Architecture MVC en PHP - Généralités

Découverte générale du patron de conception MVC

Remarques importantes à destination de l'enseignant :
- Ce support aborde de manière très succincte et très générale les composants du MVC pour en avoir une vision globale. Chaque composant sera étudié plus en détail dans les supports suivants.
- Les sections de tests contenues dans les contrôleurs et dans les modèles permettant de tester de manière indépendantes chaque module ne fonctionnent que sous environnement UNIX.

Rappel du contexte

R3st0.fr est un site web de critique de restaurant. À l'image des sites de ce type il a pour vocation le recensement des avis des consommateurs et la diffusion de ces avis aux visiteurs.

Ce site web est développé en PHP en suivant le patron de conception "modèle vue contrôleur" (pattern MVC). L'architecture retenue pour ce projet permet d'appréhender la programmation web d'une manière structurée.

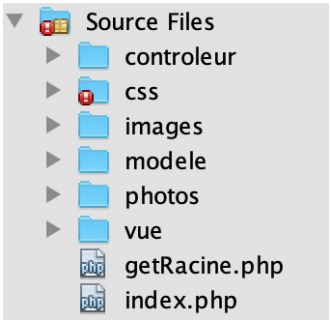
L'objectif de ce document est d'analyser de manière globale l'organisation du site, puis dans les documents suivants les différents composants de l'architecture MVC.

Ressources à utiliser
- Dossier "base de données" : fichier base.sql contenant les tables et les données de la base utilisée par l'application web étudiée.
- Dossier site : arborescence et fichiers de l'application web.

Après avoir créé la base de données (encodage utf8mb4) et importé le fichier `base.sql` , créer un nouveau projet PHP appelé SI6MVCTP1 dans Netbeans. Paramétrer le projet pour que celui-ci soit téléchargé sur le serveur lors de l'exécution.

Dans le gestionnaire de fichiers, supprimer le fichier `index.php` créé automatiquement, et remplacer le contenu du dossier "Source Files" par le contenu du dossier "site" fourni.

L'arborescence devrait être celle-ci :



Avant de commencer le TP, le site doit être paramétré pour qu'il utilise votre base de données. Dans le script `modele/bd.inc.php`, modifier les lignes suivantes afin d'indiquer les bonnes informations :

```
$login = "votre login mariaDB";
$mdp = "votre mot de passe mariaDB";
$bd = "nom de votre base de données";
```

```
$serveur = "adresse IP ou nom du serveur de base de données";
```

Afin de mieux voir l'objectif du site, et les différentes fonctionnalités que vous devrez mettre en place tout au long de ces TP, le site final est consultable à l'adresse fournie par votre professeur.

Questions 1 - Analyse de l'affichage de la liste des restaurants

Documents à utiliser
- fichiers fournis en ressources
- annexes 1,2,3,4,5,10 et 11

Exécuter le projet puis à l'aide d'un navigateur, accéder à l'url suivante :
`http://serveur/SI6MVCTP1/index.php?action=liste`

1.1. A l'aide de la documentation officielle PHP, rappeler le rôle des mots clés suivants :
`include_once`, `include`

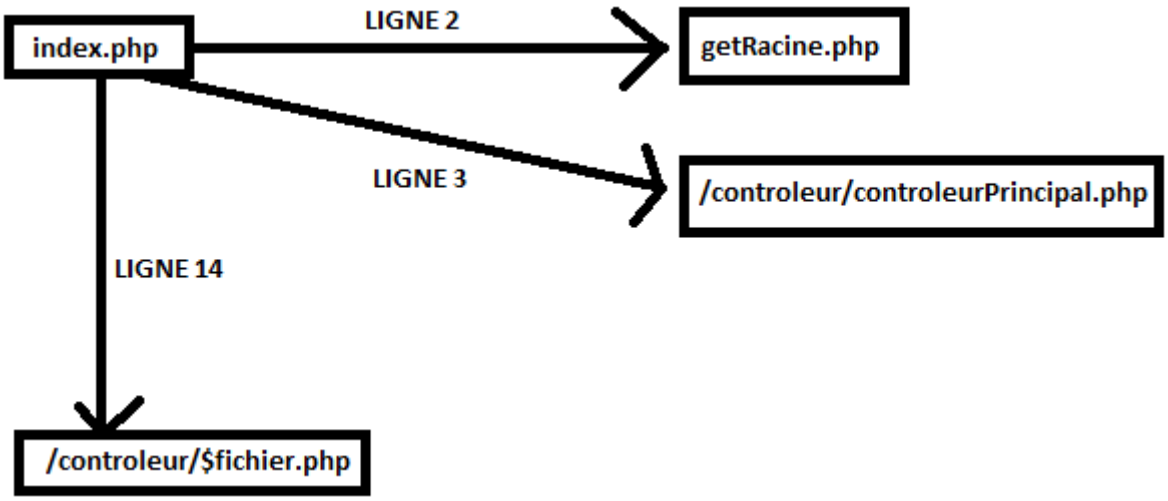
On va d'abord commencer par « include ». include permet d'inclure et d'exécuter les codes présents dans le fichier en question vers un autre. Par exemple, si je fais ''include ''header.php''; '', cela va m'inclure et exécuter l'intégralité des codes écrits dans header.php à l'intérieur du fichier qui l'appelle.

« include_once » sert quant à lui à s'assurer que le fichier inclut a bel et bien été inclus qu'une seule et unique fois. Par exemple, si je mets deux fois ''include_once ''header.php'';'' dans le même fichier, header.php ne sera exécuté qu'une fois.

1.2. Déterminer la valeur de la variable `$fichier` en affichant son contenu à la ligne précédant la balise de fin de PHP dans index.php.

La valeur de la variable $fichier équivaut au fichier du contrôleur utilisé pour afficher la page en question. Par exemple, la page d'accueil qui affiche tous les restaurants, c'est listeRestos.php, quand on clique sur un restaurant en particulier, c'est detailResto.php

1.3. En partant du fichier index.php, schématiser l'ensemble des fichiers utilisés (include ou include_once) pour afficher la liste des restaurants à l'aide de :
- rectangles portant le nom de chaque fichier,
- flèches pointant vers le fichier utilisé (inclus). Sur la flèche, indiquer la ligne qui pointe vers le fichier.



Le fichier index.php fait appel aux fichiers getRacine.php, /controleur/controleurPrincipal.php et /controleur/$fichier.php

Dans le fichier index.php, ces appels sont faits respectivement dans les lignes 2, 3 et 14.

Après avoir observé le code des scripts `bd.resto.inc.php` et `listeRestos.php` :

1.4. Relever l'ensemble des requêtes SQL contenues dans les fonctions déclarées dans le fichier `bd.resto.inc.php`.

Fonction	Requête
getRestoByIdR	SELECT * FROM resto WHERE idR=idR;
getRestos	SELECT * FROM resto;
getRestosByNomR	SELECT * FROM resto WHERE nomR LIKE :nomR;
getRestosByAdresse	SELECT * FROM resto WHERE voie AdrR LIKE :voieAdrR AND cpR LIKE :cpR AND villeR LIKE :villeR;

1.5. Quels sont les points communs de ces requêtes SQL ?

Les points communs dans ces requêtes SQL sont les suivants :
- Elles commencent toutes par « SELECT * FROM resto; » (sélectionner toutes les lignes depuis la table « resto »)

- Plus individuellement, elles ont des commandes « WHERE » et « LIKE » permettant de récupérer certaines informations précises dans ces lignes (comme l'adresse par exemple) et de les lier à ce que renvoie la page avec les LIKE.
Par exemple, getRestoByIdR (qui est le seul fonctionnel pour le moment) permet de renvoyer une page avec les informations d'un restaurant en particulier (par exemple, le restaurant « idR=4 » est la Cidrerie du fronton.

Analyse des fonctions du site à partir des annexes 5 et 6

A la fin du script présenté en annexe 5 (bd.resto.inc.php) se trouve la section suivante :

```
if ( $_SERVER["SCRIPT_FILENAME"] == __FILE__ ){
    // prog principal de test
```

Cette section permet de tester chacune des fonctions écrites dans le script (bd.resto.inc.php).
Le nom de la fonction est rappelé :

```
echo "getRestos() : \n";
```

Puis la fonction est exécutée, et son résultat est affiché :

```
print_r( getRestos());
```

Par exemple l'instruction `getRestoByIdR(1)` donne le résultat ci-dessous. Ce résultat est affiché grâce à la fonction print_r().

```
Array
(
      [idR] => 1
      [nomR] => l'entrepote
      [numAdrR] => 2
      [voieAdrR] => rue Maurice Ravel
      [cpR] => 33000
      [villeR] => Bordeaux
      [latitudeDegR] =>
      [longitudeDegR] =>
)
```

Cette section n'est exécutée que quand on teste le fichier indépendamment du reste du site. Elle n'est pas exécutée quand le fichier est inclus depuis un autre script.

Cette section ne fonctionne que dans un environnement Linux car sous Windows le test `if ( $_SERVER["SCRIPT_FILENAME"] == __FILE__ )` renvoie toujours "Faux".

1.6. En observant le nom des fonctions, le résultat affiché, et la requête contenue dans la fonction, expliquer d'une manière très générale ce que fait chacune des fonctions présente dans le script bd.resto.inc.php.
Par exemple pour la fonction getRestoByIdR() :
la requête SQL permet de récupérer les informations d'un restaurant à partir de son idR. Le résultat d'exécution de la fonction visible en annexe 6 est le suivant :

```
getRestoByIdR(idR) :
Array
(
    [idR] => 1
    [nomR] => l'entrepote
    [numAdrR] => 2
    [voieAdrR] => rue Maurice Ravel
    [cpR] => 33000
    [villeR] => Bordeaux
    [latitudeDegR] => 44.7948
    [longitudeDegR] => -0.58754
    [descR] => description
    [horairesR] => <table>...</table>
)
```

Cette fonction permet donc de récupérer les informations d'un restaurant à partir de l'identifiant passé en paramètre..

FONCTION getRestos() :

Cette requête SQL permet de récupérer les informations de TOUS les restaurants (pratique si on veut faire la liste des restaurants disponibles comme sur une page d'accueil).

Le résultat d'exécution de la fonction visible en annexe 6 est le suivant :

```
Array
(
   [0] => Array
      (
         [idR] => 1
         [nomR] => l'entrepote
         [numAdrR] => 2
         [voieAdrR] => rue Maurice Ravel
         [cpR] => 33000
         [villeR] => Bordeaux
         [latitudeDegR] => 44.7948
         [longitudeDegR] => -0.58754
         [descR] => description
         [horairesR] => <table>...</table>
      )

   [1] => Array
      (
         [idR] => 2
         [nomR] => le bar du charcutier
         [numAdrR] => 30
         [voieAdrR] => rue Parlement Sainte-Catherine
         [cpR] => 33000
         [villeR] => Bordeaux
         [latitudeDegR] =>
         [longitudeDegR] =>
         [descR] => description
         [horairesR] => <table>...</table>
      )

   [2] => Array
      (
         [idR] => 3
         [nomR] => Sapporo
         [numAdrR] => 33
         [voieAdrR] => rue Saint Rémi
         [cpR] => 33000
         [villeR] => Bordeaux
         [latitudeDegR] =>
         [longitudeDegR] =>
         [descR] => Le Sapporo propose à ses clients de délicieux plats typiques japonais.
         [horairesR] => <table>...</table>
      )
```

```
////////////// extrait tronqué ////////////

  [10] => Array
     (
        [idR] => 11
        [nomR] => Trinquet Moderne
        [numAdrR] => 60
        [voieAdrR] => Avenue Dubrocq
        [cpR] => 64100
        [villeR] => Bayonne
        [latitudeDegR] =>
        [longitudeDegR] =>
        [descR] => description
        [horairesR] => <table>...</table>
     )

)
```

Cette fonction permet alors de récupérer toutes les informations des restaurants sans paramètre précis.

FONCTION getRestosByNomR() :
Cette requête SQL permet de récupérer les informations d'un ou plusieurs restaurant(s) en particulier via la donnée passée en paramètre. Cette donnée est un texte permettant de trouver un ou plusieurs restaurant(s) en tapant son nom entier, ou simplement une partie de celui-ci.
Par exemple, si je mets `print_r(getRestosByNomR("le bar du charcutier"));`, cela me renvoie le résultat suivant :

```
[0] => Array
    (
        [idR] => 2
        [nomR] => le bar du charcutier
        [numAdrR] => 30
        [voieAdrR] => rue Parlement Sainte-Catherine
        [cpR] => 33000
        [villeR] => Bordeaux
        [latitudeDegR] =>
        [longitudeDegR] =>
        [descR] => description
        [horairesR] => <table>
```

Et si je mets uniquement `print_r(getRestosByNomR("charcut"));`, cela me renvoie encore le même résultat, cette requête ne demande pas le nom exact pour trouver le restaurant en question. Il peut même renvoyer plusieurs noms en lien dans le cas d'un nom de restaurant donné de façon très vague.

Cette fonction permet alors de récupérer toutes les informations d'un ou plusieurs restaurants selon le nom qu'on aura rentré.

FONCTION getRestosByAdresse() :
Cette requête SQL permet de récupérer les informations d'un restaurant en particulier via les trois données passées en paramètre qui sont respectivement : l'adresse (voieAdrR), le code postal (cpR) et la ville (villeR). Ces données permettent de trouver le restaurant en question via les éléments cités.
Par exemple, si je mets `print_r(getRestosByAdresse("Ravel", "33000", "Bordeaux"));`, cela me renvoie le résultat suivant :

```
[0] => Array
    (
        [idR] => 1
        [nomR] => l'entrepote
        [numAdrR] => 2
        [voieAdrR] => rue Maurice Ravel
        [cpR] => 33000
        [villeR] => Bordeaux
        [latitudeDegR] => 44.7948
        [longitudeDegR] => -0.58754
        [descR] => description
        [horairesR] => <table>
```

Cette fonction permet alors de récupérer toutes les informations d'un restaurant selon l'adresse complète fournie (avec son adresse, son code postale et sa ville).

1.7. Quelle fonction définie dans le script `bd.resto.inc.php` est utilisée dans `listeRestos.php` ?
Il s'agit de getRestos(), c'est clairement affiché dans la ligne 11 :

```
// appel des fonctions permettant de recuperer les donnees utiles a l'affichage
$listeRestos = getRestos();
```

Questions 2 - Début d'analyse de la structure du site

Documents à utiliser
- fichiers fournis en ressources
- annexes 1,2,5,7,8 et 9

L'extrait de site web fourni en ressources respecte le pattern MVC ou Modèle Vue Contrôleur Dans cette méthode de programmation, les différentes "parties" permettant de construire un site sont gérées de façon indépendante. Découvrons à quoi servent les parties "Vue" et "Modèle".

Répondre aux questions suivantes après avoir observé le contenu des scripts dans les dossiers controleur et vue du projet fourni en ressources.

2.1. Trouve-t-on des éléments de CSS ou de HTML dans les fichiers des dossiers controleur et modele ?

Nous n'en trouvons absolument aucun.

2.2. Dans quel dossier sont contenus les fichiers gérant les éléments de HTML et de CSS ?

Ils sont contenus dans les fichiers appartenant dans le dossier « vue ».

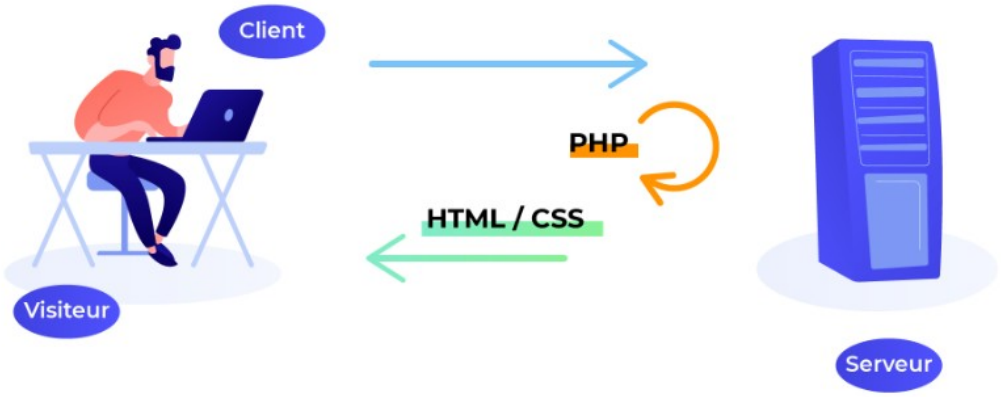
2.3. Consulter le code source de la page lors d'une visite de l'URL "`http://www.btssiobayonne.fr/~login/SI6TP1/index.php?action=liste`". Retrouve-t-on le même contenu que celui des scripts du dossier vue ?

En premier lieu, nous pouvons aisément constater la présence des fichiers « entete.html.php » et « pied.html.php » qu'on retrouve respectivement tout en haut et tout en bas du code source de la page.

Ensuite, le code PHP de vueListeRestos.php est alors adapté pour afficher du contenu HTML, mais j'expliquerai ça plus en détails dans le 2.4. Ce fichier met en place donc la façon dont sera affichée le contenu aux yeux de l'utilisateur.

2.4. Le code PHP contenu dans vueListeRestos.php est-il toujours visible après réception par le navigateur ? Par quoi est il remplacé ?

En tant que tel, le code .php n'est jamais lu par le navigateur, il est traité par le serveur



(Image disponible dans OpenClassroom) Nous constatons que le code .php n'est géré et traité uniquement que par le serveur qui renverra une réponse à l'utilisateur sous forme de HTML / CSS (ce qui est d'ailleurs pratique d'un point de vue sécuritaire).

2.4. Rappeler le point commun entre toutes les fonctions définies dans le script `bd.resto.inc.php` contenu dans le dossier modele.

Comme énoncé dans l'exercice 1.5, elles ont toutes en point commun de faire appel à l'intégralité des lignes de la table SQL « resto » (SELECT * FROM resto;)

2.5. Trouve-t-on dans d'autres fichiers que ceux du dossier modele des références à la base de données ?

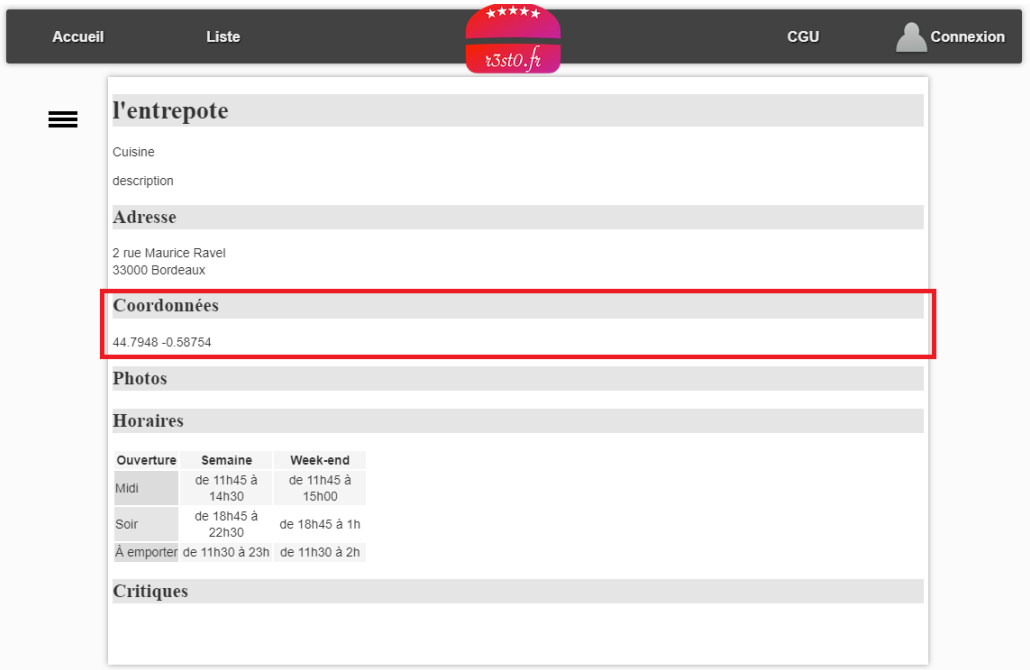
Effectivement, nous pouvons en trouver par exemple dans le fichier « detailResto.php » dans le dossier controleur. Dans la ligne 19, on trouve « `$unResto = getRestoByIdR($idR);` », getRestoByIdR qui est fonction faisant appel à des requêtes SQL.

Ensuite, dans le dossier Vue, « vueDetailResto.php » montre plusieurs lignes dans ce genre là :
`<?= $unResto['numAdrR']; ?>`
Permettant d'afficher les informations aux yeux du visiteur (dans ce cas-là, l'information présente dans « numAdrR » liée au restaurant).

Remplacez numAdrR par le nom d'une des colonnes disponibles dans la table SQL associée, et vous obtiendrez l'information souhaitée lorsque la page est chargée.

vueDetailResto.php :

```
<h2 id="coordonnees">
    Coordonnées
</h2>
<p>
    <?= $unResto['latitudeDegR']; ?>
    <?= $unResto['longitudeDegR']; ?>
</p>
```



Synthèse : expliquer globalement le rôle des scripts contenus dans les dossiers suivants

- modele

Le modèle permet de récupérer les contenus dont on a besoin depuis la base de données traitée et comment qu'est-ce qui sera affiché selon les commandes SQL qu'on voit. Une partie orientée PHP & SQL.

- vue

La vue permet de définir comment seront affichées les information aux yeux de l'utilisateur. La structure du site en général, en somme. Une partie orientée HTML & CSS.

Questions 3 – Début d'analyse de la partie contrôleur

❖ Analyse du fichier contrôleur listeRestos.php

Documents à utiliser

- fichiers fournis en ressources
- annexes 1,2,3,4 et 8

Lorsque l'on observe les commentaires présents dans le fichier, on voit que quatre grandes parties sont prévues :

- récupération des données GET, POST, et SESSION
- appel des fonctions permettant de récupérer les données utiles à l'affichage
- traitement si nécessaire des données récupérées
- appel du script de vue qui permet de gérer l'affichage des données

Les contrôleurs ont souvent ces étapes, elles peuvent parfois être dans un autre ordre, ou répétées

3.1. Quelle fonction d'accès aux données est utilisée dans ce contrôleur ? Que permet-elle de récupérer dans la base de données ?

Il s'agit de la fonction suivante qui se trouve dans le fichier « bd.inc.php » :

```php
function connexionPDO() {
    $login = "root";
    $mdp = "";
    $bd = "r3st0";
    $serveur = "127.0.0.1";

    try {
        $conn = new PDO("mysql:host=$serveur;dbname=$bd", $login, $mdp, array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES \'UTF8\''));
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $conn;
    } catch (PDOException $e) {
        print "Erreur de connexion PDO ";
        die();
    }
}
```

3.2. Les données récupérées sont-elles affichées à l'écran ? L'affichage est-il fait dans le script contrôleur ?

Non, ils ne sont pas affichés à l'écran, et l'affichage n'est pas fait non plus par le script contrôleur. Le contrôleur ne s'occupe que de mettre en place ce dont on aura besoin pour après récupérer les informations (modèle) et les afficher (vue).

3.3. Quels scripts sont inclus dans les dernières lignes du contrôleur ? Quels sont leurs rôles ?

Il s'agit de ceux-ci :

```php
if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');

    echo "getRestos() : \n";
    print_r(getRestos());

    echo "getRestoByIdR(idR) : \n";
    print_r(getRestoByIdR(1));

    echo "getRestosByNomR(nomR) : \n";
    print_r(getRestosByNomR("charcut"));

    echo "getRestosByAdresse(voieAdrR, cpR, villeR) : \n";
    print_r(getRestosByAdresse("Ravel", "33000", "Bordeaux"));
}
?>
```

```php
if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog de test
    header('Content-Type:text/plain');

    echo "connexionPDO() : \n";
    print_r(connexionPDO());
}
?>
```

Leur but est, comment le souligne le commentaire, de tester les codes si le script est effectué dans le fichier directement, sans être appelé nul part ailleurs (il me semble).

❖ Analyse du fichier contrôleur detailResto.php

Documents à utiliser
- fichiers fournis en ressources
- annexes 3,5,8 et 9

3.4. Quelle donnée est transmise au contrôleur en méthode GET ?
La donnée qui est transmise est l'IdR du restaurant en question.

3.5. Rappeler le rôle de la fonction getRestoByIdR(). Pourquoi cette méthode a-t-elle besoin d'un paramètre ?
C'est une fonction permettant de récupérer les informations du restaurant à partir de son identifiant (IdR). Cette méthode a besoin d'un paramètre car l'identifiant est une clé unique (donc primaire) permettant d'identifier chaque restaurant de façon individuelle, et ainsi afficher les informations qui sont bien liées par tel ou tel restaurant..

La variable $unResto est créée lors de l'appel à la fonction getRestoByIdR().
3.6. Explorer les fichiers "vue" inclus à la fin du fichier contrôleur et repérer les lignes où sont utilisées cette variable.
Comme on le constate dans la capture d'écran ci-dessous, plein de lignes sont du style « <?= $unResto['DonnéeEnQuestion']; ?> ». Tout cela permet d'afficher chaque information du restaurant présente au sein de la base de données, comme son nom, sa description, son adresse, ses horaires, etc.

```php
<h1><?= $unResto['nomR']; ?></h1>

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

<h2 id="coordonnees">
    Coordonnées
</h2>
<p>
    <?= $unResto['latitudeDegR']; ?>
    <?= $unResto['longitudeDegR']; ?>
</p>

<h2 id="photos">
    Photos
</h2>
<ul id="galerie">
</ul>

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

<h2 id="crit">Critiques</h2>
<ul id="critiques">
</ul>
```

3.7. Deux autres variables créées dans le contrôleur sont affichées dans la vue. Lesquelles ?
On trouve cela dans « entete.html.php », il s'agit des variables « $titre » et « $menuBurger »

3.8. Le contrôleur gère-t-il directement l'accès aux données ?

Pas vraiment, s'il fait bien appel aux données afin de récupérer les informations de ceux-ci, celui qui gère l'accès aux données est surtout le contrôleur qui s'occupe de faire les requêtes à la table de la base de données afin qu'on puisse y avoir accès, et les exploiter.

Synthèse contrôleur

Où sont affichées les données créées ou récupérées dans le contrôleur ?

Les données créées ou récupérées dans le contrôleur sont affichées dans la vue qui les traitent alors en conséquence et affiche les informations qu'il faut selon le restaurant (qui est souvent identifié à partir de son id (identifiant, qui sert de clé primaire dans une table SQL).

Les données utilisées par le contrôleur peuvent provenir de 2 sources : l'utilisateur ou la base de données.

Comment ces données sont transmises au contrôleur :

- de l'utilisateur vers le contrôleur ?

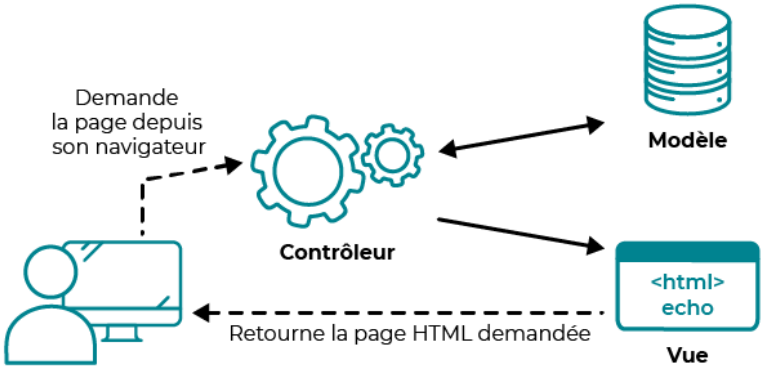
Les données sont transmises de l'utilisateur vers le contrôleur grâce à la conception MVC permettant à la vue de communiquer avec le contrôleur.

Ainsi, quand l'utilisateur fait une requête sur le site, le contrôleur se charge de lui afficher ce dont il demande (titre de la page, informations de la page, etc.).

- de la base de données vers le contrôleur ?

Les données sont transmises de la base de données vers le contrôleur grâce à la conception MVC permettant au modèle de communiquer avec le contrôleur.

Le modèle est en général représenté par la base de données dont les différentes requêtes qui ont été construites (dans le cas présent, dans diverses fonctions) sont ensuite exploitées par le contrôleur



Ce schéma d'OpenClassroom montre bien que le contrôleur est le seul à pouvoir communiquer directement avec les deux autres parties (Modèle et Vue).

Le nom du composant "contrôleur" est justifié par son rôle dans le patron de conception MVC :
il fait travailler ensemble la partie Affichage (vue) et la partie données (modèle) en récupérant les données transmises par l'utilisateur et en effectuant éventuellement des traitement sur ces données.
Le fonctionnement du contrôleur sera étudié plus en détail dans la partie suivante.

Synthèse globale

MVC est l'acronyme de Modèle Vue Contrôleur. Dans le domaine du développement d'applications, MVC est un patron de conception. Son rôle est de guider le développeur dans la création d'une application.

Le fait de décomposer une page web en 3 composants apporte plusieurs avantages :
- travail en équipe : les composants peuvent être écrits par différentes personnes ;
- test fonctionnel : en décomposant une page web en plusieurs composants, chacun des composants peut être testé séparément ;
- maintenance et évolutivité : il est possible de revoir le code, ou de changer de technologie sur un des composants (la vue par exemple) sans devoir modifier le reste du code.

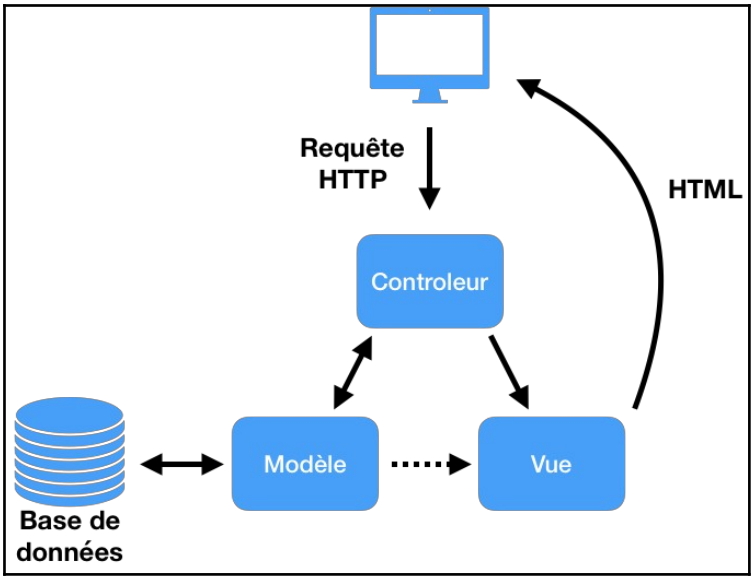
Modèle, vue et contrôleur sont les trois composants de base d'une fonctionnalité proposée par un site web. Dans le cas étudié, la fonctionnalité est l'affichage de la liste des restaurants renseignés sur le site.

Chaque fonctionnalité fait ainsi appel aux composants :
- Modèle : fonction d'accès à la base de données ;
- Vue : affichage des données à l'utilisateur ;
- Contrôleur : composant chargé de la logique applicative : récupération les données saisies par l'utilisateur, appel des fonctions du modèle, traitement des données, puis appel des vues pour l'affichage.

Le patron de conception MVC contraint et guide le programmeur dans sa manière de coder une application. Il lui impose des règles, mais en retour il permet de faciliter la maintenance, et le travail en équipe.
Ainsi il est interdit de faire le moindre affichage dans le modèle ou dans le contrôleur.

L'accès aux données, une requête SQL par exemple ne doit se trouver que dans le modèle.

La vue ne doit contenir que du code d'affichage de données provenant du contrôleur, ou éventuellement du modèle.



Pattern MVC

Annexe 1 - listeRestos.php

```
<?php
if ( $_SERVER["SCRIPT_FILENAME"] == __FILE__ ){
    $racine="..";
}
include_once "$racine/modele/bd.resto.inc.php";
// recuperation des donnees GET, POST, et SESSION
;

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

// 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";
?>
```

Annexe 2 - vueListeRestos.php

```
<h1>Liste des restaurants</h1>
<?php
for ($i = 0; $i < count($listeRestos); $i++) {
    ?>
    <div class="card">
        <div class="photoCard">
        </div>
        <div class="descrCard"><?php echo "<a href='./?action=detail&idR=" .
$listeRestos[$i]['idR'] . "'>" . $listeRestos[$i]['nomR'] . "</a>"; ?>
            <br />
            <?= $listeRestos[$i]["numAdrR"] ?>
            <?= $listeRestos[$i]["voieAdrR"] ?>
            <br />
            <?= $listeRestos[$i]["cpR"] ?>
            <?= $listeRestos[$i]["villeR"] ?>
        </div>
        <div class="tagCard">
            <ul id="tagFood">
            </ul>
        </div>
    </div>
    <?php
}
?>
```

Annexe 3 - entete.html.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
```

```
1">
        <title><?php echo $titre ?></title>
        <style type="text/css">
            @import url("css/base.css");
            @import url("css/form.css");
            @import url("css/cgu.css");
            @import url("css/corps.css");
        </style>
        <link href="https://fonts.googleapis.com/css?family=Lobster"
rel="stylesheet">
    </head>
    <body>
    <nav>

        <ul id="menuGeneral">
            <li><a href="./?action=accueil">Accueil</a></li>
            <li><a href="./?action=recherche"><img src="images/rechercher.png"
alt="loupe" />Recherche</a></li>
            <li><a href="./?action=liste">Liste</a></li>

            <li id="logo"><a href="./?action=accueil"><img
src="images/logoBarre.png" alt="logo" /></a></li>
            <li></li>
            <li><a href="./?action=cgu">CGU</a></li>

            <li><a href="./?action=connexion"><img src="images/profil.png"
alt="loupe" />Connexion</a></li>
        </ul>
    </nav>
    <div id="bouton">
        <div></div>
        <div></div>
        <div></div>
    </div>
    <ul id="menuContextuel">
        <li><img src="images/logoBarre.png" alt="logo" /></li>
        <?php if (isset($menuBurger)) { ?>
            <?php for ($i = 0; $i < count($menuBurger); $i++) { ?>
                <li>
                    <a href="<?php echo $menuBurger[$i]['url']; ?>">
                        <?php echo $menuBurger[$i]['label']; ?>
                    </a>
                </li>
            <?php } ?>
        <?php } ?>
    </ul>
    <div id="corps">
```

Annexe 4 - pied.html.php

```
</div>

</body>
</html>
```

Annexe 5 - bd.resto.inc.php

```
<?php
```

```
include_once "bd.inc.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;
}

function getRestos() {
    $resultat = array();

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

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

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();

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

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();

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

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');
    echo "getRestos() : \n";
    print_r(getRestos());

    echo "getRestoByIdR(idR) : \n";
    print_r(getRestoByIdR(1));

    echo "getRestosByNomR(nomR) : \n";
    print_r(getRestosByNomR("charcut"));

    echo "getRestosByAdresse(voieAdrR, cpR, villeR) : \n";
    print_r(getRestosByAdresse("Ravel", "33000", "Bordeaux"));
}
?>
```

Annexe 6 - Résultat de l'exécution de bd.resto.inc.php

```
getRestos() :
Array
(
    [0] => Array
        (
            [idR] => 1
            [nomR] => l'entrepote
            [numAdrR] => 2
            [voieAdrR] => rue Maurice Ravel
            [cpR] => 33000
            [villeR] => Bordeaux
            [latitudeDegR] => 44.7948
            [longitudeDegR] => -0.58754
            [descR] => description
            [horairesR] => <table>...</table>
        )

    [1] => Array
        (
            [idR] => 2
            [nomR] => le bar du charcutier
            [numAdrR] => 30
            [voieAdrR] => rue Parlement Sainte-Catherine
```

```
            [cpR] => 33000
            [villeR] => Bordeaux
            [latitudeDegR] =>
            [longitudeDegR] =>
            [descR] => description
            [horairesR] => <table>...</table>
        )

    [2] => Array
        (
            [idR] => 3
            [nomR] => Sapporo
            [numAdrR] => 33
            [voieAdrR] => rue Saint Rémi
            [cpR] => 33000
            [villeR] => Bordeaux
            [latitudeDegR] =>
            [longitudeDegR] =>
            [descR] => Le Sapporo propose à ses clients de délicieux plats
typiques japonais.
            [horairesR] => <table>...</table>
        )

////////////// extrait tronqué ////////////

    [10] => Array
        (
            [idR] => 11
            [nomR] => Trinquet Moderne
            [numAdrR] => 60
            [voieAdrR] => Avenue Dubrocq
            [cpR] => 64100
            [villeR] => Bayonne
            [latitudeDegR] =>
            [longitudeDegR] =>
            [descR] => description
            [horairesR] => <table>...</table>
        )

)
getRestoByIdR(idR) :
Array
(
    [idR] => 1
    [nomR] => l'entrepote
    [numAdrR] => 2
    [voieAdrR] => rue Maurice Ravel
    [cpR] => 33000
    [villeR] => Bordeaux
    [latitudeDegR] => 44.7948
    [longitudeDegR] => -0.58754
    [descR] => description
    [horairesR] => <table>...</table>
)
```

Annexe 7 - Extrait du résultat d'exécution de listeRestos.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
        <title>Liste des restaurants répertoriés</title>
        <style type="text/css">
            @import url("css/base.css");
            @import url("css/form.css");
            @import url("css/cgu.css");
            @import url("css/corps.css");
        </style>
        <link href="https://fonts.googleapis.com/css?family=Lobster"
rel="stylesheet">
    </head>
    <body>
    <nav>

        <ul id="menuGeneral">
            <li><a href="./?action=accueil">Accueil</a></li>
            <li><a href="./?action=recherche"><img src="images/rechercher.png"
alt="loupe" />Recherche</a></li>
            <li><a href="./?action=liste">Liste</a></li>

            <li id="logo"><a href="./?action=accueil"><img
src="images/logoBarre.png" alt="logo" /></a></li>
            <li></li>
            <li><a href="./?action=cgu">CGU</a></li>

            <li><a href="./?action=connexion"><img src="images/profil.png"
alt="loupe" />Connexion</a></li>



        </ul>
    </nav>
    <div id="bouton">
        <div></div>
        <div></div>
        <div></div>
    </div>
    <ul id="menuContextuel">
        <li><img src="images/logoBarre.png" alt="logo" /></li>
    </ul>

    <div id="corps">
<h1>Liste des restaurants</h1>

    <div class="card">
        <div class="photoCard">
        </div>
        <div class="descrCard"><a href='./?action=detail&idR=1'>l'entrepote</a>
<br />
            2            rue Maurice Ravel            <br />
            33000            Bordeaux        </div>
        <div class="tagCard">
            <ul id="tagFood">
            </ul>
        </div>
    </div>
```

```
    <div class="card">
        <div class="photoCard">
        </div>
        <div class="descrCard"><a href='./?action=detail&idR=2'>le bar du
charcutier</a>            <br />
            30            rue Parlement Sainte-Catherine            <br />
            33000            Bordeaux        </div>
        <div class="tagCard">
            <ul id="tagFood">
            </ul>
        </div>
    </div>

////////////// extrait tronqué ////////////

</div>
</body>
</html>
```

Annexe 8 - Fichier detailResto.php

```
<?php
if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    $racine = "..";
}
include_once "$racine/modele/bd.resto.inc.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
$unResto = getRestoByIdR($idR);

// traitement si necessaire des donnees recuperees
;

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

Annexe 9 - vueDetailResto.php

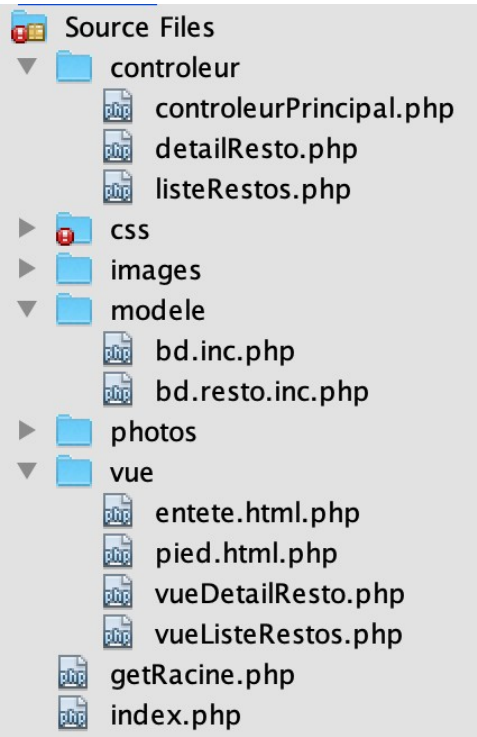
```
<h1><?= $unResto['nomR']; ?></h1>

<span id="note"></span>
<section>
    Cuisine <br />
```

```
</section>
<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">
</ul>

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

<h2 id="crit">Critiques</h2>
<ul id="critiques">
</ul>
```

Annexe 10 - index.php

```
<?php
include "getRacine.php";
include "$racine/controleur/controleurPrincipal.php";

if (isset($_GET["action"])){
    $action = $_GET["action"];
}
else{

    $action = "defaut";
}

$fichier = controleurPrincipal($action);
include "$racine/controleur/$fichier";

?>
```

Annexe 11 - Arborescence des fichiers


Source Files
controleur
controleurPrincipal.php
detailResto.php
listeRestos.php
css
images
modele
bd.inc.php
bd.resto.inc.php
photos
vue
entete.html.php
pied.html.php
vueDetailResto.php
vueListeRestos.php
getRacine.php
index.php

Architecture MVC en PHP - Contrôleur

Fonctionnement du contrôleur dans le patron de conception MVC

Remarques importantes à destination de l'enseignant
Les sections de tests contenues dans les contrôleurs et dans les modèles permettant de tester de manière indépendantes chaque module ne fonctionnent que sous environnement UNIX.

Rappel du contexte
R3st0.fr est un site web de critique de restaurant. À l'image des sites de ce type il a pour vocation le recensement des avis des consommateurs et la diffusion de ces avis aux visiteurs.

Ce site web est développé en PHP en suivant le patron de conception "modèle vue contrôleur" (pattern MVC). L'architecture retenue pour ce projet permet d'appréhender la programmation web d'une manière structurée.

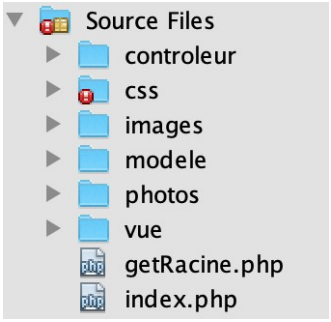
L'objectif de ce document est d'analyser le fonctionnement du contrôleur, son lien avec les autres composants de l'architecture MVC puis de mettre à jour des contrôleurs pour implémenter ou compléter des fonctionnalités.

Ressources à utiliser
- Dossier "base de données" : fichier base.sql contenant les tables et les données de la base utilisée par l'application web étudiée.
- Dossier site : arborescence et fichiers de l'application web.

Après avoir créé la base de données (encodage utf8mb4) et importé le fichier `base.sql` , créer un nouveau projet PHP appelé SI6MVCTP1 dans Netbeans. Paramétrer le projet pour que celui-ci soit téléchargé sur le serveur lors de l'exécution.

Dans le gestionnaire de fichiers, supprimer le fichier `index.php` créé automatiquement, et remplacer le contenu du dossier "Source Files" par le contenu du dossier "site" fourni.

L'arborescence devrait être celle-ci :

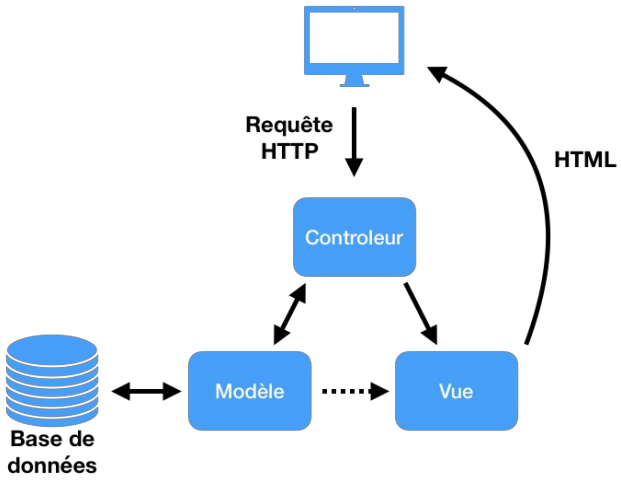


Avant de commencer le TP, le site doit être paramétré pour qu'il utilise votre base de données. Dans le script `modele/bd.inc.php`, modifier les lignes suivantes afin d'indiquer les bonnes informations :

```
$login = "votre login mariaDB";
$mdp = "votre mot de passe mariaDB";
$bd = "nom de votre base de données";
$serveur = "adresse IP ou nom du serveur de base de données";
```

Afin de mieux voir l'objectif du site, et les différentes fonctionnalités que vous devrez mettre en place tout au long de ces TP, le site final est consultable à l'adresse fournie par votre professeur.

Rappel sur les contrôleurs



Comme vu au TP précédent, un contrôleur est l'élément central d'une fonctionnalité sur un site web MVC. Chaque fonctionnalité est gérée par un contrôleur. C'est le contrôleur qui a pour rôle de :

- récupérer les données transmises par un formulaire,
- récupérer ou envoyer les données dans la base en faisant appel aux fonctions gérées par le modèle,
- traiter les données,
- appeler les vues permettant d'afficher les données récupérées, calculées, ou d'afficher les messages à destination de l'utilisateur.

Contexte

Le site r3st0.fr est un site collaboratif, les utilisateurs connectés peuvent émettre des critiques sur les restaurants et les noter. Ils peuvent aussi indiquer quel sont leurs types de cuisine préférés, ajouter des restaurants en favoris, etc...
Le site doit ainsi proposer un système d'authentification. Dans les exercices suivants, vous devrez mettre en place les contrôleurs de connexion, de déconnexion et de recherche en utilisant les vues et les modèles déjà écrits.

Question 1 - Contrôleur de connexion : connexion.php

Documents à utiliser
- fichiers fournis en ressources
- annexes 1, 2, 3, 4

Le contrôleur permettant de gérer la connexion d'un utilisateur est utilisé dans 2 cas.
- Après avoir cliqué sur le lien "connexion" du menu principal, l'utilisateur doit saisir son identifiant et son mot de passe
- L'utilisateur a saisi son login et son mot de passe dans le formulaire de connexion, l'a validé et ces deux

informations sont transmises au contrôleur de connexion.

1er cas :
le contrôleur est appelé sans paramètre, il ne peut alors pas établir la connexion et doit afficher la vue proposant le formulaire de connexion.

2eme cas :
le contrôleur est appelé avec les paramètres login et mot de passe. Il peut alors tenter la connexion. Si la connexion se passe bien, un écran de confirmation est affiché. Dans le cas contraire, le formulaire de connexion est de nouveau proposé à l'utilisateur.

Pour mettre en place ce traitement, deux scripts de la couche Modèle seront utilisés :
- `authentification.inc.php`
- `bd.utilisateur.inc.php`

Les Vues sont aussi disponibles en ressources :

- `vueAuthentification.php` : formulaire de connexion,
- `vueConfirmationAuth.php` : confirmation de connexion lorsque l'utilisateur a pu être authentifié.

Remarque
Les vues `entete.html.php` et `pied.html.php` sont toujours incluses afin de garantir l'affichage de la structure fixe du site (menus, design etc...).

L'étude des scripts de modèle et de vues sera abordée plus tard dans ce cours.

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 ?
Celui-ci est false étant donné qu'on a pas pour l'instant vérifié l'identifiant et mot de passe de l'utilisateur

1.2. Lors du 2eme appel à la fonction `isLoggedOn()` l'utilisateur est il connecté ?
Là, c'est bon, on a passé comme paramètre l'identifiant : test@bts.sio et le mot de passe : sio (le seul autorisé pour le moment)

```
not logged
logged
utilisateur connecté avec cette adresse : test@bts.sio
```

1.3. Quelle fonction permet la connexion de l'utilisateur ? Quelle est sa définition (prototype, signature) ?
La fonction qui permet la connexion de l'utilisateur est login() qui prend donc en compte deux paramètres : L'identifiant et le mot de passe (et fait une vérification que les données rentrées soient en adéquation avec ce qui est stocké (en général dans une base de données)).
Son prototype peut être noté (d'après ce que j'ai compris) : function login($mailU, $mdpU);

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()	Permet de connecter l'utilisateur s'il en fait la requête
isLoggedOn()	Vérifie si l'utilisateur est déjà connecté sur le site
getMailULogged()	Récupère l'email de la personne connectée (dans notre situation, pour l'afficher afin de l'identifier)
logout()	Permet de déconnecter l'utilisateur s'il en fait la requête.

1.5. À l'aide des annexes 2 et 3 compléter le schéma ci-dessous en indiquant :

- la méthode utilisée pour transmettre les données au contrôleur depuis le formulaire de connexion,
- le nom des indices du tableau associatif qui seront utilisés pour récupérer les données saisies dans le formulaire,
- la valeur retournée par la fonction `isLoggedOn()` orientant l'utilisateur soit sur la vue d'authentification affichant le formulaire de connexion, soit sur la vue de confirmation indiquant que l'utilisateur est bien connecté.

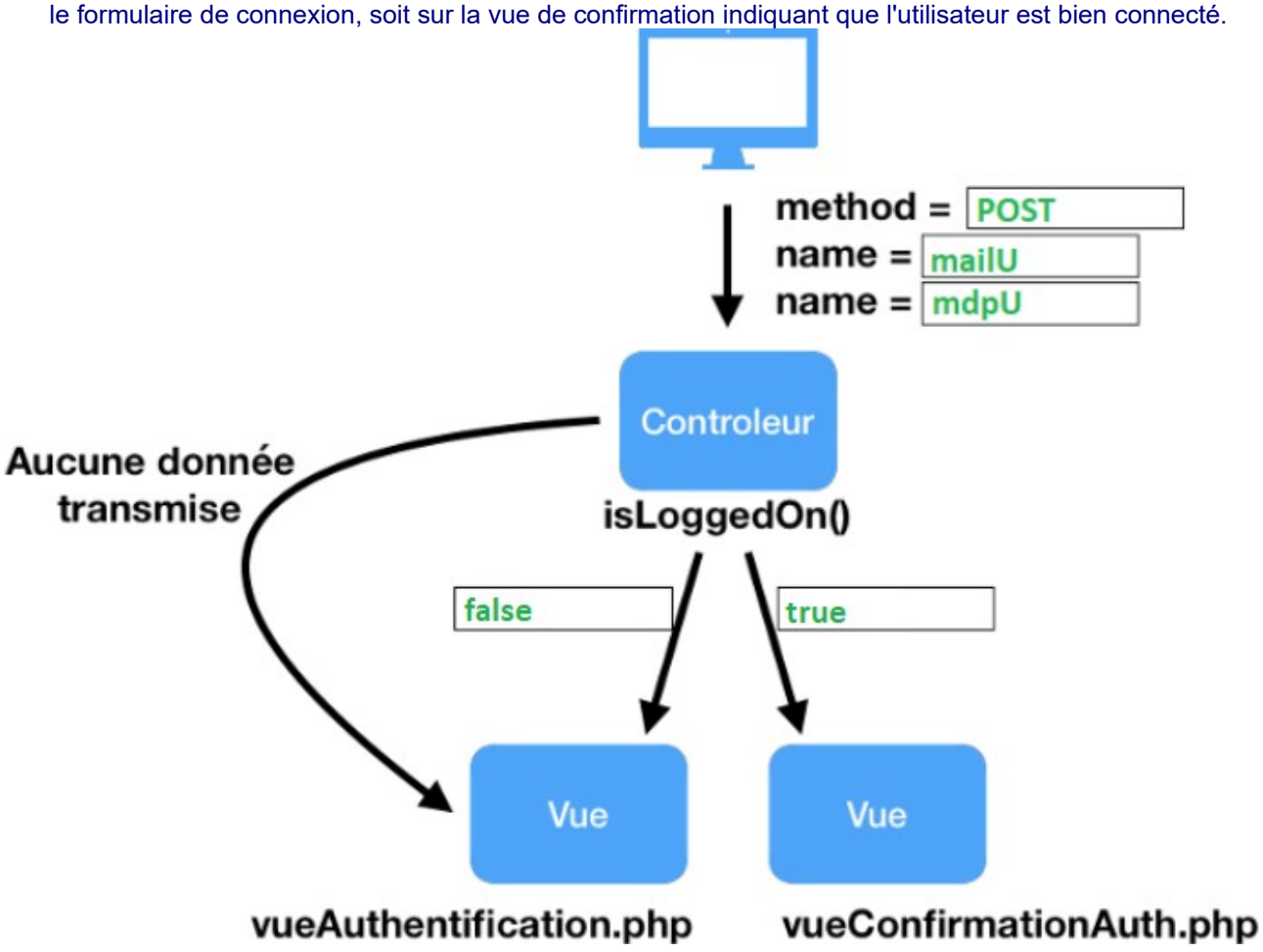


Schéma d'authentification

Remarque :

Ne pas tenir compte de la manière dont est utilisé le champ action dans la balise form du formulaire. Cette méthode spécifique au MVC sera étudiée dans l'un des cours suivants.

Répondre à la question suivante en s'inspirant de la section de test du fichier authentification.inc.php présent en annexe 1.

1.6. Compléter le code du contrôleur `connexion.php` en tenant compte des réponses données précédemment, et en respectant les indications suivantes :
Étapes du code du contrôleur de connexion :

- Si aucune donnée n'est transmise, la vue d'authentification doit être affichée
- Si les données sont transmises correctement :
    - récupérer les données transmises depuis le formulaire,
    - tenter la connexion,
    - la connexion est réussie si l'utilisateur est connecté,
        - afficher la confirmation de connexion,
        - sinon afficher à nouveau le formulaire d'authentification.

remarque : la première étape est déjà faite.

r3st0 TP2 > controleur > connexion.php > ...

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

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

// Récupération des données GET, POST et SESSION
if (!isset($_POST["mailU"]) || !isset($_POST["mdpU"])) {
    // On affiche le formulaire de connexion
    $titre = "Authentification";
    include "$racine/vue/entete.html.php";
    include "$racine/vue/vueAuthentification.php";
    include "$racine/vue/pied.html.php";
} else {
    // Récupération des données du formulaire
    $mailU = $_POST["mailU"];
    $mdpU = $_POST["mdpU"];

    // Tenter la connexion
    $util = getUtilisateurByMailU($mailU);

    if ($util && is_array($util)) {
        $mdpBD = $util["mdpU"];

        if (trim($mdpBD) == trim(crypt($mdpU, $mdpBD))) { // Une alternative à htmlspecialchars que je préfère pour les connexions
            $_SESSION["mailU"] = $mailU;
            $_SESSION["mdpU"] = $mdpBD;

            // La connexion est réussie
            $titre = "Confirmation de Connexion";
            include "$racine/vue/entete.html.php";
            include "$racine/vue/vueConfirmationAuth.php"; // Message qui dit que la connexion est réussie
            include "$racine/vue/pied.html.php";
            exit();
        }
    }

    // La connexion a échoué, afficher à nouveau le formulaire d'authentification
    $titre = "Authentification";
    include "$racine/vue/entete.html.php";
    include "$racine/vue/vueAuthentification.php"; // On doit recommencer.
    include "$racine/vue/pied.html.php";
}
?>
```

OK !

Question 2 - Contrôleur de déconnexion : deconnexion.php

Documents à utiliser
- fichiers fournis en ressources
- annexes 1, 2 et 5

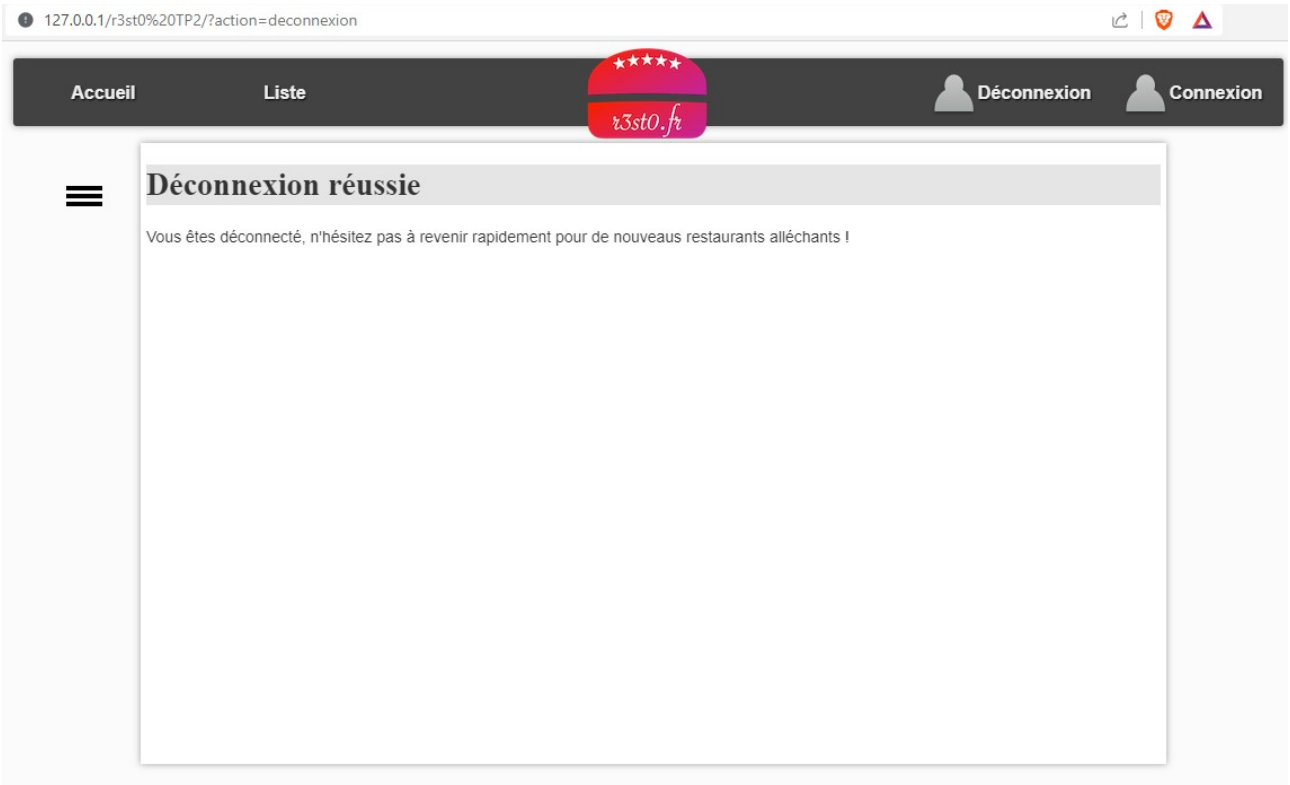
Le contrôleur `deconnexion.php` est appelé lorsque l'utilisateur clique sur le lien "déconnexion" dans le menu principal.

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

2.2. Quelle vue permettant de confirmer la déconnexion devra être appelée par ce contrôleur ?
Il s'agit de la vue vueDeconnexion.php

2.3. Lors de la déconnexion, est-il utile de transmettre une donnée au contrôleur ?
Pas vraiment, en général, il suffit simplement de faire un « session_destroy() », et le tout est joué pour la fonction logout() ;
Ou un « unset » comme dans le cas de la fonction fournie.

2.4. Compléter le code du contrôleur `deconnexion.php`. Celui-ci doit appeler la fonction appropriée du modèle, puis afficher à l'utilisateur un message de confirmation conformément aux questions précédentes.



OK !

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 : `getRestos()`, `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.

Function getRestoByIdR($idR) → Le paramètre attendu est idR qui est un « int » (nombre entier)
Function getRestosByNomR($nomR) → Le paramètre attendu est nomR qui est un « string » (chaîne de caractères)
Function getRestosByAdresse($voieAdrR, $cpR, $villeR) → Les paramètres attendus sont voieAdrR qui est un « string » (chaîne de caractères), cpR qui est un « int » (nombre entier) et villeR qui est un « string » (chaîne de caractères)

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
- recherche d'un restaurant selon l'adresse suivante : rue saint remi 33000 bordeaux

) [5] => Array ([idR] => 8 [nomR] => La table de POTTOKA [numAdrR] => 21 [voieAdrR] => Quai Amiral Dubourdieu [cpR] => 64100 [villeR] => Bayonne [latitudeDegR] => [longitudeDegR] => [descR] => description [horairesR] =>

Ouverture	Semaine	Week-end
Midi	de 11h45 à 14h30	de 11h45 à 15h00
Soir	de 18h45 à 22h30	de 18h45 à 1h
À emporter	de 11h30 à 23h	de 11h30 à 2h

3.3. Quels sont les noms de variables transmises au contrôleur en méthode `POST` lors de la recherche ?
Il s'agit de nomR si on fait une recherche par nom, ou voieAdrR, cpR et villeR si on fait une recherche par adresse

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.

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.

```php
// appel des fonctions permettant de recuperer les donnees utiles a l'affichage
if (!empty($_POST)) {
    switch ($critere) {
        case 'nom':
            // Recherche par nom
            $nomR = $_POST['nomR'];
            $critere = "nom";

            $listeRestos = getRestosByNomR($nomR);
            break;

        case 'adresse':
            // Recherche par adresse
            $voieAdrR = $_POST['voieAdrR'];
            $cpR = $_POST['cpR'];
            $villeR = $_POST['villeR'];
            $critere = "adresse";

            $listeRestos = getRestosByAdresse($voieAdrR, $cpR, $villeR);
            break;
    }
}
```

Voici le code du switch

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

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.

Car sinon, cela retourne tous les restaurants, ce qui peut être vraiment lourd à partir du moment où le site recense beaucoup de données.

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

```php
// 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 "$racine/vue/vueResultRecherche.php";
include "$racine/vue/pied.html.php";
?>
```

Question 4 - Analyse de la partie existante du contrôleur rechercheResto.php

Documents à utiliser
- fichiers fournis en ressources
- annexes 6, 7 et 8

Le même contrôleur permet de rechercher selon le nom du restaurant ou son adresse. Pourtant il n'y a qu'une seule vue qui gère l'affichage des formulaires de recherche : `vueRechercheResto.php`.

4.1. Dans le code source de la vue, quelle variable permet de choisir l'affichage du formulaire de recherche par nom ou par adresse ?
Il s'agit de $critere qui grâce au switch peut avoir des comportements différents dans le cas où on veut faire une recherche par « nom » ou par « adresse »

4.2. Quelle variable transmise en méthode GET au contrôleur permet de connaître le type de recherche - par nom ou par adresse - que l'on souhaite effectuer ?
Il s'agit de nouveau de $critere

Le rôle de la variable action sera étudié plus tard.

4.3. Rechercher où est faite cette transmission : quel script ? quelle ligne ?
Cette transmission est faite directement dans l'URL, ce qui fait qu'on a ? action=recherche&critere=nom si on fait une recherche par nom ou ? action=recherche&critere=adresse si on fait une recherche par adresse.
(J'ai aussi fait une condition par défaut qui met directement par nom).

Donc en se basant sur l'index, cela se déroule avec la script <form action=...> à la ligne 2

4.4. Sans cette transmission d'information, le contrôleur pourrait-il savoir quelle recherche effectuer ?
Je dirai que non, car sans le contrôleur, il ne saurait pas quelle recherche effectuer entre la recherche par nom ou par adresse.

4.5. Sans cette transmission d'information, le script de vue pourrait-il savoir quel formulaire afficher ?
Non plus, car la variable $critere dépend aussi du contrôleur. Sans celui-ci, $critere ne peut rien récupérer et donc, n'affiche aucun formulaire. Il suffit de mettre en commentaire //include ''$racine/vue/vueRechercheResto'' pour s'en rendre compte.

4.6. Lorsqu'une recherche est effectuée, la vue affiche à nouveau le formulaire de recherche avec des valeurs prédéfinies. Quelles variables sont alors utilisées ?
Lorsqu'une recherche est effectuée, la vue utilise les variables $nomR pour la recherche par nom,

et $voieAdrR, $cpR, $villeR pour la recherche par adresse. Ces variables en question sont alors renvoyées avec les valeurs prédéfinies dans les formulaires les correspondant.

Synthèse sur le rôle et le fonctionnement du contrôleur

Le contrôleur est l'élément central dans les fonctionnalités proposées par une application MVC. Dans notre site web il permet de :

- récupérer les actions de l'utilisateur en terme d'action sur l'interface (saisies, choix, sélections, etc.) : informations transmises en méthode GET ou POST.
- Récupérer, depuis le modèle, les données qui seront utiles pour la fonctionnalité. Une liste de restaurants par exemple.
- coder la logique applicative : traiter les données, créer les variables qui seront utiles pour le fonctionnement de l'application et l'affichage. Vérifier que l'utilisateur a bien saisi son login et son mot de passe par exemple.
- commander l'affichage des vues pour afficher à l'écran les données récupérées ou calculées. La liste des restaurants recherchés par exemple.

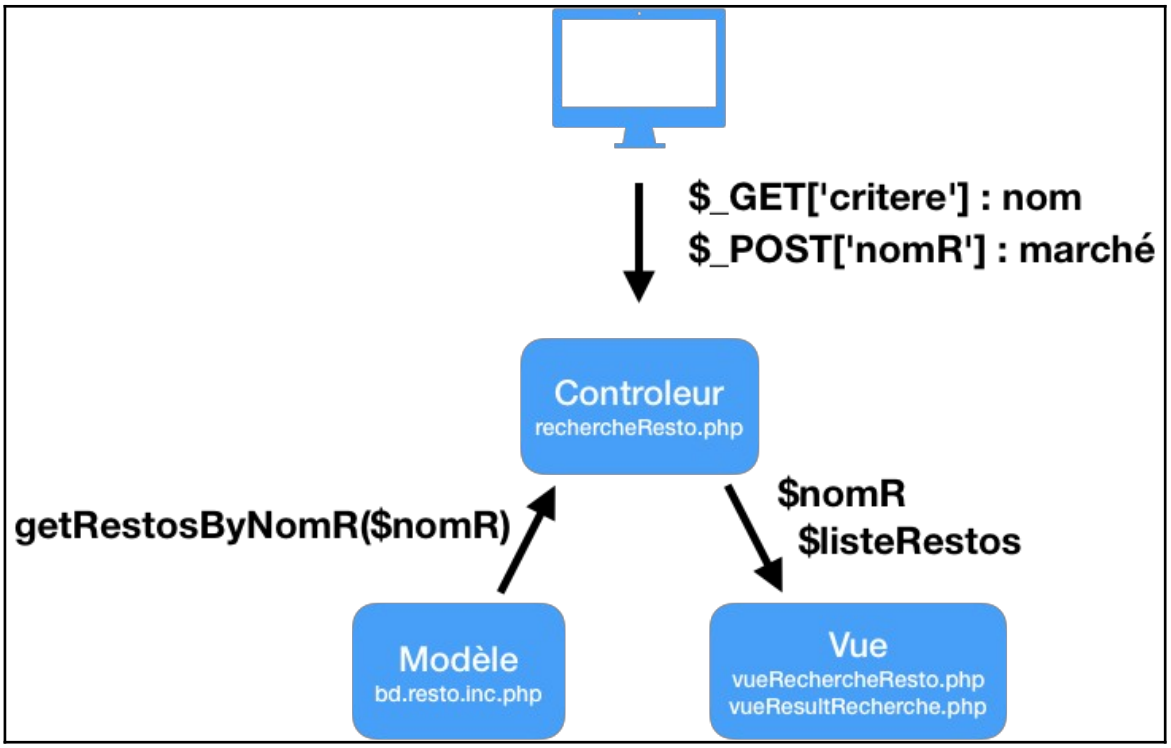


Schéma du fonctionnement du contrôleur rechercheResto.php lors d'une recherche par nom

Les fonctions du modèle sont accessibles grâce à l'inclusion des scripts nécessaires depuis le contrôleur.
Par exemple dans le script contrôleur rechercheResto.php :

```
include_once "$racine/modele/bd.resto.inc.php";
```

De même, les vues sont incluses en fin de contrôleur, une fois toute la logique applicative traitée. Les données créées ou récupérées dans le contrôleur y sont affichées.
Par exemple dans le script contrôleur listeResto.php :

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

Annexe 1 - section de test du fichier modèle authentification.inc.php

```
if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');

    // test de connexion
    if (isLoggedOn()) {
        echo "logged\n";
    } else {
        echo "not logged\n";
    }

    login("test@bts.sio", "sio");   // login de test : test@bts.sio
                                    // mot de passe de test : sio
    if (isLoggedOn()) {
        echo "logged\n";
    } else {
        echo "not logged\n";
    }

    $mail=getMailULoggedOn();
    echo "utilisateur connecté avec cette adresse : $mail \n";

    // deconnexion
    logout();
}
```

Annexe 2 - résultat d'exécution du script authentification.inc.php

```
not logged
logged
utilisateur connecté avec cette adresse : test@bts.sio
```

Annexe 3 - vueAuthentification.php

```
<h1>Connexion</h1>
<form action="./?action=connexion" method="POST">

    <input type="text" name="mailU" placeholder="Email de connexion" /><br />
    <input type="password" name="mdpU" placeholder="Mot de passe"  /><br />
    <input type="submit" />

</form>
<br />
<a href="./?action=inscription">Inscription</a>
```

Annexe 4 - contrôleur connexion.php à compléter

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

// creation 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"])){
    // on affiche le formulaire de connexion
    $titre = "authentification";
    include "$racine/vue/entete.html.php";
    include "$racine/vue/vueAuthentification.php";
    include "$racine/vue/pied.html.php";
}
else
{
    // à completer


}

?>
```

Annexe 5 - contrôleur deconnexion.php à compléter

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

// recuperation des donnees GET, POST, et SESSION

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

// traitement si necessaire des donnees recuperees


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

include "$racine/vue/pied.html.php";


?>
```

Annexe 6 - contrôleur rechercheResto.php à compléter

```
<?php

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    $racine = "..";
}
include_once "$racine/modele/bd.resto.inc.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");

// critere de recherche par defaut
$critere = "nom";
if (isset($_GET["critere"])) {
    $critere = $_GET["critere"];
}
// recuperation des donnees GET, POST, et SESSION
// recherche par nom
$nomR = "";
// recherche par adresse
$voieAdrR = "";
$cpR = "";
$villeR = "";

// appel des fonctions permettant de recuperer les donnees utiles a l'affichage
// Si on provient du formulaire de recherche : $critere indique le type de
recherche à effectuer
if (!empty($_POST)) {
    switch ($critere) {
        case 'nom':
            // recherche par nom

            break;
        case 'adresse':
            // recherche par adresse

            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 "$racine/vue/vueRechercheResto.php";
include "$racine/vue/pied.html.php";
?>
```

Annexe 7 - vue vueRechercheResto.php

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

    <?php
    switch ($critere) {
        case "nom":
            ?>
            Recherche par nom : <br />
             <input type="text" name="nomR" placeholder="nom" value="<?= $nomR ?
>" /><br />
            <?php
            break;
        case "adresse":
            ?>
            Recherche par adresse : <br />
                <input type="text" name="villeR" placeholder="ville" value="<?=
$villeR ?>"/><br />
              <input type="text" name="cpR" placeholder="code postal" value="<?=
$cpR ?>"/><br />
                <input type="text" name="voieAdrR" placeholder="rue" value="<?=
$voieAdrR ?>"/><br />
            <?php
            break;

    }
    ?>
    <br /><br />
    <input type="submit" value="Rechercher" />

</form>
```

Annexe 8 - vue vueResultRecherche.php

```
<h1>Liste des restaurants</h1>
<?php
for ($i = 0; $i < count($listeRestos); $i++) {
    ?>
    <div class="card">
          <div class="descrCard"><?php echo "<a href='./?action=detail&idR=" .
$listeRestos[$i]['idR'] . "'>" . $listeRestos[$i]['nomR'] . "</a>"; ?>
            <br />
            <?= $listeRestos[$i]["numAdrR"] ?>
            <?= $listeRestos[$i]["voieAdrR"] ?>
            <br />
            <?= $listeRestos[$i]["cpR"] ?>
            <?= $listeRestos[$i]["villeR"] ?>
        </div>
        <div class="tagCard">
            <ul id="tagFood">
            </ul>
        </div>
    </div>
    <?php
}
?>
```

Annexe 9 - extrait du modèle bd.resto.inc.php

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

function getRestoByIdR($idR) {...}

function getRestosByNomR($nomR) {...}

function getRestosByAdresse($voieAdrR, $cpR, $villeR) {...}


if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');

    echo "getRestos() : \n";
    print_r(getRestos());

    echo "getRestoByIdR(idR) : \n";
    print_r(getRestoByIdR(1));

    echo "getRestosByNomR(nomR) : \n";
    print_r(getRestosByNomR("charcut"));

    echo "getRestosByAdresse(voieAdrR, cpR, villeR) : \n";
    print_r(getRestosByAdresse("Ravel", "33000", "Bordeaux"));
}
?>
```

Architecture MVC en PHP - Vue

Fonctionnement de la vue dans le patron de conception MVC

Remarques importantes à destination de l'enseignant
Les sections de tests contenues dans les contrôleurs et dans les modèles permettant de tester de manière indépendantes chaque module ne fonctionnent que sous environnement UNIX.

Rappel du contexte
R3st0.fr est un site web de critique de restaurant. À l'image des sites de ce type il a pour vocation le recensement des avis des consommateurs et la diffusion de ces avis aux visiteurs.

Ce site web est développé en PHP en suivant le patron de conception "modèle vue contrôleur" (pattern MVC). L'architecture retenue pour ce projet permet d'appréhender la programmation web d'une manière structurée.

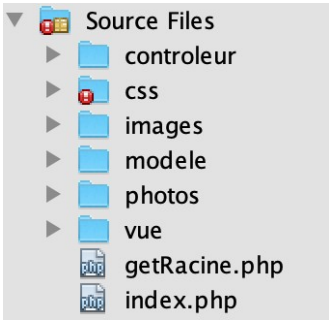
L'objectif de ce document est d'analyser le fonctionnement du composant vue dans l'architecture MVC puis de mettre à jour des vues existantes pour les améliorer.
Chaque restaurant est associé à des données dans la base : des photos, des types de cuisine, etc. L'affichage des données associées sur la fiche d'un restaurant donne plus d'informations aux utilisateurs. Ils peuvent ainsi choisir le bon restaurant où diner.

Ressources à utiliser
- Dossier "base de données" : fichier base.sql contenant les tables et les données de la base utilisée par l'application web étudiée.
- Dossier site : arborescence et fichiers de l'application web.

Après avoir créé la base de données (encodage utf8mb4) et importé le fichier `base.sql` , créer un nouveau projet PHP appelé SI6MVCTP1 dans Netbeans. Paramétrer le projet pour que celui-ci soit téléchargé sur le serveur lors de l'exécution.

Dans le gestionnaire de fichiers, supprimer le fichier `index.php` créé automatiquement, et remplacer le contenu du dossier "Source Files" par le contenu du dossier "site" fourni.

L'arborescence devrait être celle-ci :



Avant de commencer le TP, le site doit être paramétré pour qu'il utilise votre base de données. Dans le script `modele/bd.inc.php`, modifier les lignes suivantes afin d'indiquer les bonnes informations :

```
$login = "votre login mariaDB";
$mdp = "votre mot de passe mariaDB";
```

```
$bd = "nom de votre base de données";
$serveur = "adresse IP ou nom du serveur de base de données";
```

Afin de mieux voir l'objectif du site, et les différentes fonctionnalités que vous devrez mettre en place tout au long de ces TP, le site final est consultable à l'adresse fournie par votre professeur.

Partie A : Analyse d'une vue existante

Dans les trois exercices suivants vous devrez analyser le script de vue vueDetailResto.php permettant d'afficher les détails d'un restaurant. Cette vue est appelée par le contrôleur detailResto.php.

Question 1 - Analyse du script de vue vueDetailResto.php : les photos

Documents à utiliser
- fichiers fournis en ressources
- annexes 1, 2, 5 et 6

1.1. Quelles fonctions définies dans le modèle sont utilisées dans le contrôleur `detailResto.php`.

Les fonctions définies dans le contrôleur « detailResto.php » présent dans les ressources données sont :
- getRestoByIdR
- getTypesCuisinebyIdR
- getPhotosByIdR

1.2. Quelles annexes présentent l'affichage du résultat d'exécution des fonctions du modèle trouvées à la question précédente ?

Il s'agit de vueDetailResto.php (ce qui peut être difficilement autre chose, étant donné que la partie « Vue » dans le modèle MVC sert justement à… visualiser le résultat de notre code.

Rappel sur les tableaux associatifs
Un tableau associatif permet de stocker plusieurs informations dans une seule variable. L'accès aux informations ne se fait pas grâce aux numéros de cellules comme dans un tableau classique, mais avec un nom de cellule.

Exemple :
Tableau associatif : $tabEtudiant

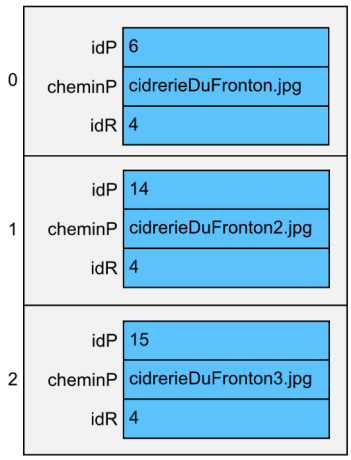
prenom	"Lionel"
nom	"Romain"
age	48

À la place d'indices, le tableau associatif fonctionne à l'aide de clés qui sont des chaînes de caractères.
La case nommée '*prenom'* contient la chaîne "lionel".
La case nommée '*nom'* contient la chaîne "Romain".
La case nommée '*age'* contient l'entier 48.

Les cellules d'un tableau associatif peuvent contenir des données de types différents.

L'affectation et l'accès à une cellule du tableau associatif fonctionnent de la même manière que pour un tableau classique :
`$tabEtudiant['prenom'] = "Patrice" ;` // permet d'affecter la chaîne "patrice" à valeur qui se réfère à la clé *'prenom'*.
`echo $tabEtudiant['prenom'] ;` // permet d'afficher la valeur de la clé '*prenom'*.

Après exécution de la fonction getPhotosByIdR() dans le contrôleur detailResto.php, la structure de la variable $lesPhotos peut être schématisée sous cette forme :



1.3. Comment est composée chacune des 3 cellules du tableau $lesPhotos ?
Les 3 cellules de $lesPhotos sont composées de :
- La première cellule est idP qui est donc la clé primaire de la table photo, permettant d'identifier chaque photo individuellement de l'autre. Il s'agit d'un int (nombre entier)
- La seconde cellule est cheminP qui représente le nom complet de la photo ainsi que son extension, ce qui peut être pratique pour après dans un <img src=photos/<?= $lesPhotos[0] [''cheminP''] ?> pour afficher la bonne image au bon endroit (on peut voir cela dans vueDetailResto.php). Il s'agit d'un string (chaîne de caractères)
- La troisième cellule est idR qui est donc une clé étrangère représentant la clé primaire de la table resto, cela permet de savoir après à quelle restaurant est lié telle image. Il s'agit d'un int (nombre entier).

1.4. Repérer dans l'annexe 1 l'instruction PHP permettant d'afficher le nom du fichier de la première photo.
L'instruction PHP permettant d'afficher le nom du fichier de la première photo est la suivante :

```php
<?php if (count($lesPhotos) > 0) { ?>
    <img src="photos/<?= $lesPhotos[0]["cheminP"] ?>" alt="photo du restaurant" />
<?php } ?>
```

On expliquera dans la question juste après le rôle de if(count($lesPhotos) > 0. On va se concentrer sur la ligne de code <img>. Pour afficher la bonne image en question, il faut faire un src=''photos/<?= $lesPhotos[0][''cheminP'']

« photos » car les images en question se trouvent dans le dossier photos

« $lesPhotos[0] » car on veut récupérer la première photo si le restaurant en possède plusieurs (on peut constater cela en affichant les détails du restaurant Cidrerie du Fronton, qui possède 3 images différentes. [0] représente la première qui lui est liée (cidrerieDuFronton.jpg), [1] représente la seconde (cidrerieDuFronton2.jpg), etc.

« [''cheminP''] car cela nous permet de récupérer la valeur de la colonne qui nous intéresse, celle qui contient le nom complet de l'image (cidrerieDuFronton.jpg, par exemple).

Si on va donc dans le detailResto de la Cidrerie du Fronton, la ligne avec des instructions en PHP seront traduites en HTML de cette façon :

**

1.5. Quel est le rôle de l'instruction if(count($lesPhotos) > 0) ? Quel est son rôle dans le code de la vue ?
Cela permet de vérifier si le restaurant qui nous intéresse a au moins une photo qui lui est liée, et si c'est le cas, de l'afficher.
C'est un bon moyen aussi d'éviter que PHP renvoie une erreur comme quoi il n'a pas pu trouver d'images, etc.

1.6. Quelle est la syntaxe générale permettant d'accéder à un champ (idP, CheminP ou idR) contenu dans la variable $lesPhotos ?
Pour chacun des trois champs cités dans la question, il s'agit simplement de :
idP = $lesPhotos[''idp'']
CheminP = $lesPhotos[''CheminP'']
idR = $lesPhotos[''idR'']

Question 2 - vueDetailResto.php : les types de cuisine

Documents à utiliser
- fichiers fournis en ressources
- annexes 1, 2, 7 et 8

2.1. Quel contrôleur produit la variable $lesTypesCuisine ? Préciser le nom de la fonction et du modèle.
Le contrôleur qui produit la variable $lesTypesCuisine est detailResto.php, le nom de la fonction qui lui est liée est « getTypesCuisineByIdR » qu'on peut trouver dans le fichier modèle « bd.typecuisine.inc.php ».

2.2. Schématiser le contenu de la variable $lesTypesCuisine

0	idTC	1
	libelleTC	sud ouest
1	idTC	1
	libelleTC	sud ouest
2	idTC	3
	libelleTC	orientale

La contenu de la variable $lesTypesCuisine est composé de : idTC et libelleTC

Note intéressante : Un restaurant peut être lié à plusieurs types de cuisine différentes, celui-ci n'est pas forcément limité à la cuisine « sud ouest » ou « orientale ».

2.3. Combien de types de cuisine sont contenus dans cette variable ?

Étant donné que pas tous les types de cuisine ont un restaurant attribué, pour le moment, il y a 6 types de cuisine contenus dans la variable, malgré les 11 proposés dans la table « typecuisine ».

2.4. Quelle est la syntaxe permettant d'accéder au libelle d'un type de cuisine contenu dans la variable ?
On peut tout simplement faire $lesTypesCuisine[''libelleTC''], ce qu'on peut constater dans la ligne 11 de vueDetailResto, par ailleurs.

Question 3 : vueDetailResto.php - le restaurant

Documents à utiliser
- fichiers fournis en ressources
- annexes 1, 2, 3 et 4

3.1. Quel contrôleur produit la variable $unResto ? Préciser le nom de la fonction et du modèle.
Le contrôleur qui produit la variable $unResto est detailResto.php. $unResto utilise la fonction se nommant getRestoByIdR se trouvant dans le modèle bd.resto.inc.php.

3.2. Schématiser le contenu de la variable $unResto

0	idR	1
	nomR	l'entrepote
	numAdrR	2
	voieAdrR	Rue Maurice Ravel
	cpR	33000
	villeR	Bordeaux
	latitudeDegR	44,7948
	longitudeDegR	-0,58754
	descR	description
	horairesR	<table>…</table>
1	idR	2
	nomR	le bar du charcutier

	numAdrR	30
	voieAdrR	Rue Parlement Sainte-Catherine
	cpR	33000
	villeR	Bordeaux
	latitudeDegR	NULL
	longitudeDegR	NULL
	descR	description
	horairesR	<table>…</table>
2	idR	3
	nomR	Sapporo
	numAdrR	33
	voieAdrR	Rue Saint Rémi
	cpR	33000
	villeR	Bordeaux
	latitudeDegR	NULL
	longitudeDegR	NULL
	descR	Le Sapporo propose à ses...
	horairesR	<table>…</table>

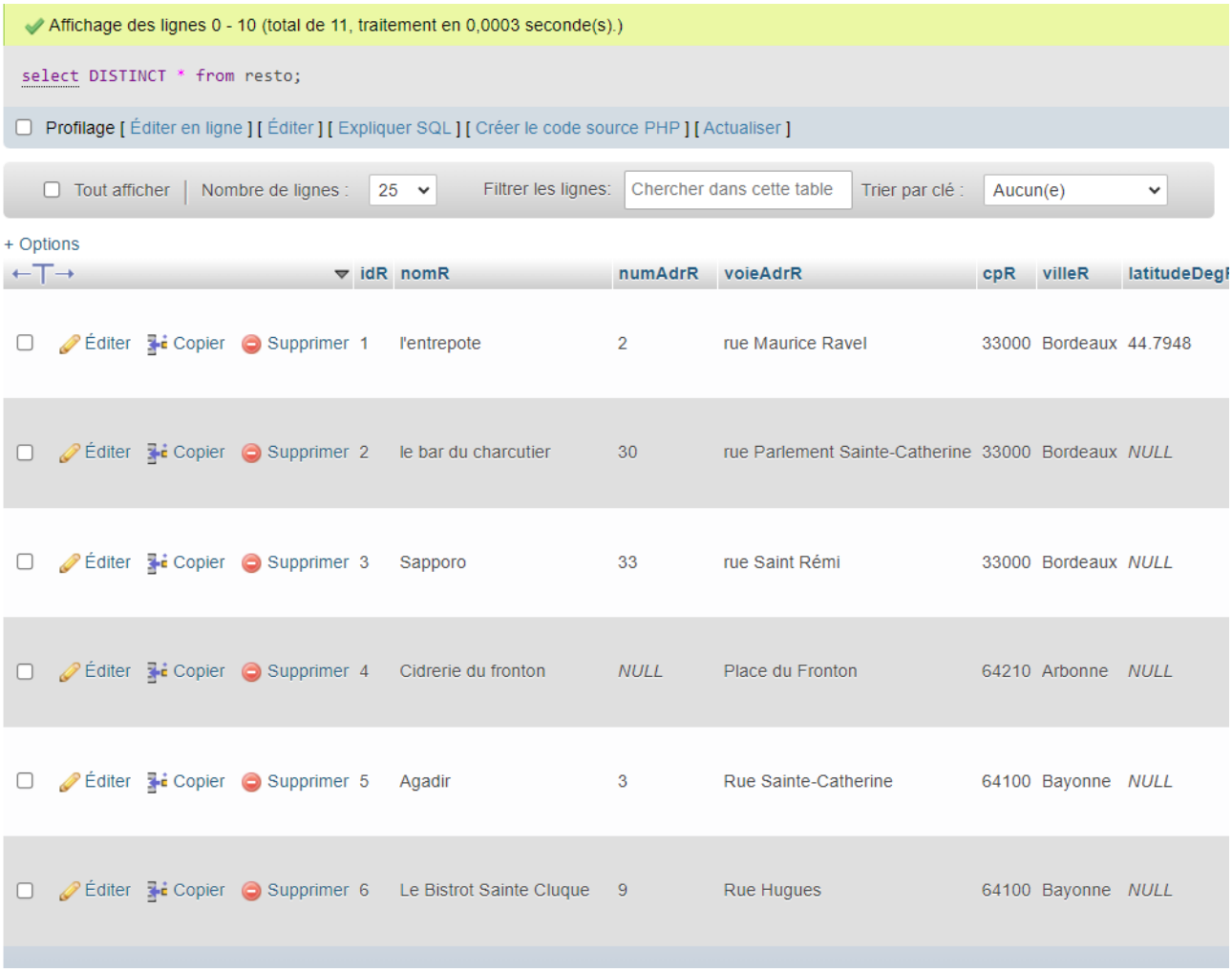
La contenu de la variable $unResto est composé de : idR, nomR, numAdR, voieAdrR, cpR, villeR, latitudeDegR, longitudeDegR, descR, horairesR. Certaines informations peuvent être inconnues, comme latitudeDegR ou longitudeDegR

3.3. Quelles sections de code dans la vue utilisent cette variable ?

Les sections de code dans la vue « vueDetailResto.php » utilisant la variable sont :

$unResto[‘’nomR’’];
$unResto[‘’descR’’];
$unResto[‘’numAdrR’’];
$unResto[‘’voieAdrR’’];
$unResto[‘’cpR’’];
$unResto[‘’villeR’’];
$unResto[‘’horaireR’’];

3.4. Combien de restaurants sont contenus dans cette variable ?



Le même nombre de restaurants affichés en page d'accueil, donc la variable $unResto contient un total de 11 restaurants.

3.5. Quelle est la syntaxe permettant d'accéder au nom d'un restaurant contenu dans la variable ?
La syntaxe permettant d'accéder au nom d'un restaurant contenu dans la variable est la suivante : $unResto[''nomR'']

nomR contenant bien évidemment le nom de ladite donnée nous intéressant (le nom).

Synthèse

La vue utilise des données créées ou récupérées par le contrôleur. Certaines fonctions du modèles sont parfois appelées directement dans la vue. Ces données sont utilisées de manière brute, sans traitement.
Si un traitement est à faire, il doit être déporté dans le contrôleur.

Un code propre devrait toujours récupérer les données dans le contrôleur, et parfois construire des variables complexes. Faire appel à des fonctions du modèle dans les vue est une manière de simplifier le travail, mais si trop de code est présent dans la vue, il est probable que sa conception est à revoir.
Le seul code PHP qui devrait être présent dans la vue devrait être directement lié à la vue. Ainsi on ne doit pas trouver de code permettant de gérer la logique applicative, ni d'accès direct aux données de la base.

Le choix des langages HTML et PHP dans la vue n'est qu'un exemple. Il est possible d'utiliser des frameworks de vue ou des moteurs de templates pour faciliter l'écriture du code : Twig par exemple.

D'une manière générale, une vue contient essentiellement du code HTML qui intègre du code PHP (et pas l'inverse). C'est la manière de procéder la plus propre, on évite ainsi :

- les multiples appels à la fonction echo,
- les guillemets parfois complexes à gérer lors des concaténations, etc.

Pour pouvoir utiliser les variables créées dans le contrôleur, il faut connaître leur structure. Deux possibilités dans ce cas :

- la documentation du modèle et des fonctions est telle qu'il est facile de comprendre comment accéder aux informations ;
- la documentation n'est pas assez claire ou inexistante. Dans ce cas PHP nous donne des outils de debuggage (print_r, echo, etc) permettant de connaître la structure et le contenu des variables.

Partie B : Adaptation de vues

Question 4 : Adaptation du menu général

Documents à utiliser
- fichiers fournis en ressources
- annexe 9

Lors de la connexion de l'utilisateur sur le site, le lien connexion présent dans le menu général doit être modifié pour faire appel à la consultation du profil utilisateur à la place du formulaire de connexion.
Le menu est présent dans la vue entete.html.php. Il est composé d'une liste HTML (voir annexe 9)

4.1. Rappeler quelle fonction du modèle authentification.inc.php permet de connaître l'état de connexion du visiteur du site.
La fonction du modèle permettant de connaître l'état de connexion du visiteur du site est isLoggedOn()

4.2. Quel type de donnée est renvoyé par cette fonction ? Donner des exemples.
Le type de donnée qui est renvoyé par cette fonction se trouve dans $ret qui est un booléen (true ou false). Il peut être à false si l'utilisateur n'est simplement pas connecté (if (!isset($_SESSION)), ou true s'il est bien connecté et que la variable $_SESSION gérée avec la fonction getUtilisateurByMailU s'est correctement exécutée.

Lorsque l'utilisateur est connecté sur le site, l'option "Connexion" du menu doit être remplacée par l'option "Mon Profil" qui pourra être produite par le code ci-dessous :

```
<li>
      <a href="./?action=profil">
            <img src="images/profil.png" alt="loupe" />
            Mon Profil
      </a>
</li>
```

4.3. Quel élément de la liste du menu général est affiché lorsque l'utilisateur n'est pas connecté (comportement par défaut) ?
L'élément de la liste du menu général qui est affiché par défaut quand l'utilisateur n'est pas connecté est la liste « Connexion » qui permet à l'utilisateur de justement, se connecter (il sera remplacé par « Mon profil » quand il sera connecté ».

4.4. Adapter le code de la vue entete.html.php en utilisant la fonction trouvée à la question 1 afin de faire en sorte que l'option "Mon Profil" soit affichée lorsque l'utilisateur est connecté. L'option "Connexion" est affichée lorsque aucun utilisateur n'est connecté.

```php
<nav>
    <?php
        if(isLoggedOn() == true) {
            echo '
        <ul id="menuGeneral">
                <li><a href="./?action=accueil">Accueil</a></li>
                <li><a href="./?action=recherche"><img src="images/rechercher.png" alt="loupe" />Recherche</a></li>
                <li></li>
                <li id="logo"><a href="./?action=accueil"><img src="images/logoBarre.png" alt="logo" /></a></li>
                <li></li>
                <li><a href="./?action=cgu">CGU</a></li>
                <li>
                    <a href="./?action=profil">
                        <img src="images/profil.png" alt="loupe" />
                        Mon Profil
                    </a>
                </li>
            </ul>
            ';
        } else {
            echo '
        <ul id="menuGeneral">
                <li><a href="./?action=accueil">Accueil</a></li>
                <li><a href="./?action=recherche"><img src="images/rechercher.png" alt="loupe" />Recherche</a></li>
                <li></li>
                <li id="logo"><a href="./?action=accueil"><img src="images/logoBarre.png" alt="logo" /></a></li>
                <li></li>
                <li><a href="./?action=cgu">CGU</a></li>
                <li><a href="./?action=connexion"><img src="images/profil.png" alt="loupe" />Connexion</a></li>
            </ul>
        ';
        }
    ?>
</nav>
```

On affiche deux menus différents selon l'état de connexion de l'utilisateur, s'il est pas connecté, ce sera le menu avec l'élément de connexion. S'il est connecté, il sera remplacé par le profil.

Remarques :
- on observe que l'URL pointée par le lien ainsi que le texte affiché sont différents dans les deux cas ;
- les fonctions du modèle authentification.inc.php sont accessibles depuis n'importe quelle page du site, ce fichier étant inclus dans index.php. Il n'est donc pas nécessaire d'ajouter une inclusion dans la vue.

Question 5 : Profil utilisateur - affichage des types de cuisine préférés par l'utilisateur

Documents à utiliser
- fichiers fournis en ressources
- annexes 10 et 12

Pour tester le code que vous allez mettre en place vous devez être connecté sur le site et consulter la page "Mon Profil" accessible en cliquant sur le lien dans le menu général.
Le contrôleur associé à cette fonctionnalité est `monProfil.php`.

L'instruction suivante crée la variable `$mesTypeCuisineAimes` :
```
$mesTypeCuisineAimes = getTypesCuisinePreferesByMailU($mailU);
```

5.1. Afficher la variable `$mesTypeCuisineAimes` dans le contrôleur à l'aide de la fonction `print_r()` puis schématiser sa structure.

```
Array ( [0] => Array ( [idTC] => 1 [libelleTC] => sud ouest ) [1] => Array
( [idTC] => 2 [libelleTC] => japonaise ) [2] => Array ( [idTC] => 3
[libelleTC] => orientale ) [3] => Array ( [idTC] => 10 [libelleTC] =>
viande ) [4] => Array ( [idTC] => 11 [libelleTC] => grillade ) )
```

0	idTC	1
	libelleTC	sud ouest
1	idTC	2
	libelleTC	japonaise
2	idTC	3
	libelleTC	orientale
3	idTC	10
	libelleTC	viande
4	idTC	11
	libelleTC	grillade

5.2. Quelle syntaxe permet d'atteindre le libelle d'un type de cuisine contenu dans la variable `$mesTypeCuisineAimes` ?
La syntaxe permettant d'atteindre le libellé d'un type de cuisine dans la variable en question est : $mesTypeCuisineAimes[''libelleTC''']

5.3. Repérer dans la vue le code HTML permettant d'afficher les types de cuisine.

```
les types de cuisine que j'aime :
<ul id="tagFood">
    <li class="tag"><span class="tag">#</span>sud ouest</li>
    <li class="tag"><span class="tag">#</span>viande</li>
    <li class="tag"><span class="tag">#</span>grillade</li>
</ul>
```

C'est dans cette partie là que les types de cuisine appréciés de l'utilisateur sont affichées.

5.4. Quelle partie de code est répétée ?

C'est la partie qui commence par <li class=''tag''><span class=''tag''># te qui finit par

5.5. Écrire le code permettant de parcourir la variable $mesTypeCuisineAimes et d'afficher pour chaque type le libellé associé.

```
les types de cuisine que j'aime :
<ul id="tagFood">
    <?php for ($j = 0; $j < count($mesTypeCuisineAimes); $j++) { ?>
        <li class="tag"><span class="tag">#</span><?= $mesTypeCuisineAimes[$j]["libelleTC"] ?></li>
    <?php } ?>
</ul>
```

Nickel !

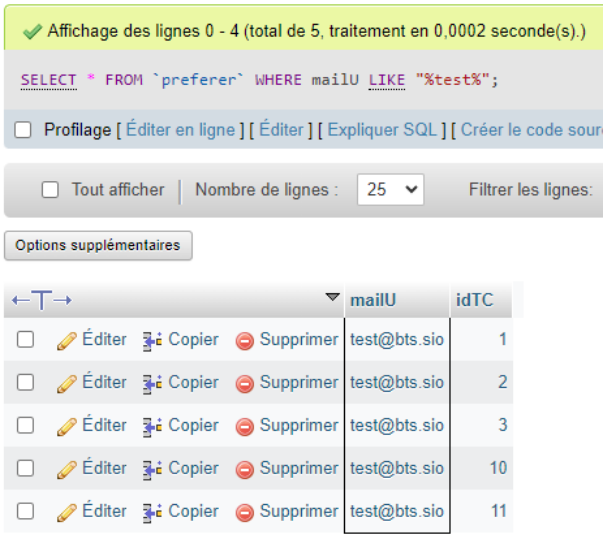
5.6. Modifier le code produit à la question précédente afin de faire en sorte que chaque type de cuisine soit affiché en respectant en respectant les balises HTML trouvées à la question 5.3. Intégrer ce code à la vue `vueMonProfil.php`.
Le code HTML généré visible depuis le navigateur doit se rapprocher de celui de l'annexe 10.

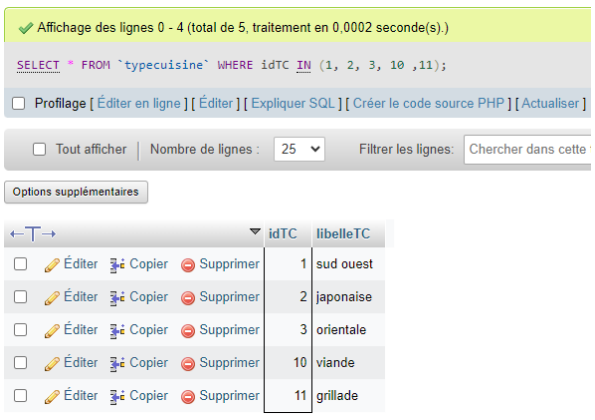
```
les types de cuisine que j'aime :
<ul id="tagFood">
    <?php for ($j = 0; $j < count($mesTypeCuisineAimes); $j++) { ?>
        <li class="tag"><span class="tag">#</span><?= $mesTypeCuisineAimes[$j]["libelleTC"] ?></li>
    <?php } ?>
</ul>
```

Voici le code source qui permettra de parcourir les types de cuisine appréciés du profil en question (le même que dans ma question 5.5, ça n'a pas changé).

les types de cuisine que j'aime : #sud ouest #japonaise #orientale #viande #grillade

Voici les types de cuisines appréciés de l'utilisateur testeur SIO.





Je vérifie via diverses commandes SQL si les valeurs retournées sont cohérentes et… c'est le cas !

Question 6 : Profil utilisateur - affichage des restaurants aimés par l'utilisateur

Documents à utiliser
- fichiers fournis en ressources
- annexes 11 et 12

Pour tester le code que vous allez mettre en place vous devez être connecté sur le site et consulter la page "Mon Profil" accessible en cliquant sur le lien éponyme dans le menu général.
Le contrôleur associé à cette fonctionnalité est monProfil.php.

L'instruction suivante crée la variable `$mesRestosAimes` :
```
$mesRestosAimes = getRestosAimesByMailU($mailU);
```

L'objectif de cette vue est d'afficher la liste des restaurants aimés par l'utilisateur. Cette liste apparaît sous forme de liens permettant d'accéder à aux détails de chaque restaurant.

Dans le code suivant figurent des informations provenant de la variable $mesRestosAimes.
```
<a href="./?action=detail&idR=4">Cidrerie du fronton</a>
```

On retrouve ainsi l'identifiant et le nom du restaurant.

L'identifiant du restaurant est passé en paramètre pour le contrôleur qui gère la consultation des données du restaurant. Chaque lien doit respecter cette structure.

6.1. En procédant comme pour l'exercice précédent, expliquer comment accéder dans la variable `$mesRestosAimes` à l'identifiant et au nom de chaque restaurant.

Pour accéder dans la variable $mesRestosAimes ainsi que les deux données qui nous intéressent, il faut :
- Pour l'identifiant du restaurant : $mesRestosAimes[''idR'']
- Pour le nom du restaurant : $mesRestosAimes[''nomR'']

6.2. Écrire le code permettant de parcourir la variable `$mesRestosAimes` et d'afficher pour chaque restaurant le nom associé.

```
les restaurants que j'aime : <br>
    <?php for ($i = 0; $i < count($mesRestosAimes); $i++) { ?>
        <?php echo '<a href="./?action=detail&idR=' .  $mesRestosAimes[$i]["idR"] . '">' . $mesRestosAimes[$i]["nomR"] . '</a><br>'; ?>
    <?php } ?>

<hr>
```

Autant le dire, il faut bien plisser les yeux et se concentrer pour poser les guillemets aux bons endroits.

6.3. Adapter le code de la vue vueMonProfil.php afin de remplacer l'affichage statique des trois restaurants par un affichage dynamique en fonction des restaurants contenus dans la variable `$mesRestosAimes`.

vueMonProfil.php M | detailResto.php | entete.html.php

r3st0 TP3 EN COURS > site > vue > vueMonProfil.php > ul#tagFood

```
1
2   <h1>Mon profil</h1>
3
4   Mon adresse électronique : <?= $util["mailU"] ?> <br />
5   Mon pseudo : <?= $util["pseudoU"] ?> <br />
6
7   <hr>
8
9   les restaurants que j'aime : <br>
10      <?php for ($i = 0; $i < count($mesRestosAimes); $i++) { ?>
11          <?php echo '<a href="./?action=detail&idR=' .  $mesRestosAimes[$i]["idR"] . '">' . $mesRestosAimes[$i]["nomR"] . '</a><br>'; ?>
12      <?php } ?>
13
14  <hr>
15
16  les types de cuisine que j'aime :
17  <ul id="tagFood">
18      <?php for ($j = 0; $j < count($mesTypeCuisineAimes); $j++) { ?>
19          <li class="tag"><span class="tag">#</span><?= $mesTypeCuisineAimes[$j]["libelleTC"] ?></li>
20      <?php } ?>
21  </ul>
22  <hr>
23  <a href="./?action=deconnexion">se deconnecter</a>
```

Le code complet (les lignes qui nous intéressent dans le cadre de la partie 6 se situent entre les lignes 10 à 12).

Mon profil

Mon adresse électronique : test@bts.sio
Mon pseudo : testeur SIO

---

les restaurants que j'aime :
Cidrerie du fronton
Agadir
Le Bistrot Sainte Cluque
la petite auberge
La table de POTTOKA

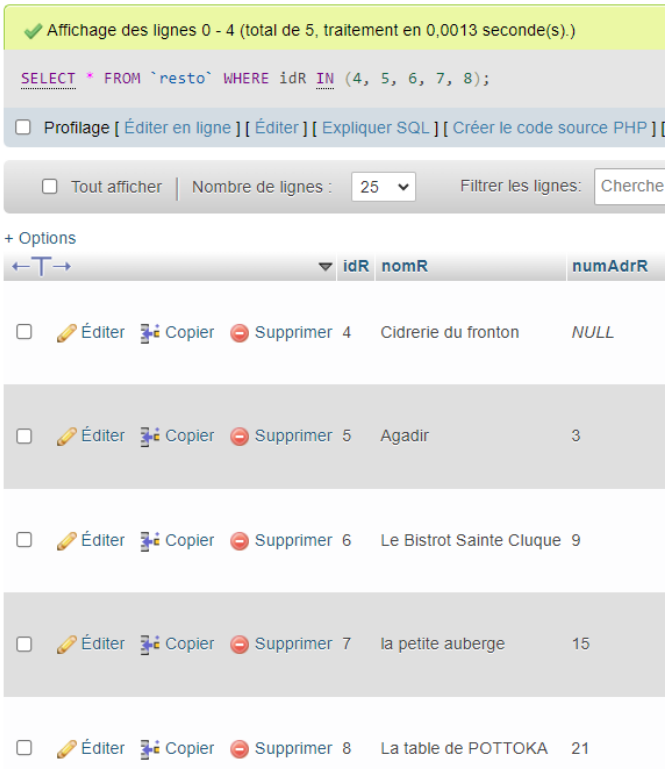
---

les types de cuisine que j'aime : #sud ouest #japonaise #orientale #viande #grillade

---

se deconnecter

Voici le visuel final.



Une fois encore, après vérification via des commandes SQL, l'affichage des restaurants préférés de testeur SIO est cohérent avec la base de données.

Annexe 1 - vueDetailResto.php

```
<h1><?= $unResto['nomR']; ?>
</h1>

<span id="note">
</span>
<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>
<p id="principal">
    <?php if (count($lesPhotos) > 0) { ?>
        <img src="photos/<?= $lesPhotos[0]["cheminP"] ?>" alt="photo du
restaurant" />
    <?php } ?>
    <br />
    <?= $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>
         <img class="galerie" src="photos/<?= $lesPhotos[$i]["cheminP"] ?>"
               alt="" />
      </li>
    <?php } ?>

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

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

<ul id="critiques">
</ul>
```

Annexe 2 - controleur detailResto.php

```
<?php

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    $racine = "..";
}
include_once "$racine/modele/bd.resto.inc.php";
include_once "$racine/modele/bd.typecuisine.inc.php";
include_once "$racine/modele/bd.photo.inc.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
$unResto = getRestoByIdR($idR);
$lesTypesCuisine = getTypesCuisineByIdR($idR);
$lesPhotos = getPhotosByIdR($idR);


// traitement si necessaire des donnees recuperees
;

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

Annexe 3 - extrait du modele bd.resto.inc.php

```
<?php
include_once "bd.inc.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;
}
```

```
...


if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');
...

    echo "getRestoByIdR(idR) : \n";
    print_r(getRestoByIdR(1));

...
}
?>
```

Annexe 4 - extrait du résultat d'exécution de bd.resto.inc.php

```
getRestoByIdR(idR) :
Array
(
    [idR] => 1
    [nomR] => l'entrepote
    [numAdrR] => 2
    [voieAdrR] => rue Maurice Ravel
    [cpR] => 33000
    [villeR] => Bordeaux
    [latitudeDegR] => 44.7948
    [longitudeDegR] => -0.58754
    [descR] => description
    [horairesR] => <table>...</table>
)
```

Annexe 5 - extrait du modele bd.photo.inc.php

```
<?php
include_once "bd.inc.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;
}
```

```
if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');
...
    echo "\n getPhotosByIdR(4) : \n";
    print_r(getPhotosByIdR(4));
...
}
?>
```

Annexe 6 - extrait du résultat d'exécution de bd.photo.inc.php

```
getPhotosByIdR(4) :
Array
(
    [0] => Array
        (
            [idP] => 6
            [cheminP] => cidrerieDuFronton.jpg
            [idR] => 4
        )

    [1] => Array
        (
            [idP] => 14
            [cheminP] => cidrerieDuFronton2.jpg
            [idR] => 4
        )

    [2] => Array
        (
            [idP] => 15
            [cheminP] => cidrerieDuFronton3.jpg
            [idR] => 4
        )
)
```

Annexe 7 - extrait du modele bd.typecuisine.inc.php

```
<?php
include_once "bd.inc.php";
function getTypesCuisineByIdR($idR){
    $resultat = array();
    try {
        $cnx = connexionPDO();
        $req = $cnx->prepare("select typeCuisine.* from typeCuisine,proposer
where typeCuisine.idTC = proposer.idTC and proposer.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;
}

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');
...
    echo "getTypesCuisineByIdR(idR) : \n";
    print_r(getTypesCuisineByIdR(4));
...
}
?>
```

Annexe 8 - extrait du résultat d'exécution de bd.typecuisine.inc.php

```
getTypesCuisineByIdR(idR) :
Array
(
    [0] => Array
        (
            [idTC] => 1
            [libelleTC] => sud ouest
        )

    [1] => Array
        (
            [idTC] => 8
            [libelleTC] => sandwich
        )

    [2] => Array
        (
```

```
            [idTC] => 11
            [libelleTC] => grillade
        )
)
```

Annexe 9 - extrait de la vue entete.html.php - menu général

```
<ul id="menuGeneral">
   <li><a href="./?action=accueil">Accueil</a></li>
   <li><a href="./?action=recherche"><img src="images/rechercher.png"
alt="loupe" />Recherche</a></li>
   <li></li>
   <li id="logo"><a href="./?action=accueil"><img src="images/logoBarre.png"
alt="logo" /></a></li>
   <li></li>
   <li><a href="./?action=cgu">CGU</a></li>
   <li><a href="./?action=connexion"><img src="images/profil.png"
alt="loupe" />Connexion</a></li>
</ul>
```

Annexe 10 - extrait code code généré affichant les types de cuisine préférés par l'utilisateur

```
<ul id="tagFood">
    <li class="tag"><span class="tag">#</span>sud ouest</li>
    <li class="tag"><span class="tag">#</span>viande</li>
    <li class="tag"><span class="tag">#</span>grillade</li>
</ul>
```

Annexe 11 - extrait code code généré affichant les restaurants aimés par l'utilisateur

```
les restaurants que j'aime : <br />
    <a href="./?action=detail&idR=4">Cidrerie du fronton</a><br />
    <a href="./?action=detail&idR=6">Le Bistrot Sainte Cluque</a><br />
    <a href="./?action=detail&idR=8">La table de POTTOKA</a><br />
```

Annexe 12 - vue à modifier : vueMonProfil.php

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

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

<hr>

les restaurants que j'aime : <br>
    <a href="./?action=detail&idR=4">Cidrerie du fronton</a><br>
    <a href="./?action=detail&idR=6">Le Bistrot Sainte Cluque</a><br>
    <a href="./?action=detail&idR=8">La table de POTTOKA</a><br>
<hr>
```

```
les types de cuisine que j'aime :
<ul id="tagFood">
    <li class="tag"><span class="tag">#</span>sud ouest</li>
    <li class="tag"><span class="tag">#</span>viande</li>
    <li class="tag"><span class="tag">#</span>grillade</li>
</ul>
<hr>
<a href="./?action=deconnexion">se deconnecter</a>
```

Architecture MVC en PHP - Modèle

Fonctionnement du modèle dans le patron de conception MVC

Remarques importantes à destination de l'enseignant
Les sections de tests contenues dans les contrôleurs et dans les modèles permettant de tester de manière indépendantes chaque module ne fonctionnent que sous environnement UNIX.

Rappel du contexte
R3st0.fr est un site web de critique de restaurant. À l'image des sites de ce type il a pour vocation le recensement des avis des consommateurs et la diffusion de ces avis aux visiteurs.

Ce site web est développé en PHP en suivant le patron de conception "modèle vue contrôleur" (pattern MVC). L'architecture retenue pour ce projet permet d'appréhender la programmation web d'une manière structurée.

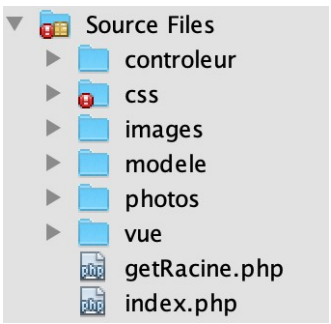
L'objectif de ce document est d'analyser le fonctionnement du composant modèle dans l'architecture MVC puis de compléter et mettre à jour des fonctions du modèle.
La base de données utilisée dans ce projet permet aux utilisateurs d'aimer des restaurants comme on pourrait les mettre en favoris, et de les critiquer. Le principe d'un site web communautaire est de mettre à disposition des utilisateurs des fonctionnalités leur permettant de s'exprimer.
Chaque restaurant peut être évalué à l'aide d'une note allant de 1 à 5. De même un commentaire peut être laissé sur sa fiche pour argumenter la note qui a été donnée.

Ressources à utiliser
- Dossier "base de données" : fichier base.sql contenant les tables et les données de la base utilisée par l'application web étudiée.
- Dossier site : arborescence et fichiers de l'application web.

Après avoir créé la base de données (encodage utf8mb4) et importé le fichier `base.sql` , créer un nouveau projet PHP appelé SI6MVCTP1 dans Netbeans. Paramétrer le projet pour que celui-ci soit téléchargé sur le serveur lors de l'exécution.

Dans le gestionnaire de fichiers, supprimer le fichier `index.php` créé automatiquement, et remplacer le contenu du dossier "Source Files" par le contenu du dossier "site" fourni.

L'arborescence devrait être celle-ci :



Avant de commencer le TP, le site doit être paramétré pour qu'il utilise votre base de données. Dans le script `modele/bd.inc.php`, modifier les lignes suivantes afin d'indiquer les bonnes informations :

```
$login = "votre login mariaDB";
$mdp = "votre mot de passe mariaDB";
$bd = "nom de votre base de données";
$serveur = "adresse IP ou nom du serveur de base de données";
```

Afin de mieux voir l'objectif du site, et les différentes fonctionnalités que vous devrez mettre en place tout au long de ces TP, le site final est consultable à l'adresse fournie par votre professeur.

A - Analyse du fonctionnement du modèle

Question 1 - Analyse de la fonction getRestoByIdR() du modèle bd.resto.inc.php

Documents à utiliser
- fichiers fournis en ressources
- annexes 1, 2, 4, 5 et 7

Lors du test de cette fonction en annexe 4, la valeur passée en paramètre est utilisée dans la requête SQL.
1.1. Indiquer le nom du paramètre de la fonction et sa valeur dans l'exemple d'utilisation de l'annexe 4.

Le nom du paramètre de la fonction (qui est alors getRestoByIdR()) est $idR, la valeur passée dans l'exemple d'utilisation de l'annexe 4 est 1.

L'instruction prepare() de la fonction contient la requête, ainsi qu'un tag '`:idR`' à l'emplacement où devrait se situer cette valeur.
```
$req = $cnx->prepare("select * from resto where idR=:idR");
```

Le rôle du tag est d'indiquer qu'une valeur doit être utilisée à cet emplacement. Lors de l'appel à la fonction `bindValue()`, le tag va être associé à une valeur qui peut être :

- soit une variable,
- soit une valeur littérale.

Lors de l'instruction suivante, le tag contenu dans la requête est associé à la variable $idR.
```
$req->bindValue(':idR', $idR, PDO::PARAM_INT);
```

Le contenu de la variable $idR est aussi vérifié à l'aide de la constante PDO::PARAM_INT. Le rôle de cette constante est d'indiquer que la variable $idR est un entier. Dans le cas contraire la requête ne pourra pas être exécutée.

Si il y avait différents tags dans la requête, on ferait plusieurs fois appel à la fonction bindValue() en précisant pour chaque tag la valeur associée.

1.2. À l'aide des annexes, trouver la requête exécutée par la fonction getRestoByIdR().

La requête exécutée par la fonction getRestoByIdR() est :
SELECT * FROM resto WHERE idR=:idR;

Lorsque l'on exécute cette fonction dans l'interpréteur SQL (onglet SQL de phpmyadmin) on obtient le résultat mentionné en annexe 5.

1.3. Expliquer pourquoi la requête SQL ne retourne qu'une seule ligne.

La requête SQL en question ne retourne qu'une seule ligne car idR est la clé primaire de la table resto, ça veut dire qu'elle permet de rendre unique chaque ligne et donc, de bien différencier

chaque restaurant de l'autre.
Imaginons que deux restaurants portent le même nom dans la base de données, on pourra tout de même aisément les identifier grâce à la clé unique (l'une pourrait avoir un idR = 11 et l'autre idR = 37).
Chaque restaurant ayant sa propre idR, et la requête ne pouvant recueillir qu'un seul idR, cette requête SQL ne peut retourner qu'uniquement une ligne.

Dans notre base de données, il suffit de se rendre dans la table « resto » pour se rendre compte que s'il y a quelque chose de certain qui différencie chaque ligne, c'est bien le idR (qui est en plus « auto-incrémental, ce qui fait que si on rajoute un restaurant, il aura la même valeur + 1 de la ligne précédente).

Le résultat de la requête SQL doit être transmis au programme PHP dans un format de données qu'il est capable de traiter. Le système de gestion de base de données relationnelles (SGBDR : MariaDB, MySQL, Postgres etc.) traite les données dans un format qui lui est propre.
Lorsque l'on exécute une requête SQL, le SGBDR renvoie un jeu de données - un curseur - où les données sont organisées en lignes et en colonnes. Chaque colonne possède un nom provenant des propriété d'une table et les lignes correspondent aux occurrences des tables.

Par exemple pour la requête
```
select idR, nomR, numAdrR, voieAdrR, cpR, villeR from resto
```
on obtient le résultat (curseur) suivant. Dans le code PHP, le curseur se trouve dans la variable $req.

idR	nomR	numAdrR	voieAdrR	cpR	villeR
1	l'entrepote	2	rue Maurice Ravel	33000	Bordeaux
2	le bar du charcutier	30	rue Parlement Sainte-Catherine	33000	Bordeaux
3	Sapporo	33	rue Saint Rémi	33000	Bordeaux

Ce curseur contient trois lignes et 6 colonnes.

Un curseur est lu ligne par ligne. C'est le rôle de l'instruction fetch().

L'instruction suivante permet de placer le pointeur de lecture du curseur sur le 1er élément du curseur, et de renvoyer dans résultat le contenu de cette ligne sous la forme d'un tableau associatif.
```
$resultat = $req->fetch(PDO::FETCH_ASSOC);
```

Ce tableau associatif est alors retourné à la fin de la fonction getRestoById() :
```
return $resultat;
```

Pour rappel, cette fonction est appelée dans le contrôleur detailResto.php et affichée dans la vue vueDetailResto.php étudiée dans le TP précédent.

1.4. Rappeler la syntaxe utilisée pour accéder à un champ d'un tableau associatif.

Pour accéder à un champ dans un tableau associatif, il faut utiliser la syntaxe suivante :
$nom_de_variable[''nom_du_champ''];

1.5. Rappeler comment accéder au nom du restaurant dans la variable retournée par la fonction

getRestoByIdR().

Pour accéder au nom du restaurant dans la variable retournée dans la fonction getRestoByIdR(), il faut utiliser la syntaxe suivante :
$listeRestos['nomR'];

Lorsque le curseur contient plusieurs lignes comme pour la requête au dessus, les lignes doivent être lues une à une. Lorsque l'instruction fetch() est appelée une deuxième fois, le pointeur de lecture passe à l'enregistrement suivant.
L'instruction fetch() est appelée autant de fois que nécessaire. La valeur retournée à chaque appel est :

- soit la ligne pointée dans le curseur,
- soit la valeur false indiquant qu'on est arrivé en fin de curseur.

idR	nomR	numAdrR	voieAdrR	cpR	villeR
1	l'entrepote	2	rue Maurice Ravel	33000	Bordeaux
2	le bar du charcutier	30	rue Parlement Sainte-Catherine	33000	Bordeaux
3	Sapporo	33	rue Saint Rémi	33000	Bordeaux

Parcours d'un curseur à l'aide de la fonction fetch()

Pour la lecture de trois lignes dans le curseur, la boucle de parcours fait appel quatre fois à fetch(). La 4ème tentative de lecture n'étant pas possible ici, la valeur false est retournée.
On peut donc utiliser la valeur retournée par fetch() pour savoir si la boucle de lecture doit continuer ou s'arrêter.

Le code de la fonction `getRestosByNomR()` contient les instructions suivantes :

```
$ligne = $req->fetch(PDO::FETCH_ASSOC) ;
while ($ligne) {
      $resultat[] = $ligne;
      $ligne = $req->fetch(PDO::FETCH_ASSOC) ;
}
```

Les appels successifs à la fonction `fetch()` retournent dans `$ligne` un tableau associatif.
Au dernier appel, lorsque la fin du curseur a été atteinte, la valeur `false` est placée dans `$ligne`, la condition d'entrée dans la boucle n'est plus satisfaite, et la boucle s'arrête.

Question 2 - Analyse de la fonction getRestosByNomR()

Documents à utiliser
- fichiers fournis en ressources
- annexes 2, 4, 6 et 7

La fonction getRestosByNomR() prend en paramètre une chaîne de caractère. Cette chaîne est utilisée dans la requête SQL exécutée par cette fonction.

2.1. Quelle requête SQL est envoyée à la fonction prepare() dans getRestosByNomR() ?

La requête SQL envoyée à la fonction prepare() dans getRestosByNomR() est la suivante :
SELECT * FROM resto WHERE nomR LIKE :nomR;

Le résultat de l'exécution de la fonction getRestosByNomR() présentée en annexe 4 illustre les données récupérées lorsque l'on envoie 'charcut' comme paramètre à la fonction.

2.2. Quelle requête SQL est réellement exécutée après l'appel à bindValue() ?

Oui, cette requête SQL est bien exécutée après l'appel à bindValue, car sinon, on a une erreur qui dit qu'il y a une violation de la syntaxe des données.

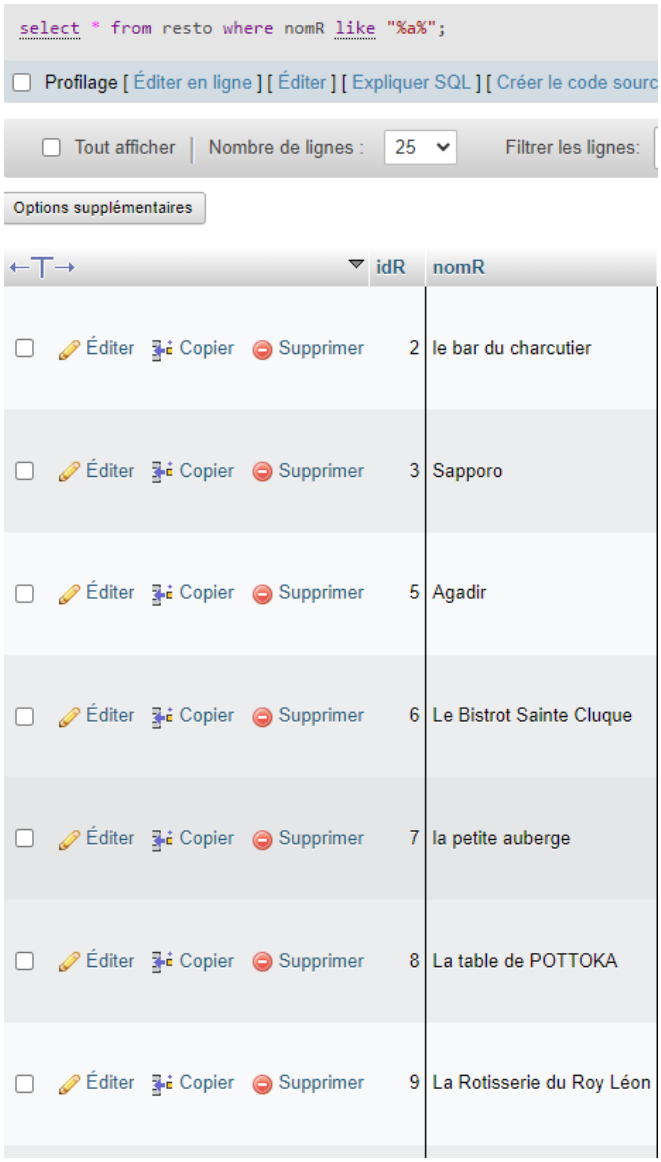
```
getRestosByNomR('charcut') :
Erreur !: SQLSTATE[42000]: Syntax error or access violation: 1064 Erreur de syntaxe près de ':nomR' à la ligne 1
```

L'erreur en question après avoir mis en commentaire bindValue()… bon, eh bien, on va la réactiver de nouveau.

2.3. Cette requête est-elle susceptible de retourner plusieurs lignes ?

Effectivement, c'est possible qu'elle puisse retourner plusieurs lignes. Donnons un exemple.

Au lieu de mettre « charcut », on va juste mettre… « a », ce qui fait que tous les restaurants avec un nom contenant un « a » dedans seront retournés (et potentiellement affichés après traitement)



Nous constatons alors que tous les restaurants avec au moins un « a » dans leur nom sont retournés par la commande liée à la fonction. Pour faire très simple, cette fonction (et la requête SQL) permet de récupérer tous les restaurants comprenant la chaîne de caractères liée au nom.

Pour les plus observateurs, dans la partie 1, « getRestoByIdR » n'a pas de « s » après Resto, alors que dans la requête getRestosByIdR, on observe un « s » après Resto… Ca ne sert pas à grand-chose que je le dise, mais c'est là.

Dans le cas de « charcut », il est normal qu'elle ne retourne qu'une ligne car un seul restaurant référencé contient « charcut » (le bar du charcutier) dans son nom (cette commande peut autant retourner aucune ligne qu'elle peut en retourner une ou plusieurs).

2.4. Parmi les 2 propositions suivantes une seule est vraie, justifier votre proposition à l'aide du code de la fonction présent en annexe 2, de vos connaissances en PHP et en SQL :
a) la variable `$resultat` retournée par la fonction `getRestosByNomR()` est un tableau associatif contenant les informations d'un restaurant.
b) la variable `$resultat` retournée par la fonction `getRestosByNomR()` est un tableau dont chaque case

est un tableau associatif. Cette variable peut contenir les informations sur plusieurs restaurants.

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

La seule bonne réponse dans la fonction getRestosByNomR() est donc la réponse b) pour plusieurs raisons :
- Premièrement, on vient de dire et de prouver que cette fonction peut contenir les informations sur plusieurs restaurants, $resultat a comme but de montrer le résultat qui ne diffère en aucun cas de notre raisonnement de base.
- Deuxièmement, on pourra voir en-dessous que chaque case est un tableau associatif (ce qui veut dire qu'on pourra accéder indépendamment à chaque information après, si on veut juste le nom, la ville ou encore les horaires d'un restaurant).

```
getRestosByNomR('a') :
Array
(
    [0] => Array
        (
            [idR] => 2
            [nomR] => le bar du charcutier
            [numAdrR] => 30
            [voieAdrR] => rue Parlement Sainte-Catherine
            [cpR] => 33000
            [villeR] => Bordeaux
            [latitudeDegR] =>
            [longitudeDegR] =>
            [descR] => description
            [horairesR] => <table>
    <thead>
        <tr>
            <th>Ouverture</th><th>Semaine</th>  <th>Week-end</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td class="label">Midi</td>
            <td class="cell">de 11h45 à 14h30</td>
            <td class="cell">de 11h45 à 15h00</td>
        </tr>
        <tr>
            <td class="label">Soir</td>
            <td class="cell">de 18h45 à 22h30</td>
            <td class="cell">de 18h45 à 1h</td>
        </tr>
        <tr>
            <td class="label">À emporter</td>
            <td class="cell">de 11h30 à 23h</td>
            <td class="cell">de 11h30 à 2h</td>
        </tr>
    </tbody>
</table>
        )

    [1] => Array
        (
            [idR] => 3
            [nomR] => Sapporo
            [numAdrR] => 33
            [voieAdrR] => rue Saint Rémi
            [cpR] => 33000
            [villeR] => Bordeaux
            [latitudeDegR] =>
            [longitudeDegR] =>
            [descR] => Le Sapporo propose à ses clients de délicieux plats typiques japonais.
```

Oui, j'ai encore utilisé « a » pour prouver la véracité de mes propos.

Question 3 - Analyse de la fonction getNoteMoyenneByIdR() du modèle bd.critiquer.inc.php

Documents à utiliser
- fichiers fournis en ressources
- annexe 8

3.1. Expliquer le rôle de la requête ci-dessous présente dans la fonction getNoteMoyenneByIdR()

```
select avg(note) from critiquer where idR=:idR
```

Le rôle de la requête présente dans la fonction getNoteMoyenneByIdR() est de faire la moyenne des avis utilisateurs d'un restaurant (une moyenne pouvant aller de 1 (Très mauvais) à 5 (Excellent)).

3.2. Combien de résultats sont attendus de la requête SQL ?

Seulement une, étant donné qu'on fait une moyenne de toutes les notes. Mes souvenirs de mes cours en CM1 sont plutôt limitées, mais lorsqu'on calcule la moyenne de quelque chose, cela ne

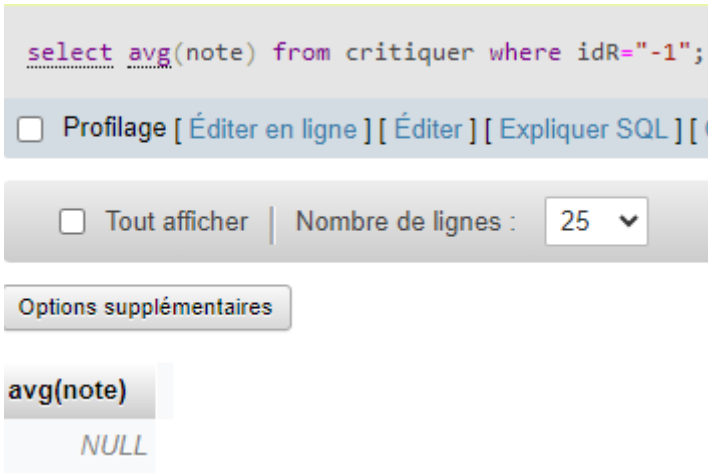
retourne qu'un seul résultat (la moyenne de « quelque chose », la seule et unique donnée nous intéressant).

Exécuter la requête en donnant la valeur 2 à la propriété idR.
(Voir en-dessous)

3.3. Comment est nommée la colonne affichée dans le résultat

La colonne affichant le résultat de la moyenne des notes du restaurant portant l'idR = 2 se nomme avg(note).

Exécuter la requête en donnant une valeur impossible à la propriété idR. Par exemple 0 ou -1.

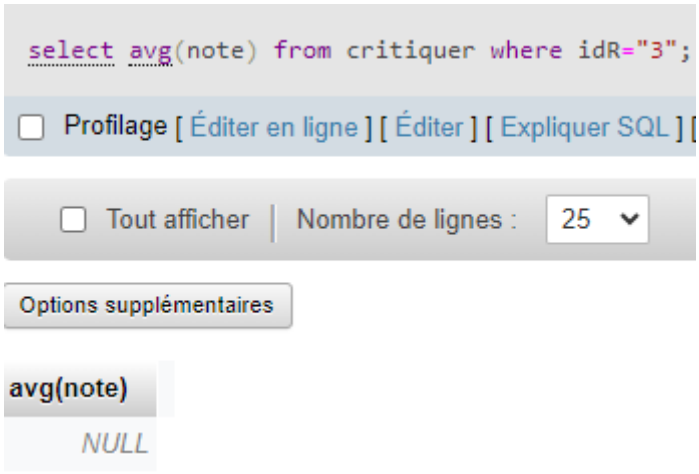


NULL, pas que le restaurant soit très mauvais, juste qu'aucun restaurant a un idR = 0 ou -1. Qui dit aucun restaurant, dit forcément aucun avis, dit aucune moyenne possible.
(Dans le cas contraire, appelez un exorciste dans les plus brefs délais.)

3.4. Quelle valeur est obtenue dans le cas ou le calcul peut être fait ? Quelle valeur est obtenue dans le cas ou le calcul ne peut être fait ?

Donc la valeur qui peut être obtenue dans le cas où le calcul peut être fait est un nombre à virgules (float) pouvant osciller entre 1.0000 et 5.000.

Dans le cas où le calcul ne peut pas être fait (si le restaurant en question « n'existe pas » dans la base de données ou si celui-ci n'a encore aucun avis pour le moment), cela retourne simplement un NULL.



Le restaurant portant l'idR = 3 (Sapporo) n'ayant aucun avis utilisateur pour le moment, on ne peut en aucun cas calculer une moyenne (diviser par 0 est impossible).

3.5. À partir des questions précédentes, et en consultant le code de la fonction indiquer quel résultat sera retourné par la fonction lorsque l'identifiant d'un restaurant passé en paramètre existe ou n'existe pas dans la base de données.

Si l'identifiant d'un restaurant passé en paramètre existe :

Cela retournera un résultat oscillant entre 1.0000 et 5.0000 (similaire à ce qu'on peut voir dans la base de données).

```
 getNoteMoyenneByIdR(1)
3.8000
```

Si l'identifiant d'un restaurant passé n'existe pas :

Cela retournera le résultat de 0 (ce qui diffère avec la base de données phpmyadmin qui renverrait le résultat NULL)

```
 getNoteMoyenneByIdR(0)
0
```

3.6. Expliquer la condition suivante située à la fin de la fonction :

```
    if ($resultat["avg(note)"] != NULL) {
        return $resultat["avg(note)"];
    } else {
        return 0;
    }
```

La condition suivante à la fin de la fonction qui est alors :

```
} else {
        return 0;
```

}

consiste à dire que le résultat, dans le cas où le restaurant n'a pas de note ($resultat[''avg(note)''] == NULL), qu'au lieu d'afficher brutalement un NULL à l'utilisateur, d'afficher un simple « 0 » (ce qui ne veut pas dire que le restaurant est très mauvais, sachant que l'échelle des notes se situent entre 1 et 5) à la place.

Cette condition affiche donc la valeur « 0 » à la place du « NULL » comme il aurait été affiché à la base si cette condition n'avait jamais existé dans le cas où il n'y aurait aucune note (rendant donc le calcul de la moyenne impossible).

Question 4 - insertion de données, fonction addAimer() du modèle bd.aimer.inc.php

Documents à utiliser
- fichiers fournis en ressources
- annexes 9, 10 et 11

Dans la base de données, la table aimer a pour rôle d'indiquer quel utilisateur aime quels restaurants. Une occurrence dans la table signifie que l'utilisateur aime le restaurant associé. L'absence d'occurrence signifie que l'utilisateur n'aime pas encore le restaurant.

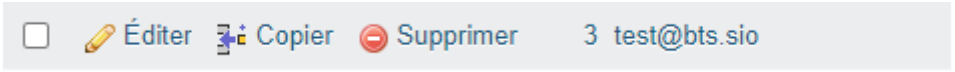
La liste des restaurants aimés par l'utilisateur connecté est accessible dans la section profil.
Pour aimer un restaurant il suffit de cliquer sur l'icône en forme d'étoile dans la fiche descriptive d'un restaurant. Lorsque l'utilisateur n'aime pas encore le restaurant, l'étoile est transparente.
Ces fonctionnalités sont visibles sur la version finale du site:
`http://etu.btssiobayonne.fr/~mathieu.capliez/resto.fr/`

Pour accéder à ces fonctionnalités, il faut évidemment être connecté.

4.1. Expliquer le rôle de la requête présente dans la fonction addAimer().
Le rôle de la requête présente dans la fonction addAimer() est de pouvoir insérer l'adresse email de l'utilisateur connecté ainsi que l'idR du restaurant qu'il veut ajouter dans ses favoris dans la base de données.
Par exemple, j'insère cette ligne de commande SQL dans la table « aimer ».

```
1 insert into aimer (mailU, idR) values("test@bts.sio","3")
```

Cette ligne s'est insérée dans la table avec les données qu'on a spécifié aux bons endroits après values :



test@bts.sio a désormais comme favori le restaurant portant l'idR 3 (Sapporo).

Voyons s'il y a une modification au niveau de l'utilisateur :

les restaurants que j'aime :
Sapporo
Cidrerie du fronton
Agadir
Le Bistrot Sainte Cluque
la petite auberge
La table de POTTOKA

Bingo

4.2. Expliquer pourquoi les deux appels à bindValue() n'utilisent pas la même constante PDO en 3ème paramètre.

Tout simplement car idR est un INT (nombre entier), donc il est normal qu'il y ait PDO::PARAM_INT alors que mailU est une chaîne de caractères (STR) (string), donc pour lui, il lui faut PDO::PARAM_STR.

4.3. A l'aide de l'annexe 11, déterminer quelle valeur est retournée par la fonction execute() en cas de réussite ou d'échec.

En cas de réussite, la valeur retournée est alors TRUE, mais en cas d'échec, la valeur retournée sera alors de FALSE.

4.4. Déduire de la question précédente le type de données retourné par la fonction addAimer().

Donc le type de données retourné par la fonction addAimer() est un booléen.

4.5. Si un utilisateur tente d'aimer un restaurant qu'il aime déjà, quelle sera la valeur retournée par la fonction addAimer() ?

La valeur retournée sera FALSE car on ne peut pas ajouter deux fois exactement la même chose (le même utilisateur aime le même restaurant).

Synthèse

Avant de pouvoir exécuter des requêtes SQL, le script PHP doit se connecter au SGBDR, c'est le rôle de la fonction connexionPDO(). Cette fonction retourne un descripteur d'accès à la base de données : la variable $cnx.
Cette variable est créée par l'instruction :
`new PDO("mysql:host=$serveur;dbname=$bd", $login, $mdp);`

Les paramètres de la fonction `PDO()` permettent de spécifier :

- le driver de base de données à utiliser : `mysql`
- l'adresse ou le nom du serveur : `$serveur`
- le nom de la base de données à utiliser : `$bd`
- le nom d'utilisateur pour se connecter au SGBDR : `$login`
- le mot de passe de l'utilisateur : `$mdp`

La ligne suivante permet de configurer la connexion en mode debug.
`$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);`

Dans ce mode, les erreurs SQL et les erreurs d'accès à la base de données sont interceptées et peuvent être affichées à l'utilisateur.
Ce mode est préférable pendant les phases de développement ou de maintenance. Cette instruction peut être désactivée lorsque le site passe en production.

PDO est une interface d'accès aux bases de données en PHP. PDO fonctionne avec de multiples SGBDR, il peut fonctionner avec MySQL, MariaDB, Postgres etc. L'avantage d'une telle solution est qu'il n'est pas utile de ré-écrire toutes les fonctions d'accès aux données si on change de SGBDR.

PDO fournit un ensemble de fonctions permettant d'exécuter des requêtes SQL dans un programme PHP. On peut ainsi exécuter des requêtes, récupérer leur résultat, connaître le nombre de lignes dans ce résultat, annuler une transaction, récupérer les messages d'erreur SQL etc.

```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;
}
```

Prépare une requête SQL en prévision de son exécution

Attribue une valeur à un tag dans la requête et vérifie le type de données

Exécution de la requête sur le SGBDR et récupération du curseur dans $req

Parcours du curseur, récupération des données sous forme d'un tableau associatif dans $ligne

Fonctions de base d'accès a une base de données

Gestion des exceptions
L'accès à un SGBDR dans une application n'est pas garanti. Il faut donc prévoir des mécanismes en cas de perte de la connexion, d'erreur d'authentification à la base de données, de droits d'accès insuffisants, etc.

Le mécanisme d'exception permet de prendre en compte ces situations.

Ce document n'explique pas le fonctionnement des exceptions, mais illustre comment on peut s'en servir dans une fonction du modèle.
On observe deux blocs bien distincts dans l'exemple au dessus :

- try : bloc de code à exécuter sous condition qu'il n'y ait pas d'erreur PDO ;
- catch : interception du type d'erreur PDOException dans le bloc try. Si une erreur est détectée, le bloc d'instruction du catch est exécuté. Le code présent ici permet d'afficher le message d'erreur puis d'arrêter l'exécution du programme.

Par exemple, si une erreur de type PDOException se produit pendant la préparation de la requête, le bloc d'instruction catch est exécuté.

B - Ajout de fonctionnalités au modèle

Question 5 - mise en place de la fonction delAimer()

Documents à utiliser
- fichiers fournis en ressources
- annexes 9 et 10

Les utilisateurs inscrits sur le site ont la possibilité de supprimer un restaurant de leur liste de restaurants aimés.
Dans la base de données, cette action se traduit par la suppression de l'occurrence associée dans la table aimer.

5.1. Quelle est la clé primaire de la table aimer ?

La clé primaire de la table aimer peut sembler être les deux données, mais l'une d'entre elle est aussi une clé étrangère (mailU) qui est donc clé primaire d'une autre table.
Donc la clé primaire de la table aimer est idR.

MailU est clé primaire, mais surtout clé étrangère d'une autre table. Alors que idR est bien clé primaire de la table « aimer » dont on s'intéresse.

5.2. Rappeler la syntaxe de la requête de suppression en SQL.

La syntaxe de suppression en SQL est :

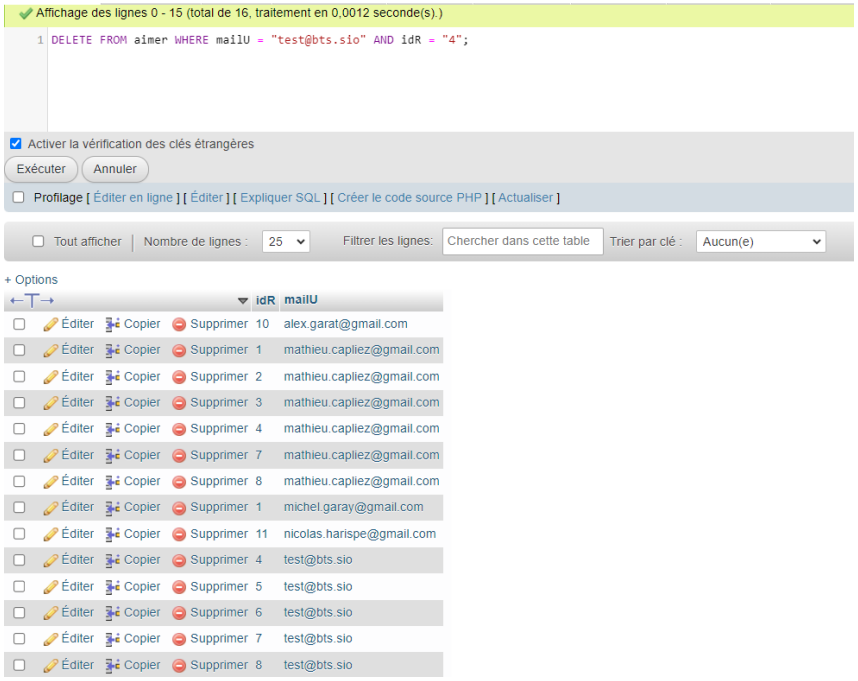
DELETE FROM [table en question] WHERE [‘donnée en question’] = [valeur de la donnée en question]

(Voir : https://sql.sh/cours/delete)

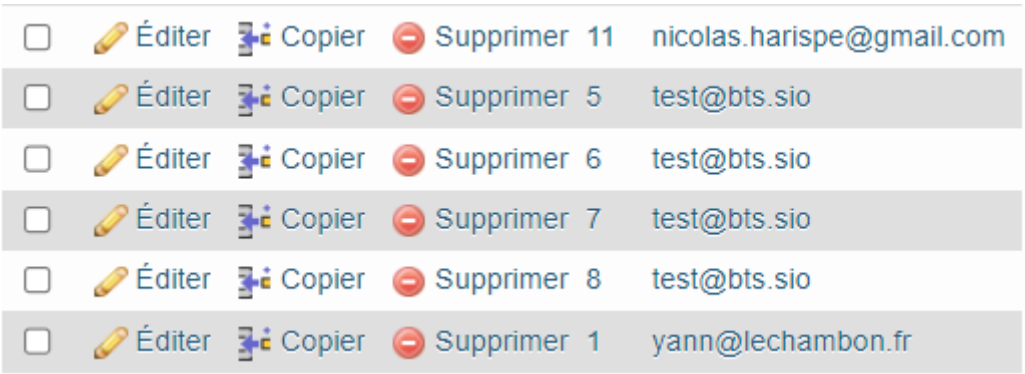
5.3. Écrire la requête SQL permettant de supprimer une occurrence précise de la table aimer.
La commande permettant de supprimer une occurrence précise dans la table aimer est la suivante :

DELETE FROM aimer WHERE mailU = :mailU AND idR = :idR;

5.4. Se connecter à phpmyadmin et tester la requête à l'aide d'un exemple.



Le but de cette ligne est d’effacer la ligne contenant l’idR = 4 et mailU = test@bts.sio



La ligne qui nous intéressait a bien été supprimée.

5.5. Dans quel script du modèle doit être placée la fonction delAimer() ? Justifier.
La fonction delAimer doit être placée dans le fichier modèle « bd.aimer.inc.php », car cette fonction a un impact sur la table aimer.

(Globalement, les fichiers modèles ont pour nom « bd.[nom_de_la_table_affectée].inc.php »)

5.6. Quels sont les paramètres à transmettre à la fonction delAimer() permettant de supprimer précisément une occurrence de la base de données ? En déduire le prototype de la fonction.

Les paramètres à transmettre à la fonction delAimer() sont $mailU et $idR.

Le prototype de la fonction serait alors :

function delAimer($mailU, $idR)

5.7. Écrire le code de la fonction, puis ajouter un appel à cette fonction dans la section de test du script modèle approprié.

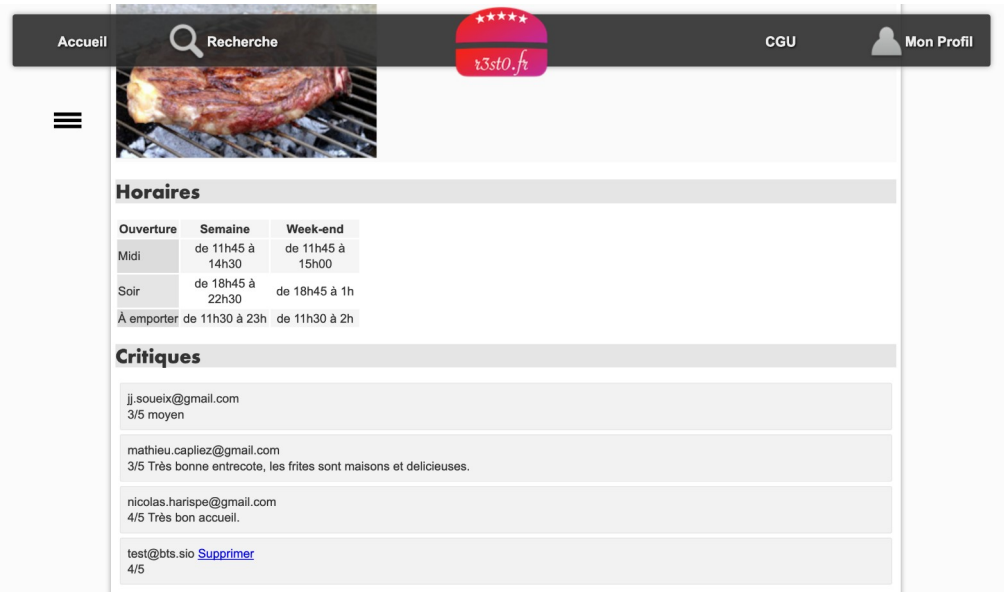
```php
function delAimer($mailU, $idR) {
    try {
        $cnx = connexionPDO();
        $req = $cnx->prepare("DELETE FROM aimer WHERE mailU = :mailU AND idR = :idR");
        $req->bindValue(':idR', $idR, PDO::PARAM_INT);
        $req->bindValue(':mailU', $mailU, PDO::PARAM_STR);

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

```php
if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');


    echo "\n getAimerByMailU(\"mathieu.capliez@gmail.com\") : \n";
    print_r(getAimerByMailU("mathieu.capliez@gmail.com"));

    echo "\n getAimerByIdR(1) : \n";
    print_r(getAimerByIdR(1));

    echo "\n getAimerById(mailU, idR) : \n";
    print_r(getAimerById("mathieu.capliez@gmail.com", 1));

    echo "\n addAimer(\"mathieu.capliez@gmail.com\",7) : \n";
    print_r(addAimer("mathieu.capliez@gmail.com", 7));

    echo "\n delAimer(\"mathieu.capliez@gmail.com\",7) : \n";
    print_r(delAimer("mathieu.capliez@gmail.com", 7));

}
?>
```

L'implémentation de la fonctionnalité - en plus du modèle - dans le site sera faite au TP suivant.

Question 6 - consultation des critiques - notes et commentaires - d'un restaurant.

Documents à utiliser
- fichiers fournis en ressources
- annexes 12



Fiche descriptive d'un restaurant et commentaires associés

La fiche descriptive d'un restaurant contient en bas de page la liste des critiques (notes et commentaires) émises par les utilisateurs sur ce restaurant.
La vue et le contrôleur sont déjà codés pour cette fonctionnalité, mais la fonction du modèle se contente pour le moment de retourner une liste vide.
Dans le contrôleur, la ligne suivante permet d'appeler la fonction du modèle que vous devez écrire dans cet exercice.

```
$critiques = getCritiquerByIdR($idR);
```

Cette ligne nous informe du nom que doit prendre la méthode ainsi que du paramètre attendu.
De même le nom de la variable à gauche de l'affectation ($critiques) nous informe que la valeur qui doit être retournée est une liste de critiques.
Une critique étant composée de tous les éléments de la table critiquer.

Dans la vue, la section de code suivante affiche les critiques attribuées au restaurant contenues dans la variable `$critiques`.

```
<ul id="critiques">
   <?php for ($i = 0; $i < count($critiques); $i++) { ?>
      <li>
         <span>
            <?= $critiques[$i]["mailU"] ?>
            <?php if ($critiques[$i]["mailU"] == $mailU) { ?>
               <a href='./?action=supprimerCritique&idR=<?= $unResto['idR']; ?>'>Supprimer</a>
            <?php } ?>
         </span>
         <div>
            <span>
               <?php
               if ($critiques[$i]["note"]) {
                  echo $critiques[$i]["note"] . "/5";
```

```
            }
            ?>
          </span>
          <span><?= $critiques[$i]["commentaire"] ?> </span>
        </div>

    </li>
  <?php } ?>
</ul>
```

6.1. Écrire la définition (prototype) de la fonction getCritiquerByIdR().

La définition (prototype) de la fonction getCritiquerByIdR() est la suivante :

function getCritiquerBy($IdR)

6.2. Écrire puis tester la requête SQL permettant de récupérer les critiques associées à un restaurant.

La requête SQL permettant de récupérer les critiques associées à un restaurant est la suivante :

SELECT * FROM `critiquer` WHERE idR =:idR;

Affichage des lignes 0 - 4 (total de 5, traitement en 0,0003 seconde(s).)

```
SELECT * FROM `critiquer` WHERE idR = "1";
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

+ Options

	idR	mailU	note	commentaire
Éditer Copier Supprimer	1	jj.soueix@gmail.com	3	moyen
Éditer Copier Supprimer	1	mathieu.capliez@gmail.com	3	Très bonne entrecote, les frites sont maisons et d...
Éditer Copier Supprimer	1	nicolas.harispe@gmail.com	4	Très bon accueil.
Éditer Copier Supprimer	1	test@bts.sio	4	5/5 parce que j'aime les entrecotes
Éditer Copier Supprimer	1	yann@lechambon.fr	5	*NULL*

Un exemple sur phpmyadmin (on peut voir qu'il y a eu quelques petites modifications entre l'annexe 13 et la base de données).

La fonction getCritiquerByIdr() doit renvoyer un résultat similaire à celui présenté dans l'annexe 13.

6.3. À l'aide de la section de code issue de la vue et de l'annexe 13 décrire la structure de données retournée par la fonction getCritiquerByIdR().

0	idR	1
	mailU	jj.soueix@gmail.com
	note	3
	commentaire	moyen
1	idR	1
	mailU	mathieu@capliez@gmail.com
	note	3
	commentaire	Très bonne entrecote, les frites sont maisons et délicieuses.
2	idR	1

	mailU	nicolas.harispe@gmail.com
	note	4
	commentaire	Très bon accueil
3	idR	1
	mailU	test@bts.sio
	note	5
	commentaire	

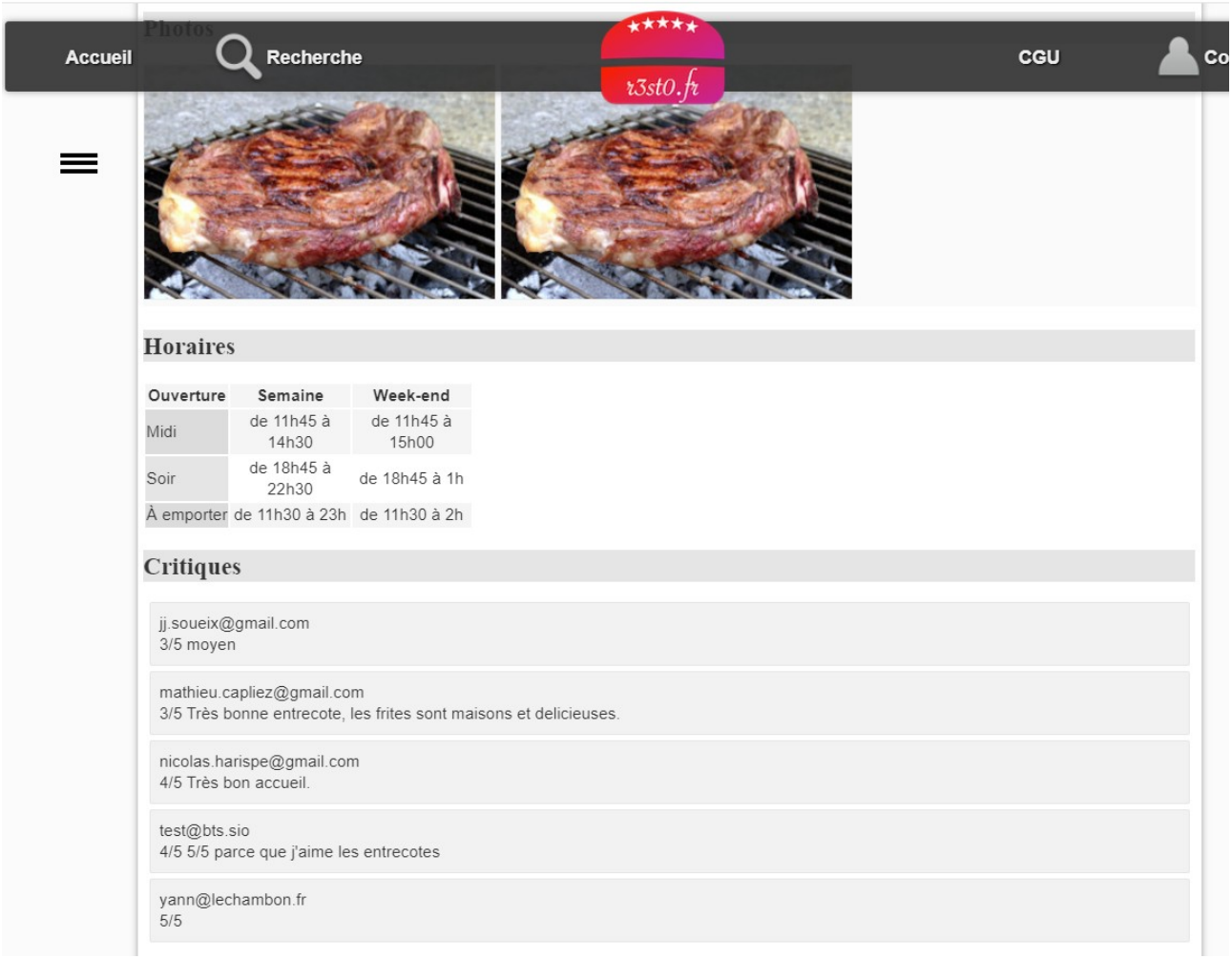
6.4. La fonction getCritiquerByIdr() actuellement présente dans le script modèle bd.critiquer.inc.php ne retourne qu'une liste vide. Compléter son code afin que celle-ci retourne la liste des critiques du restaurant dont l'identifiant est passé en paramètre.
Vérifier le bon fonctionnement de la fonction en ajoutant un appel dans la section de test du script bd.critiquer.inc.php puis en consultant la fiche d'un restaurant associé à des critiques.

```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;
}
```

Le code de la fonction en question (à noter que $resultat = array(); ne me sert au final pas vraiment).

```php
if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');

    echo "\n getCritiquerByIdR(1) \n";
    print_r(getCritiquerByIdR(1));

    echo "\n getNoteMoyenneByIdR(1) \n";
    print_r(getNoteMoyenneByIdR(1));
}
?>
```

J'ai rajouté un test de script.



Les critiques s'affichent correctement sur chaque page !

Annexe 1 - bd.inc.php

```
<?php

function connexionPDO() {
    $login = "votre login";
    $mdp = "votre mot de passe";
    $bd = "nom de la base de données";
    $serveur = "adresse IP ou nom du serveur de base de données";

    try {
        $conn = new PDO("mysql:host=$serveur;dbname=$bd", $login, $mdp,
array(PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES \'UTF8\''));
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $conn;
    } catch (PDOException $e) {
        print "Erreur de connexion PDO ";
        die();
    }
}

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog de test
    header('Content-Type:text/plain');

    echo "connexionPDO() : \n";
    print_r(connexionPDO());
}
?>
```

Annexe 2 - extrait du modèle bd.resto.inc.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;
}

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

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // prog principal de test
    header('Content-Type:text/plain');

    echo "getRestoByIdR(1) : \n";
    print_r(getRestoByIdR(1));

    echo "getRestosByNomR('charcut') : \n";
    print_r(getRestosByNomR("charcut"));

}
?>
```

Annexe 3 - résultat d'exécution du script bd.inc.php

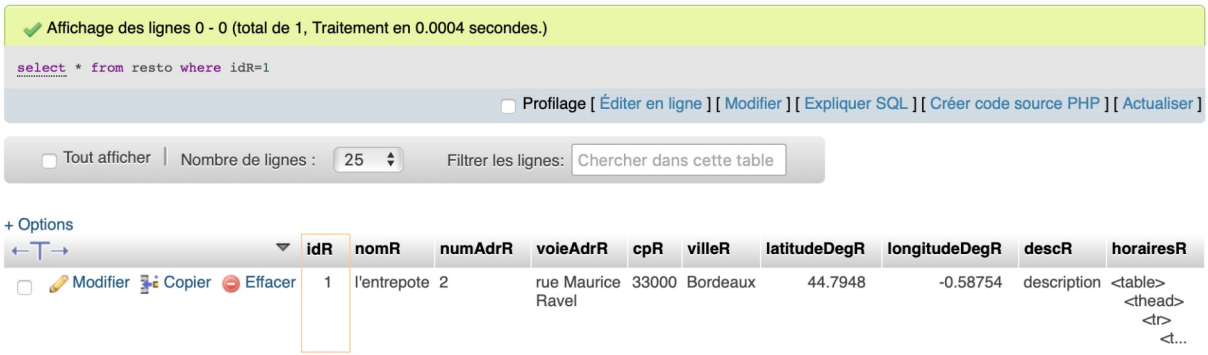
```
connexionPDO() :
PDO Object
(
)
```

Annexe 4 - extrait du résultat d'exécution du script bd.resto.inc.php

```
getRestoByIdR(1) :
Array
(
   [idR] => 1
   [nomR] => l'entrepote
   [numAdrR] => 2
   [voieAdrR] => rue Maurice Ravel
   [cpR] => 33000
   [villeR] => Bordeaux
   [latitudeDegR] => 44.7948
   [longitudeDegR] => -0.58754
   [descR] => description
   [horairesR] => <table>...</table>
)

getRestosByNomR('charcut') :
Array
(
```

```
  [0] => Array
     (
        [idR] => 2
        [nomR] => le bar du charcutier
        [numAdrR] => 30
        [voieAdrR] => rue Parlement Sainte-Catherine
        [cpR] => 33000
        [villeR] => Bordeaux
        [latitudeDegR] =>
        [longitudeDegR] =>
        [descR] => description
        [horairesR] => <table>...</table>
     )
)
```

Annexe 5 - résultat d'exécution de la requête SQL : select * from resto where idR=1



Affichage des lignes 0 - 0 (total de 1, Traitement en 0.0004 secondes.)

```
select * from resto where idR=1
```

Profilage [Éditer en ligne] [Modifier] [Expliquer SQL] [Créer code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table

+ Options

	idR	nomR	numAdrR	voieAdrR	cpR	villeR	latitudeDegR	longitudeDegR	descR	horairesR
Modifier Copier Effacer	1	l'entrepote	2	rue Maurice Ravel	33000	Bordeaux	44.7948	-0.58754	description	<table> <thead> <tr> <t...

Annexe 6 - résultat d'exécution de la requête SQL : select * from resto where nomR like '%charcut%'

Affichage des lignes 0 - 0 (total de 1, Traitement en 0.0004 secondes.)

```
select * from resto where nomR like '%charcut%'
```

Profilage [Éditer en ligne] [Modifier] [Expliquer SQL] [Créer code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table

+ Options

	idR	nomR	numAdrR	voieAdrR	cpR	villeR	latitudeDegR	longitudeDegR	descR	horairesR
Modifier Copier Effacer	2	le bar du charcutier	30	rue Parlement Sainte-Catherine	33000	Bordeaux	*NULL*	*NULL*	description	<table> <thead> <tr> <t...

Annexe 7 - extrait du script SQL de création de la base de données

```
create table resto (
      idR bigint,
      nomR varchar(255),
      numAdrR varchar(20),
      voieAdrR varchar(255),
```

```
      cpR char(5),
      villeR varchar(255),
      latitudeDegR float,
      longitudeDegR float,
      descR text,
      horairesR text,
      primary key (idR)
);
```

Annexe 8 - extrait du modèle bd.critiquer.inc.php

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

        $req->execute();

        $resultat = $req->fetch(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage();
        die();
    }
    if ($resultat["avg(note)"] != NULL) {
        return $resultat["avg(note)"];
    } else {
        return 0;
    }
}
```

Annexe 9 - extrait du modèle bd.aimer.inc.php

```
function addAimer($mailU, $idR) {
    try {
        $cnx = connexionPDO();
        $req = $cnx->prepare("insert into aimer (mailU, idR)
values(:mailU,:idR)");
        $req->bindValue(':idR', $idR, PDO::PARAM_INT);
        $req->bindValue(':mailU', $mailU, PDO::PARAM_STR);

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

Annexe 10 - extrait du script de création de la base de données

```
create table aimer (
      idR bigint,
      mailU varchar(100),
      primary key (idR,mailU),
      foreign key(idR) references resto(idR),
      foreign key(mailU) references utilisateur(mailU)
);
```

Annexe 11 - extrait de la documentation PHP

PDOStatement::execute
(PHP 5 >= 5.1.0, PHP 7, PECL pdo >= 0.1.0)

PDOStatement::execute — Exécute une requête préparée

Description
public PDOStatement::execute ([array $input_parameters]) : bool

Exécute une requête préparée. Si la requête préparée inclut des marqueurs de positionnement, vous pouvez :

- PDOStatement::bindParam() et/ou PDOStatement::bindValue() doit être appelé pour lier des variables ou des valeurs (respectivement) aux marqueurs de paramètres. Les variables liées passent leurs valeurs en entrée et reçoivent les valeurs de sortie, s'il y en a, de leurs marqueurs de positionnement respectifs
- ou passer un tableau de valeurs de paramètres, uniquement en entrée

Liste de paramètres
input_parameters
Un tableau de valeurs avec autant d'éléments qu'il y a de paramètres à associer dans la requête SQL qui sera exécutée. Toutes les valeurs sont traitées comme des constantes PDO::PARAM_STR.

Les valeurs multiples ne peuvent pas être liées à un seul paramètre; par exemple, il n'est pas autorisé de lier deux valeurs à un seul paramètre nommé dans une clause IN().

La liaison de plus de valeurs que spécifié n'est pas possible ; s'il y a plus de clés dans input_parameters que dans le code SQL utilisé pour PDO::prepare(), alors la requête préparée échouera et une erreur sera levée.

Valeurs de retour
Cette fonction retourne TRUE en cas de succès ou FALSE si une erreur survient.

Annexe 12 - extrait du script de création de la base de données

```
create table critiquer (
      idR bigint,
      mailU varchar(100),
      note integer,
      commentaire varchar(4096),
      primary key (idR,mailU),
      foreign key(idR) references resto(idR),
      foreign key(mailU) references utilisateur(mailU)
);
```

Annexe 13 - exemple de valeur retournée par la fonction getCritiquerByIdR(1)

```
Array
(
    [0] => Array
        (
            [idR] => 1
            [mailU] => jj.soueix@gmail.com
            [note] => 3
            [commentaire] => moyen
        )

    [1] => Array
        (
            [idR] => 1
            [mailU] => mathieu.capliez@gmail.com
            [note] => 3
            [commentaire] => Très bonne entrecote, les frites sont maisons et
delicieuses.
        )

    [2] => Array
        (
            [idR] => 1
            [mailU] => nicolas.harispe@gmail.com
            [note] => 4
            [commentaire] => Très bon accueil.
        )

    [3] => Array
        (
            [idR] => 1
            [mailU] => test@bts.sio
            [note] => 4
            [commentaire] =>
        )

)
```

Architecture MVC en PHP - Contrôleur principal

Fonctionnement du contrôleur principal dans le patron de conception MVC

Remarques importantes à destination de l'enseignant
Les sections de tests contenues dans les contrôleurs et dans les modèles permettant de tester de manière indépendantes chaque module ne fonctionnent que sous environnement UNIX.

Rappel du contexte
R3st0.fr est un site web de critique de restaurant. À l'image des sites de ce type il a pour vocation le recensement des avis des consommateurs et la diffusion de ces avis aux visiteurs.

Ce site web est développé en PHP en suivant le patron de conception "modèle vue contrôleur" (pattern MVC). L'architecture retenue pour ce projet permet d'appréhender la programmation web d'une manière structurée.

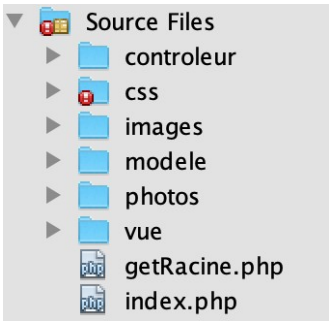
L'objectif de ce document est d'analyser le fonctionnement du contrôleur principal dans l'architecture MVC puis d'intégrer de nouveaux modules fournis.

Ressources à utiliser
- Dossier "base de données" : fichier base.sql contenant les tables et les données de la base utilisée par l'application web étudiée.
- Dossier site : arborescence et fichiers de l'application web.

Après avoir créé la base de données (encodage utf8mb4) et importé le fichier `base.sql` , créer un nouveau projet PHP appelé SI6MVCTP1 dans Netbeans. Paramétrer le projet pour que celui-ci soit téléchargé sur le serveur lors de l'exécution.

Dans le gestionnaire de fichiers, supprimer le fichier `index.php` créé automatiquement, et remplacer le contenu du dossier "Source Files" par le contenu du dossier "site" fourni.

L'arborescence devrait être celle-ci :



Avant de commencer le TP, le site doit être paramétré pour qu'il utilise votre base de données. Dans le script `modele/bd.inc.php`, modifier les lignes suivantes afin d'indiquer les bonnes informations :

```
$login = "votre login mariaDB";
$mdp = "votre mot de passe mariaDB";
$bd = "nom de votre base de données";
$serveur = "adresse IP ou nom du serveur de base de données";
```

Afin de mieux voir l'objectif du site, et les différentes fonctionnalités que vous devrez mettre en place tout au long de ces TP, le site final est consultable à l'adresse fournie par votre professeur.

A - Analyse du fonctionnement du contrôleur principal

Question 1 - Analyse de l'existant

Documents à utiliser
- fichiers fournis en ressources
- annexes 1, 2 et 3

Rappel : transmission de paramètres en méthode GET
Lorsque des paramètres sont transmits en méthode GET, ils sont visibles dans la barre d'URL. Les paramètres sont spécifiés en fin d'URL après le point d'interrogation.
`http://serveur/dossier/script.php?prenom=jean&age=20`

Dans l'exemple ci-dessus, les paramètres transmis sont : `prenom` et `age`.
`prenom` a pour valeur `jean`
`age` a pour valeur `20`

1.1. Relever les URL en navigant successivement sur les écrans de connexion, de recherche et d'accueil. indiquer quelle partie de l'URL change, en spécifiant le nom de paramètre et la valeur envoyés en méthode GET.

Écran	URL	Paramètre	Valeur
connexion	/?action=connexion	action	connexion
recherche	/?action=recherche	action	recherche
accueil	/?action=accueil	action	accueil

Le fichier index.php est utilisé par défaut par le serveur web Apache lorsqu'une URL pointant sur un répertoire est appelé par le navigateur. C'est donc lui qui a reçu les paramètres envoyés en méthode GET dans la question 1.1.

1.2. A l'aide de l'annexe 1 et de la réponse à la question précédente, indiquer les valeurs pouvant être prises par la variable `$action`.

Les valeurs pouvant être prises par la variable $action sont actuellement visibles dans controleurPrincipal.php et sont : defaut, liste, detail, recherche, connexion, deconnexion et profil.

La fonction controleurPrincipal() visible en annexe 2 est appelée par le fichier index.php. Cette fonction a un rôle de routage (d'aiguillage). Elle donne accès aux différents contrôleurs de l'application.

1.3. Schématiser la structure de la variable $lesActions créée dans la fonction controleurPrincipal().
La structure de la variable $lesActions est la suivante :
defaut => listeRestos.php
liste => listeRestos.php
detail => detailResto.php
recherche => rechercheResto.php
connexion => connexion.php
deconnexion => deconnexion.php
profil => monProfil.php

1.4. Quelle partie de cette structure est similaire aux donnés trouvées en question 1.1 ?

On retrouve après les «?action= » les données en question qui redirigent vers les pages qu'on souhaite (si ?action=connexion, on se retrouvera sur la page connexion.php).

Excepté pour accueil qui renvoie à la vue listeRestos.php.

Lorsque l'on accède à la variable $lesActions, on peut à partir d'une action obtenir un nom de fichier. Par exemple l'instruction suivante place la chaîne "listeRestos.php" dans la variable $fichier.

```
$fichier = $lesActions['liste'];
```

1.5. À partir de vos réponses aux questions précédentes, indiquer quelles valeurs peuvent être transmises à la fonction controleurPrincipal() ?

Les valeurs qui peuvent être transmises à la fonction controleurPrincipal() sont donc : connexion, recherche et accueil (qui n'est pas traité dans le tableau associatif, mais dans ce cas-là, la valeur sera remplacée par « defaut », qui est la page d'accueil « listeRestos.php »).

1.6. Pour chacune des valeurs transmises à la fonction indiquer quelle sera la valeur retournée.

Valeur transmise	Valeur retournée
connexion	connexion.php
recherche	rechercheResto.php
accueil	listeRestos.php

1.7. Que se passe-t-il si l'action transmise a la fonction controleurPrincipal() n'existe pas dans la variable $lesActions ? Quelle valeur est retournée ?

Comme énoncé dans la question 1.5, si l'action transmise dans la fonction controleurPrincipal() n'existe pas dans la variable $lesActions, la valeur retournée sera listeRestos.php.

Lorsque la fonction controleurPrincipal() est appelée dans le fichier index.php, elle retourne le nom d'un fichier contrôleur à inclure. Les deux lignes suivantes issues du script index.php prennent en charge la sélection du contrôleur demandé, et son chargement.

```
$fichier = controleurPrincipal($action);
include "$racine/controleur/$fichier";
```

1.8. Après avoir consulté l'annexe 3, expliquer à quoi sert la condition utilisant l'instruction array_key_exists() dans la fonction controleurPrincipal().

La condition utilisant l'instruction en question dans la fonction sert à vérifier si la valeur rentrée dans $action est bien dans le tableau associatif (car c'est un array comme en témoigne la ligne 4). Si c'est bien le cas (TRUE), alors $action aura bien la valeur comme prévue qui sera appliquée. Sinon, la valeur de $action sera ''defaut'' qui renverra à listeRestos.php (notre page d'accueil).

1.9. Quel script contrôleur est appelé par index.php lorsque la variable GET action n'est pas renseignée ?

Comme en témoigne ce code suivant :

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

```
    $action = $_GET["action"];
}
else {
    $action = "defaut";
}
```

Si la variable GET action n'est pas renseignée, $action aura forcément une valeur de ''defaut'' et renverra donc à la page d'accueil, donc le script contrôleur appelé par index.php est listeRestos.php.

```
function controleurPrincipal($action) {
    $lesActions = array();
    $lesActions["defaut"] = "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";
```

1.10. Quel script contrôleur est appelé par index.php lorsque la variable GET action contient le mot clé "liste"

```
if (isset($_GET["action"])) {
    $action = $_GET["action"];
}
else {
    $action = "defaut";
}
```

Si la variable GET action a bien une valeur (le rôle de isset qui permet de vérifier si une variable a été définie), le mot clé est alors transmis vers la fonction controleurPrincipal() qui va donner le contrôleur souhaité.

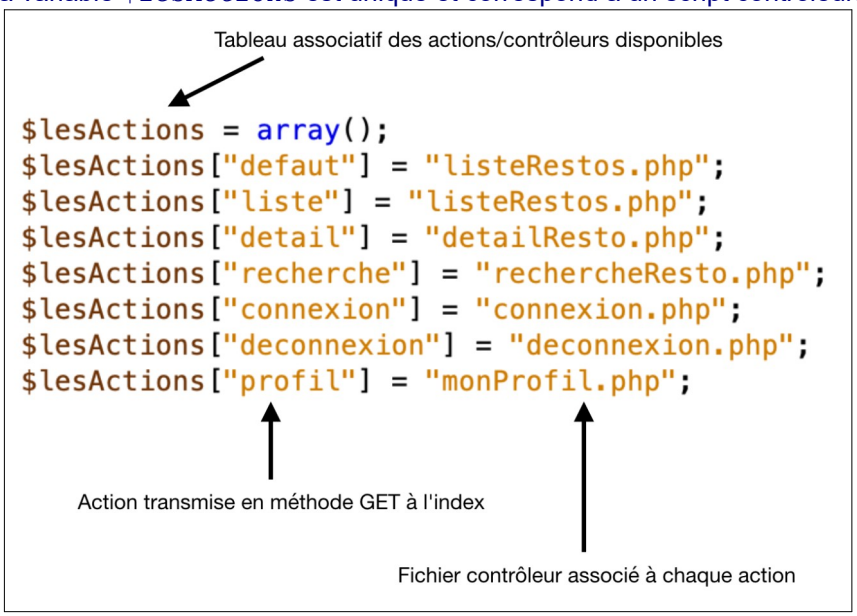
```
function controleurPrincipal($action) {
    $lesActions = array();
    $lesActions["defaut"] = "listeRestos.php";
    $lesActions["liste"] = "listeRestos.php";
    $lesActions["detail"] = "detailResto.php";
    $lesActions["recherche"] = "rechercheResto.php";
    $lesActions["connexion"] = "connexion.php";
    $lesActions["deconnexion"] = "deconnexion.php";
    $lesActions["profil"] = "monProfil.php";
```

Donc le script contrôleur qui est appelé par index.php lorsque la variable GET action contient le mot clé ''liste'' est listeRestos.php

Synthèse

La variable `$lesActions` de la fonction `controleurPrincipal()` contient l'ensemble des contrôleurs accessibles sur le site. Un contrôleur est l'élément central d'une fonctionnalité.
Lorsqu'une nouvelle fonctionnalité est développée sur l'application, il faut l'intégrer au contrôleur principal en ajoutant une nouvelle valeur dans la variable `$lesActions`.

Chaque clé dans la variable `$lesActions` est unique et correspond à un script contrôleur.



Structure de la variable $lesActions

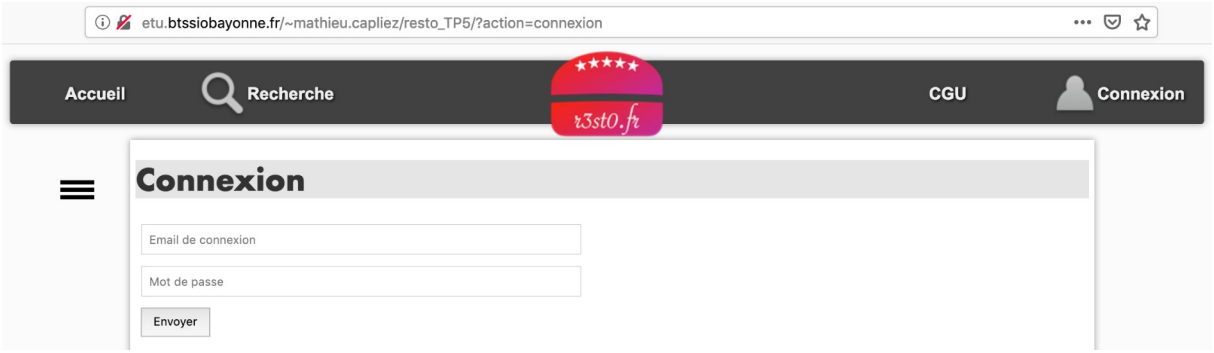
Lorsqu'on ajoute une nouvelle fonctionnalité dans l'application, il faut créer le script contrôleur dans le dossier du même nom, puis déclarer cette nouvelle fonctionnalité dans la fonction `controleurPrincipal()` en ajoutant l'action correspondante (clé) et le fichier associé.

L'instruction ci-dessous permet d'ajouter une nouvelle action associée à un contrôleur dans la variable `$lesActions`.

```
$lesActions['uneAction'] = "scriptControleur.php" ;
```

Accéder à une fonctionnalité sur l'application web
Le fichier index.php est la porte d'entrée pour l'accès aux différentes fonctionnalités. C'est lui qui réceptionne l'action envoyée en méthode GET puis charge le contrôleur associé.
Dès lors, la navigation sur l'application web ne fait apparaître que le fichier index.php dans l'URL. Chaque fonctionnalité est demandée par l'intermédiaire de la variable action transmise.

Lorsqu'on souhaite accéder à une fonctionnalité par l'intermédiaire d'un lien, l'URL pointée doit faire référence à l'action souhaitée.



Accès au contrôleur de connexion

```php
<ul id="menuGeneral">
    <li><a href="./?action=accueil">Accueil</a></li>
    <li><a href="./?action=recherche"><img src="images/rechercher.png" alt="loupe" />Recherche</a></li>
    <li></li>

    <li id="logo"><a href="./?action=accueil"><img src="images/logoBarre.png" alt="logo" /></a></li>
    <li></li>
    <li><a href="./?action=cgu">CGU</a></li>
    <?php if(isLoggedOn()){ ?>
    <li><a href="./?action=profil"><img src="images/profil.png" alt="loupe" />Mon Profil</a></li>
    <?php }
    else{ ?>
    <li><a href="./?action=connexion"><img src="images/profil.png" alt="loupe" />Connexion</a></li>
    <?php } ?>
</ul>
```

Action transmise au fichier index.php

Le nom du fichier contrôleur est masqué à l'utilisateur qui ne voit que l'action.

Dans l'exemple ci dessus, le nom du fichier index.php n'est pas visible, on peut soit l'indiquer, soit faire référence à l'emplacement "/" qui désigne le fichier index.php dans la configuration du serveur web Apache .

Les contrôleurs ne sont jamais directement appelés dans l'URL, on doit toujours passer par le contrôleur principal, et donc par index.php. Il en est de même avec les formulaires, comme celui de connexion.

Action transmise au fichier index.php en méthode GET (visible dans l'URL)

Données transmises en POST au fichier index.php puis au contrôleur connexion.php inclus

```php
<h1>Connexion</h1>
<form action="./?action=connexion" method="POST">

    <input type="text" name="mailU" placeholder="Email de connexion" /><br />
    <input type="password" name="mdpU" placeholder="Mot de passe"  /><br />
     <input type="submit" />

</form>
```

Extrait de la vue affichant le formulaire de connexion

On remarque que les données du formulaire sont transmises en méthode POST à index.php. Lors de l'inclusion du fichier contrôleur (connexion.php), ces données seront aussi accessibles dans le contrôleur.

Pour intégrer une nouvelle fonctionnalité au site web, il faut donc :

- créer un nouveau contrôleur dans le dossier approprié,
- créer la vue et le modèle associé si besoin,
- ajouter une action dans la variable $lesActions associant le nom de l'action au nom du script contrôleur,
- donner l'accès à ce contrôleur par l'intermédiaire d'un lien ou d'un formulaire transmettant l'action en méthode GET.

B - Intégration de contrôleurs pré-existants

Question 2 - CGU

Documents à utiliser
- fichiers fournis en ressources
- annexe 4

Les conditions générales d'utilisations ont déjà été écrites. La vue et le contrôleur associés sont disponibles. De même, le menu général propose un lien vers celles ci.

Lorsque l'on clique sur ce lien les CGU ne sont pas affichées, le contrôleur par défaut est chargé à la place.

2.1. Repérer dans le menu général l'action correspondant au contrôleur ayant en charge l'affichage des CGU

Il s'agit de ?action=cgu.

2.2. Placer les fichiers fournis en ressource dans les dossiers appropriés.

C'est fait (les fichiers vueCgu et cgu).

2.3. Rédiger l'instruction à ajouter à la fonction controleurPrincipal() pour ajouter la nouvelle action dans la variable $lesActions.

```php
function controleurPrincipal($action) {
    $lesActions = array();
    $lesActions["defaut"] = "listeRestos.php";
    $lesActions["liste"] = "listeRestos.php";
    $lesActions["detail"] = "detailResto.php";
    $lesActions["recherche"] = "rechercheResto.php";
    $lesActions["cgu"] = "cgu.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["defaut"];
    }
}
```

On a alors rajouté la ligne suivante : $lesActions[''cgu''] = ''cgu.php''.

2.4. Ajouter la nouvelle action à la fonction puis tester le bon fonctionnement du lien CGU dans le menu général.

C'est fonctionnel !

Question 3 - aimer un restaurant

Documents à utiliser
- fichiers fournis en ressources
- Annexe 5

Lorsqu'un utilisateur connecté consulte la fiche descriptive d'un restaurant, il a la possibilité d'ajouter celui-ci dans la liste des restaurants qu'il aime. Cette action se fait en cliquant sur l'étoile située à droite du nom du restaurant. Tant que l'utilisateur n'a pas aimé ce restaurant, l'étoile est grisée.

Lien sous forme d'une étoile pour aimer un restaurant

Lorsqu'on clique sur ce lien, le traitement n'est pas effectué, rien n'est enregistré dans la table aimer. Le contrôleur par défaut est chargé à la place.

3.1. Quelle vue permet d'afficher la fiche descriptive d'un restaurant ?

La vue permettant d'afficher la fiche descriptive d'un restaurant est vueDetailResto.php

3.2. Repérer dans le code de la vue le lien correspondant à l'étoile.

Il s'agit de ''./?action=aimer&idR=<?= $unResto['idR']; ?> qu'on trouve respectivement aux lignes 5 et 7 du fichier.

3.3. Quels sont les paramètres envoyés en méthode GET lorsque l'on clique sur le lien ? Préciser le nom et la valeur de chaque paramètre.

Il y a deux paramètres qui sont envoyés en méthode GET lorsque l'on clique sur le lien :

1 > Nom du paramètre : ''action'' dont sa valeur est ''aimer''.
2 > Nom du paramètre : ''idR'' dont sa valeur est ''$unResto['idR']'' (ce dernier étant un int).

3.4. À partir de vos connaissances et du script contrôleur fourni en ressource, déterminer dans quels scripts sont utilisés chacune des deux variables transmises par le lien en méthode GET (celles trouvées à la question précédente).

Les scripts qui sont utilisés pour les deux variables transmises par le lien en méthode GET sont donc $mailU et $idR.

3.5. Placer le fichier fourni en ressource dans le dossier approprié.

Le dossier approprié pour aimer.php est le dossier « controleur ».

3.6. Rédiger l'instruction à ajouter à la fonction controleurPrincipal() pour ajouter la nouvelle action dans la variable $lesActions.

```php
function controleurPrincipal($action) {
    $lesActions = array();
    $lesActions["defaut"] = "listeRestos.php";
    $lesActions["liste"] = "listeRestos.php";
    $lesActions["detail"] = "detailResto.php";
    $lesActions["recherche"] = "rechercheResto.php";
    $lesActions["cgu"] = "cgu.php";
    $lesActions["aimer"] = "aimer.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["defaut"];
    }
}
```

J'ai rajouté la ligne $lesActions[''aimer''] = ''aimer.php'';

3.7. Ajouter la nouvelle action à la fonction puis tester le bon fonctionnement du contrôleur en cliquant sur l'étoile. (Pour tester, il faut être authentifié sur le site)

Lorsqu'on clique sur l'étoile, on observe que la page affichée reste la même. La page chargée affiche aussi la même URL, pourtant le lien pointé est bien différent.

Le test fonctionne, si je clique sur l'étoile, l'image change et je constate que la liste des restaurants que l'utilisateur aime subit une modification aussi (que ce soit en retirant ou ajoutant le restaurant dans la liste des favoris).

3.8. Le contrôleur aimer.php fait-il appel à une vue ?

Non, le contrôleur aimer.php ne fait appel à aucune vue (alors que d'autres documents ont des

include en fin de fichier appelant divers fichiers « vue »).

En fin de fichier il est fait mention en commentaire du referer. La dernière instruction du contrôleur mentionne aussi ce terme :

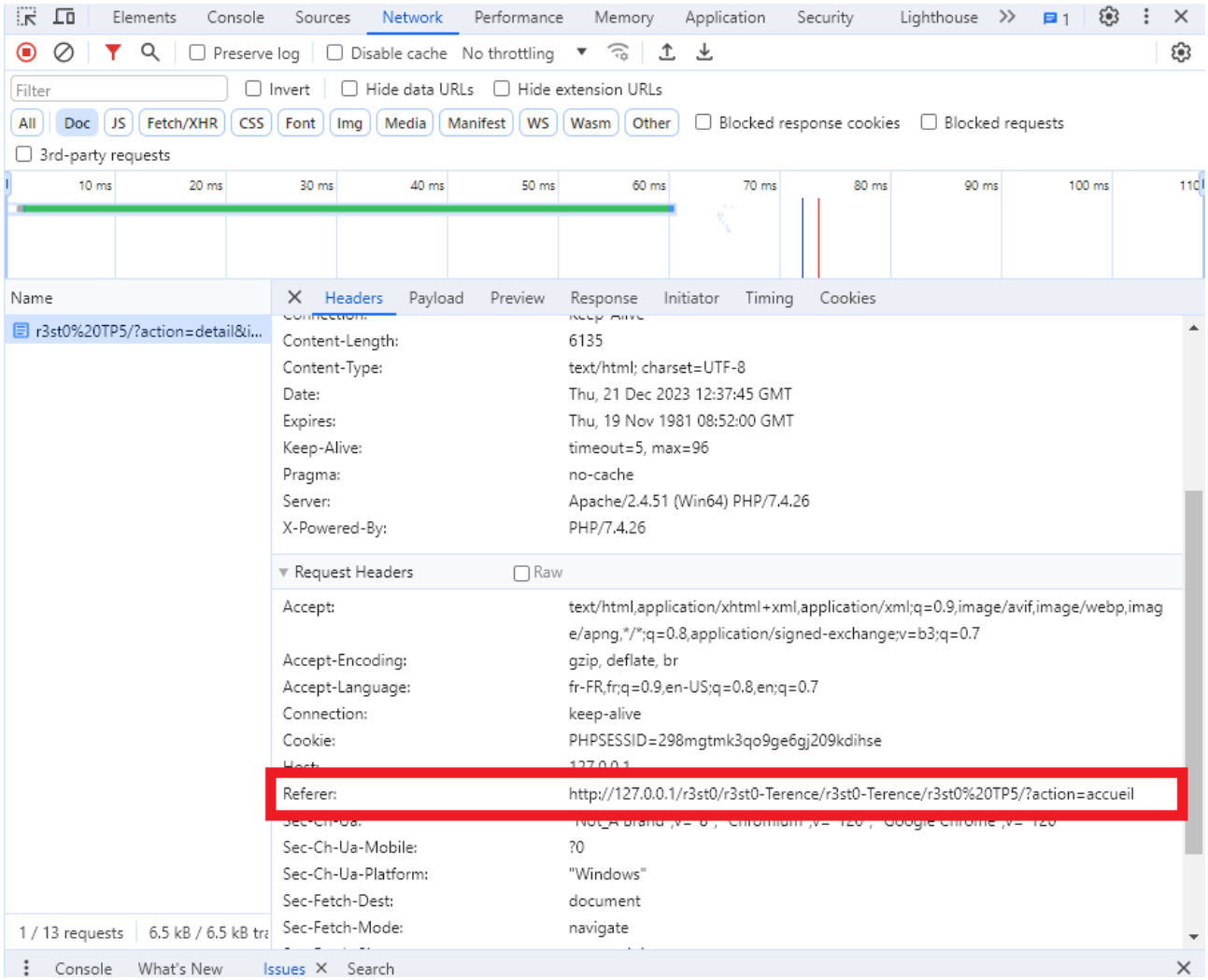
```
header('Location: ' . $_SERVER['HTTP_REFERER']);
```

3.9. Rechercher sur internet la signification du terme referer.

« L'en-tête de requête Referer contient l'adresse de la page web précédente à partir de laquelle un lien a été suivi pour demander la page courante. L'en-tête Referer permet aux serveurs d'identifier la provenance des visiteurs d'une page et cette information peut être utilisée à des fins d'analyse, de journalisation ou pour améliorer la politique de cache par exemple. »

Source : mdn web docs (https://developer.mozilla.org/fr/docs/Web/HTTP/Headers/Referer)

Il s'agit donc de pouvoir connaître le cheminement de la navigation du site web d'un utilisateur. Par exemple, s'il est sur la page d'accueil (/index.php) et qu'il va sur la page du détail d'un restaurant (/detailResto.php), la requête HTTP envoyée au serveur inclut dans l'en-tête ''Referer'' l'URL de la page d'accueil.



Avec l'inspecteur d'éléments, dans l'onglet « Network », on peut constater ce point (effectivement, la page précédente que je venais de visiter était la page d'accueil).

Mettre en commentaire l'instruction header() et afficher à la place la valeur de la variable $_SERVER['HTTP_REFERER'].
Tester de nouveau l'action d'aimer sur un restaurant.

3.10. Que contient la variable $_SERVER['HTTP_REFERER'] ?

http://127.0.0.1/r3st0/r3st0-Terence/r3st0-Terence/r3st0%20TP5/?action=detail&idR=1

Comme prévu, le lien de la page précédente avant de cliquer sur le bouton (qui amène à une nouvelle page du fait du <a href>)

3.11. Déduire de vos observations le rôle de l'instruction ci-dessous :

```
header('Location: ' . $_SERVER['HTTP_REFERER']);
```

Le rôle de l'instruction est donc de ramener l'utilisateur de nouveau à la page où il était quand il appuie sur le bouton Aimer.

Sans cette ligne de code, étant donné que aimer.php n'a aucune vue, l'utilisateur se retrouverait sur une page vide.

L'instruction permet donc de recharger (rester sur ?) la page au moment où on appuie sur le bouton avec la modification en conséquence (ajouter ou supprimer le restaurant de ses favoris), sans devoir être redirigé sur une page vide (car dénuée de vue).

Question 4 - inscription

Le formulaire d'inscription est accessible par l'intermédiaire d'un lien dans la section "connexion" du site.

4.1 Analyser ce lien, puis à l'aide des fichiers en ressource apporter les modifications nécessaires pour rendre l'inscription fonctionnelle.

Voici les modifications effectuées :

Tout d'abord, j'ai importé les documents inscription.php et vueInscription.php aux bons endroits (respectivement dans le dossier « contrôleur » et dans le dossier « vue »).

Ensuite, j'ai ajouté dans le controleurPrincipal.php une ligne de code pour rajouter inscription.php

```php
$lesActions["inscription"] = "inscription.php";
```

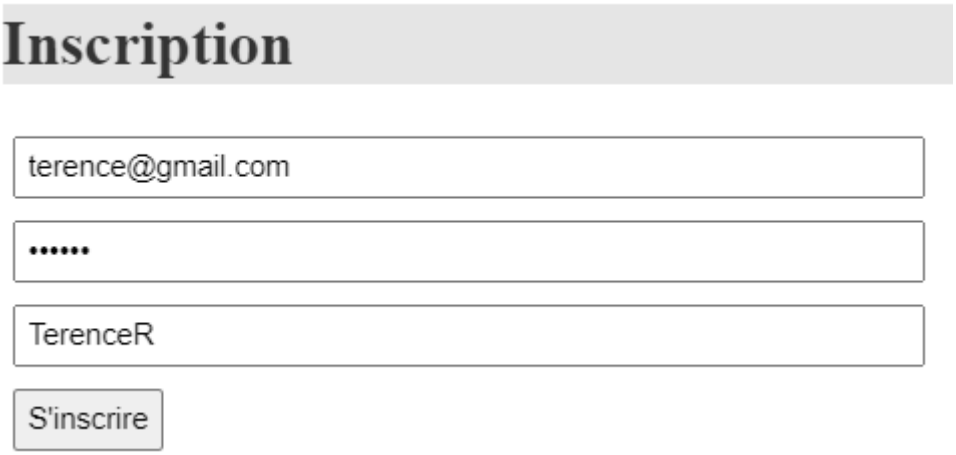
Au moment de m'inscrire, un message d'erreur est apparu car vueConfirmationInscription.php n'existait pas et ne pouvait donc pas être inclus, j'ai donc remédié à ce problème en créant ledit fichier.

r3st0 TP5 > vue > vueConfirmationInscription.php > ...

```
1   <h1>Confirmation d'inscription</h1>
2
3   Vous êtes bien inscrit sur r3sto.fr, vous pouvez vous connecter à présent !
```

Une fois encore, j'ai connu plus difficile.

Créons un compte et amusons-nous à tester les fonctionnalités implémentées lors de ce TP 5 !



Note : le mot de passe est azerty

Bien, nous allons nous connecter dorénavant.

Ajoutons des restaurants en favoris.

Parfait.

Remarques :
- le modèle, la vue et le contrôleur sont déjà prêts. Ils sont soit fournis en ressource, soit déjà en place dans le code existant ;
- reprendre la même logique que pour les exercices précédents.

Annexe 1 - index.php

```
<?php
include "getRacine.php";
include "$racine/controleur/controleurPrincipal.php";
include_once "$racine/modele/authentification.inc.php";

if (isset($_GET["action"])) {
    $action = $_GET["action"];
}
else {
    $action = "defaut";
}

$fichier = controleurPrincipal($action);
include "$racine/controleur/$fichier";
?>
```

Annexe 2 - controleurPrincipal.php

```
<?php

function controleurPrincipal($action) {
    $lesActions = array();
    $lesActions["defaut"] = "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["defaut"];
    }
}

?>
```

Annexe 3 - extrait de la documentation PHP

array_key_exists

(PHP 4 >= 4.0.7, PHP 5, PHP 7)
`array_key_exists` — Vérifie si une clé existe dans un tableau

Description
`array_key_exists ( mixed $key , array $array ) : bool`

array_key_exists() retourne TRUE s'il existe une clé du nom de key dans le tableau array. key peut être n'importe quelle valeur valide d'index de tableau.

Liste de paramètres
key

Valeur à vérifier.

array
Un tableau contenant les clés à vérifier.

Valeurs de retour

Cette fonction retourne TRUE en cas de succès ou FALSE si une erreur survient.

Annexe 4 - extrait du fichier entete.html.php

```
<ul id="menuGeneral">
    <li><a href="./?action=accueil">Accueil</a></li>
    <li><a href="./?action=recherche"><img src="images/rechercher.png"
alt="loupe" />Recherche</a></li>
    <li></li>

    <li id="logo"><a href="./?action=accueil"><img src="images/logoBarre.png"
alt="logo" /></a></li>
    <li></li>
    <li><a href="./?action=cgu">CGU</a></li>
    <?php if(isLoggedOn()){ ?>
    <li><a href="./?action=profil"><img src="images/profil.png" alt="loupe"
/>Mon Profil</a></li>
    <?php }
    else{ ?>
    <li><a href="./?action=connexion"><img src="images/profil.png"
alt="loupe" />Connexion</a></li>
    <?php } ?>

</ul>
```

Annexe 5 - contrôleur aimer.php

```
<?php

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    $racine = "..";
}
include_once "$racine/modele/bd.aimer.inc.php";

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

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

$mailU = getMailULoggedOn();
if ($mailU != "") {
    $aimer = getAimerById($mailU, $idR);

// traitement si necessaire des donnees recuperees
    ;
    if ($aimer == false) {
        addAimer($mailU, $idR);
    } else {
        delAimer($mailU, $idR);
    }
}
```

```
// redirection vers le referer
header('Location: ' . $_SERVER['HTTP_REFERER']);
?>
```