TP MVC User Authentification

Table des matières

1. OBJECTIFS DE LA SEQUENCE : ARCHITECTURE MVC POUR SE PREPARER A SYMFONY

1.1. CAHIER DES CHARGES

Gérer la connexion des utilisateurs à une application quelconque

Sécuriser l'authentification

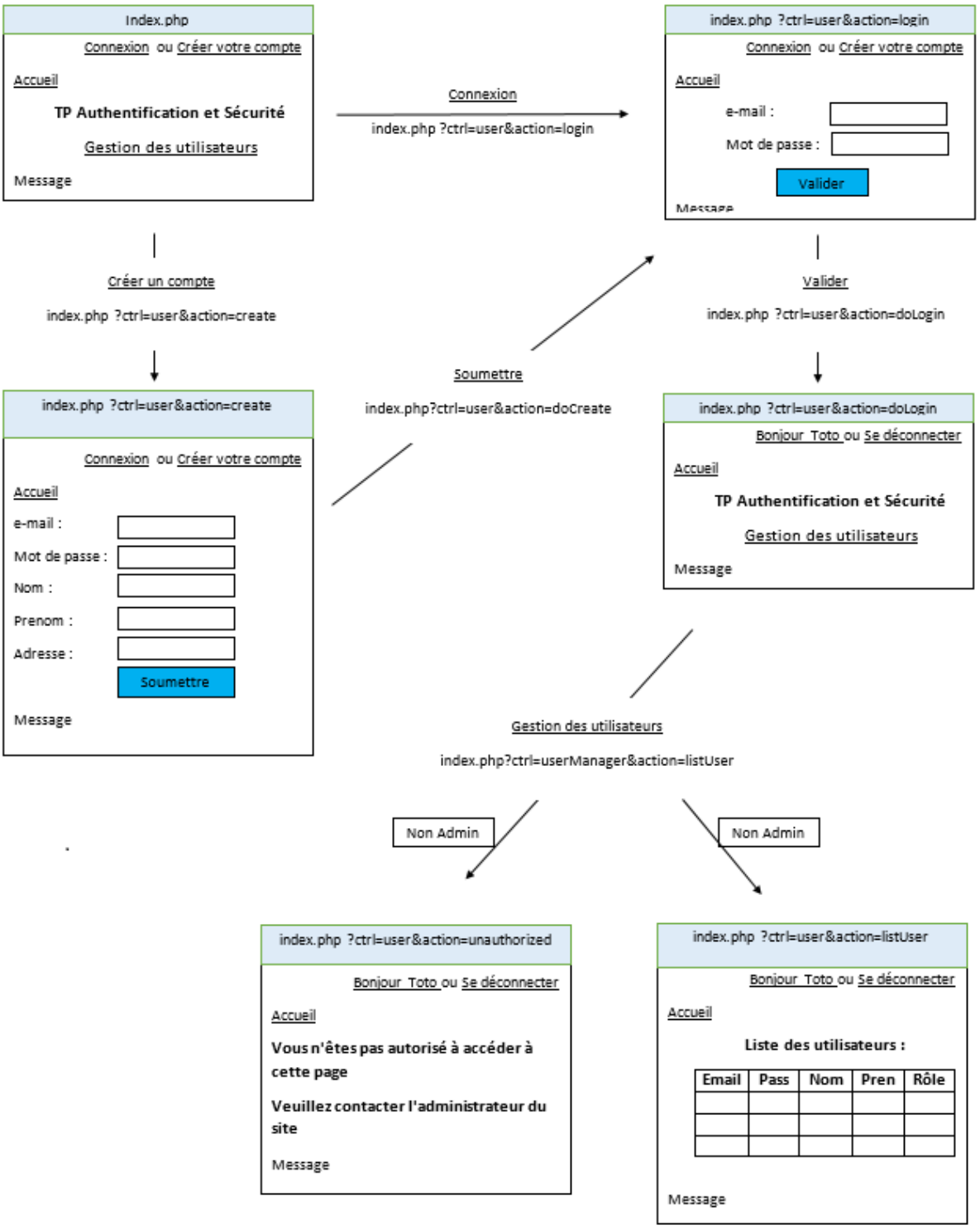
Il est demandé de rendre une application parcourant fonctionnellement :

- L'accueil avec son message connexion ou salutation si authentifié
- Le formulaire d'authentification
- Le formulaire de création d'un utilisateur
- Les pages de gestion des utilisateurs sont facultatives

1.2. UNE DEMO EN LIGNE :

http://lp-projet-web.ovh/userAuthentification/

1.3. ARBORESCENCE DE L'APPLICATION



2. RAPPEL DES CLASSES CONSTRUIRES AU TP PRECEDENT

2.1. LISTE DES FONCTIONS ASSUREES PAR LA CLASSE USER

1. Disposer dans ses attributs, des données stockées en base de données [déjà réalisé dans le TP précédent] :
   (1) id,
   (2) email,
   (3) password,
   (4) firstName,
   (5) lastName,
   (6) address,
   (7) postalCode,
   (8) city,
   (9) country,
   (10)rôle (admin ou ghest)
2. s'enregistrer en vérifiant la composition de l'email, en stockant le password sous forme cryptée de la meilleure façon
3. se loguer, se déloguer (le user connecté sera stocké en variable de session S_SESSION)

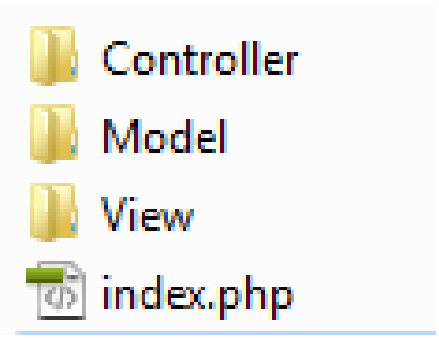
2.2. LISTE DES FONCTIONS ASSUREES PAR LA CLASSE USERMANAGER

1. Sélectionner un user suivant son id
2. Insérer un nouvel user en BdD
3. Modifier les données d'un utilisateur enregistré
4. Supprimer un user
5. Sélectionner tous les users
6. Vérifier le login / password lors de la connexion de l'utilisateur
7. ...

2.3. LISTE DES FONCTIONS ASSUREES PAR LA CLASSE CONNECTION

1. Disposer en attribut des identifiants, password de connexion pour se connecter à la base de données.
2. Créer à l'aide de la librairie PDO, une connexion à la base de données

3. MVC : STRUCTURE DES DOSSIERS A REPRODUIRE :



Nous allons détailler le contenu attendu de chaque dossier.

3.1. LES CLASSES DU DOSSIER MODEL

3.1.1. UNE CLASSE TECHNIQUE : "CONNECTION.PHP"

La classe "Connection" (dans le dossier Model) permettant d'établir la connexion avec la base de données.

```php
<?php
class Connection {

    private $host;
    private $dbname;
    private $username;
    private $password;
    private $db;

    public function __construct() {

        $this->host = 'localhost';
        $this->dbname = 'dream_seller';
        $this->username = 'root';
        $this->password = 'mysql';
        try
        {
            $this->db = new PDO('mysql:host=' . $this->host . ';dbname='
. $this->dbname . ';charset=utf8', $this->username, $this->password);
        }
        catch(PDOException $e)
        {
            echo $e->getMessage();
        }
    }

    public function getDb() {
        return $this->db;
    }
}
```

3.2. LES CLASSES USER (USER.PHP) ET USERMANAGER (USERMANAGER.PHP)

➔ Reprendre les classes User et UserManager produites au TP précédent pour la placer dans le dossier model

3.3. LE CONTRÔLEUR PRINCIPAL : INDEX.PHP ET LE USERCONTROLLER

3.3.1. LE CONTRÔLEUR PRINCIPAL INDEX.PHP

La navigation dans l'application s'effectue en adressant à la page index.php deux variables $_GET : ctrl et action.

```
index.php?ctrl=user&action=login
```

L'appel du contrôleur et de son action est produit alors par :

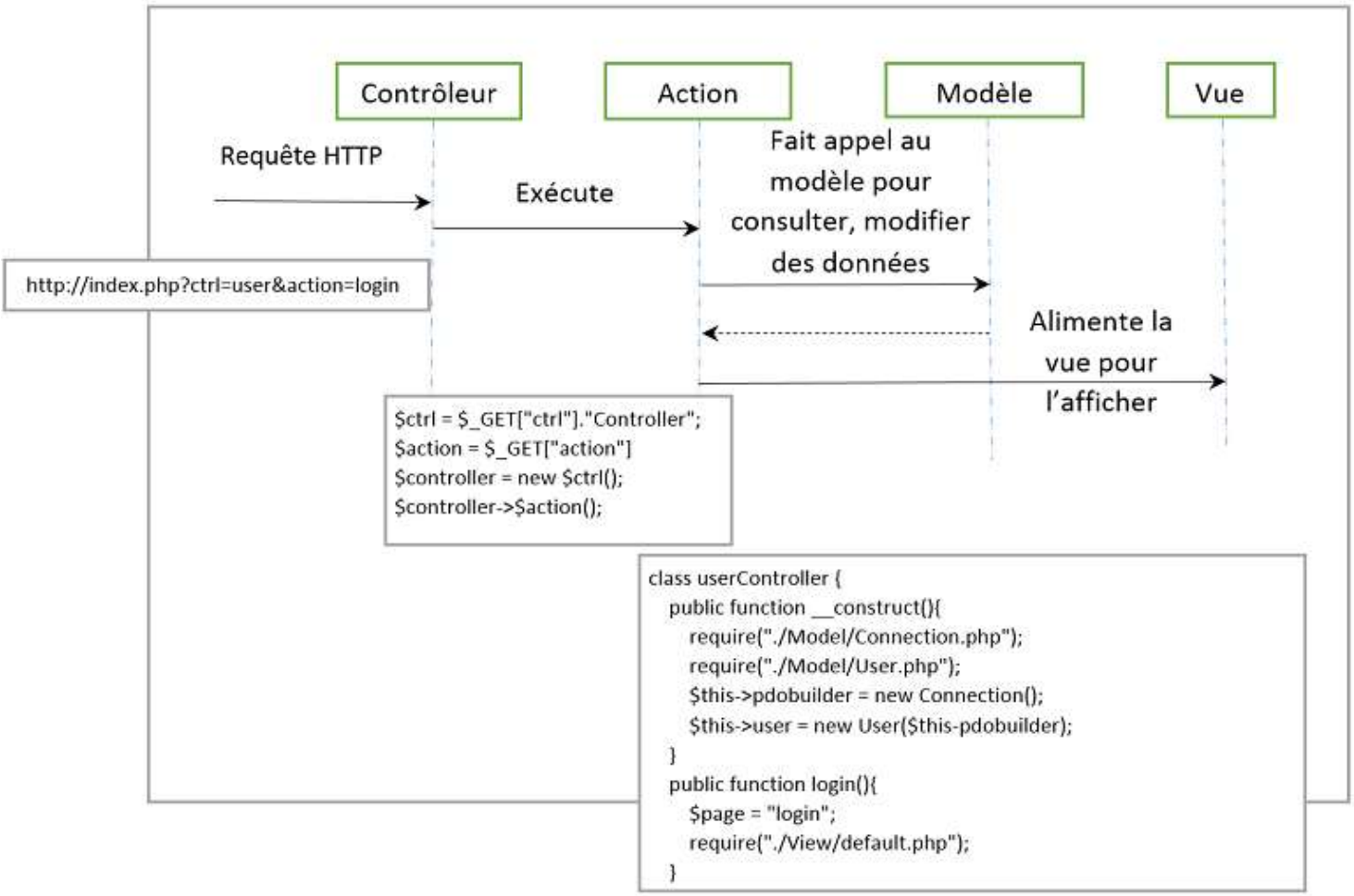
```php
<?php
session_start();
require_once('./Model/Connection.class.php');
$pdoBuilder = new Connection();
$db = $pdoBuilder->getDb();

if (
    ( isset($_GET['ctrl']) && !empty($_GET['ctrl']) ) &&
    ( isset($_GET['action']) && !empty($_GET['action']) )
) {

    $ctrl = $_GET['ctrl'];
    $action = $_GET['action'];
}
else {

    $ctrl = 'category';
    $action = 'display';
}

require_once('./controller/' . $ctrl  . 'Controller.class.php');

$ctrl = $ctrl . 'Controller';
$controller = new $ctrl($db);
$controller->$action();

?>
```

3.3.2. LE CONTRÔLEUR USERCONTROLLER

Comment construire la classe userController et son action login ?

```php
<?php
class userController {

    private $userManager;
    private $user;

    public function __construct($db1) {

        require('./Model/User.class.php');
        require_once('./Model/UserManager.class.php');
        $this->userManager = new UserManager($db1);

        $this->db = $db1 ;
    }

    public function login() {

        $page = 'login';

        require('./View/main.php');
    }


    public function doLogin() {

        $this->user = new User();

        // Cette action teste l'existence d'un utilisateur de email $_POST['email'] et de password $_POST['password']
        // Le user extrait par le UserManager est renvoyé dans $result
        // A vous d'écrire les 3 lignes correspondantes


        $result = //_____  ;

       if ( $result ) :
            $info = "Connexion reussie";
            $_SESSION['user'] = $result;
            $page = 'home';

        else :

            $info = "Identifiants incorrects.";
        endif;

          require('./View/main.php');

    }
```

3.3.3. CIRCULATION DANS L'ARCHITECTURE MVC POUR LA REQUÊTE : INDEX.PHP?CTRL=USER&ACTION=LOGIN



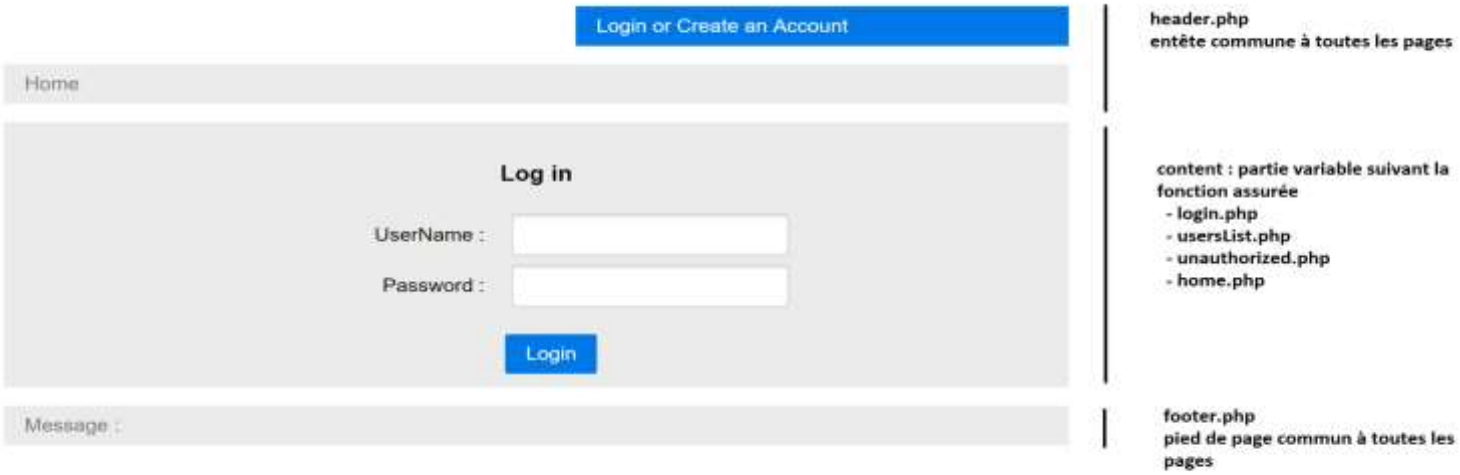
➔ A vous de placer ces dernières lignes du userController.php dans le dossier Controller.

3.3.4. L'ACTION DOCREATE DU USERCONTROLLER

```php
public
function doCreate()
{
    if (
        isset($_POST['email']) &&
        isset($_POST['password']) &&
        isset($_POST['lastName']) &&
        isset($_POST['firstName']) &&
        isset($_POST['address']) &&
        isset($_POST['postalCode']) &&
        isset($_POST['city'])
    ) {
        $alreadyExist = $this->userManager->findByEmail($_POST['email']);

        if (!$alreadyExist) {
            $newUser = new User($_POST);
            $this->userManager->create($newUser);
            $page = 'login';
        } else {
            $error = "ERROR : This email (" . $_POST['email'] . ") is used by another user";
            $page = 'create';
        }
    }

    require('./View/default.php');
}
```

4. LES VUES

Les pages peuvent organisées de la manière suivante :



On trouvera donc dans le dossier View les fichiers suivants :

1. header.php
2. footer.php
3. login.php
4. createAccount.php
5. home.php
6. unauthorized.php
7. usersList.php
8. et une page default.php qui assembler suivant la demande du contrôleur les extraits précédents de la manière suivante

```php
<?php
include_once "header.php" ?>

    <section id="main-section">
        <?php
        if(isset($page)) {
            if($page == 'home')
                require("./View/home.php");
            else
                require("./View/".$page.".php");
        }
        ?>
    </section>

<?php include_once "footer.php" ?>
```

Les vues pourront être copiées de la démo en ligne :

http://lp-projet-web.ovh/userAuthentification/

5. ETAT CONNECTE : VARIABLE DE SESSION $_SESSION

Etat connecté, un poste de $_SESSION stocke l'enregistrement du user issu de la base de données :

```
$_SESSION["user"] = $req[0]
```

La déconnexion se traduit par

```
unset($_SESSION["user"]);
```

6. ENCRYPTAGE DU MOT DE PASSE

Le mot de passe devra être crypté en base de données.

On pourra utiliser la fonction sha1().

On notera que ce hashage n'est pas considéré comme suffisant par les experts du piratage. On pourra envisager la fonction crypt() si vous avez le temps.