2024/2025	AP
BTS SIO 2SLAM	Auteur(s) : Loric Worms Date de rédaction : 30/09/2024

Compte-rendu ticket 2

I - 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.

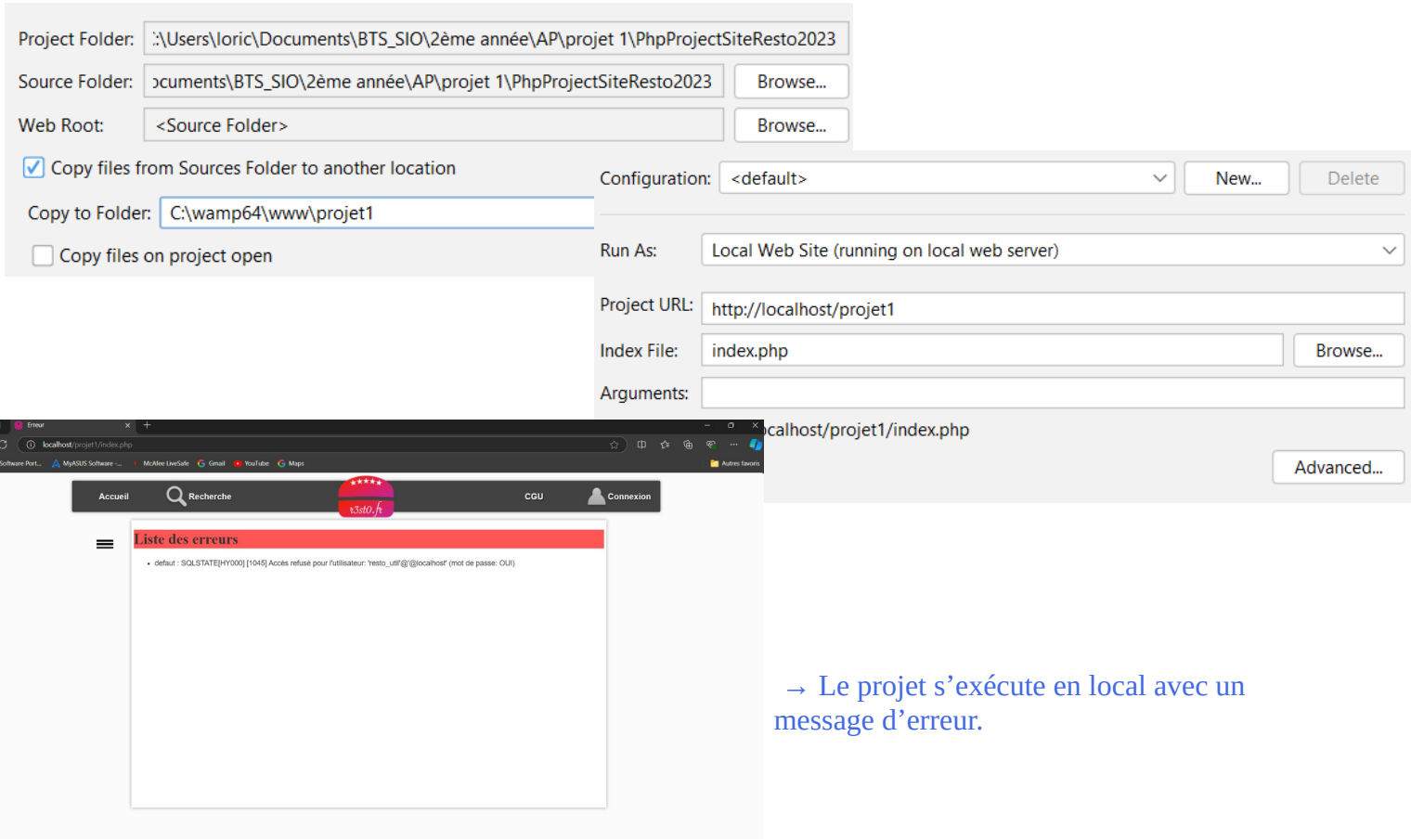
Afin de mieux voir l'objectif du site, et les différentes fonctionnalités, le site final est consultable en interne à l'adresse suivante : https://srvweb.jolsio.net:8443/test2slam/siteresto/

Étude des fonctionnalités existantes

Ressources à utiliser
- le code source du site, sous la forme d'un projet NetBeans, est à télécharger depuis Moodle.

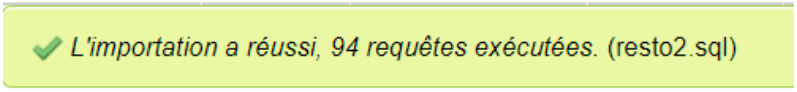
Travail à faire

1. Paramétrer le projet pour que celui-ci soit exécutable sur le serveur web local ;



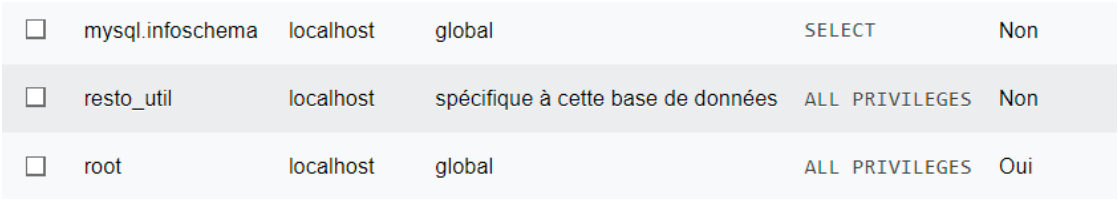
→ Le projet s'exécute en local avec un message d'erreur.

2. Créer la base de données « resto2 » en exécutant les scripts du dossier « sql » du projet ;



→ Création de la bdd.

☐	aimer	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	16	InnoDB	utf8mb4_general_ci	32,0 kio
☐	critiquer	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	19	InnoDB	utf8mb4_general_ci	32,0 kio
☐	photo	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	14	InnoDB	utf8mb4_general_ci	32,0 kio
☐	resto	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	11	InnoDB	utf8mb4_general_ci	16,0 kio
☐	utilisateur	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	7	InnoDB	utf8mb4_general_ci	32,0 kio
	5 tables	Somme							67	MyISAM	utf8mb4_unicode_ci	144,0 kio



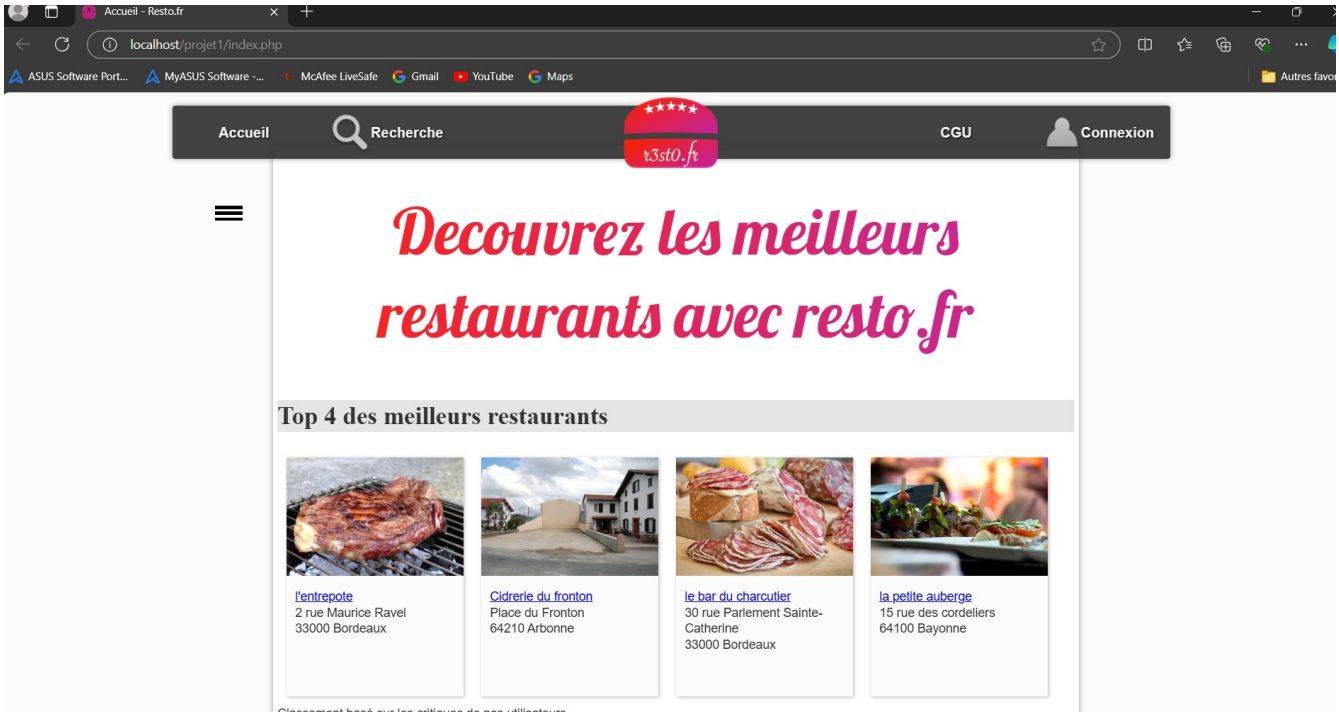
☐	mysql.infoschema	localhost	global	SELECT	Non
☐	resto_util	localhost	spécifique à cette base de données	ALL PRIVILEGES	Non
☐	root	localhost	global	ALL PRIVILEGES	Oui

→ création de l'utilisateur « resto_util »

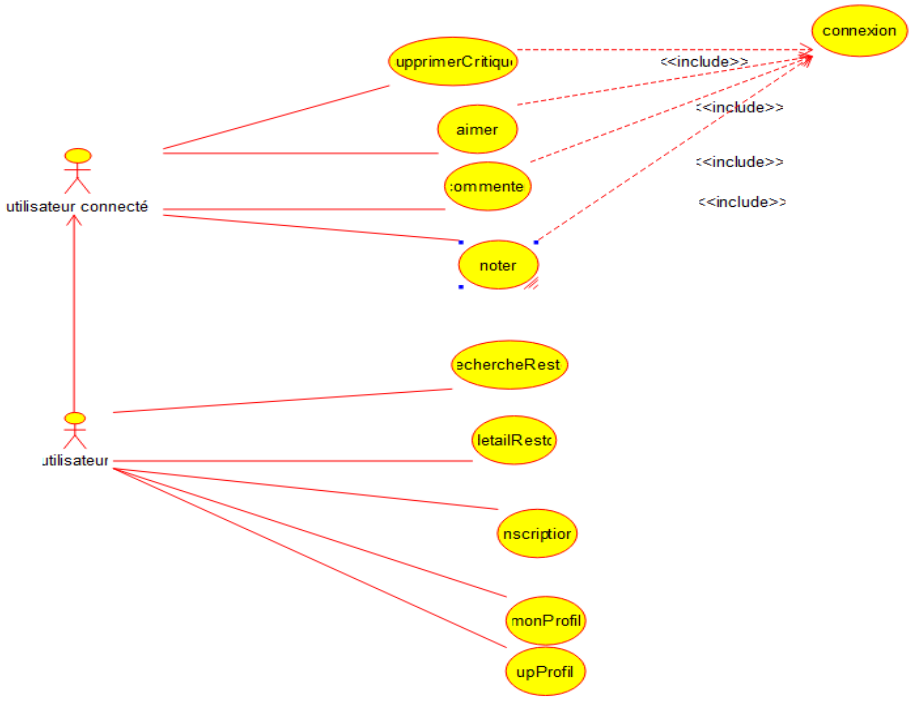
3. Paramétrer l'accès à la base de données par l'application :
Dans le script modele/dao/Bdd.class.php, compléter les lignes suivantes afin d'indiquer les bonnes informations :
private static $login = "";
// login utilisateur de la BDD
private static $mdp = ""; // mdp utilisateur de la BDD
private static $bd = ""; // nom de la BDD
private static $serveur = "";
// nom de domaine du serveur de BDD

```
private static ?PDO $pdo=null;
private static $login = "resto_util";    // login utilisateur de la BDD
private static $mdp = "secret";          // mdp  utilisateur de la BDD
private static $bd = "resto2";           // nom de la BDD
private static $serveur = "localhost";   // nom de domaine du serveur de BDD
private static $pdoOptions = array  (
    PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8"    // pour récupérer les données en UTF8
    ,PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION        // permet la gestion des exceptions
    //,PDO::ATTR_CASE => PDO::CASE_UPPER                  // pour compatibilité avec Oracle
                    );
```

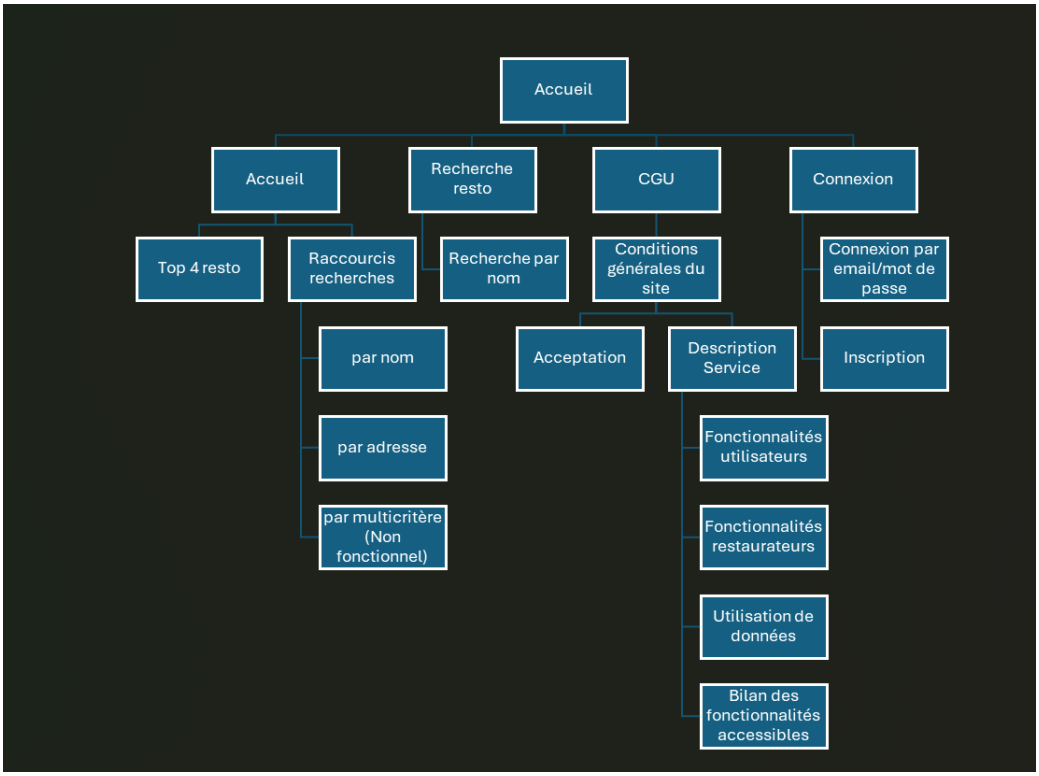
4.



4. Réaliser un diagramme de cas d'utilisation des fonctionnalités existantes



5. Tracer l'arborescence du site en utilisant l'outil dessin de LibreOffice (LibreOffice Draw), ou bien SmartArt sous Word par exemple



II - Architecture MVC en PHP

II.1 Analyse de l'affichage de la liste des restaurants

Travail à faire

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

- include_once : C'est une instruction qui inclut et évalue le fichier spécifié lors de l'exécution du script PHP. Si le fichier a déjà été inclus auparavant dans le même script, il ne le réinclura pas.

- include : Similaire à include_once, mais il peut inclure le même fichier plusieurs fois si nécessaire pendant l'exécution du script.

- require_once : Comme include_once, mais génère une erreur fatale (E_COMPILE_ERROR) si le fichier spécifié n'est pas trouvé. Utile lorsque le fichier inclus est essentiel au bon fonctionnement du script.

- require : Similaire à require_once, mais peut être utilisé plusieurs fois pendant l'exécution du script. Génère une erreur fatale si le fichier spécifié n'est pas trouvé.

- use : Ce mot clé est utilisé pour importer des classes, des fonctions ou des constantes d'un espace de noms spécifique dans le fichier courant, facilitant ainsi l'accès aux éléments définis dans d'autres fichiers ou bibliothèques.

Ces instructions sont essentielles pour l'inclusion de fichiers externes et la gestion des dépendances dans les scripts PHP.

2. Déterminer la valeur de la variable $fichier à la ligne précédant la balise de fin de PHP dans index.php ; moyen : affichage intermédiaire ou bien mode débogage + point d'arrêt

La valeur de la variable $fichier à la ligne précédant la balise de fin de PHP dans index est cgu.php.

J'ai ajouté cette ligne pour afficher la valeur de $fichier :

// Ajout d'un affichage intermédiaire pour débogage

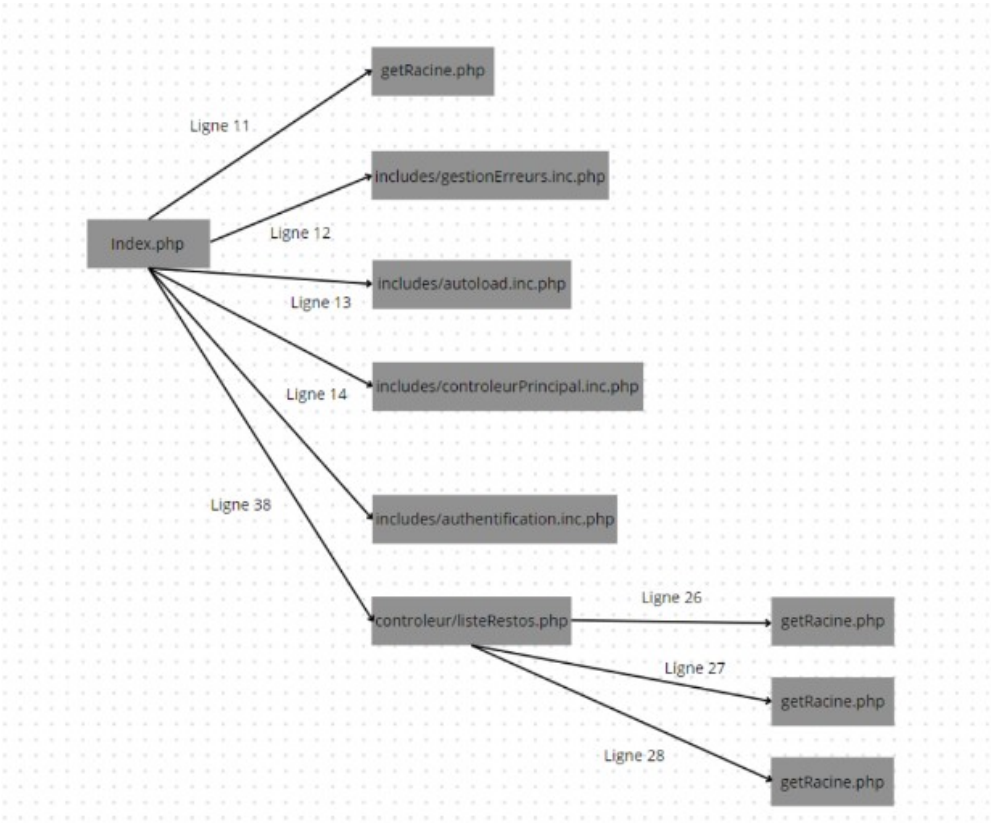
echo "Valeur de \$fichier : $fichier";

Résultat obtenu :

Valeur de $fichier : cgu.php

3. En partant du fichier index.php, schématiser l'ensemble des fichiers utilisés (require ou require_once) pour afficher la liste des restaurants (URL : index.php?action=liste) ;
Le schéma comportera :
- des rectangles portant le nom de chaque fichier,
- des flèches pointant vers le fichier inclus. Sur la flèche, indiquer le n°de ligne de code de l'inclusion

Pour vous aider, utilisez le mode débogage et avancez pas à pas en utilisant step over (F8) ou step into (F7) pour les include.



4. Après avoir observé le code de la classe RestoDAO.class.php, relever l'ensemble des requêtes SQL contenues dans ses méthodes.

Nom de la méthode	Requête SQL
getOneById	"SELECT * FROM resto WHERE idR = :idR"
getAll	"SELECT * FROM resto"
getTop4	"SELECT SUM(note) AS NotesCumulees, r.idR, nomR, numAdrR, voieAdrR, cpR, villeR, latitudeDegR, longitudeDegR, descR, horairesR FROM resto r INNER JOIN critiquer c ON r.idR = c.idR GROUP BY r.idR, nomR, numAdrR, voieAdrR, cpR, villeR, latitudeDegR, longitudeDegR, descR, horairesR ORDER BY NotesCumulees DESC LIMIT 4"
getAllByNomR	"SELECT * FROM resto WHERE nomR LIKE :nomR"
getAllByAdresse	"SELECT * FROM resto WHERE voieAdrR LIKE :voieAdrR AND cpR LIKE :cpR AND villeR LIKE :villeR
getAllMultiCriteres	"SELECT DISTINCT r.* FROM resto r INNER JOIN proposer p ON r.idR = p.idR WHERE (:filtre) OR nomR LIKE :nomR OR voieAdrR LIKE :voieAdrR AND cpR LIKE :cpR AND villeR LIKE :villeR ORDER BY nomR"
getAimesByIdU	"SELECT resto.* FROM resto INNER JOIN aimer ON resto.idR = aimer.idR WHERE idU = :idU"

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

Les points communs de ces requêtes SQL sont :

Table principale : Toutes les requêtes utilisent la table resto.
PDO : Utilisation de la bibliothèque PDO pour préparer et exécuter les requêtes.
Paramètres liés : Utilisation de bindParam pour sécuriser les paramètres des requêtes.
Gestion des erreurs : Utilisation de blocs try-catch pour gérer les exceptions.
Transformation en objets : Conversion des résultats des requêtes en objets Resto.
Sélection : Utilisation principalement de requêtes SELECT.
Filtrage : Utilisation de clauses WHERE pour filtrer les données.
Itération sur les résultats : Bouclage avec while pour traiter les résultats multi-lignes.

Ces points communs montrent une approche cohérente et sécurisée pour interagir avec la base de données, en respectant les bonnes pratiques de programmation en PHP.

II.2 Analyse d’un script du dossier test

Cette section n’est exécutée que lorsque l’on veut effectuer un test unitaire, indépendamment de l’affichage du site web.
Exécutez le script suivant : testRestoDAO.php et observez le code correspondant.

Travail à faire

1. Pour chaque méthode DAO testée, observer le résultat affiché, la requête SQL exécutée, puis expliquer son rôle d’une manière très générale.
Par exemple, la méthode getRestoByIdR() :
la requête SQL permet de récupérer les informations d'un restaurant à partir de son identifiant passé en paramètre ($idR).

- getAll : la requête SQL permet de récupérer les informations de tous les restaurants. Cette fonction permet donc de récupérer les informations de tous les restaurants et d’instancier un objet de la classe métier Resto.

- getTop4 : la requête SQL permet de récupérer la somme des notes, l’id du restaurant, son nom, son numéro, son adresse, code postal, sa ville, sa latitude en degré, sa longitude en degré, sa description, ses horaires des 4 premiers restaurants. Cette fonction permet donc de récupérer les informations principales des 4 premiers restaurants et d’instancier un objet de la classe métier Resto.

- getAllByNomR : la requête SQL permet de récupérer les informations des restaurants filtrés par leur nom. Cette fonction permet donc de récupérer les informations des restaurants par nom et d’instancier un objet de la classe métier Resto.

- getAllByAdresse : la requête SQL permet de récupérer les informations des restaurants filtrés par leur adresse. Cette fonction permet donc de récupérer les informations des restaurants filtrés par adresse et d’instancier un objet de la classe métier Resto.

- getRestosByMultiCriteres : aucun affichage.

Cette fonction permet donc de récupérer les informations d'un restaurant donné et d’instancier un objet de la classe métier Resto.

2. Quelle méthode de la classe RestoDAO est utilisée dans listeRestos.php ?

La méthode de la classe RestoDAO «getAll » est utilisée dans listeRestos.php.

II.3 Les couches « vue et « modèle »

L’application web fournie en ressources respecte le « design pattern » MVC ou Modèle Vue Contrôleur. Dans ce patron de conception, les différentes "couches" permettant de construire un site sont gérées de façon indépendante. Découvrons à quoi servent les couches "Vue" et "Modèle".

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

Travail à faire

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

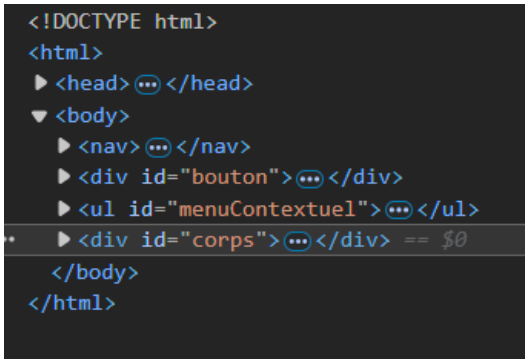
Les fichiers contenues dans les dossiers controleur et modele (dao/metier) sont entièrement rédigés en php, sans aucune trace d'éléments de CSS ou de HTML.

2. Dans quel dossier sont contenus les fichiers produisant du code HTML et CSS ?

Dans le ficher « css » pour le CSS et « vue » pour le HTML.

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

En inspectant la page affichant le listes des restaurants, nous pouvons constater la présence d'éléments HTML et mais pas de PHP.



4. Rappeler le point commun entre toutes les méthodes de classe RestoDAO. du dossier modele/dao.

Ces fonctions sont déclarées comme « static ».

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

Seuls les fichiers du dossiers dao font référence à la base de données

Synthèse

6. Expliquer globalement le rôle des scripts contenus dans les dossiers suivants

◦ modele/dao

Gère l'accès aux données de l'application (CRUD, requêtes SQL, etc.)

◦ modele/metier

Définit les objets métier de l'application (entités du domaine, propriétés et méthodes)

◦ vue

Présente les données de l'application à l'utilisateur (templates HTML, scripts JavaScript, etc.)

II.4 La couche contrôleur

II.4.1 - Le contrôleur listeRestos.php

Travail à faire

1. Quelle méthode DAO est utilisée dans ce contrôleur ? Que permet-elle de récupérer dans la base de données ?

Dans ce fichier est utilisé la méthode getAll() du fichier RestoDAO et permet de retourner tous les restaurants.

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

La liste (array) des restaurant est affiché à l'écran à l'aide du fichier « vueListeRestos.php ».

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

```
// Construction de la vue
$titre = "Liste des restaurants répertoriés";
require_once "$racine/vue/entete.html.php";
require_once "$racine/vue/vueListeRestos.php";
require_once "$racine/vue/pied.html.php";
```

Ces scriptes permettent de construire la vue de la page (l'entête, la liste et le pied de page).

II.4.2 - Le contrôleur detailResto.php

Travail à faire

1. Quelle donnée est transmise au contrôleur en méthode GET ?

La donnée transmise au contrôleur en méthode GET est idR, qui est l'identifiant du restaurant à afficher.

2. Rappeler le rôle de la méthode getOneById. Pourquoi cette méthode a-t-elle besoin d'un paramètre ?
La variable $unResto reçoit le résultat de l'appel à getOneById.

Le rôle de la méthode getOneById est de récupérer un restaurant à partir de son identifiant.
Elle a besoin d'un paramètre ($idR) pour sélectionner le restaurant spécifique.

3. Explorer les fichiers "vue" inclus à la fin du fichier contrôleur et repérer les lignes où cette variable est utilisée.

vueAccueil.php : ligne 29

vueListeRestos.php : ligne 26

4. Le contrôleur gère-t-il directement l’accès aux données ?

Non, il délègue l'accès aux données aux classes DAO (RestoDAO, CritiqueDAO).

II.4.3 - Synthèse

1. 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.

Les données créées ou récupérées dans le contrôleur sont transmises à la vue, qui est responsable de les afficher à l'utilisateur.

2. Comment ces données sont transmises au contrôleur :
◦de l'utilisateur vers le contrôleur ?

Les données sont transmises via des requêtes HTTP par exemple, lorsque l'utilisateur soumet un formulaire). Le contrôleur reçoit ces données via des variables superglobales

◦de la base de données vers le contrôleur ?

Les données sont transmises via des objets DAO (Data Access Object), qui gèrent l'accès à la base de données. Le contrôleur utilise ces objets DAO pour récupérer les données de la base de données.

3. Résumer le rôle de la couche « contrôleur »

Le contrôleur est la couche qui gère les interactions entre l'utilisateur et l'application. Il reçoit les données de l'utilisateur et de la base de données, les traite et les transmet à la vue pour affichage. Le contrôleur est responsable de :

- Recevoir les requêtes de l'utilisateur
- Récupérer les données de la base de données via les objets DAO
- Traiter les données et les règles métier
- Transmettre les données à la vue pour affichage
- Gérer les erreurs et les exceptions

III - Synthèse globale

MVC est l'acronyme de Modèle Vue Contrôleur. Dans le domaine du développement d'applications, MVC est un patron de conception. C’est une bonne pratique de développement d'une application.
Son application apporte plusieurs avantages :
• faciliter le travail en équipe : les composants peuvent être écrits par différentes personnes ;
• faciliter le test unitaire : 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.

Chaque fonctionnalité, par exemple l'affichage de la liste des restaurants, 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.
Principes de base du MVC :
• le contrôleur et le modèle ne procèdent à aucun affichage : c’est le rôle de la vue ;
• les requêtes SQL sont exclusivement situées dans les méthodes de la couche DAO ;
• les vues ne doivent contenir que du code d'affichage de données provenant du contrôleur, ou éventuellement du modèle (des classes métier)