Observation du modèle d'authentification

à l'aide des annexes 1 et 2

L'annexe 1 est la section de test du fichier authentification.inc.php. Cette section montre l'endroit où sont exécutées et testées les fonctions définies dans ce fichier.

L'annexe 2 correspond à l'affichage obtenu lors de l'exécution de l'annexe 1.

1.1. La fonction `isLoggedOn()` retourne un booléen, soit true, soit false selon que l'utilisateur est connecté ou pas sur le site.

Lors du 1er appel à la fonction `isLoggedOn()` l'utilisateur est il connecté ?

Non, la fonction login n'a pas encore été utilisée, donc il ne l'est pas.

1.2. Lors du 2eme appel à la fonction `isLoggedOn()` l'utilisateur est il connecté ?

Oui

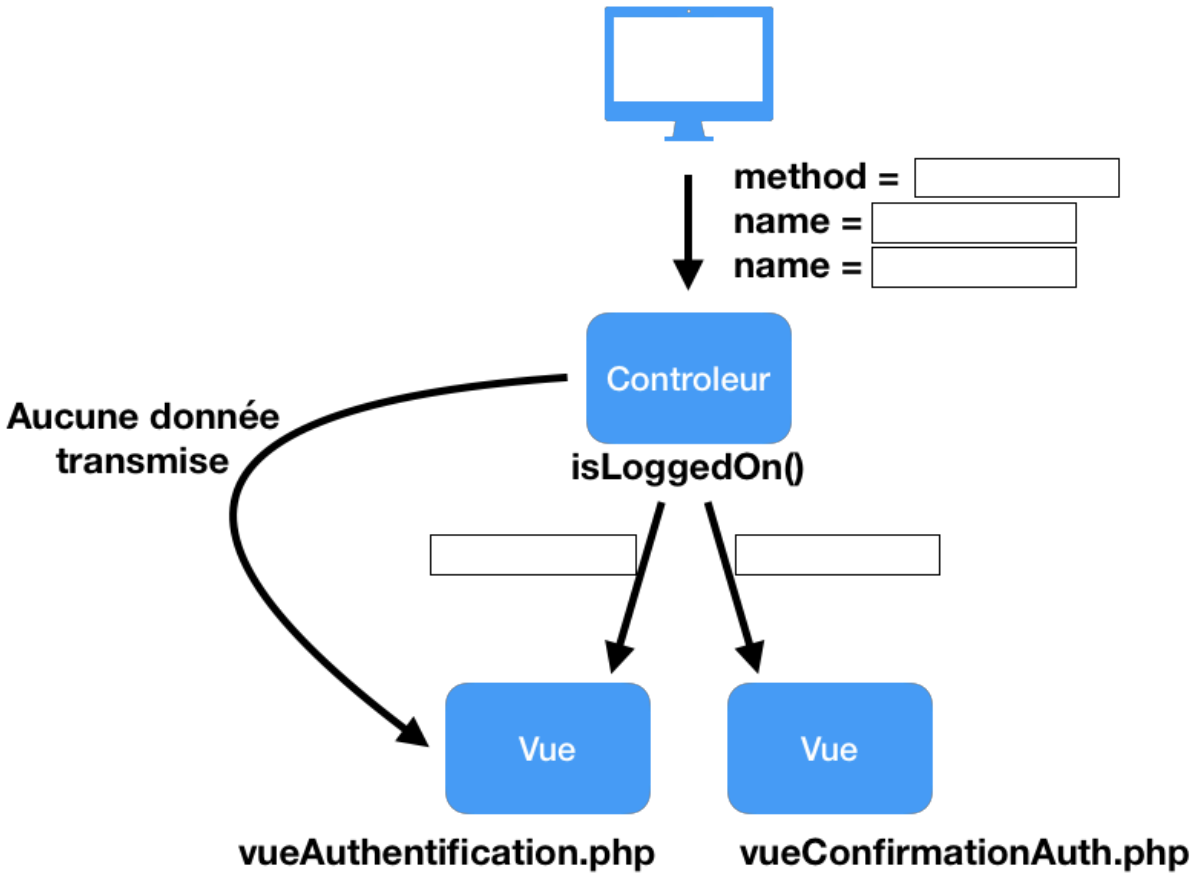
1.3. Quelle fonction permet la connexion de l'utilisateur ? Quelle est sa définition (prototype, signature) ?

```
function login($mailU, $mdpU) {
    if (!isset($_SESSION)) {
        session_start();
    }
    $util = getUtilisateurByMailU($mailU);
    $mdpBD = $util["mdpU"];
    if (trim($mdpBD) == trim(crypt($mdpU, $mdpBD))) {
        // le mot de passe est celui de l'utilisateur dans la base de donnees
        $_SESSION["mailU"] = $mailU;
        $_SESSION["mdpU"] = $mdpBD;
    }
```

1.4. À l'aide des questions précédentes et en observant la section de test et le résultat d'exécution du script `authentification.inc.php`, indiquer le rôle des fonctions suivantes :

fonction	rôle

login()	Connecter l'utilisateur et son compte.
isLoggedOn()	Vérifier si l'utilisateur est connecté.
getMailULogged()	Obtenir le mail à partir duquel l'utilisateur s'est connecté
logout()	Deconnecter l'utilisateur



1.5
method = POST, name = mailU, name = mdpU
isLoggedOn() => True => vueConfirmationAuth.php
isLoggedOn() => False => vueAuthentification.php

Question 2 - Contrôleur de déconnexion : deconnexion.php

Documents à utiliser

- fichiers fournis en ressources

- annexes 1, 2 et 5

Le contrôleur `deconnexion.php` est appelé lorsque l'utilisateur clique sur le lien "déconnexion" dans le menu principal.

2.1 Quelle fonction du modèle permet de déconnecter l'utilisateur actuellement connecté sur le site ?
logout()

2.2. Quelle vue permettant de confirmer la déconnexion devra être appelée par ce contrôleur ?

vueDeconnexion.php

2.3. Lors de la déconnexion, est-il utile de transmettre une donnée au contrôleur ?

Non, pas vraiment.

Question 3 - Analyse et adaptation du contrôleur de recherche : rechercheResto.php

Documents à utiliser

- fichiers fournis en ressources

- annexes 6,7, 8 et 9

En version finale, le contrôleur rechercheResto.php permet différents modes de recherche. Pour le moment il s'agit de se focaliser sur les deux recherches suivantes : par nom et par adresse.

Le contrôleur rechercheResto.php peut être appelé dans 2 situations :

- soit pour demander l'affichage du formulaire de recherche par l'intermédiaire de la vue `vueRechercheResto.php`,
- soit pour effectuer la recherche selon les valeurs saisies dans le formulaire et afficher les résultats à l'aide de la vue `vueResultRecherche.php`.

Lorsque l'utilisateur a rempli puis validé le formulaire de recherche, le contrôleur doit effectuer la bonne recherche en utilisant la fonction du modèle appropriée.

Le script de vue `vueResultRecherche.php` est capable d'afficher la variable appelée `$listeRestos`.

Le type de données de cette variable est compatible avec ce que retournent les fonctions du modèle :

`getRestoByIdR()`, `getRestosByNomR()` et `getRestosByAdresse()`.

3.1. En consultant le script `bd.resto.inc.php`, indiquer pour chacune de ces trois fonctions leurs signatures (prototype ou définition). Préciser le nom des paramètres attendus en plus de leurs types.

La fonction getRestoByIdR prend comme parametre $idR (l'id d'un resto) qui devrait etre un entier.

La fonction getRestoByNomR prend comme parametre $nomR, qui est un string

getRestoByAdresse prend comme parametres $voieAdresse, $cpR et $villeR, respectivement la voie, le code postal et la ville d'un Resto, qui sont un string, un entier, et un string

```php
function getRestoByIdR($idR) {

    try {

        $cnx = connexionPDO();

        $req = $cnx->prepare("select * from resto where idR=:idR");

        $req->bindValue(':idR', $idR, PDO::PARAM_INT);



        $req->execute();



        $resultat = $req->fetch(PDO::FETCH_ASSOC);

    } catch (PDOException $e) {

        print "Erreur !: " . $e->getMessage();

        die();

    }

    return $resultat;
```

```php
}


function getRestosByNomR($nomR) {
    $resultat = array();

    try {
        $cnx = connexionPDO();
            $req = $cnx->prepare("select * from resto where nomR like :nomR");
        $req->bindValue(':nomR', "%".$nomR."%", PDO::PARAM_STR);

        $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;
}
```

```php
function getRestosByAdresse($voieAdrR, $cpR, $villeR) {

    $resultat = array();



    try {

        $cnx = connexionPDO();

          $req = $cnx->prepare("select * from resto where voieAdrR like
:voieAdrR and cpR like :cpR and villeR like :villeR");

                      $req->bindValue(':voieAdrR',   "%".$voieAdrR."%",
PDO::PARAM_STR);

        $req->bindValue(':cpR', $cpR."%", PDO::PARAM_STR);

        $req->bindValue(':villeR', "%".$villeR."%", PDO::PARAM_STR);

        $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;

}
```

À l'aide de la fonction `print_r()`, afficher le contenu de la variable `$_POST` dans le contrôleur `rechercheResto.php`.

3.2. Quel est le contenu de la variable $_POST dans les 2 situations suivantes :

recherche d'un nom de restaurant : Array ([nomR] => (nom de la recherche))

recherche d'un restaurant selon l'adresse suivante : rue saint remi 33000 bordeaux : Array ([villeR] => rue saint remi [cpR] => 33000 [voieAdrR] => bordeaux)

3.3. Quels sont les noms de variables transmises au contrôleur en méthode `POST` lors de la recherche ?

```
// recherche par nom

$nomR = "";

// recherche par adresse

$voieAdrR = "";

$cpR = "";

$villeR = "";
```

3.4. Compléter la section de récupération des données `POST` dans le contrôleur afin de faire en sorte que les variables `$nomR`, `$voieAdrR`, `$cpR` et `$villeR` soient valorisées correctement en fonction de la recherche effectuée. Par défaut ces variables sont initialisées à chaîne vide.

exemple : `$nomR` ne peut recevoir `$POST['nomR']` que si `$POST['nomR']` existe. Dans le cas contraire, `$nomR` doit se voir affecté chaîne vide.

```
if (!empty($_POST)) {

    switch ($critere) {

        case 'nom':

            // recherche par nom

            $nomR = $_POST['nomR'] ;

            break;

        case 'adresse':
```

```
            // recherche par adresse

            $voieAdrR = $_POST['voieAdrR'] ;

            $cpR = $_POST['cpR'];

            $villeR = $_POST['villeR'];

            break;

    }

}
```

Les paramètres de recherche sont donc contenus dans les variables mentionnées au-dessus.

3.5. Compléter le code de chaque cas du switch dans le contrôleur en faisant appel à la fonction appropriée du modèle. Utiliser les paramètres tels qu'ils ont été décrits en question 3.1.

Rappel : Les données récupérées doivent être placées dans la variable `$listeRestos` pour que la vue puisse l'afficher.

```
if (!empty($_POST)) {

    switch ($critere) {

        case 'nom':

            // recherche par nom

            $nomR = $_POST['nomR'] ;

            getRestosByNomR($nomR);



            break;

        case 'adresse':

            // recherche par adresse

            $voieAdrR = $_POST['voieAdrR'] ;

            $cpR = $_POST['cpR'];
```

```
            $villeR = $_POST['villeR'];

            getRestosByAdresse($voieAdrR, $cpR, $villeR);

            break;

    }

}
```

3.6. Pourquoi l'appel aux fonctions du modèle est fait lorsque la condition `!empty($_POST)` est vérifiée ?

Parce que sinon les variables du switch ne seront pas remplies, et que sans celles-ci, lors de l'appel des fonctions du contrôleur, ça va renvoyer un message d'erreur.

La vue affichant le résultat de la recherche (`vueResultRecherche.php`) doit être appelée dans le bloc de fin du contrôleur.

La syntaxe d'appel de vue est similaire aux autres vues incluses. Par exemple :

```
include "$racine/vue/entete.html.php";
```

L'affichage du résultat de la recherche n'est pas systématique, il faut que des données aient été trouvées.

Dans tous les cas, le formulaire de recherche est affiché afin de permettre à l'utilisateur de modifier sa recherche.

Consulter la version définitive du site pour avoir un aperçu du comportement attendu lorsque l'utilisateur recherche un restaurant.

3.7. Ajouter dans le contrôleur l'appel à la vue `vueResultRecherche.php`.