Presentation PHP MVC

Plan

I. Rappel sur les Sessions

A. Session

1. Définition

Une session en PHP correspond à une façon de stocker des données différentes pour chaque utilisateur en utilisant un identifiant de session unique.

Les identifiants de session vont généralement être envoyés au navigateur via des cookies de session et vont être utilisés pour récupérer les données existantes de la session.

2. Intérêt

Un des grands intérêts des sessions:

- conserver des informations pour un utilisateur lorsqu'il navigue d'une page à une autre.
- Les informations de session sont stockées côté serveur ce qui fait que les sessions vont pouvoir être beaucoup plus sûres que les cookies.

NB : Notez toutefois que le but des sessions n'est pas de conserver des informations indéfiniment mais simplement durant une « session » de navigation.

3. Etapes D'utilisation de la session

Les étapes d'utilisation de la session sont:

- Démarrer la session
- Définir et récupérer les variables de sessions
- Terminer la session et détruire les variables de sessions

a) Démarrer la session

Pour pouvoir utiliser les variables de session, il va avant tout falloir qu'une session soit démarrée à un moment ou à un autre.Pour démarrer une session en PHP, on va utiliser la fonction **session_start()**.

Cette fonction va générer un identifiant de session unique qui va généralement être envoyé au navigateur sous forme de cookie sous le nom **PHPSESSID**.

NB: Par ailleurs, notez qu'il va falloir appeler **session_start()** dans chaque page où on souhaite pouvoir accéder aux variables de session.

b) Définir et récupérer les variables de sessions

Pour définir et récupérer les valeurs des variables de session, nous allons pouvoir utiliser la variable superglobale **$_SESSION.**

Cette superglobale est un tableau associatif qui stocke les différentes variables de sessions **avec leurs noms en index du tableau et leurs valeurs en valeurs du tableau.**

#definir.session.php

```php
<?php
  //On démarre une nouvelle session
   session_start();

  //On définit des variables de session
   $_SESSION['prenom'] = 'Pierre';
   $_SESSION['age'] = 29;
?>
```

#recuper.session.php

```php
 <?php
//On démarre une nouvelle session
   session_start();
           echo 'Bonjour ' .$_SESSION['prenom']. ',
           tu as ' .$_SESSION['age']. ' ans';
?>
```

c) Terminer la session et détruire les variables de sessions

Une session PHP se termine généralement automatiquement lorsqu'un utilisateur ferme la fenêtre de son navigateur.

Il peut cependant parfois être souhaitable de terminer une session avant. Pour faire cela, nous allons pouvoir utiliser les fonctions `session_destroy()` qui détruit toutes les données associées à la session courante et `session_unset()` qui détruit toutes les variables d'une session.

La fonction `session_destroy()` va supprimer le fichier de session dans lequel sont stockées toutes les informations de session. Cependant, cette fonction ne détruit pas les variables globales associées à la session (c'est-à-dire le contenu du tableau `$_SESSION`) ni le cookie de session.

Pour supprimer totalement la session on peut utiliser :

- `session_unset()` : qui détruit automatiquement toutes les variables d'une session.
- `unset( $_SESSION[key])` : qui détruit la variable associée cette clé dans la session.

II. Gestion de Projet

A. Backlog Quizz SA

Lien
https://docs.google.com/document/d/1rsyXPQKOqdl1LNyd8xUT0jbpW_fvu_-1G9iFX5hjSLY/edit?usp=sharing

B. Création Backlog Quizz SA sur Trello

C. Planification du Sprint 1

Sprint	ID	Description	Tâches		Délai	Membres	Démonstration
			Num	Description			
Sprint 1	Us1	En tant Développeur,j'initialise les outils de Gestion projet et de versionning	T1	Création des Tableaux Trello	2min	Wane	
			T2	Création du repository du Projet	2min		
	Us2	En tant Développeur,je crée la structure du projet avec l'approche MVC	T1	Création de la structure MVC du Projet	10min		
			T2	Définir le fichier Json	20min		
			T3	Définition des constantes du projet	15min		
			T4	Création du front controller	15min		
			T5	Création du router	15min		
			T6	Création des controllers	20 min		
			T7	Création de la couche accès aux données(models)	10min		

	Us2	En tant qu'utilisateur (Admin ou Joueur) je peux me connecter au jeux	T1	Réaliser les fonctionnalités du modèle	20 min		**1. Scenario Nominal** a. Recuperer les donnees saisies b. Valider les Données c. Vérification de l'existence de ce login et de ce mot de passe dans le fichier json d. Stockage des données de l'utilisateur dans la session e. Redirection vers le menu accueil. **2. Alternance** b.1 stockage des erreurs de validation dans la session b.2 redirection vers la vue de connexion c.1 stockage de l'erreur non existence de l'utilisateur dans la session c.2 redirection vers la vue de connexion
			T2	Réaliser les fonctionnalités du contrôleur 1. Valider les des Données 1. Faire la Validation Front 2. Faire la Validation Back 2. Règles de Validation 1. Login et mot de passe obligatoire 2. Login doit etre un email 3. Le Mot de doit contenir au moins 6 caractères et est formé au moins une lettre et au moins un chiffre 3. Gérer les autorisation	60min	

<table>
<tr><td rowspan="4"></td><td></td><td></td><td>T3</td><td>Intégration de la page de connexion</td><td>30 min</td><td rowspan="3"></td><td></td></tr>
<tr><td rowspan="3">Us3</td><td rowspan="2">En tant qu'Admin je peux lister les Joueurs</td><td>T1</td><td>Réaliser les fonctionnalités du modèle</td><td>10 min</td><td rowspan="3">1. Scénario Nominal
a. Récupérer les joueurs a partir du modèle
b. charger la vue liste des joueurs
c. afficher les joueurs dans une table
2. Alternance
c.1 afficher pas de joueur lorsqu’il n'y a pas de joueurs.</td></tr>
<tr><td>T2</td><td>Réaliser les fonctionnalités du contrôleur</td><td>15min</td></tr>
<tr><td></td><td>T3</td><td>Intégration de la page liste des joueurs</td><td>20min</td><td></td></tr>
</table>

III. Architecture MVC

B. Présentation

Le modèle MVC décrit une manière d'organiser une application web en la décomposant en trois sous-parties :

- la partie Modèle ;
- la partie Vue ;
- la partie Contrôleur.

Ce modèle de conception (« design pattern ») a été imaginé à la fin des années 1970 pour le langage Smalltalk afin de bien séparer le code de l'interface graphique de la logique applicative. Il est utilisé dans de très nombreux langages : bibliothèques Swing et Model 2 (JSP) de Java, frameworks PHP, ASP.NET MVC, etc.

C. Rôle de chaque Composant

1. Composant Modèle

Ce composant de l' architecture MVC gère l'accès aux données. Il peut s'agir d'un ensemble de fonctions dans la programmation procédurale.

2. La partie Vue

Ce composant s'occupe des interactions avec l'utilisateur : présentation, saisie et validation des données.Il est formé par des pages html,css et Js.

3. La partie Contrôleur

Ce composant gère la logique métier (« business logic ») c'est-à-dire l'ensemble des services offerts par notre application . Il fait le lien entre l'utilisateur et le reste de l'application en recevant toutes les requêtes (GET ou POST) puis de produire une réponse(les vues) .

NB: Dans le Modèle MVC , on peut aussi avoir les composants tel que:

- Front Controller : Le contrôleur frontal est le composant du modèle MVC par lequel transite toutes les requêtes (GET ou POST)de notre application.Son rôle est de charger les constantes,les fonctions de validation ,les rôles et le principalement Router.

- **Router :** Le routeur est le composant du modèle MVC qui reçoit la requête transférer par le front controller puis charge le controller correspondant à la requête.

D. Structure d'un Projet MVC

- *config*
  - validator.php
  - contantes.php
  - router.php
  - role.php
- public
  - index.php
  - uploads
  - img
  - css
  - js
- templates
  - include
- data
  - db.json
- src
  - models
  - controllers

Dossier	*Description*
config	Ce dossier contient : • Les constantes du projet(*constantes.php*) • Les fonctions de validation (*validator.php*) • Le router (*router.php*) • Les fonction de gestion des autorisations (*role.php*)
public	Ce dossier est le point d'entrée de l'application . Chaque requête(*GET ou POST*)passe forcément par le fichier *index.php* qui représente le *front controller.* Étant accessible par tous, il est généralement utilisé pour mettre à disposition des fichiers de ressources, principalement les *style(CSS)*,les *scripts(JS)* et les *images.*
src	C'est le cœur du projet ! L'endroit où vous passerez le plus de temps à coder. Il regroupe tout le code PHP de votre

	application, c'est ici que vous mettrez en place toute la logique de votre application. Les dossiers qui seront obligatoires à utiliser pour le fonctionnement de l'application sont : • ***models*** • ***controllers***
controllers	Ce dossier contient tous les controllers de notre application web.Un controller est un fichier PHP qui reçoit les requêtes (GET ou POST) envoyées par un ***router*** et l'associe à des services ou actions. Lors du traitement d'une action, il peut charger des données à partir des ***modèles*** .Une fois le traitement terminé, le contrôleur génère une réponse qui peut être (HTML,JSON, XML, ...). **NB:Aucun traitement accès aux données ne doit se faire depuis un Controller (très important)**
models	Ce dossier contient les fichiers PHP qui définissent les fonctions d'accès aux données . **NB: Dans ce projet les données sont stockées dans un fichier Json.**
data	Ce dossier contient les fichiers de données.Les fichiers de données sont en Json.
templates	Ce dossier contient les vues de notre application.on distingue deux types de vues : • Les Vues de Présentation ou Layout ou template ou accueil

	• Les Vues d'action **NB: Dans le dossier template les dossiers sont organisés par controller.**
include	Ce dossier contient les parties réutilisables dans les vues.Généralement ses parties sont : • importation des fichiers fichiers css et js • le menu

V. Réalisation du Sprint 1

A. Réalisation du US2

1. Réalisation T1 : Création de la structure MVC du Projet

- ***config***
  - validator.php
  - contantes.php
  - router.php
  - role.php
  - orm.php
- **public**
  - index.php
  - uploads
  - img
  - css
  - js
- **templates**
  - **include**
    - header.htm.php
    - footer.html.php
    - menu.html.php
  - **securite**
    - connexion.html.php
  - **user**
    - accueil.html.php
    - liste.joueur.html.php
- **data**
  - db.json
- **src**
  - **models**
    - user.model.php
  - **controllers**
    - securite.controllers.php
    - user.controllers.php

2. Réalisation T2 : Définir le fichier Json

a) Format le Json

Le JavaScript Object Notation (JSON) est un format standard utilisé pour représenter des données structurées de façon semblable aux objets Javascript.
Il est habituellement utilisé pour structurer et transmettre des données sur des applications web (par exemple, envoyer des données depuis un serveur vers un client afin de les afficher sur une page web ou vice versa).

b) Définition des données

#db.json

```
{
 "users":[
        {
           "nom_complet":"Birane Baila Wane",
           "login":"joeur@gmail.com",
           "password":"joeur",
           "role":"ROLE_JOUEUR",
           "score":15
        },
        {
           "nom_complet":"Aissatou faye",
           "login":"admin@gmail.com",
           "password":"admin",
           "role":"ROLE_ADMIN",
           "score":0
        }

 ]
}
```

3. Réalisation T3: Définition des constantes du projet

a) Super Global $_SERVER

$_SERVER est un tableau contenant des informations comme les en-têtes, dossiers et chemins du script.

$_SERVER est manipulé à travers les constantes suivantes .

- **SCRIPT_FILENAME:** constante qui retourne le chemin absolu vers le fichier contenant le script en cours d'exécution.
- **SCRIPT_NAME:** constante qui retourne le nom du script courant.On utilise pour générer des liens dynamiques d'appel de page ou de chargement image.
- **REQUEST_METHOD:** constante qui retourne la méthode de requête utilisée pour accéder à la page ; par exemple 'GET', 'POST'.

b) Quelques constantes et fonctions utiles

- **DIRECTORY_SEPARATOR :** constante qui retourne le séparateur de répertoire utilisé dans ce système(/ ou \).

#constantes.php

```php
<?php
/**
* Chemin sur dossier racine du projet
*/

define("ROOT",str_replace("public".DIRECTORY_SEPARATOR."index.php","",$_SERVER['SCR
IPT_FILENAME']));
/**
 * Chemin sur dossier src qui contient les controllers et les modeles
*/
define("PATH_SRC",ROOT."src".DIRECTORY_SEPARATOR);
/**
 * Chemin sur dossier templates du projet
*/
 define("PATH_VIEWS",ROOT."templates".DIRECTORY_SEPARATOR);
/**
 * Chemin sur data qui contient le fichier Json db.json
*/
 define("PATH_DB",ROOT."data/db.json");



/**
* Chemin sur le dossier public , pour inclusion des images,css et js
```

```
*/

 define("WEBROOT",str_replace("index.php","",$_SERVER['SCRIPT_NAME']));
/**
* Chemin sur l'action des formulaires
*/
define("PATH_POST","http://localhost:8001");
```

4. Réalisation T4: Création du front controller

a) Quelques constantes et fonctions utiles

- **__FILE__**: constante qui renvoie le chemin complet et le nom du fichier PHP en cours d'exécution.
- **dir_name()**: fonction qui Renvoie le chemin du dossier parent

#index.php

```
<?php
//Demarrage de la sesion
if(session_status()==PHP_SESSION_NONE){
 session_start();
}
//inclusion des constantes
require_once dirname(dirname(__FILE__))."/config/constantes.php";
//inclusion du Validator
require_once dirname(dirname(__FILE__))."/config/validator.php";
//
require_once dirname(dirname(__FILE__))."/config/orm.php";
//inclusion des roles
require_once dirname(dirname(__FILE__))."/config/role.php";


//Chargement du router
require_once dirname(dirname(__FILE__))."/config/router.php";
```

5. Réalisation T5: Création du router

a) Super Global $_REQUEST

#router.php

```php
<?php
   if(isset($_REQUEST['controller']) ){
       switch ($_REQUEST['controller']) {
               case "securite" :
                   require_once(PATH_SRC."controllers/securite.controllers.php");
               break;
               case "user" :
                   require_once(PATH_SRC."controllers/user.controllers.php");
               break;
       }

   }else{
          require_once(PATH_SRC."controllers/securite.controllers.php");
   }
```

6. Réalisation T6: Création des controllers

#securite.controllers.php

```php
<?php

/**
* Traitement des Requetes  POST
*/

 if($_SERVER['REQUEST_METHOD']=="POST"){
     if(isset($_POST['action'])){
     }
 }

/**
* Traitement des Requetes GET
*/
 if($_SERVER['REQUEST_METHOD']=="GET"){
     if(isset($_GET['action'])){

     }
 }
```

7. Réalisation T7: Création de la couche accès aux données(models)

a) Quelques constantes et fonctions utiles

- **file_get_contents():** fonction qui retourne tout le contenu d' un fichier sous forme d'une chaîne.
- **json_decode():** fonction qui convertit du json en tableau
- **file_put_contents():** fonction qui écrit une chaîne dans un fichier
- **json_encode():** fonction qui convertit un tableau en Json

b) Fonctions de manipulation du fichier

#orm.php

```php
///Recuperation  des donnees du fichier
 function find_data(string $key):array{
   $dataJson=file_get_contents(PATH_DB);
   $data=json_decode($dataJson,true);
   return $data[$key];
}

//Enregistrement et Mis a jour des donnees du fichier
function save_data(string $key,array $data):array{
      return [];
}
```

B. Réalisation du US2

1. Réalisation T1: Réaliser les fonctionnalités du modèle

a) Identification fonctions

- find_user_login_password()

#user.model.php

```
function find_user_login_password(string $login,string $password):array{
   $users=find_data("users");
   foreach ($users as $user) {
       if( $user['login']==$login && $user['password']==$password)
         return $user;
   }
   return [];
}
```

2. Réalisation T2: Réaliser les fonctionnalités du contrôleur

a) Quelques constantes et fonctions utiles

- filter_var() : fonction qui retourne valide une variable avec un filtre spécifique .On peut citer comme filtre :
  - FILTER_VALIDATE_EMAIL : constante qui valide un email

b) Identification fonctions de validation back

- champ_obligatoire()
- valid_email()
- valid_password()

#validator.php

```
function champ_obligatoire(string $key,string $data,array &$errors,string
$message="ce champ est obligatoire"){
  if(empty($data)){
    $errors[$key]=$message;
  }
}
function valid_email(string $key,string $data,array &$errors,string $message="email
invalid"){

 if(!filter_var($data,FILTER_VALIDATE_EMAIL)){
       $errors[$key]=$message;
   }
}
```

```
function valid_password(string $key,string $data,array &$errors,string
$message="password invalid"){
}
```

c) Identification fonctions de validation Front

d) Identification fonctions du controller

- `connexion()`
- `déconnexion()`

#securite.controllers.php

```
  function connexion(string $login,string $password):void {
        $errors=[];
           champ_obligatoire("login",$login,$errors);
           if(!isset($errors['login'])){
              valid_email("login",$login,$errors);
           }
           champ_obligatoire("password",$password,$errors);
           if(!isset($errors['login'])){
              valid_password("password",$password,$errors);
           }
              if(count($errors)==0){
                $userConnect=find_user_login_password($login,$password);
                if(count($userConnect)!=0){
                    $_SESSION[USER_KEY]=$userConnect;

header("location:".WEBROOT."?controller=user&action=accueil");
                     exit();

                 }else{
                  $errors['connexion']="Login ou Mot de passe incorrect";
                  $_SESSION['errors']=$errors;
                  header("location:".WEBROOT);
                  exit();
                 }
              }else{
                $_SESSION['errors']=$errors;
                header("location:".WEBROOT);
                 exit();
              }


        }
```

```
        function logout():void{
            $_SESSION['user_connect']=array();
           unset($_SESSION['user_connect']);
           session_destroy();
           header("location:".WEBROOT);
           exit();
        }
```

e) Définition des requêtes GET dans le controller user

#user.controller.php

```
if($_SERVER['REQUEST_METHOD']=="GET"){

     if(isset($_GET['action'])){
     }
 }
```

f) Définition des autorisations

#role.php

```
<?php
define("USER_KEY","user_connect");
define("ROLE_JOUEUR","ROLE_JOUEUR");
define("ROLE_ADMIN","ROLE_ADMIN");
function is_connect():bool{
  return isset($_SESSION[USER_KEY]);
}
function is_joueur():bool{
   return is_connect() && $_SESSION[USER_KEY]['role']==ROLE_JOUEUR;
}

function is_admin():bool{
   return is_connect() && $_SESSION[USER_KEY]['role']==ROLE_ADMIN;
}
```

3. Réalisation T3: Intégration de la page de connexion

- Lien Template

a) Définition des fichiers inclusion

- header.html.php

```
<!DOCTYPE html>
```

```
<html>
<head>
   <title>QUIZZ APP</title>
   <link rel="stylesheet" type="text/css" href="<?=WEBROOT."css/login.css"?>">
   <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">
   <script src="https://kit.fontawesome.com/a81368914c.js"></script>
   <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
```

- footer.html.php

```
<script type="text/javascript" src="<?=WEBROOT."js/login.js"?>"></script>
</body>
</html>
```

b) Intégration de la page de connexion

- Importation du css

```
   <link rel="stylesheet" type="text/css"
href="<?=WEBROOT."css/login.css"?>">
```

- Importation du js

```
<script type="text/javascript"
src="<?=WEBROOT."js/login.js"?>"></script>
```

- Ajouter des images

```
<img class="wave" src="<?=WEBROOT."img/wave.png"?>">
```

- Formulaire de connexion
  - action

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

  - champ input hidden pour le controller

```
<input type="hidden" name="controller" value="securite">
```

- champ input hidden de l'action

```
  <input type="hidden" name="action" value="connexion">
```

- **Affichage des Message Erreur**

```
<?php
  if(isset($_SESSION['errors'])){
    $errors=$_SESSION['errors'];
    unset($_SESSION['errors']);
  }

?>

<?php if(isset($errors['login'])):?>
                  <span style="color:red;margin-left:-200px;
font-size:13px"><?=$errors['login']?></span>
               <?php endif ?>
               </div>


<?php if(isset($errors['password'])):?>
                  <span style="color:red;margin-left:-200px;
font-size:13px"><?=$errors['password']?></span>
               <?php endif ?>
```

#connexion.html.php

```
<?php
  require_once(PATH_VIEWS."include/header.html.php");
  if(isset($_SESSION['errors'])){
    $errors=$_SESSION['errors'];
```

```
    unset($_SESSION['errors']);
  }

?>
   <img class="wave" src="<?=WEBROOT."img/wave.png"?>">
   <div class="container">
       <div class="img">
           <img src="<?=WEBROOT."img/login form.svg"?>">
       </div>
       <div class="login-content">
           <form action="<?=PATH_POST?>" method="post">
              <input type="hidden" name="controller" value="securite">
              <input type="hidden" name="action" value="connexion">
               <img src="img/avatar.svg">
               <h2 class="title">QUIZZ SA</h2>
               <?php if(isset($errors['connexion'])):?>
                 <span style="color:red;
font-size:13px"><?=$errors['connexion']?></span>
               <?php endif ?>
               <div class="input-div one">
                  <div class="i">
                       <i class="fas fa-user"></i>
                  </div>
                  <div class="div">
                       <h5>Username</h5>
                       <input type="text" name="login" class="input">

                  </div>

               </div>
               <div>
               <?php if(isset($errors['login'])):?>
                 <span style="color:red;margin-left:-200px;
font-size:13px"><?=$errors['login']?></span>
               <?php endif ?>
               </div>
               <div class="input-div pass">
                  <div class="i">
                       <i class="fas fa-lock"></i>
                  </div>
                  <div class="div">
                       <h5>Password</h5>
                       <input type="password" name="password" class="input">
                  </div>
               </div>
```

```
                <?php if(isset($errors['password'])):?>
                  <span style="color:red;margin-left:-200px;
font-size:13px"><?=$errors['password']?></span>
                <?php endif ?>
                <button type="submit" class="btn" name="action"
value="connexion">Connexion</button>
            </form>
        </div>
    </div>

<?php
  require_once(PATH_VIEWS."include/footer.html.php");
?>
```

c) Création du Menu

- Lien Template
- menu.html.php
- application des autorisations

#menu.html.php

```
<section id="sidebar">
        <a href="#" class="brand">
            <i class='bx bxs-smile'></i>
            <span class="text">Quizz SA</span>
        </a>
        <ul class="side-menu top">
            <li class="active">
                <a href="#">
                    <i class='bx bxs-dashboard' ></i>
                    <span class="text">Dashboard</span>
                </a>
            </li>
        <?php if(is_admin()):?>
            <li>
                <a href="#">
                    <i class='bx bxs-shopping-bag-alt' ></i>
                    <span class="text">Liste des Joueurs</span>
                </a>
            </li>
        <?php endif?>
        </ul>
        <ul class="side-menu">

            <li>
```

```
                <a href="#" class="logout">
                    <i class='bx bxs-log-out-circle' ></i>
                    <span class="text">Logout</span>
                </a>
            </li>
        </ul>
    </section>
    <!-- SIDEBAR -->
```

d) Application du lien de déconnexion

#menu.html.php

```
<li>
                <a href="<?=WEBROOT."?controller=securite&action=deconnexion"?>"
class="logout">
                    <i class='bx bxs-log-out-circle' ></i>
                    <span class="text">Logout</span>
                </a>
            </li>
```

#user.controllers.php

```
if($_SERVER['REQUEST_METHOD']=="GET"){

    if(isset($_GET['action'])){
        if(!is_connect()) header("location:".WEBROOT);
```

e) Importation du fichier css dans header.html.php

- style.css

#header.html.php

```
    <link rel="stylesheet" type="text/css" href="<?=WEBROOT."css/style.css"?>">
```

f) Importation du fichier js dans footer.html.php

- script.js

#footer.html.php

```
<script type="text/javascript" src="<?=WEBROOT."js/script.js"?>"></script>
```

g) Définition de la requête GET dans le controller user

#user.controller.php

```
if($_SERVER['REQUEST_METHOD']=="GET"){

    if(isset($_GET['action'])){
        if ($_GET['action']=="accueil") {
```

```
            $first_lien="Dashboard";
            $sub_lien="Home";
          //Chargement des Vues de Users
         require_once(PATH_VIEWS."user/accueil.html.php");
      }
    }
 }
```

h) Création de la page Accueil

- accueil.html.php

#accueil.html.php

```
<?php
  require_once(PATH_VIEWS."include/header.html.php");
  require_once(PATH_VIEWS."include/menu.html.php");

?>

<!-- CONTENT -->
<section id="content">
     <!-- NAVBAR -->
     <nav>
        <i class='bx bx-menu' ></i>


        <a href="#" class="profile">
           <img src="img/people.png">
        </a>
     </nav>
     <!-- NAVBAR -->

     <!-- MAIN -->
     <main>
        <div class="head-title">
           <div class="left">
              <h1>Menu</h1>
              <ul class="breadcrumb">
                 <li>
                    <a href="#"><?=$first_lien?></a>
                 </li>
                 <li><i class='bx bx-chevron-right' ></i></li>
                 <li>
```

```
                    <a class="active" href="#"><?=$sub_lien?></a>
                </li>
            </ul>
        </div>

    </div>


    <?php
        echo $content_for_template ;
      ?>

  </main>
  <!-- MAIN -->
 </section>
 <!-- CONTENT -->
<?php
 require_once(PATH_VIEWS."include/footer.html.php");
?>
```

C. Réalisation du US3

1. Réalisation T1: Réaliser les fonctionnalités du modèle

a) Identification fonctions du modèle

- find_users()

#user.model.php

```
function find_users(string $role):array{
   $users=find_data("users");
   $result=[];
   foreach ($users as $user) {
       if( $user['role']==$role)
          $result[]= $user;
   }
   return  $result;
}
```

2. Réalisation T2: Réaliser les fonctionnalités du contrôleur

a) Identification fonctions du controller

- lister_joueur()

#user.controllers.php

```
function lister_joueur():array {
            return find_users("ROLE_JOUEUR");
      }
```

b) Définition de la requête GET dans le controller user

3. Réalisation T3: Intégration de la page liste des joueurs

a) Quelques constantes et fonctions utiles

- ob_start(): fonction qui stocke temporairement un flux(html,chaîne ..) dans la mémoire tampon.
  Tant qu'elle est enclenchée, aucune donnée, hormis les en-têtes, n'est envoyée au navigateur, mais temporairement mise en tampon.
- ob_get_clean(): fonction qui récupère le contenu du tampon puis l'affecte à une variable;

b) Chargement des vues dans le controller user

#user.controller.php

```
if($_SERVER['REQUEST_METHOD']=="GET"){


     if(isset($_GET['action'])){
     if ($_GET['action']=="accueil") {
       $first_lien="Dashboard";
       $sub_lien="Home";
          //Chargement des Vues de Users
         if(isset($_GET['view'])){
               switch ($variable) {
                 case 'liste.joueur':
                    ob_start();


require_once(PATH_VIEWS."user/liste.joueur.html.php");
                     $content_for_template= ob_get_clean();
                  break;
```

```
                }
            }else{
              ob_start();
                $data=lister_joueur();
               require_once(PATH_VIEWS."user/liste.joueur.html.php");
               $content_for_template= ob_get_clean();
            }
        require_once(PATH_VIEWS."user/accueil.html.php");
    }
  }

}
```

c) Génération de la table HTML

#liste.joueur.html.php

```
<div class="table-data">
            <div class="order">
                <div class="head">
                    <h3>Liste des Joueurs</h3>
                    <i class='bx bx-search' ></i>
                    <i class='bx bx-filter' ></i>
                </div>
                <table>
                    <thead>
                        <tr>
                            <th>Nom et Prenom</th>
                            <th>Score</th>
                        </tr>
                    </thead>
                    <tbody>
                    <?php foreach($data as $value):?>
                        <tr>
                            <td>
                                <img src="">
                                <p><?=$value['nom_complet']?></p>
                            </td>
                            <td><?=$value['score']?></td>

                        </tr>
                    <?php endforeach?>
```

```
                    </tbody>
                </table>
            </div>

        </div>
```

d) Ajout du lien liste des joueurs dans le menu

#user.controller.php