.

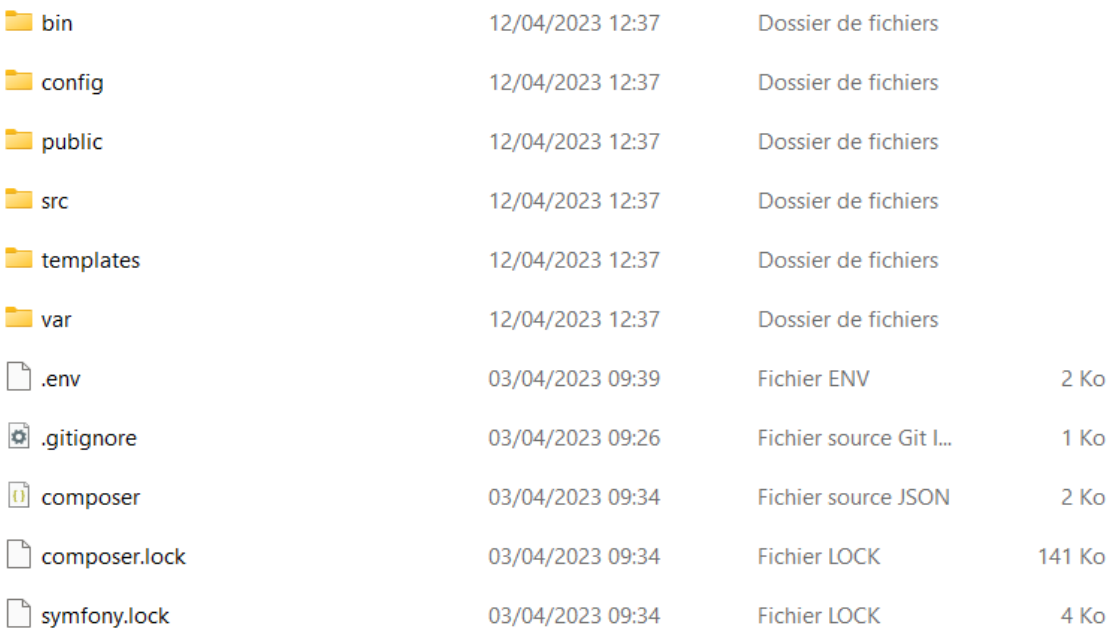
6) Mission 4 :

a) Objectif

L'objectif de cette partie était comme la partie précédente d'adapter le code à un framework (extension qui contient plusieurs, module, composant qui servent d'aide ou autre, …) symfony qu'il est un framework php en mvx permettant de limiter le code fastidieux, réduire le temps de développement.

b) Symfony

Pour mettre en place symfony nous avons dû télécharger composer et twig et adapter les modèle des fichier vue en twig, déjà nous devons créer un projet avec symfony qui se présentera sur cette forme



Les fichier vue se trouverons eux dans template, ainsi les fichier contenant image, css et js seront stocker dans les assets afin de pouvoir les réutiliser plus facilement ; quand au controller dans src et controller, chaque controller devra contenir ceci, ceci est soit fait automatiquement quand on utilise une des variable ou pourra être gérer grâce au bundle make, maker est outil de symfony qui permet de simplifier la création de fichier, il permet par exemple de générer des formulaire de connexion avec ces contrôleur et vue ; permet aussi de générer des contrôleur avec toute les dépendance nécessaire ainsi il créa aussi un fichier dans templates qui correspondra à la vue du contrôleur.

```php
// src/Controller/GenresController.php
namespace App\Controller;
use Symfony\Component\HttpFoundation\Response;
require_once 'modele/class.PdoJeux.inc.php';
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Session\SessionInterface;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Bundle\FrameworkBundle\Controller\RedirectController;
use PdoJeux;
```

Ceci représentant tous les fichier nécessaire on fonctionnement du controller l'index sera remplacer, le fichier pour être exécuter devra être démarrer via symfony en démarrer un serveur local et démarrer aussi wamp pour que notre serveur puisse accéder à la base de données, symfony permet de multiple chose comme la possibilité de voir les logs, voir les présentent travaillent sur le projet, une meilleur gestion des rôle (admin, user) .

Symfony permet une gestion plus simple des require et des dépendance grâce à aux routes, render et la fonction assets ; les routes permettent de définir les différentes URL auxquelles les utilisateurs peuvent accéder dans l'application. Chaque route est associée à une action spécifique, qui est exécutée lorsque l'URL correspondante est demandée et permettent de gérer quelle utilisateurs auront le droit d'accéder aux route en configurant le fichier security.yaml.

```php
/**
* @Route("/connexion/valider", name="connexion_valider")
*/
```

```yaml
# Note: Only the *first* access control that matches will be used
access_control:
    - { path: ^/app_home, roles: ROLE_USER }
    # - { path: ^/profile, roles: ROLE_USER }
```

En ce qui concerne le rendu des vues, Symfony fournit la fonction render. Cette fonction permet d'afficher une vue spécifique dans le navigateur. Lorsque la fonction render est appelée, Symfony se charge de récupérer le fichier de template correspondant à la vue, d'exécuter le rendu en y injectant les données fournies par le contrôleur, puis d'envoyer le résultat final au navigateur de l'utilisateur.

```php
return $this->render('connexion.html.twig');
```

Quand à la fonction assets dans Symfony elle permet de gérer les fichiers images, les fichiers CSS et les scripts JavaScript. Elle facilite l'accès à ces ressources en fournissant un moyen simple de spécifier leur emplacement dans le projet. Par exemple, au lieu d'écrire manuellement le chemin complet d'une image ou d'un fichier CSS, on peut utiliser la fonction assets pour générer automatiquement le chemin approprié jusqu'au dossier public. Cela rend le code plus maintenable et facilite le déplacement ou le changement d'emplacement des ressources.

```twig
<img src="{{asset('/assets/img/logo2.png')}}" alt="" />
```

c) Mise en place

Donc pour pouvoir changer agora à symfony nous avons du adapter les appellation des template en utilisant render, créer des routes pour chaques contrôleur ainsi que de les récréer en reprenant une parties du code, quand au modèle celui ce vue inchangé , ainsi que modifier l'appelations des images, nous avons aussi du lier le projet à la base de données ceci s'effectue dans le fichier .env

```
###< symfony/framework-bundle ###
AGORA_DSN='mysql:dbname=agora;host=localhost'
AGORA_DB_USER='userAgoraBo'
AGORA_DB_PWD='r4qW9ihvtPQPwLdS'
```

Nous avons utilisé la méthodes de mettre les information de connexion dans des variables afin de pouvoir les réutiliser dans le fichier modèle la connexion à la bdd peut s'effectuer des cette manière aussi

```
DATABASE_URL="mysql://admin:W_ESwNpg*k_bWUrB@127.0.0.1:3306/covoiturage?serverVersion=8&charset=utf8mb4"
```

Cette méthode est moins pratique car elle met toute les information dans une même variable

Puis dans chaque contrôleur et pour chaque fonction on du être préciser des routes afin de pouvoir réutiliser celle-ci dans le template ou dans d'autre fichier ; par exemple ici nous allons chercher à utiliser la fonction associer à la route jeu_ajouter, ce qui nous permet ensuite de la réutiliser

```
<form action="{{ path('jeux_ajouter') }}" method="post">
```

d) Conclusion

En conclusion, Symfony nous simplifie le développement d'un projet en fournissant des fonctionnalités telles que la gestion des dépendances, la configuration des routes et le rendu des vues. Il permet également de maintenir une structure claire et de gagner du temps grâce à l'utilisation de Composer, Twig et des outils comme Maker, Cependant, son apprentissage et sa complexité peuvent rendre sa prise en main initiale difficile pour les développeurs. Une compréhension approfondie des concepts clés de Symfony et une connaissance avec sa structure de projet sont nécessaires.

7) Difficulté rencontrée :

a) Mission 1

Dans la mission 1 j'ai pu rencontrer plusieurs difficultés, comme le développement de la page jeux et pegis, qu'ils eux ont plus de données dans leurs tables que les autres donc avait besoin d'avoir son code adapter afin de tous les afficher et que le css ne déborde pas à cause de , et ayant aussi besoin d'adapter les fonctions de suppression, modification, etc. J'ai pu rencontrer quelque problème avec l'index et les autres pages qui m'afficher rien quelquefois à cause de virgule ou de fonction avec de mauvais appelle.

b) Mission 2

Dans la mission 2 j'ai rencontré quelque erreur comme les problèmes dans la fonction get un membre pour hash le mot de passe avec le technique du sel et la vérification avec le mot de passe, et j'ai aussi pu rencontrer plusieurs erreur à cause la modification de l'index.

c) Mission 3

Les difficultés que j'ai pu rencontrer dans la mission 3 on était d'adapter les fonctions et la vue, comme objectif était aussi d'afficher toute les tables de la base de données agora donc certaine contrôleur on du voir s'adapter ceci à était le problème, j'ai pu rencontrer aussi un problème avec le menu qui s'afficher en double mais cela venait d'un double appelle du menu.

d) Mission 4

Dans la mission 4 j'ai pu rencontrer comme difficulté, a la mise en place et l'installation de symfony avec la reconnaissance des variable local celle-ci ne fut pas reconnu dans l'invité de commande, ce problème à pu être régler en déplacent symfony, puis j'ai rencontré un problème avec l'adaptation des différent fichier contrôleur.

8) Conclusion :

En conclusion dans ce projet nous avons du mettre en place une application web respectant des contrainte, toute en ayant des demande évolutif au cours de la mission en respectant de nouvelle contrainte, dans ce projet nous avons pu aussi surpasser des difficulté en cours de projet.