

Architecture MVC en PHP - Généralités

Partie 1 : généralités

Partie 2 : contrôleur

Partie 3 : vue

Partie 4 : modèle

Partie 5 : contrôleur principal et routage

Découverte générale du patron de conception MVC

- 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 GNU/Linux.

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.

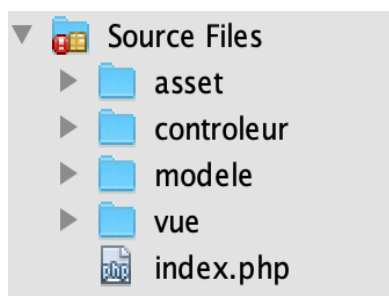
Ressources à utiliser téléchargeable

- 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.

Préparations *[ne pas recréer / importer une base de données si elle a été créée précédemment]*

Après avoir créé la base de données nommée « **resto** » (encodage utf8mb4) et importé le fichier base.sql sur PhpMyAdmin, créer un nouveau dossier nommé **MVCTP1** à la racine de votre serveur web. Copier le contenu du dossier « **site** » dans ce dossier. Importer le dossier **MVCTP1** dans Visual Studio Code.

L'arborescence devrait être semblable celle-ci :



Avant de commencer le TP, le site doit être paramétré afin 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 = "localhost"; //le SGBRD est installé sur le serveur
```

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 [cette adresse](#).

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

Documents à utiliser

- fichiers fournis en ressources
- annexes 1,2,3,4,5, et 10

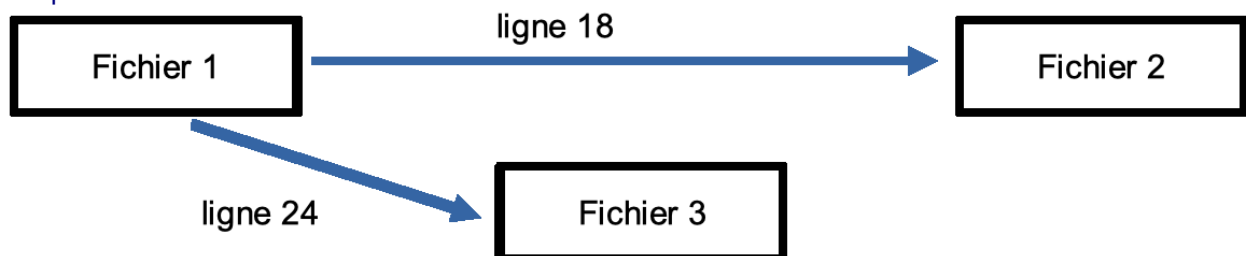
Exécuter le projet puis à l'aide d'un navigateur, accéder à l'url suivante :
`http://mon_serveur_virtuel/MVCTP1/index.php?action=liste`

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

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`.

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.

Exemple :



Le fichier 1 fait appel au fichier 2.
Dans le fichier 1, cet appel est fait à la ligne 18

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 |
|----------|---------|
| | |
| | |
| | |
| | |

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

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..

1.7. Quelle fonction définie dans le script `bd.resto.inc.php` est utilisée dans `listeRestos.php` ?



Ces fonctions et le rôle du script `bd.resto.inc.php` seront étudiées plus en détail dans la 4ème partie de cette suite de TP.

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 ?

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

2.3. Consulter le code source de la page lors d'une visite de l'URL "http://server/MVCTP1/index.php?action=liste". Retrouve-t-on le même contenu que celui des scripts du dossier vue ?

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

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

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

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

- modele

- vue

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 ?

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

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

❖ 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 ?

3.5. Rappeler le rôle de la fonction `getRestoByIdR()`. Pourquoi cette méthode a-t-elle besoin d'un paramètre ?

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.

3.7. Deux autres variables créées dans le contrôleur sont affichées dans la vue. Lesquelles ?

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

Synthèse contrôleur

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

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 ?

- de la base de données vers le contrôleur ?

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 traitements 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.

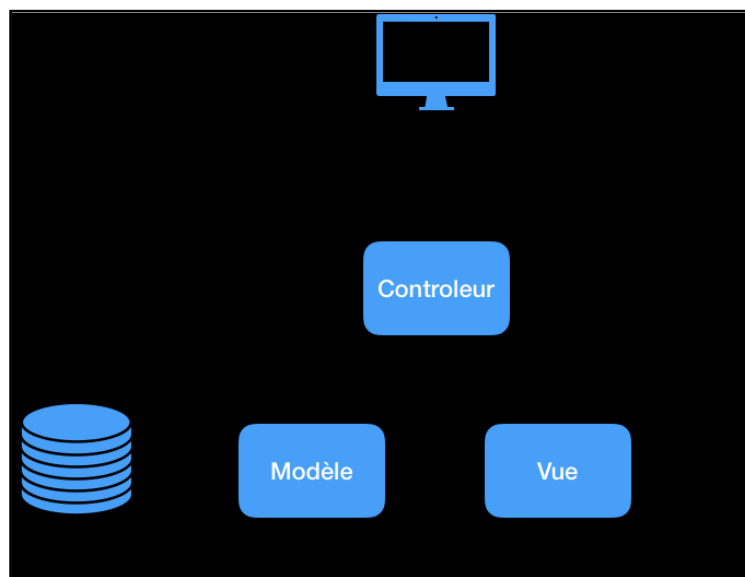
Chaque fonctionnalité fait ainsi appel aux composants :

- Modèle : fonction d'accès à la base de données ;
- Vue : affichage des données à l'utilisateur ;
- Contrôleur : composant chargé de la logique « métier » applicative : récupération des 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
/**
 *    Controleur secondaire : listeResto
 */

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // Un MVC utilise uniquement ses requêtes depuis le contrôleur principal :
    index.php
    die('Erreur : '.basename(__FILE__));
}

require_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

```
<?php

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // Un MVC utilise uniquement ses requêtes depuis le contrôleur principal :
    index.php
    die('Erreur : '.basename(__FILE__));
}

?>

<h1>Liste des restaurants</h1>

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

    <div class="card">
        <div class="photoCard">
        </div>
        <div class="descrCard"><?= "<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

```

<?php

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // Un MVC utilise uniquement ses requêtes depuis le contrôleur principal :
    index.php
    die('Erreur : ' . basename(__FILE__));
}

?>
<!DOCTYPE html>
<html>
    <?php include('head.html.php'); ?>

    <body>
    <nav>

        <ul id="menuGeneral">
            <li><a href="./?action=accueil">Accueil</a></li>
            <li><a href="./?action=liste">Liste</a></li>
            <li></li>
            <li id="logo"><a href="./?action=accueil"></a></li>
            <li></li>
            <li><a href="./?action=cgu">CGU</a></li>

            <li><a href="./?action=connexion">Connexion</a></li>
        </ul>
    </nav>
    <div id="bouton">
        <div></div>
        <div></div>
        <div></div>
    </div>
    <ul id="menuContextuel">
        <li></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] => 1'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>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta charset="UTF-8" />
    <link href="https://fonts.googleapis.com/css?family=Lobster"
rel="stylesheet" />
    <link href="asset/font/awesome/css/font-awesome.min.css"
rel="stylesheet" />
    <!-- static local -->
    <link href="asset/css/base.css" rel="stylesheet" />
    <link href="asset/css/form.css" rel="stylesheet" />
    <link href="asset/css/cgu.css" rel="stylesheet" />
    <link href="asset/css/corps.css" rel="stylesheet" />
    <title><?= $titre ?></title>
  </head>
  <body>
    <nav>

      <ul id="menuGeneral">
        <li><a href="./?action=accueil">Accueil</a></li>
        <li><a href="./?action=recherche">Recherche</a></li>
        <li><a href="./?action=liste">Liste</a></li>

        <li id="logo"><a href="./?action=accueil"></a></li>
        <li></li>
        <li><a href="./?action=cgu">CGU</a></li>

        <li><a href="./?action=connexion">Connexion</a></li>

      </ul>
    </nav>
    <div id="bouton">
      <div></div>
      <div></div>
      <div></div>
    </div>
    <ul id="menuContextuel">
      <li></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

/**
 *      Controleur secondaire : detailResto
 */

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // Un MVC utilise uniquement ses requêtes depuis le contrôleur principal :
    index.php
    die('Erreur : '.basename(__FILE__));
}

require_once RACINE . "/modele/bd.resto.inc.php";

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

```
<?php

if ($_SERVER["SCRIPT_FILENAME"] == __FILE__) {
    // Un MVC utilise uniquement ses requêtes depuis le contrôleur principal :
    index.php
    die('Erreur : ' . basename(__FILE__));
}
?>

<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

/**
 *    Controleur principal
 */

require dirname(__FILE__) . "/controleur/config.php";

require RACINE . "/controleur/routage.php";

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

//Ajoute un controleur secondaire ($fichier) en fonction du metier ($action) :
$fichier = redirigeVers($action);
require RACINE . "/controleur/" . $fichier;

?>
```