

Projet Resto



Nom de la fonction : controleurPrincipal

Objectif :

Cette fonction sert de contrôleur principal pour router les actions des utilisateurs vers les fichiers PHP correspondants dans une application web. Elle associe chaque action à son fichier PHP respectif.

Paramètres :

- \$action : Une chaîne de caractères représentant l'action demandée par l'utilisateur.

Valeur de retour :

- Une chaîne de caractères contenant le nom de fichier du script PHP correspondant à l'action demandée.

Fonctionnalité :

1. Initialise un tableau associatif nommé \$lesActions, qui associe chaque action à son fichier PHP correspondant.
2. Vérifie si l'action demandée existe dans le tableau \$lesActions en utilisant array_key_exists().
3. Si l'action existe, récupère le nom de fichier associé à l'action depuis le tableau \$lesActions.
4. Si l'action n'existe pas, utilise par défaut l'action "default", qui représente généralement la page d'accueil ou la page par défaut de l'application.
5. Retourne le nom de fichier du script PHP correspondant à l'action demandée.

Exemple d'utilisation :

- Pour router l'action d'un utilisateur vers le fichier PHP approprié :

```
$action = "detail"; // Exemple d'action
```

```
$filename = controleurPrincipal($action);
```

```
// $filename contient maintenant le nom de fichier du script PHP  
"detailResto.php".
```

```
<?php  
function controleurPrincipal($action){  
    $lesActions = array();  
    $lesActions["default"] = "listeRestos.php";  
    $lesActions["liste"] = "listeRestos.php";  
    $lesActions["detail"] = "detailResto.php";  
    $lesActions["recherche"] = "rechercheResto.php";  
    $lesActions["connexion"] = "connexion.php";  
    $lesActions["deconnexion"] = "deconnexion.php";  
    $lesActions["profil"] = "monProfil.php";  
  
    if (array_key_exists ( $action , $lesActions )){  
        return $lesActions[$action];  
    }  
    else{  
        return $lesActions["default"];  
    }  
}
```

?>

Model et vue

Afficher une liste de restaurants répertoriés.

1. Inclusion des fichiers de modèle :

- Les fichiers de modèle sont inclus pour accéder aux fonctions de base de données nécessaires.
- Ces fichiers incluent probablement des fonctions pour récupérer des informations sur les restaurants, les photos et les types de cuisine depuis la base de données.

2. Récupération des données :

- Aucune donnée GET, POST ou SESSION n'est explicitement récupérée dans ce script. Les données nécessaires sont probablement récupérées dans les fonctions de modèle incluses.

3. Appel des fonctions de récupération de données :

- La fonction `getLesRestos()` est appelée pour récupérer la liste des restaurants.

4. Traitement des données :

- Aucun traitement explicite des données n'est effectué dans ce script. Cela peut être effectué dans les fonctions de modèle appelées précédemment.

5. Affichage des données :

- Le titre de la page est défini comme "Liste des restaurants répertoriés".
- Les fichiers d'en-tête HTML (`entete.html.php`), de pied de page HTML (`pied.html.php`) et de vue pour la liste des restaurants (`vueListeRestos.php`) sont inclus.
- Chaque restaurant est affiché sous forme de carte (élément "div" avec la classe "card").
- Pour chaque restaurant, la première photo et les types de cuisine associés sont affichés.
- Les restaurants sont liés à la page de détail correspondante avec un lien dynamique.

```
<?php
include_once "$racine/modele/bd.resto.php";
include_once "$racine/modele/bd.photo.php";
include_once "$racine/modele/bd.typecuisine.php";

// recuperation des donnees GET, POST, et SESSION
;

// appel des fonctions permettant de recuperer les donnees utiles a
l'affichage
$lesRestos = getLesRestos();

// traitement si necessaire des donnees recuperees
;

// appel du script de vue qui permet de gerer l'affichage des donnees
$titre = "Liste des restaurants répertoriés";
include "$racine/vue/entete.html.php";
include "$racine/vue/vueListeRestos.php";
include "$racine/vue/pied.html.php";
?>

<h1>Liste des restaurants</h1>

<?php foreach ($lesRestos as $resto) : ?>

    <?php
        $lesPhotos = getPhotosByIdR($resto['idR']);
        $lesTypesCuisine = getLesTypesCuisineByIdR($resto['idR']);
```

```

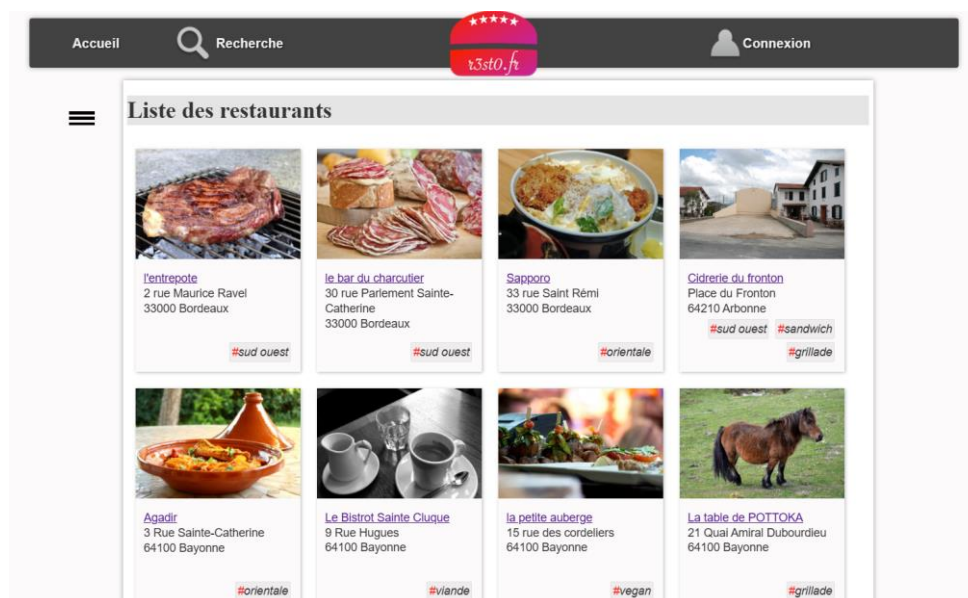
?>

<div class="card">
  <div class="photoCard">

    <?php if (count($lesPhotos) > 0) : ?>
      "
alt="photo du restaurant" />
    <?php endif; ?>
  </div>
  <div class="descrCard">
    <a href='./?action=detail&idR=<?= $resto['idR'] ?>'><?=
$resto['nomR'] ?></a><br />
    <?= $resto["numAdrR"] ?>
    <?= $resto["voieAdrR"] ?><br />
    <?= $resto["cpR"] ?>
    <?= $resto["villeR"] ?>
  </div>
  <div class="tagCard">
    <ul id="tagFood">
      <?php foreach ($lesTypesCuisine as $typeCuisine) : ?>
        <li class="tag"><span class="tag">#</span><?=
$typeCuisine["libelleTC"] ?></li>
      <?php endforeach; ?>
    </ul>
  </div>
</div>

<?php endforeach; ?>

```



Afficher les détails d'un restaurant, y compris sa description, ses photos, ses horaires, et les critiques associées.

1. Inclusion des fichiers de modèle :

- Les fichiers de modèle sont inclus pour accéder aux fonctions de base de données nécessaires.
- Ces fichiers incluent probablement des fonctions pour récupérer des informations sur les restaurants, les photos, les types de cuisine, les critiques, ainsi que pour l'authentification des utilisateurs.

2. Initialisation du menu burger :

- Un menu burger est créé sous forme d'un tableau contenant des liens vers différentes sections de la page détaillée du restaurant.

3. Récupération des données GET :

- L'identifiant du restaurant (idR) est récupéré à partir des données GET pour déterminer quel restaurant afficher.

4. Appel des fonctions de récupération de données :

- Les informations sur le restaurant (unResto), les types de cuisine associés (lesTypesCuisine), les photos (lesPhotos), la note moyenne (noteMoy), et les critiques (critiques) sont récupérées à partir de la base de données.

5. Vérification de l'action 'aimer' :

- Si l'action 'aimer' est définie dans les paramètres de l'URL et que l'utilisateur est connecté, le script vérifie si l'utilisateur aime déjà le restaurant. Selon le cas, il ajoute ou supprime l'aimer en appelant les fonctions appropriées.

6. Affichage des données :

- Les détails du restaurant, y compris son nom, son type de cuisine, sa description, son adresse, ses photos, ses horaires, et les critiques sont affichés de manière structurée.

7. Interaction utilisateur :

- Les utilisateurs peuvent aimer ou ne pas aimer un restaurant en cliquant sur une icône appropriée.
- Les utilisateurs peuvent également noter un restaurant en cliquant sur les étoiles et ajouter leurs commentaires.

```
<?php
if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    $racine = "..";
}

include_once "$racine/modele/bd.resto.php";
include_once "$racine/modele/bd.photo.php";
include_once "$racine/modele/bd.typecuisine.php";
include_once "$racine/modele/bd.critiquer.php";
include_once "$racine/modele/authentication.php";
include_once "$racine/modele/bd.aimer.php";

// creation du menu burger
$menuBurger = array();
$menuBurger[] = array("url"=>"#top", "label"=>"Le restaurant");
$menuBurger[] = array("url"=>"#adresse", "label"=>"Adresse");
$menuBurger[] = array("url"=>"#photos", "label"=>"Photos");
$menuBurger[] = array("url"=>"#horaires", "label"=>"Horaires");
$menuBurger[] = array("url"=>"#crit", "label"=>"Critiques");

// recuperation des donnees GET, POST, et SESSION
$idR = $_GET["idR"];

// appel des fonctions permettant de recuperer les donnees utiles a
l'affichage
$unResto = getLeRestoByIdR($idR);
$lesTypesCuisine = getLesTypesCuisinebyIdR($idR);
$lesPhotos = getPhotosByIdR($idR);
$noteMoy = getNoteMoyenneByIdR($idR);
if ($noteMoy !== null) {
    $noteMoy = round($noteMoy, 0);
} else {
    // Gérer le cas où la note moyenne est null
}
```

```

    // Par exemple, vous pouvez définir une valeur par défaut ou afficher
    un message d'avertissement
}
$critiques = getCritiquerByIdR($idR);

$mailU = getMailULoggedOn();

$aimer = null;
if ($mailU) {
    $aimer = getAimerById($mailU, $idR);
}

// Vérification si l'action 'aimer' a été définie dans les paramètres de
l'URL et si l'utilisateur est connecté
if (isset($_GET['action']) && $_GET['action'] == 'aimer' && isset($mailU))
{
    // Vérifier si l'utilisateur aime déjà le restaurant
    $aimer = getAimerById($mailU, $unResto['idR']);
    if ($aimer) {
        // Si l'utilisateur aime déjà le restaurant, supprimer l'aimer
        delAimer($mailU, $unResto['idR']);
    } else {
        // Si l'utilisateur n'aime pas encore le restaurant, ajouter
l'aimer
        addAimer($mailU, $unResto['idR']);
    }
}

// traitement si necessaire des donnees recuperees
;

// appel du script de vue qui permet de gerer l'affichage des donnees
$titre = "Détail d'un restaurant";
include "$racine/vue/entete.html.php";
include "$racine/vue/vueDetailResto.php";
include "$racine/vue/pied.html.php";
?>

<h1><?= $unResto['nomR']; ?>
<?php if (isset($mailU)): ?>
    <a href="./?action=aimer&idR=<?= $unResto['idR'] ?>">
        <?php
            $aimer = getAimerById($mailU, $unResto['idR']);
            if ($aimer) {
                echo "<img src='images/aime.png' alt='J'aime' />";
            } else {
                echo "<img src='images/aimepas.png' alt='Je n'aime pas' />";
            }
        ?>
    </a>
<?php endif; ?>
</h1>
<h2>type de cuisine

    <span id="note">
        <?php for ($i = 1; $i <= 5; $i++) { ?>
            <a class="aimer" href="./?action=noter&note=<?= $i ?>&idR=<?=
$unResto['idR']; ?>" >
                <?php if ($i <= $noteMoy) { ?>
                    

```

```

        <?php } else {
            ?>
            
        <?php } ?>
    </a>
    <?php } ?>
</span>
</h2>
<section>
    Cuisine <br />
    <ul id="tagFood">
        <?php for ($j = 0; $j < count($lesTypesCuisine); $j++) { ?>
            <li class="tag"><span class="tag">#</span><?>
$lesTypesCuisine[$j]["libelleTC"] ?></li>
            <?php } ?>
        </ul>
    </section>
<h2>Description</h2>
<p id="principal">

    <?= $unResto['descR']; ?>
</p>
<h2 id="adresse">
    Adresse
</h2>
<p>
    <?= $unResto['numAdrR']; ?>
    <?= $unResto['voieAdrR']; ?><br />
    <?= $unResto['cpR']; ?>
    <?= $unResto['villeR']; ?>

</p>

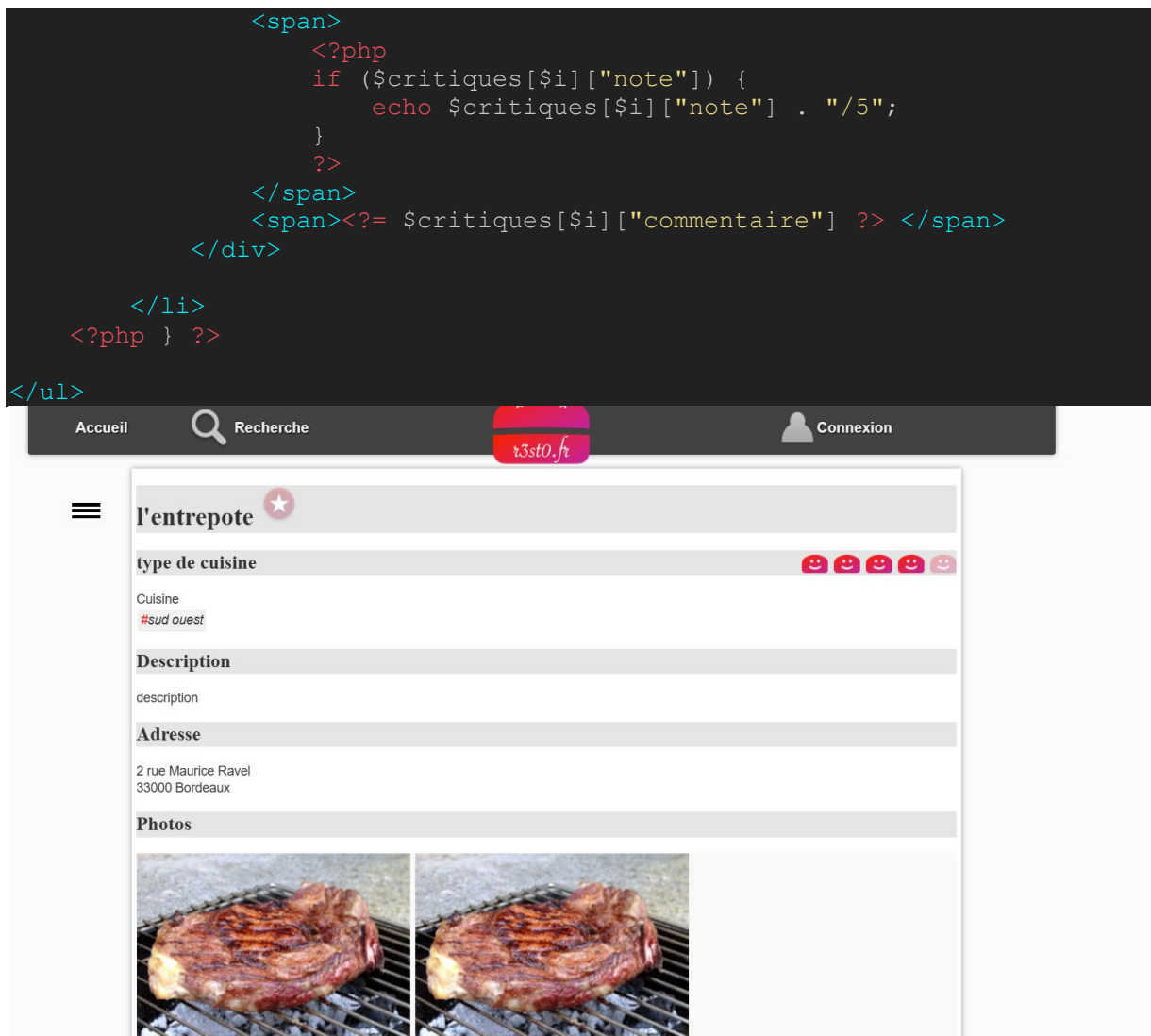
<h2 id="photos">
    Photos
</h2>
<ul id="galerie">
    <?php for ($i = 0; $i < count($lesPhotos); $i++) { ?>
        <li> " alt="" /></li>
        <?php } ?>
    </ul>

<h2 id="horaires">
    Horaires
</h2>
<?= $unResto['horairesR']; ?>

<h2 id="crit">Critiques</h2>

<ul id="critiques">
    <?php for ($i = 0; $i < count($critiques); $i++) { ?>
        <li>
            <span>
                <?= $critiques[$i]["mailU"] ?>
            </span>
            <div>

```



La recherche de restaurants selon différents critères tels que le nom, l'adresse ou le type de cuisine.

1. Inclusion des fichiers de modèle :

- Les fichiers de modèle sont inclus pour accéder aux fonctions de base de données nécessaires.
- Ces fichiers incluent probablement des fonctions pour récupérer des informations sur les restaurants, les types de cuisine, etc.

2. Initialisation du menu burger :

- Un menu burger est créé sous forme d'un tableau contenant des liens vers différentes options de recherche : par nom, par adresse ou par type de cuisine.

3. Récupération du critère de recherche :

- Le critère de recherche par défaut est le nom du restaurant.
- Si le critère est spécifié dans les données GET, il est utilisé pour la recherche.

4. Récupération des données POST :

- Les données POST sont récupérées selon le critère de recherche choisi : nom, adresse ou types de cuisine.

5. Appel des fonctions de récupération de données :

- Selon le critère de recherche sélectionné, les fonctions de modèle appropriées sont appelées pour récupérer les restaurants correspondants depuis la base de données.

6. Traitement des données :

- Aucun traitement supplémentaire des données n'est effectué dans ce script.

7. Affichage des résultats de recherche :

- Les résultats de la recherche sont affichés en utilisant le script de vue correspondant.
- Si le formulaire de recherche est soumis (données POST non vides), les résultats sont affichés dans la vue "vueListeRestos.php".

8. Affichage du formulaire de recherche :

- Le formulaire de recherche est affiché avec les champs appropriés en fonction du critère de recherche sélectionné.
- Pour la recherche par type de cuisine, des cases à cocher sont générées en récupérant les types de cuisine depuis la base de données.

```
<?php

include_once "$racine/modele/bd.resto.php";
include_once "$racine/modele/bd.typecuisine.php";
include_once "$racine/modele/bd.photo.php";
include_once "$racine/modele/bd.critiquer.php";

// creation du menu burger
$menuBurger = array();
$menuBurger[] =
Array("url"=>"./?action=recherche&critere=nom", "label"=>"Recherche par
nom");
$menuBurger[] =
Array("url"=>"./?action=recherche&critere=adresse", "label"=>"Recherche par
adresse");
$menuBurger[] =
Array("url"=>"./?action=recherche&critere=type", "label"=>"Recherche par
type de cuisine");

// critere de recherche par default
$critere = "nom";
if (isset($_GET["critere"])) {
    $critere = $_GET["critere"];
}

// recuperation des donnees GET, POST, et SESSION
if (isset($_GET["critere"])) {
    $critere = $_GET["critere"];
}
$nomR="";
if (isset($_POST["nomR"])) {
    $nomR = $_POST["nomR"];
}
$voieAdrR="";
if (isset($_POST["voieAdrR"])) {
    $voieAdrR = $_POST["voieAdrR"];
}
$cpr="";
```

```

if (isset($_POST["cpR"])) {
    $cpR = $_POST["cpR"];
}
$villeR="";
if (isset($_POST["villeR"])) {
    $villeR = $_POST["villeR"];
}
$tabIdTC = array();
if(isset($_POST["tabIdTC"])){
    $tabIdTC = $_POST["tabIdTC"];
}

// appel des fonctions permettant de recuperer les donnees utiles a
l'affichage

switch($critere){
    case 'nom':
        // recherche par nom
        $lesRestos = getLesRestosByNomR($nomR);
        break;
    case 'adresse':
        // recherche par adresse
        $lesRestos = getLesRestosByAdresse($voieAdrR, $cpR, $villeR);
        break;
    case 'type':
        // recherche par type de cuisine
        if(!empty($tabIdTC)){
            $lesRestos = getLesRestosByTC($tabIdTC);
        }
        else{
            $lesRestos = getLesRestos();
        }
        break;
}

// traitement si necessaire des donnees recuperees
;

// appel du script de vue qui permet de gerer l'affichage des donnees
$titre = "Recherche d'un restaurant";
include "$racine/vue/entete.html.php";
include "$racine/vue/vueRechercheResto.php";
if (!empty($_POST)) {
    // affichage des resultats de la recherche
    include "$racine/vue/vueListeRestos.php";
}
include "$racine/vue/pied.html.php";

?>

<h1>Recherche d'un restaurant</h1>
<form action="./?action=recherche&critere=<?= $critere ?>" method="POST">

    <?php
    switch ($critere) {

```

```
</form>
```

Recherche d'un restaurant

Recherche par nom :

Rechercher


Liste des restaurants

Recherche d'un restaurant


Recherche par adresse :

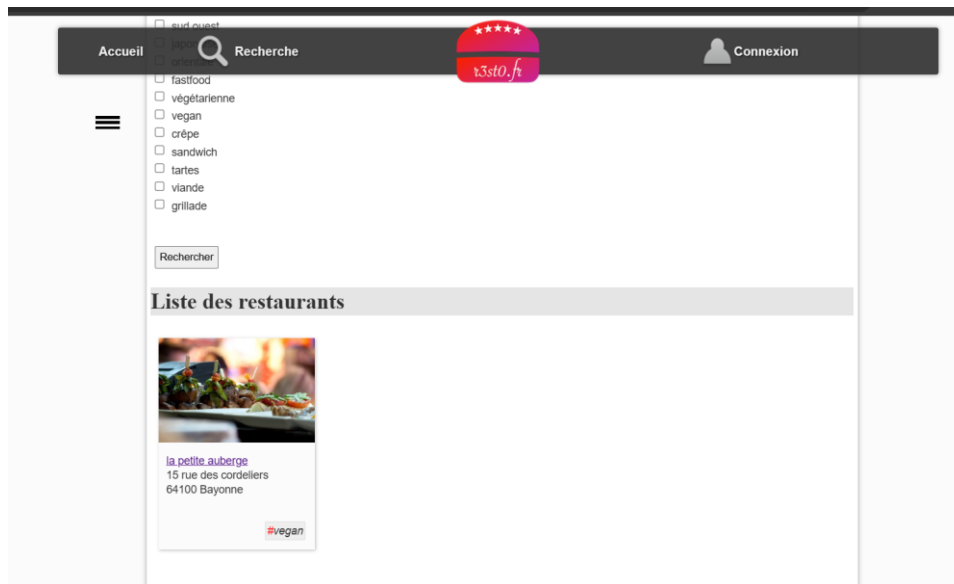
Rechercher

Liste des restaurants



[l'entrepote](#)
2 rue Maurice Ravel
33000 Bordeaux

 sud ouest



l'authentification des utilisateurs et affiche le formulaire de connexion.

1. **Inclusion des fichiers de modèle :**
 - Le fichier d'authentification est inclus pour accéder aux fonctions de connexion.
2. **Création du menu burger :**
 - Le menu burger est créé avec des liens vers les pages de connexion et d'inscription.
3. **Récupération des données POST :**
 - Les données POST sont récupérées pour le courriel de l'utilisateur et son mot de passe.
4. **Traitement des données :**
 - Les données de connexion sont envoyées à la fonction de connexion login().
5. **Vérification de l'état de connexion :**
 - Si l'utilisateur est connecté, la page "monProfil.php" est incluse pour afficher le profil de l'utilisateur.
 - Sinon, le formulaire de connexion est affiché.
6. **Affichage du formulaire de connexion :**
 - Le formulaire de connexion est affiché avec les champs pour le courriel et le mot de passe.
 - En cas d'erreur d'authentification, un message d'erreur est affiché.
7. **Informations supplémentaires :**
 - Des informations de test sont fournies pour permettre la connexion avec un utilisateur de test.

```

• <?php
  include_once "$racine/modele/authentification.php";

  // création du menu burger
  $menuBurger = array();
  $menuBurger[] =
  Array("url"=>"./?action=connexion","label"=>"Connexion");
  $menuBurger[] =
  Array("url"=>"./?action=inscription","label"=>"Inscription");

  // recuperation des donnees GET, POST, et SESSION
  if(isset($_POST["mailU"]) && isset($_POST["mdpU"])){
    $mailU=$_POST["mailU"];
  }

```

```

        $mdpU=$_POST["mdpU"];
    }
    else
    {
        $mailU="";
        $mdpU="";
    }
    // appel des fonctions permettant de recuperer les donnees utiles a
    l'affichage

    // traitement si necessaire des donnees recuperees
    login($mailU,$mdpU);

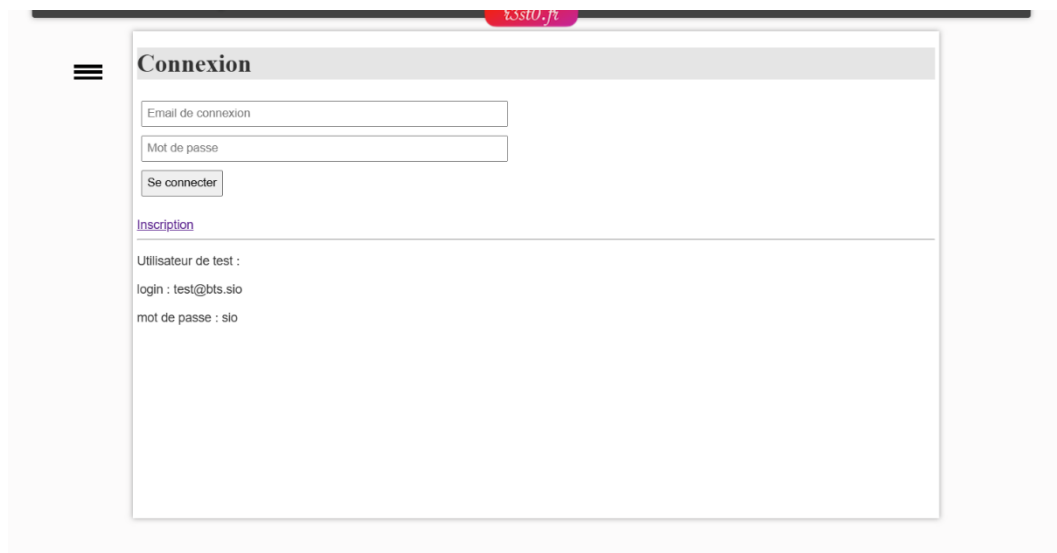
    if (isLoggedIn()){
        include "$racine/controleur/monProfil.php";
    }
    else{
        $titre = "authentification";
        include "$racine/vue/entete.html.php";
        include "$racine/vue/vueAuthentification.php";
        include "$racine/vue/pied.html.php";
    }
    ?>

```

```

<h1>Connexion</h1>
    <?php if (isset($erreur)): ?>
        <p style="color: red;"><?= $erreur ?></p>
    <?php endif; ?>
    <form action="./?action=connexion" method="POST">
        <input type="text" name="mailU" placeholder="Email de connexion"
required /><br />
        <input type="password" name="mdpU" placeholder="Mot de passe"
required /><br />
        <input type="submit" value="Se connecter" />
    </form>
    <br />
    <a href="./?action=inscription">Inscription</a>
    <hr />
    <p>Utilisateur de test :</p>
    <p>login : test@bts.sio</p>
    <p>mot de passe : sio</p>

```



The screenshot shows a web browser window with a URL bar displaying 'vssio.fr'. The page has a header with a hamburger menu icon on the left and the title 'Connexion' in a grey bar. The main content area contains a login form with two text input fields labeled 'Email de connexion' and 'Mot de passe', and a 'Se connecter' button. Below the form is a link labeled 'Inscription'. At the bottom of the page, there is a section for test user credentials: 'Utilisateur de test :', 'login : test@bts.sio', and 'mot de passe : sio'.

l'affichage du profil utilisateur.

1. **Inclusion des fichiers de modèle :**
 - Les fichiers d'authentification et les modèles de base de données nécessaires sont inclus.
2. **Création du menu burger :**
 - Le menu burger est créé avec des liens vers les actions de consultation et de modification du profil.
3. **Récupération des données utilisateur :**
 - Si l'utilisateur est connecté, ses informations sont récupérées à partir de la session.
4. **Récupération des données associées au profil :**
 - Les restaurants aimés par l'utilisateur et les types de cuisine préférés sont récupérés à partir de la base de données.
5. **Affichage des données du profil :**
 - Les informations de base de l'utilisateur, tels que son adresse électronique et son pseudo, sont affichées.
 - Les restaurants aimés sont affichés sous forme de cartes avec leurs détails.
 - Les types de cuisine aimés sont affichés sous forme de liste.
6. **Déconnexion :**
 - Un lien est fourni pour permettre à l'utilisateur de se déconnecter.

```
7. <?php
if ( $_SERVER["SCRIPT_FILENAME"] == __FILE__ ) {
    $racine="..";
}
include_once "$racine/modele/authentification.php";
include_once "$racine/modele/bd.utilisateur.php";
include_once "$racine/modele/bd.typecuisine.php";
include_once "$racine/modele/bd.resto.php";

// creation du menu burger
$menuBurger = array();
$menuBurger[] = Array("url"=>"./?action=profil","label"=>"Consulter
mon profil");
$menuBurger[] = Array("url"=>"./?action=updProfil","label"=>"Modifier
mon profil");

// recuperation des donnees GET, POST, et SESSION

// appel des fonctions permettant de recuperer les donnees utiles a
l'affichage
if (isLoggedIn()) {
    $mailU = getMailULoggedIn();
    $util = getLeUtilisateurByMailU($mailU);

    $mesRestosAimes = getRestosAimesByMailU($mailU);

    $mesTypeCuisineAimes = getTypesCuisinePreferesByMailU($mailU);
    // traitement si necessaire des donnees recuperees

    // appel du script de vue qui permet de gerer l'affichage des
    donnees
    $titre = "Mon profil";
    include "$racine/vue/entete.html.php";
```

```

        include "$racine/vue/vueMonProfil.php";
        include "$racine/vue/pied.html.php";
    }
    else{
        $titre = "Mon profil";
        include "$racine/vue/entete.html.php";
        include "$racine/vue/pied.html.php";
    }

    ?>

```

```
<h1>Mon profil</h1>
```

```
Mon adresse électronique : <?=$util["mailU"] ?> <br />
```

```
Mon pseudo : <?=$util["pseudoU"] ?> <br />
```

```
<hr>
```

```
les restaurants que j'aime : <br />
```

```
<?php for($i=0;$i<count($mesRestosAimes);$i++){ ?>
```

```
<div class="card">
```

```
    <a href="./?action=detail&idR=<?=$mesRestosAimes[$i]["idR"] ?>">
```

```
    <p>
```

```
        <?=$mesRestosAimes[$i]["nomR"] ?><br />
```

```
        <?=$mesRestosAimes[$i]["numAdrR"] ?>
```

```
        <?=$mesRestosAimes[$i]["voieAdrR"] ?><br />
```

```
        <?=$mesRestosAimes[$i]["cpR"] ?>
```

```
        <?=$mesRestosAimes[$i]["villeR"] ?>
```

```
    </p>
```

```
    </a>
```

```
</div>
```

```
<?php } ?>
```

```
<hr>
```

```
les types de cuisine que j'aime :
```

```
<ul id="tagFood">
```

```
    <?php for($i=0;$i<count($mesTypeCuisineAimes);$i++){ ?>
```

```
        <li class="tag"><span class="tag">#</span><?=$
```

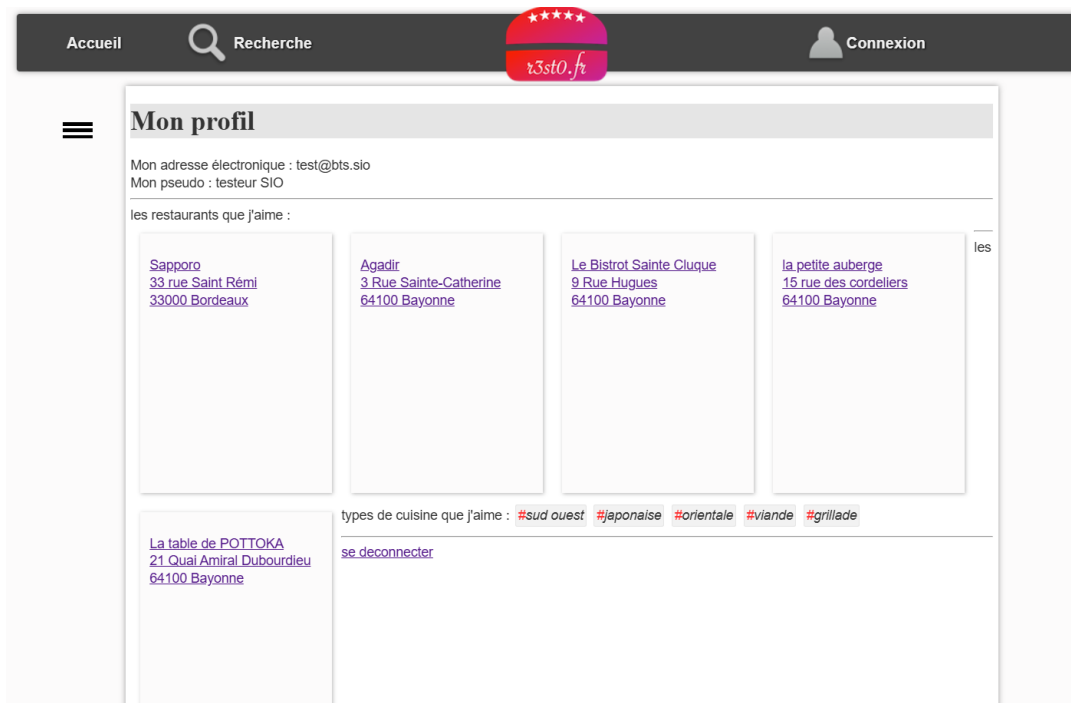
```
$mesTypeCuisineAimes[$i]["libelleTC"] ?></li>
```

```
    <?php } ?>
```

```
</ul>
```

```
<hr>
```

```
<a href="./?action=deconnexion">se deconnecter</a>
```



Contrôleur

bd.resto

Des fonctions permettant d'effectuer des opérations sur la table "resto" dans une base de données. Voici un aperçu des fonctions disponibles :

1. **getLesRestos()** : Cette fonction récupère tous les restaurants de la table "resto" et les renvoie sous forme de tableau associatif.
2. **getLeRestoByIdR(\$idR)** : Cette fonction récupère les détails d'un restaurant spécifique en fonction de son identifiant (idR) et les renvoie sous forme de tableau associatif.
3. **getLesRestosByNomR(\$nomR)** : Cette fonction recherche les restaurants dont le nom (nomR) contient une chaîne spécifique, passée en paramètre. Elle renvoie les résultats sous forme de tableau associatif.
4. **getLesRestosByAdresse(\$voieAdrR, \$cpR, \$villeR)** : Cette fonction recherche les restaurants en fonction de l'adresse, en filtrant par la voie (voieAdrR), le code postal (cpR) et la ville (villeR). Les résultats sont renvoyés sous forme de tableau associatif.
5. **getLesRestosByTC(\$tabIdTC)** : Cette fonction recherche les restaurants en fonction des types de cuisine spécifiés dans le tableau d'identifiants de type de cuisine (\$tabIdTC). Elle utilise une jointure avec la table "proposer" pour effectuer la recherche et renvoie les résultats sous forme de tableau associatif.
6. **getRestosAimesByMailU(\$mailU)** : Cette fonction récupère les restaurants aimés par un utilisateur spécifique, identifié par son adresse électronique (mailU). Elle utilise une jointure avec la table "aimer" pour effectuer la recherche et renvoie les résultats sous forme de tableau associatif.

```
<?php
include_once "bd.pdo.php";
```



```

function getLesRestos() {
    try {
        $conn = connexionPDO();
        $sql = "SELECT * FROM resto";
        $stmt = $conn->query($sql);
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

function getLeRestoByIdR($idR) {
    try {
        $conn = connexionPDO();
        $sql = "SELECT * FROM resto WHERE idR = :idR";
        $stmt = $conn->prepare($sql);
        $stmt->bindParam(':idR', $idR, PDO::PARAM_INT);
        $stmt->execute();
        return $stmt->fetch(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

function getLesRestosByNomR($nomR) {
    try {
        $conn = connexionPDO();
        $sql = "SELECT * FROM resto WHERE nomR LIKE :nomR";
        $stmt = $conn->prepare($sql);
        $stmt->bindValue(':nomR', "%$nomR%", PDO::PARAM_STR);
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

function getLesRestosByAdresse($voieAdrR, $cpR, $villeR) {
    try {
        $conn = connexionPDO();
        $sql = "SELECT * FROM resto WHERE voieAdrR LIKE :voieAdrR AND cpR
LIKE :cpR AND villeR LIKE :villeR";
        $stmt = $conn->prepare($sql);
        $stmt->bindValue(':voieAdrR', "%$voieAdrR%", PDO::PARAM_STR);
        $stmt->bindValue(':cpR', "%$cpR%", PDO::PARAM_STR);
        $stmt->bindValue(':villeR', "%$villeR%", PDO::PARAM_STR);
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

function getLesRestosByTC($tabIdTC) {
    try {
        $conn = connexionPDO();
        $sql = "SELECT r.* FROM resto r

```

```

        INNER JOIN proposer p ON r.idR = p.idR
        WHERE p.idTC IN (".implode(",", $tabIdTC).");
    $stmt = $conn->query($sql);
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
} catch (PDOException $e) {
    print "Erreur !: " . $e->getMessage() . "<br/>";
    die();
}

}

function getRestosAimesByMailU($mailU) {
    try {
        $conn = connexionPDO();
        $sql = "SELECT r.* FROM resto r
            INNER JOIN aimer a ON r.idR = a.idR
            WHERE a.mailU = :mailU";
        $stmt = $conn->prepare($sql);
        $stmt->bindParam(':mailU', $mailU, PDO::PARAM_STR);
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}
?>

```

Bd.typedecuisine

des fonctions pour interagir avec la table "typecuisine" dans une base de données. Voici un résumé des fonctions disponibles :

1. **getLesTypesCuisine()** : Cette fonction récupère tous les types de cuisine de la table "typecuisine" et les renvoie sous forme de tableau associatif.
2. **getLesTypesCuisinebyIdR(\$idR)** : Cette fonction récupère les types de cuisine associés à un restaurant spécifique, identifié par son ID (idR). Elle utilise une jointure avec la table "proposer" pour obtenir ces informations et renvoie les résultats sous forme de tableau associatif.
3. **getTypesCuisinePreferesByMailU(\$mailU)** : Cette fonction récupère les types de cuisine préférés d'un utilisateur spécifique, identifié par son adresse électronique (mailU). Elle utilise une jointure avec la table "preferer" pour obtenir ces informations et renvoie les résultats sous forme de tableau associatif.

```

4. <?php
include_once "bd.pdo.php";

function getLesTypesCuisine()
{
    try {
        $conn = connexionPDO();
        if ($conn) {
            $sql = "SELECT * FROM typecuisine";
            $stmt = $conn->query($sql);
            $typesCuisine = $stmt->fetchAll(PDO::FETCH_ASSOC);
            return $typesCuisine;
        } else {
            return false;
        }
    }
}

```

```

    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

function getLesTypesCuisinebyIdR($idR) {
    try {
        $conn = connexionPDO();
        if ($conn) {
            $sql = "SELECT tc.libelleTC FROM typecuisine tc
                    INNER JOIN proposer p ON tc.idTC = p.idTC
                    WHERE p.idR = :idR";
            $stmt = $conn->prepare($sql);
            $stmt->bindParam(':idR', $idR, PDO::PARAM_INT);
            $stmt->execute();
            $typesCuisine = $stmt->fetchAll(PDO::FETCH_ASSOC);
            return $typesCuisine;
        } else {
            return false; // Retourner false en cas d'échec de la
connexion
        }
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

function getTypesCuisinePreferesByMailU($mailU) {
    try {
        $conn = connexionPDO();
        if ($conn) {
            $sql = "SELECT tc.libelleTC FROM typecuisine tc
                    INNER JOIN preferer p ON tc.idTC = p.idTC
                    WHERE p.mailU = :mailU";
            $stmt = $conn->prepare($sql);
            $stmt->bindParam(':mailU', $mailU, PDO::PARAM_STR);
            $stmt->execute();
            $typesCuisine = $stmt->fetchAll(PDO::FETCH_ASSOC);
            return $typesCuisine;
        } else {
            return false; // Retourner false en cas d'échec de la
connexion
        }
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

?>

```

Bd.photos

une fonction pour récupérer les photos associées à un restaurant spécifique, identifié par son ID (idR). Voici un aperçu de la fonction :

1. **getPhotosByIdR(\$idR)** : Cette fonction récupère les photos liées à un restaurant spécifique à partir de la table "photo" dans la base de données. Elle prend l'ID du

restaurant en paramètre et exécute une requête SQL pour sélectionner toutes les lignes de la table "photo" où l'ID du restaurant correspond à celui fourni. Les résultats sont renvoyés sous forme de tableau associatif contenant toutes les informations sur les photos trouvées pour ce restaurant.

```
2. <?php
include_once "bd.pdo.php";

function getPhotosByIdR($idR) {
    $resultat = array();

    try {
        $cnx = connexionPDO();
        $req = $cnx->prepare("select * from photo where idR=:idR");
        $req->bindValue(':idR', $idR, PDO::PARAM_INT);

        $req->execute();

        $ligne = $req->fetch(PDO::FETCH_ASSOC);
        while ($ligne) {
            $resultat[] = $ligne;
            $ligne = $req->fetch(PDO::FETCH_ASSOC);
        }
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage();
        die();
    }
    return $resultat;
}

?>
```

Bd.critiquer

gérer les critiques (commentaires et notes) des restaurants dans une base de données. Voici un récapitulatif des fonctions et le code consolidé :

1. **getCritiquerByIdR(\$idR) :**
 - Cette fonction récupère toutes les critiques d'un restaurant spécifique identifié par son identifiant (`idR`).
 - Elle exécute une requête SQL pour sélectionner toutes les lignes de la table `critiquer` où `idR` correspond à l'identifiant du restaurant, et retourne les résultats sous forme de tableau associatif.
2. **delCritiquer(\$idR, \$mailU) :**
 - Cette fonction supprime une critique d'un utilisateur pour un restaurant donné.
 - Elle utilise une requête SQL préparée pour supprimer une ligne de la table `critiquer` où `idR` et `mailU` correspondent à l'identifiant du restaurant et l'email de l'utilisateur respectivement.
3. **addCritiquer(\$idR, \$mailU, \$note, \$commentaire) :**
 - Cette fonction ajoute une nouvelle critique d'un utilisateur pour un restaurant donné.

- Elle utilise une requête SQL préparée pour insérer une nouvelle ligne dans la table `critiquer` avec les informations fournies (`idR`, `mailU`, `note`, `commentaire`).
4. **getNoteMoyenneByIdR(\$idR) :**
- Cette fonction calcule et retourne la note moyenne des critiques pour un restaurant spécifique.
 - Elle utilise une requête SQL préparée pour calculer la moyenne des notes dans la table `critiquer` où `idR` correspond à l'identifiant du restaurant, et retourne la note moyenne.

```
5. <?php
include_once "bd.pdo.php";

function getCritiquerByIdR($idR) {
    $resultat = array();

    try {
        $cnx = connexionPDO();
        $req = $cnx->prepare("select * from critiquer where
idR=:idR");
        $req->bindValue(':idR', $idR, PDO::PARAM_INT);

        $req->execute();

        $ligne = $req->fetch(PDO::FETCH_ASSOC);
        while ($ligne) {
            $resultat[] = $ligne;
            $ligne = $req->fetch(PDO::FETCH_ASSOC);
        }
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage();
        die();
    }
    return $resultat;
}

// Fonction pour supprimer la critique d'un utilisateur pour un
restaurant donné
function delCritiquer($idR, $mailU){
    try {
        $conn = connexionPDO();
        if ($conn) {
            $sql = "DELETE FROM critiquer WHERE idR = :idR AND mailU
= :mailU";
            $stmt = $conn->prepare($sql);
            $stmt->bindParam(':idR', $idR, PDO::PARAM_INT);
            $stmt->bindParam(':mailU', $mailU, PDO::PARAM_STR);
            $stmt->execute();
            return true;
        } else {
            return false;
        }
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

// Fonction pour ajouter la critique d'un utilisateur pour un
restaurant donné
```

```

function addCritiquer($idR, $mailU, $note, $commentaire){
    try {
        $conn = connexionPDO();
        if ($conn) {
            $sql = "INSERT INTO critiquer (idR, mailU, note,
commentaire) VALUES (:idR, :mailU, :note, :commentaire)";
            $stmt = $conn->prepare($sql);
            $stmt->bindParam(':idR', $idR, PDO::PARAM_INT);
            $stmt->bindParam(':mailU', $mailU, PDO::PARAM_STR);
            $stmt->bindParam(':note', $note, PDO::PARAM_INT);
            $stmt->bindParam(':commentaire', $commentaire,
PDO::PARAM_STR);
            $stmt->execute();
            return true;
        } else {
            return false;
        }
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

// Fonction pour récupérer la critique d'un utilisateur pour un
restaurant donné
function getNoteMoyenneByIdR($idR) {
    try {
        $conn = connexionPDO();
        $sql = "SELECT AVG(note) as noteMoyenne FROM critiquer WHERE
idR = :idR";
        $stmt = $conn->prepare($sql);
        $stmt->bindParam(':idR', $idR, PDO::PARAM_INT);
        $stmt->execute();
        $noteMoyenne = $stmt->fetch(PDO::FETCH_ASSOC);
        return $noteMoyenne['noteMoyenne'];
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}
?>

```

Bd.aimer

Gérer les actions d'aimer (like) des restaurants pour des utilisateurs spécifiques. Voici le code pour ces fonctions consolidé :

1. **getAimerById(\$mailU, \$idR) :**

- Cette fonction vérifie si un utilisateur spécifique (mailU) a aimé un restaurant donné (idR).
- Elle utilise une requête SQL préparée pour sélectionner la ligne correspondante dans la table `aimer` et retourne le résultat sous forme de tableau associatif.

2. **addAimer(\$mailU, \$idR) :**

- Cette fonction ajoute un enregistrement indiquant qu'un utilisateur a aimé un restaurant.
- Elle utilise une requête SQL préparée pour insérer une nouvelle ligne dans la table `aimer` avec les informations fournies (mailU, idR).

3. **delAimer(\$mailU, \$idR) :**

- Cette fonction supprime l'enregistrement indiquant qu'un utilisateur a aimé un restaurant.
- Elle utilise une requête SQL préparée pour supprimer la ligne correspondante dans la table `aimer` où `mailU` et `idR` correspondent aux informations fournies.

```
4. <?php

function getAimerById($mailU, $idR){
    try {
        $conn = connexionPDO();
        if ($conn) {
            $sql = "SELECT * FROM aimer WHERE mailU = :mailU AND idR
= :idR";
            $stmt = $conn->prepare($sql);
            $stmt->bindParam(':mailU', $mailU, PDO::PARAM_STR);
            $stmt->bindParam(':idR', $idR, PDO::PARAM_INT);
            $stmt->execute();
            $aimer = $stmt->fetch(PDO::FETCH_ASSOC);
            return $aimer;
        } else {
            return false;
        }
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

function addAimer($mailU, $idR){
    try {
        $conn = connexionPDO();
        if ($conn) {
            $sql = "INSERT INTO aimer (mailU, idR) VALUES (:mailU,
:idR)";
            $stmt = $conn->prepare($sql);
            $stmt->bindParam(':mailU', $mailU, PDO::PARAM_STR);
            $stmt->bindParam(':idR', $idR, PDO::PARAM_INT);
            $stmt->execute();
            return true;
        } else {
            return false;
        }
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br/>";
        die();
    }
}

function delAimer($mailU, $idR){
    try {
        $conn = connexionPDO();
        if ($conn) {
            $sql = "DELETE FROM aimer WHERE mailU = :mailU AND idR =
:idR";
            $stmt = $conn->prepare($sql);
            $stmt->bindParam(':mailU', $mailU, PDO::PARAM_STR);
            $stmt->bindParam(':idR', $idR, PDO::PARAM_INT);
            $stmt->execute();
            return true;
        } else {
```

```

        return false;
    }
} catch (PDOException $e) {
    print "Erreur !: " . $e->getMessage() . "<br/>";
    die();
}
}

?>

```

Bd.utilisateur

- **etUtilisateurs()** :

- Cette fonction récupère tous les utilisateurs de la table `utilisateur`.
- Elle utilise une requête SQL pour sélectionner toutes les lignes de la table `utilisateur` et retourne les résultats sous forme de tableau associatif.

- **getLeUtilisateurByMailU(\$mailU)** :

- Cette fonction récupère un utilisateur spécifique en fonction de son adresse e-mail (`mailU`).
- Elle utilise une requête SQL préparée pour sélectionner la ligne correspondant à `mailU` dans la table `utilisateur` et retourne le résultat sous forme de tableau associatif.

- **addUtilisateur(\$mailU, \$mdpU, \$pseudoU)** :

- Cette fonction ajoute un nouvel utilisateur dans la table `utilisateur`.
- Elle utilise une requête SQL préparée pour insérer une nouvelle ligne avec les informations fournies (`mailU`, `mdpU`, `pseudoU`).
- Le mot de passe (`mdpU`) est crypté avant d'être inséré dans la base de données.

```

• <?php
include_once "bd.pdo.php";

function getUtilisateurs() {

    try {
        $cnx = connexionPDO();
        $req = $cnx->prepare("select * from utilisateur");
        $req->execute();

        $ligne = $req->fetch(PDO::FETCH_ASSOC);
        while ($ligne) {
            $resultat[] = $ligne;
            $ligne = $req->fetch(PDO::FETCH_ASSOC);
        }
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage();
        die();
    }
    return $resultat;
}

function getLeUtilisateurByMailU($mailU){

```



```

try {
    $conn = connexionPDO();
    $sql = "SELECT * FROM utilisateur WHERE mailU = :mailU";
    $stmt = $conn->prepare($sql);
    $stmt->bindParam(':mailU', $mailU, PDO::PARAM_STR);
    $stmt->execute();
    return $stmt->fetch(PDO::FETCH_ASSOC);
} catch (PDOException $e) {
    print "Erreur !: " . $e->getMessage() . "<br/>";
    die();
}

}

function addUtilisateur($mailU, $mdpU, $pseudoU) {
    try {
        $cnx = connexionPDO();

        $mdpUCrypt = crypt($mdpU, "sel");
        $req = $cnx->prepare("insert into utilisateur (mailU, mdpU,
pseudoU) values (:mailU, :mdpU, :pseudoU)");
        $req->bindValue(':mailU', $mailU, PDO::PARAM_STR);
        $req->bindValue(':mdpU', $mdpUCrypt, PDO::PARAM_STR);
        $req->bindValue(':pseudoU', $pseudoU, PDO::PARAM_STR);

        $resultat = $req->execute();
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage();
        die();
    }
    return $resultat;
}

?>

```

Authentification

les opérations de connexion, de déconnexion et de vérification de session pour les utilisateurs. Voici une explication détaillée de chaque fonction et quelques suggestions pour améliorer la sécurité et la robustesse du code.

Explications des Fonctions

1. login(\$mailU, \$mdpU)

- Cette fonction authentifie un utilisateur en vérifiant son adresse e-mail et son mot de passe.
- Si la combinaison e-mail/mot de passe est correcte, elle initialise une session et stocke les informations de l'utilisateur.
- Utilise `password_verify` pour vérifier le mot de passe haché.

2. logout()

- Cette fonction déconnecte un utilisateur en détruisant la session.

3. getMailULoggedIn()

- Cette fonction renvoie l'adresse e-mail de l'utilisateur connecté.
- Si aucun utilisateur n'est connecté, elle renvoie une chaîne vide.

4. isLoggedIn()

- Cette fonction vérifie si un utilisateur est actuellement connecté.
- Elle compare les informations de session avec celles de la base de données.

```

5. <?php
include_once "bd.utilisateur.php";

function login($mailU, $mdpU) {
    if (!isset($_SESSION)) {
        session_start();
    }

    $util = getLeUtilisateurByMailU($mailU);

    if ($util) { // Check if user exists
        $mdpBD = $util["mdpU"];

        // Using password_verify for checking password
        if (password_verify($mdpU, $mdpBD)) {
            // Password is correct, set session variables
            $_SESSION["mailU"] = $mailU;
            $_SESSION["mdpU"] = $mdpBD;
            return true;
        }
    }

    return false;
}

function logout() {
    if (!isset($_SESSION)) {
        session_start();
    }
    unset($_SESSION["mailU"]);
    unset($_SESSION["mdpU"]);
}

function getMailULoggedIn() {
    if (isLoggedIn()) {
        return $_SESSION["mailU"];
    }
    return "";
}

function isLoggedIn() {
    if (!isset($_SESSION)) {
        session_start();
    }

    if (isset($_SESSION["mailU"])) {
        $util = getLeUtilisateurByMailU($_SESSION["mailU"]);
        if ($util && $util["mailU"] == $_SESSION["mailU"] &&
            $util["mdpU"] == $_SESSION["mdpU"]) {
            return true;
        }
    }

    return false;
}
?>

```