

# Projet site web R3st0.fr

## Itération n°1 – Ticket n°2 – étude de l’existant

### Table des matières

I - Rappel du contexte.....2

II - Étude des fonctionnalités existantes.....2

III - Architecture MVC en PHP .....3

    III.1 Analyse de l’affichage de la liste des restaurants.....3

    III.2 Analyse d’un script du dossier test.....4

    III.3 Les couches « vue et « modèle » .....5

    III.4 La couche contrôleur.....5

        III.4.1 - Le contrôleur listeRestos.php .....5

        III.4.2 - Le contrôleur detailResto.php.....5

        III.4.3 - Synthèse.....6

IV - Synthèse globale.....6

## I - Rappel du contexte

R3st0.fr est un site web de critique de restaurants. 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://www3.jolsio.net/test2slam/public/siteresto> (adresse interne uniquement)

## II - 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 ;

Nous avons bien paramétré le projet pour qu'il puisse être lancé sur :

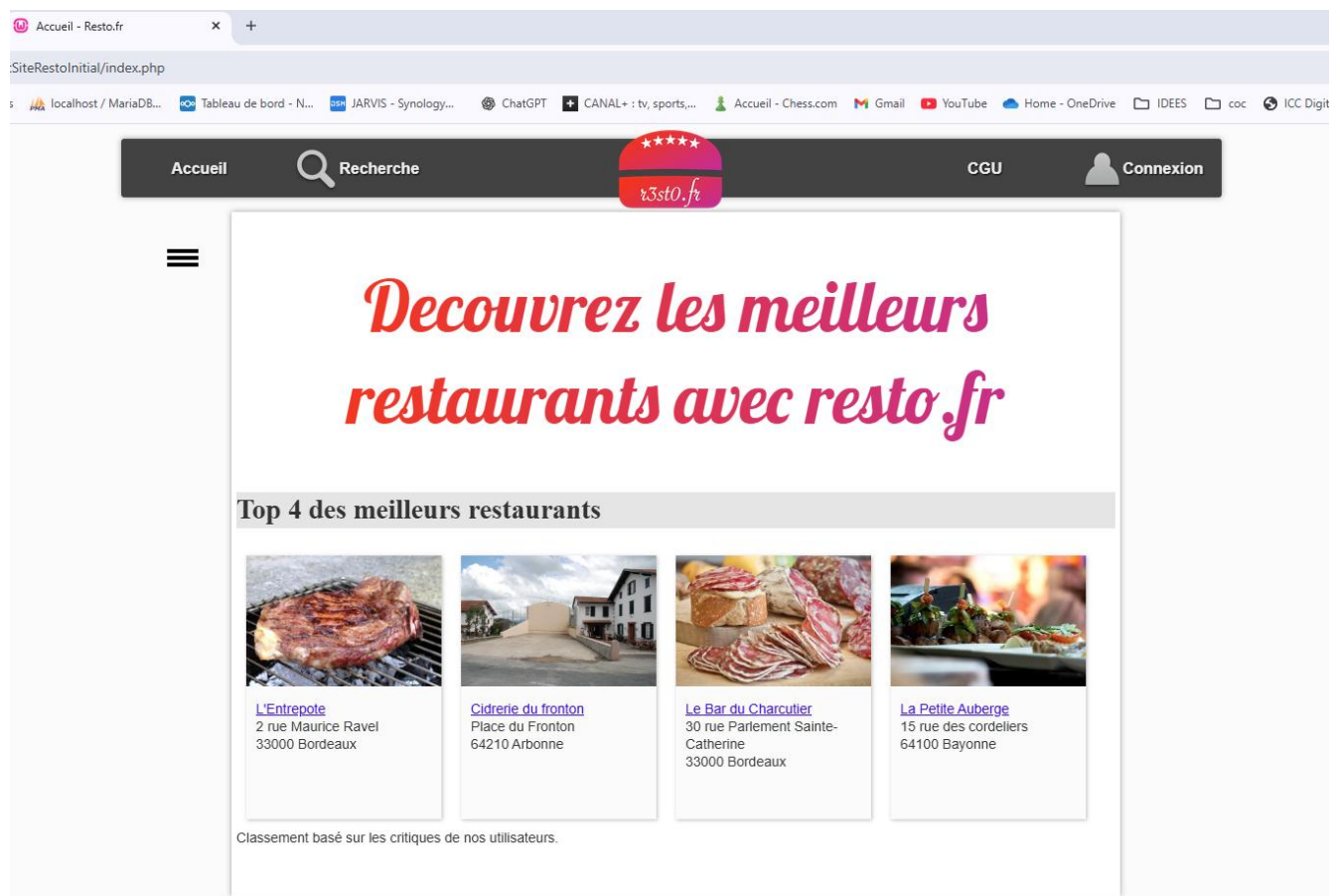


Figure 1 : Vue de la page d'accueil

Dans NetBeans, il faut aller dans service et se connecter à la base de données. Pour que le site s'exécute correctement il faut mettre la copie du projet sous wamp64/www.

Source Folder: C:\Users\jade\OneDrive\Bureau\La Joiverie\2. annee\AP\Projet\PHP\ProjectSiteRestoInitial

Web Root: <Source Folder>

☒ Copy files from Sources Folder to another location

Copy to Folder: C:\wamp64\www\PhpProjectSiteRestoInitial

☐ Copy files on project open

Encoding: UTF-8

PHP Version: PHP 7.4

Configuration: PHP version is used only for hints

Run As: Local Web Site (running on local web server)

Project URL: http://localhost/PhpProjectSiteRestoInitial

Index File: index.php

Arguments:

http://localhost/PhpProjectSiteRestoInitial/index.php

Figure 2 : Réglage du projet

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

Nous avons bien créé la base de données resto2 comme on peut le voir ci-dessous en prenant ce fichier du projet fournit :

```
1  -- phpMyAdmin SQL Dump
2  -- version 5.0.4
3  -- https://www.phpmyadmin.net/
4  --
5  -- Hôte : localhost
6  -- Généré le : mer. 07 juil. 2021 à 14:01
7  -- Version du serveur : 10.4.17-MariaDB-log
8  -- Version de PHP : 7.4.14
9
10 SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
11 START TRANSACTION;
12 SET time_zone = "+00:00";
13
14
15 /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
16 /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
17 /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
18 /*!40101 SET NAMES utf8mb4 */;
19
20 --
21 -- Base de données : resto2
22 -- évolution de la base resto :
23 -- le champ mailU n'est plus la clef primaire de la table utilisateur, c'est un champ autoincrémenté (idU)
24 -- les tables aimer, critiquer, preferer ont été modifiées en conséquence
25 -- les contraintes de clef étrangères de ces tables ont été également adaptées
26 --
27 CREATE DATABASE IF NOT EXISTS resto2 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
28 USE resto2;
```

Figure 3 : Script sql à exécuter pour bdd resto2

Puis on a lancé sur l'icône jaune pour l'exécuter.

On peut voir qu'on voit la bdd est bien exécuter et qu'on peut l'avoir sur phpMyAdmin :

The screenshot shows the PHPMyAdmin interface for a MySQL database named 'resto2'. The 'Structure' tab is selected, displaying a table of database tables. The table has columns: Table, Action, Lignes, Type, Interclassement, Taille, and Perte. It lists five tables: 'aimer', 'critiquer', 'photo', 'resto', and 'utilisateur', each with its respective row count, storage engine (InnoDB), character set (utf8mb4\_0900\_ai\_ci), and size. A summary row at the bottom shows the total size of the database as 144,0 kio.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
aimer	Parcourir, Structure, Rechercher, Insérer, Vider, Supprimer	16	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
critiquer	Parcourir, Structure, Rechercher, Insérer, Vider, Supprimer	19	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
photo	Parcourir, Structure, Rechercher, Insérer, Vider, Supprimer	14	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
resto	Parcourir, Structure, Rechercher, Insérer, Vider, Supprimer	11	InnoDB	utf8mb4_0900_ai_ci	16,0 kio	-
utilisateur	Parcourir, Structure, Rechercher, Insérer, Vider, Supprimer	7	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	-
5 tables	Somme	67	MyISAM	utf8mb4_unicode_ci	144,0 kio	0 o

Figure 4 : vue de la BDD resto2 sur PHPMyAdmin

Voici la vue MLD :

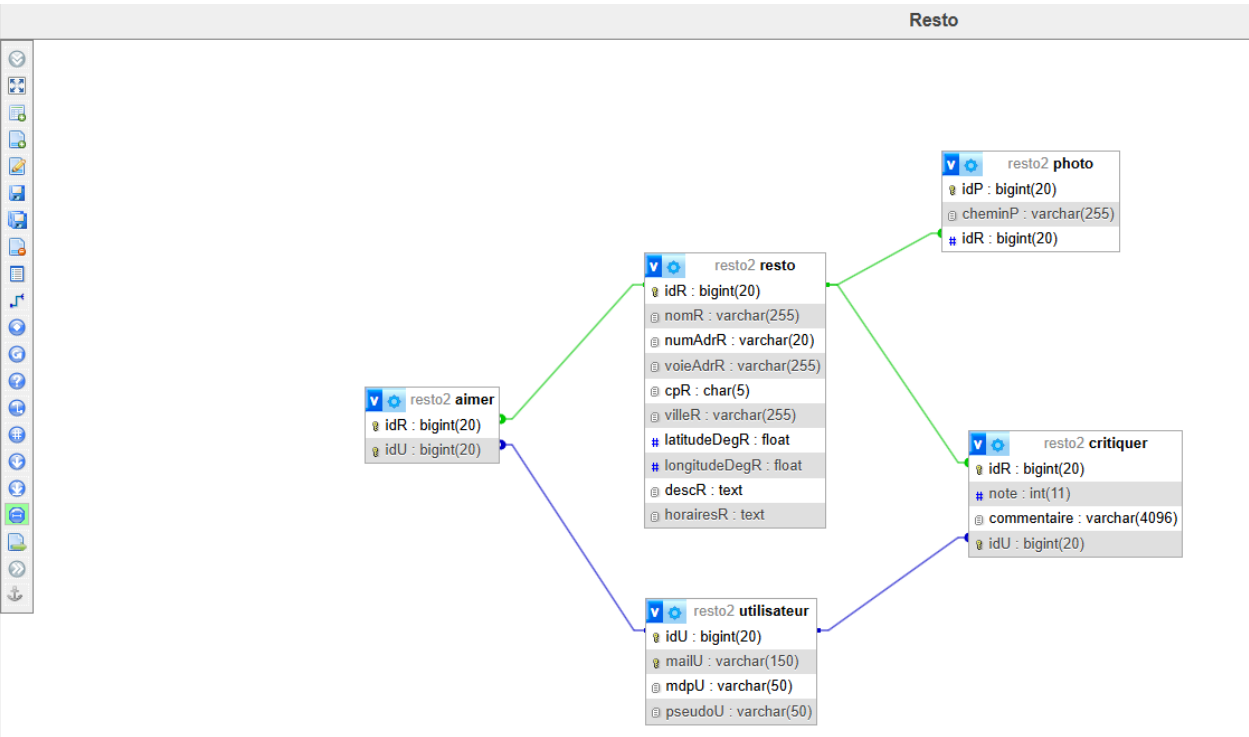
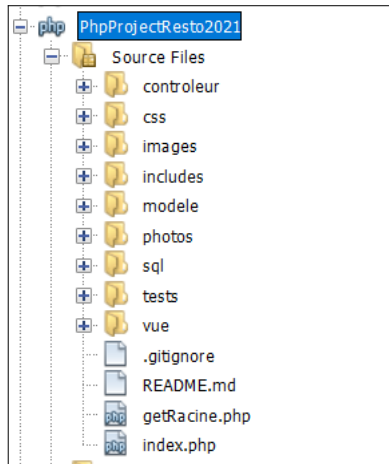


Figure 5 : Vue MLD de la bdd sur PHPMyAdmin

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
```



Arborescence des dossiers du projet

En allant voir ce fichier dans l'arborescence du projet, cette partie était déjà complétée comme on peut le voir ci-dessous :

```
/**
 * @var PDO Objet de type PDO, dépositaire de la connexion courante à la 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 $port = "3307";
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 database (noms de champs transcrits en majuscules)
);
```

Figure 6 : partie configuration projet

Mais en rencontrant des problèmes sur un autre PC on a dû rajouter le port 3307 ou 3306 selon les configurations comme on peut le voir ci-dessous (rajouter la ligne : `self::$pdo .....`) et également ci-dessus la ligne `$port = "3307"` ou 3306 si autre :

```
/**
 * Crée un objet de type PDO et ouvre la connexion
 * @return \PDO un objet de type PDO pour accéder à la base de données
 * @throws PDOException
 */
public static function connecter() : ?PDO {
    // on ne crée une connexion que si elle n'existe pas déjà ...
    if (is_null(self::$pdo)) {
        // instantiation d'un objet de connexion PDO
        self::$pdo = new PDO("mysql:host=".self::$serveur.";dbname=".self::$bd.";port=".self::$port, self::$login, self::$mdp, self::$pdoOptions);
    }
    return self::$pdo;
}
```

Figure 7 : partie configuration projet 2

Et si on teste le fichier de test de BDD on peut voir que cela fonctionne bien :

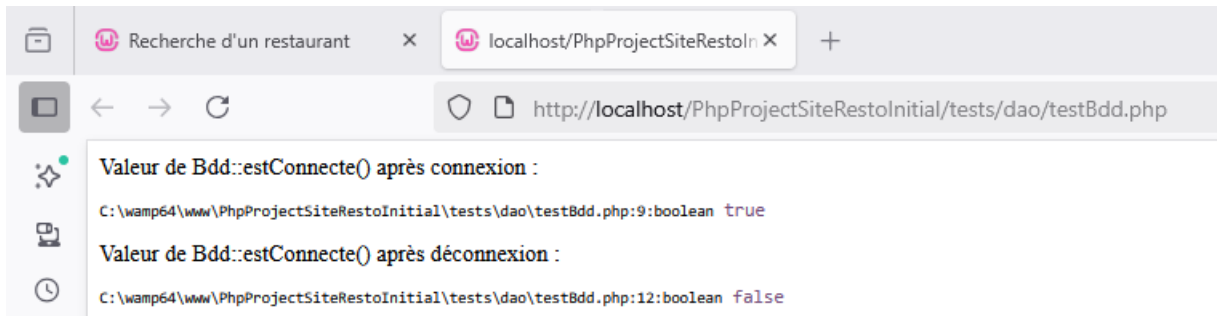


Figure 8 : test de connexion avec le fichier testBDD.php

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

**(début à vérifier avec le groupe)**

Voici le diagramme de cas d'utilisation sur les différentes fonctionnalités existantes :

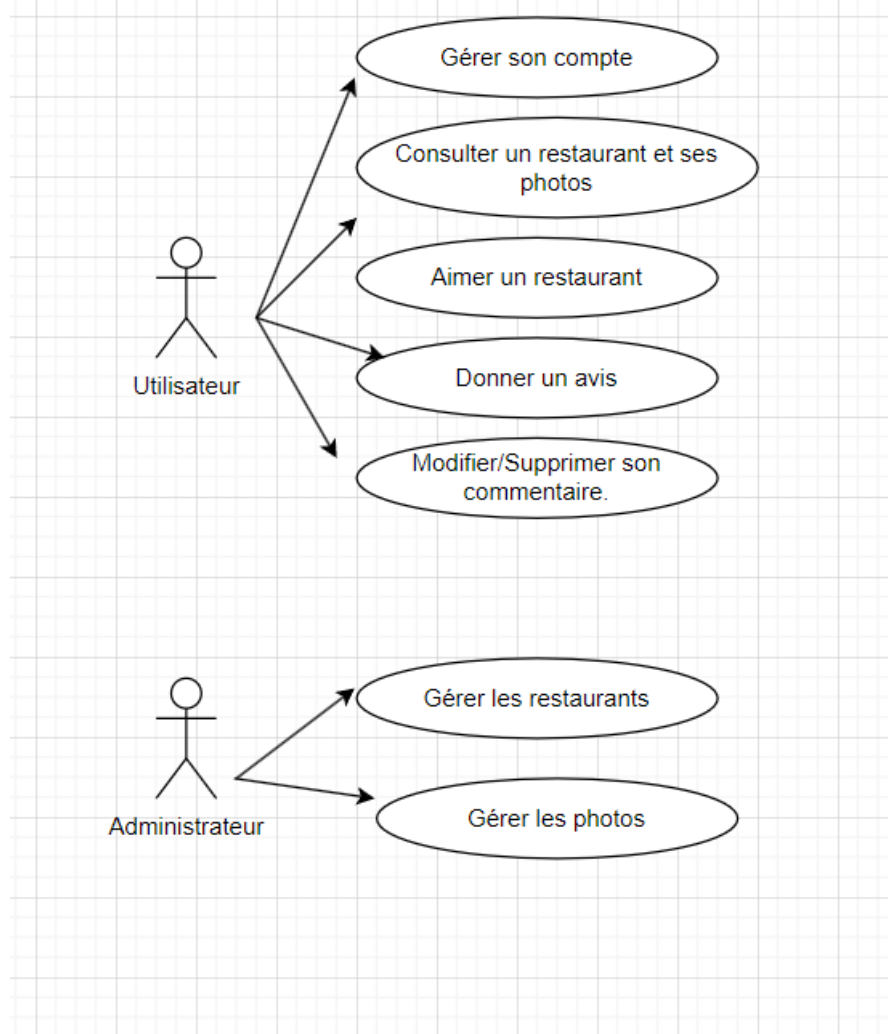


Figure 9 : Diagramme des cas d'utilisation

(voir fichier sur git pour le télécharger sur l'itération 1 du ticket 2.

## III - Architecture MVC en PHP

Découverte générale du patron de conception MVC.

### III.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** : Inclut et exécute le fichier spécifié **une seule fois** pendant l'exécution du script. Si le fichier a déjà été inclus auparavant, PHP ne l'inclura pas de nouveau, ce qui permet d'éviter les erreurs de re-déclaration de fonctions, classes ou constantes. Si le fichier n'existe pas, PHP génère un **warning** (`E_WARNING`) et le script continue son exécution.

**include** : Inclut et exécute le fichier spécifié dans le script courant. Contrairement à `include_once`, le même fichier peut être inclus plusieurs fois, ce qui peut provoquer des erreurs si des fonctions, classes ou constantes sont redéclarées. Si le fichier n'existe pas, PHP génère un **warning** (`E_WARNING`) mais continue l'exécution du script.

**require\_once** : Fonctionne comme `require`, mais **n'inclut le fichier qu'une seule fois**. Utile pour éviter la re-déclaration de classes, fonctions ou constantes. Si le fichier n'existe pas, PHP génère une **erreur fatale** (`E_COMPILE_ERROR` avant PHP 8, `E_ERROR` depuis PHP 8) et **arrête l'exécution du script**.

**require** : Inclut et exécute le fichier spécifié dans le script courant. Contrairement à `include`, si le fichier n'existe pas, PHP génère une **erreur fatale** et arrête le script.

**use** : Permet d'importer un **namespace** pour accéder à une classe, fonction ou constante sans écrire son chemin complet et d'inclure un **trait** dans une classe pour réutiliser ses méthodes et propriétés. `use` **n'inclut pas de fichiers**, il est uniquement lié à la résolution de noms dans le code.

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*

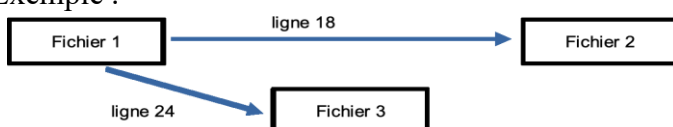
La valeur de la variable `$fichier` correspond au contrôleur principal du paramètre `action`, c'est-à-dire que `fichier` est remplacé par la valeur de l'action, donc par le nom d'un fichier. Par exemple : si `action = accueil.php`, alors le chemin `$racine/contrôleur/$fichier = $racine/contrôleur/accueil.php`.

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](http://index.php?action=liste)) ;

Le schéma comportera :

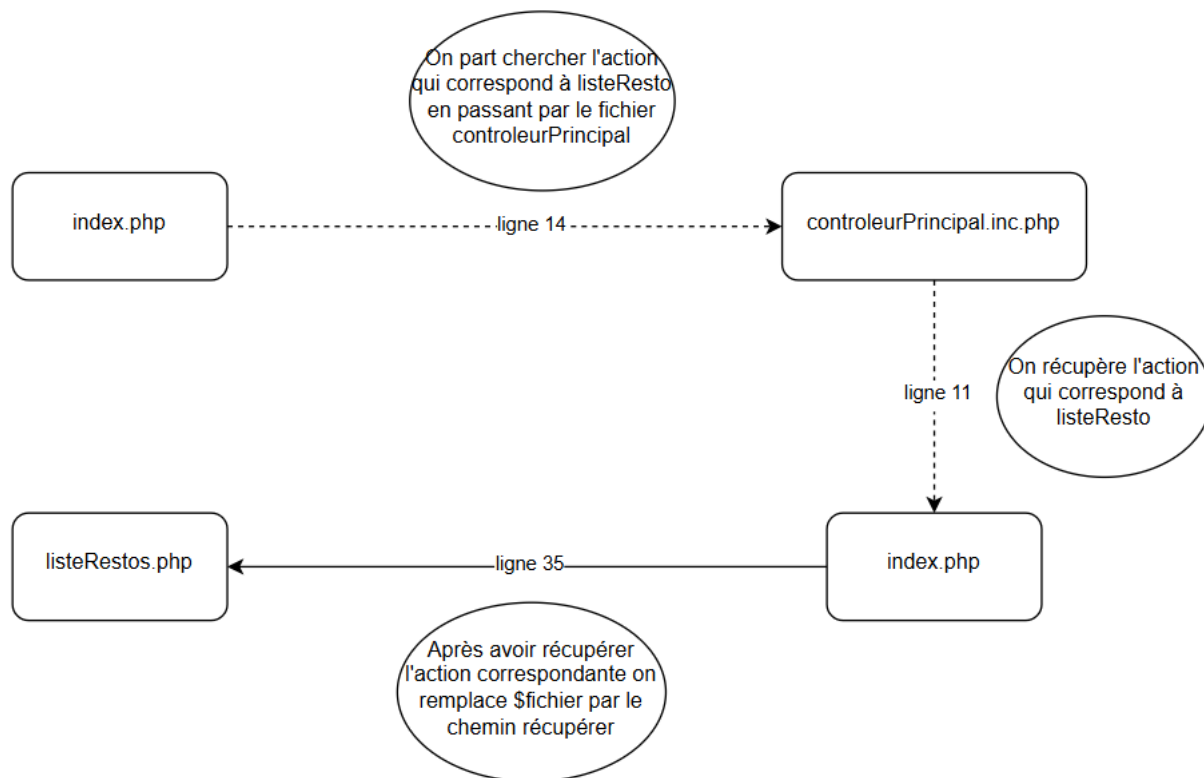
- a. des rectangles portant le nom de chaque fichier,
- b. des flèches pointant vers le fichier inclus. Sur la flèche, indiquer le n° de ligne de code de l'inclusion

Exemple :



*Commentaire : Le fichier 1 inclus le fichier 2 à la ligne 18*

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



- 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
enregistrementVersObjet()	/

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

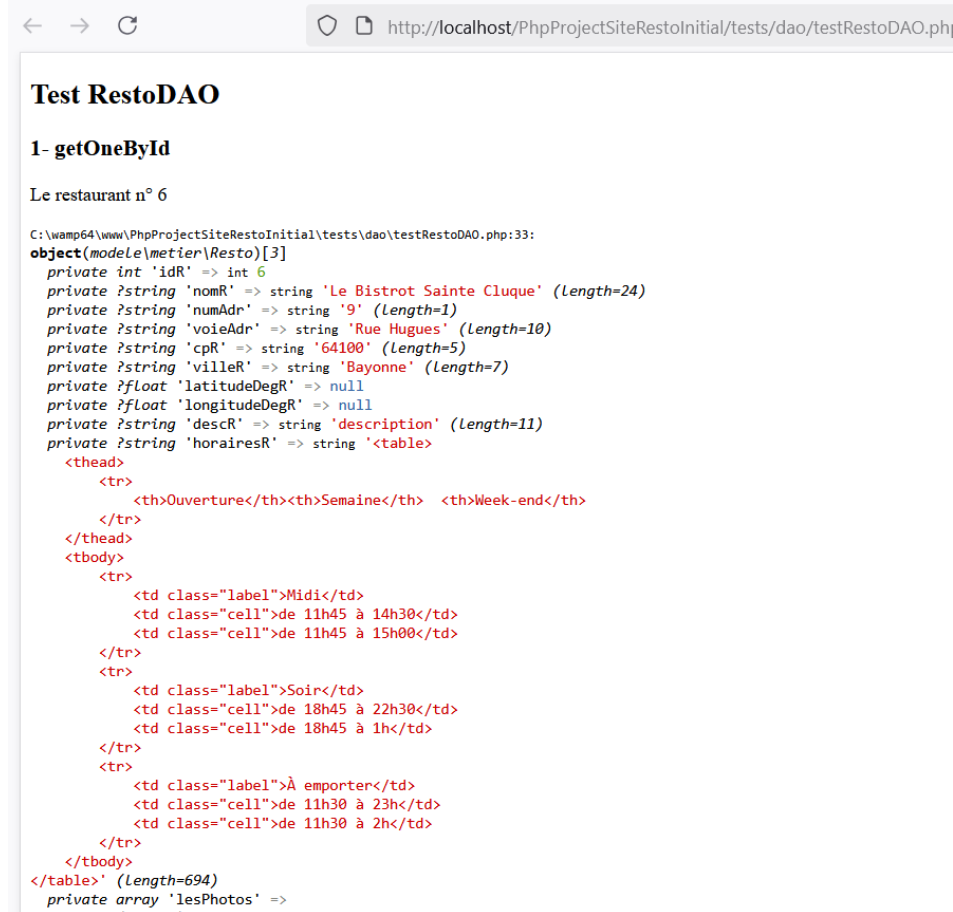
Toutes les requêtes SQL de RestoDAO portent sur la table resto, sélectionnent soit toutes les colonnes soit des colonnes spécifiques, utilisent des filtres via WHERE et parfois des jointures (INNER JOIN) avec d'autres tables (critiquer, proposer, aimer). Elles sont préparées avec des paramètres PDO pour sécuriser les entrées et certaines appliquent des GROUP BY ou ORDER BY. L'objectif commun est de récupérer les données nécessaires pour instancier des objets Resto.

### III.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.

On a bien testé la page testRestoDAO.php comme on peut le voir ci-dessous. Cette page une fois exécuter elle teste toutes les méthodes de RestoDAO et affiche leur contenu complet pour vérification.



```
C:\wamp64\www\ PhpProjectSiteRestoInitial\tests\dao\testRestoDAO.php:33:
object(model\metier\Resto)[3]
  private int 'idR' => int 6
  private ?string 'nomR' => string 'Le Bistrot Sainte Cluque' (length=24)
  private ?string 'numAdr' => string '9' (length=1)
  private ?string 'voieAdr' => string 'Rue Hugues' (length=10)
  private ?string 'cpR' => string '64100' (length=5)
  private ?string 'villeR' => string 'Bayonne' (length=7)
  private ?float 'latitudeDegR' => null
  private ?float 'longitudeDegR' => null
  private ?string 'descR' => string 'description' (length=11)
  private ?string 'horairesR' => string '<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>' (length=694)
  private array 'lesPhotos' =>
    ...
  }
```

Figure 10 : début de page TestRetsoDao

#### 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). Le résultat d'exécution du test unitaire est le suivant :*

```
1- getOneById
Le restaurant n° 6
/testes/dao/testRestoDAO.php:33:
object(model\metier\Resto)[3]
  private int 'idR' => int 6
  private ?string 'nomR' => string 'Le Bistrot Sainte Cluque' (length=24)
  private ?string 'numAdr' => string '9' (length=1)
  private ?string 'voieAdr' => string 'Rue Hugues' (length=10)
  private ?string 'cpR' => string '64100' (length=5)
  private ?string 'villeR' => string 'Bayonne' (length=7)
  private ?float 'latitudeDegR' => null
  private ?float 'longitudeDegR' => null
  private ?string 'descR' => string 'description' (length=11)
  private ?string 'horairesR' => string '<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>
  [...]
</tbody>
</table>' (length=694)
private array 'lesPhotos' =>
  array (size=1)
    0 =>
      object (modele\metier\Photo) [7]
        private int 'idP' => int 8
        private string 'cheminP' => string 'leBistrotSainteCluque.jpg' (length=25)
private array 'lesCritiques' =>
  array (size=1)
    0 =>
      object (modele\metier\Critique) [6]
        private ?int 'note' => int 4
        private ?string 'commentaire' => string 'Cuisine de qualité.' (length=20)
        private modele\metier\Utilisateur 'leUtilisateur' =>
          object (modele\metier\Utilisateur) [5]
            private int 'idU' => int 5
            private string 'mailU' => string 'nicolas.harispe@gmail.com' (length=25)
            private ?string 'mdpU' => string '$1$zvNDSFQsdfqsDfQsdfsT.' (length=24)
            private ?string 'pseudoU' => string 'Nico40' (length=6)
            private array 'lesRestosAimes' =>
              array (size=0)
                empty
```

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

```
try {
    Bdd::connecter();
    ?>
    <h2>Test RestoDAO</h2>

    <h3>1- getOneById</h3>
    <?php $idR = 6; ?>
    <p>Le restaurant n° <?= $idR ?></p>
    <?php
    $leResto = RestoDAO::getOneById($idR);
    var_dump($leResto);

    ?>

    <h3>2- getAll</h3>
    <p>Tous les restaurants</p>
    <?php
    $lesRestos = RestoDAO::getAll();
    var_dump($lesRestos);

    ?>

    <h3>3- getTop4</h3>
    <p>Les quatre restaurants les mieux notés</p>
    <?php
    $lesRestos = RestoDAO::getTop4();
    var_dump($lesRestos);

    ?>

    <h3>5- getAllByNom</h3>
    <p>Filtrage des restaurants sur le nom</p>
    <?php
    $lesRestos = RestoDAO::getAllByNomR("pot");
    var_dump($lesRestos);

    ?>

    <h3>6- getAllByAdresse</h3>
    <p>Filtrage des restaurants sur l'adresse</p>
    <?php
    $lesRestos = RestoDAO::getAllByAdresse("du", "", "Bay");
    var_dump($lesRestos);

    ?>

    ?>

    <h3>7- getRestosByMultiCriteres</h3>
    <p>Filtrage des restaurants multicritères</p>
    <?php
    $lesRestos = RestoDAO::getAllMultiCriteres("pot", "du", "", "Bay", array(3, 6));
    var_dump($lesRestos);

    ?>

    <h3>8- getAimesByIdU</h3>
    <?php $unIdU = 7; ?>
    <p>Liste des restaurants aimés par l'utilisateur <?= $unIdU ?> (chargement de type "lazy")</p>
    <?php
    $lesRestos = RestoDAO::getAimesByIdU($unIdU);
    var_dump($lesRestos);

    ?>

    <?php
    Bdd::deconnecter();
} catch (Exception $ex) {
    ?>
    <h4>*** Erreur récupérée : <br/> <?= $ex->getMessage() ?> <br/>***</h4>
    <?php
}
?>
```

Le test DAO exécute toutes les requêtes SQL écrites dans le tableau partie 3.1 question 4.

## Test AimerDAO

### 1- estAimeByIdU

L'utilisateur d'id 3 aime le restaurant d'id 1 ?

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testAimerDAO.php:26:boolean true
```

L'utilisateur d'id 6 aime le restaurant d'id 1 ?

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testAimerDAO.php:31:boolean false
```

### 2- insert

Réussite de l'ajout :

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testAimerDAO.php:40:boolean true
```

Après ajout, L'utilisateur d'id 6 aime-t-il le restaurant d'id 1 ?

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testAimerDAO.php:43:boolean true
```

### 3- delete

Réussite de la suppression :

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testAimerDAO.php:52:boolean true
```

Après suppression, L'utilisateur d'id 6 aime-t-il le restaurant d'id 1 ?

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testAimerDAO.php:55:boolean false
```

Le résultat afficher par le teste DAO aimer, teste pour certain id utilisateur si, il aime le restaurant de l'id restaurant choisi. Un boolean est renvoyer True si l'id de l'utilisateur aime l'id restaurant et False s'il n'aime pas.

<?php

```
use modele\dao\Bdd;

require_once '../includes/autoload.inc.php';
require_once '../includes/authentication.inc.php';

Bdd::connecter();
// test d'authentification applicative
//login("mathieu.capliez@gmail.com", "Passel?");
$mailU = "test@bts.sio";
echo "Tentative de connexion de $mailU<br/>";
login($mailU, "sio");
if (isLoggedIn()) {
    echo "Ok, $mailU is logged<br/>";
} else {
    echo "Problème, $mailU is not logged<br/>";
}

// deconnexion
echo "Tentative de déconnexion<br/>";
logout();
if (isLoggedIn()) {
    echo "Problème, $mailU is always logged<br/>";
} else {
    echo "Ok, $mailU is no more logged<br/>";
}

echo "Tentative de connexion avec un login inexistant<br/>";
$mailU = "inconnu@bts.sio";
login($mailU, "1234");
if (isLoggedIn()) {
    echo "Problème, $mailU ne devrait pas être connecté<br/>";
} else {
    echo "Ok, $mailU n'est pas connecté<br/>";
}
```

```
Tentative de connexion de test@bts.sio
Ok, test@bts.sio is logged
Tentative de déconnexion
Ok, test@bts.sio is no more logged
Tentative de connexion avec un login inexistant
Ok, inconnu@bts.sio n'est pas connecté
```

Teste de connexion avec login existant puis inexistant, si le login existe affiche Ok, is log, sinon affiche

Ok,....., n'est pas connecté.

```
<?php

] use modele\dao\Bdd;
  require_once '../includes/autoload.inc.php';

] try{
  $cnx = Bdd::connecter();
  echo "Valeur de Bdd::estConnecte() après connexion :";
  var_dump(Bdd::estConnecte());
  Bdd::deconnecter();
  echo "Valeur de Bdd::estConnecte() après déconnexion :";
  var_dump(Bdd::estConnecte());
} catch (PDOException $ex) {
    echo "*** échec de la connexion à la BDD *** : ".$ex->getMessage();
}
}
```

Valeur de Bdd::estConnecte() après connexion :

C:\wamp64\www\PhpProjectsSiteRestoInitial\tests\dao\testBdd.php:9:boolean true

Valeur de Bdd::estConnecte() après déconnexion :

C:\wamp64\www\PhpProjectsSiteRestoInitial\tests\dao\testBdd.php:12:boolean false

Le teste permet de vérifier si la base de données est connectée, renvoie un boolean, si la base de données est connectée renvoie True, sinon renvoie False.

```
?>
<h2>Test CritiqueDAO</h2>

<h3>1- getNoteMoyenneByIdR</h3>
<?php $id = 1; ?>
<p>La note moyenne des critiques du restaurant n° <?= $id ?></p>
<?php
$noteMoyenne = CritiqueDAO::getNoteMoyenneByIdR($id);
var_dump($noteMoyenne);
?>

<?php $id = 7; ?>
<p>La note moyenne des critiques du restaurant n° <?= $id ?></p>
<?php
$noteMoyenne = CritiqueDAO::getNoteMoyenneByIdR($id);
var_dump($noteMoyenne);
?>

<h3>2- getOneById</h3>
<?php $idR = 6;
$idU = 5;
?>
<p>La critique de l'utilisateur n° <?= $idU ?> pour le restaurant n° <?= $idR ?></p>
<?php
$laCritique = CritiqueDAO::getOneById($idR, $idU);
var_dump($laCritique);
?>

<h3>3- getAllByResto</h3>
<?php $idR = 2; ?>
<p>Les critiques pour le restaurant n° <?= $idR ?></p>
<?php
$desCritiques = CritiqueDAO::getAllByResto($idR);
var_dump($desCritiques);
?>
```

```
<h3>4- insert</h3>
<?php
$unIdR = 1;
$unIdU = 1;
$user = UtilisateurDAO::getOneById($unIdU);
$laCritique = new Critique(4, "Supercalifragilistic", $user);
$ok = CritiqueDAO::insert($unIdR, $laCritique);
?>
Réussite de l'ajout :
<?php var_dump($ok); ?>
<p>Après ajout, voici la nouvelle critique pour le restaurant d'id <?= $unIdR ?> ? </p>
<?php
var_dump(CritiqueDAO::getOneById($unIdR, $user->getIdU()));
?>

<h3>5- update</h3>
<?php
// On modifie la critique
$laCritique->setNote(3);
$laCritique->setCommentaire("expidelilicious");
$ok = CritiqueDAO::update($unIdR, $laCritique);
?>
Réussite de la modification :
<?php var_dump($ok); ?>
<p>Après modification, voici la critique pour le restaurant d'id <?= $unIdR ?> ? </p>
<?php
var_dump(CritiqueDAO::getOneById($unIdR, $user->getIdU()));
?>
```



# TAILLE JADE

## DELAUNAY-GUITTON Benjamin

## PORTOLLEAU Anaïs

### Test CritiqueDAO

#### 1- getNoteMoyenneByIdR

La note moyenne des critiques du restaurant n° 1

```
C:\wamp64\www\PHPProjectSiteRestoInitial\tests\dao\testCritiqueDAO.php:29:float 3.8
```

La note moyenne des critiques du restaurant n° 7

```
C:\wamp64\www\PHPProjectSiteRestoInitial\tests\dao\testCritiqueDAO.php:35:float 4.5
```

#### 2- getOneById

La critique de l'utilisateur n° 5 pour le restaurant n° 6

```
C:\wamp64\www\PHPProjectSiteRestoInitial\tests\dao\testCritiqueDAO.php:45:
object(modele\metier\Critique)[4]
  private ?int 'note' => int 4
  private ?string 'commentaire' => string 'Cuisine de qualité.' (length=20)
  private modele\metier\Utilisateur 'leUtilisateur' =>
    object(modele\metier\Utilisateur)[3]
      private int 'idu' => int 5
      private string 'mailU' => string 'nicolas.harispe@gmail.com' (length=25)
      private ?string 'mdpU' => string '$1$zVND5FQ5dfqsDFQsdfst.' (length=24)
      private ?string 'pseudoU' => string 'Nico40' (length=6)
      private array 'lesRestosAimes' =>
        array (size=0)
          empty
```

#### 3- getAllByResto

Les critiques pour le restaurant n° 2

```
C:\wamp64\www\PHPProjectSiteRestoInitial\tests\dao\testCritiqueDAO.php:53:
array (size=4)
  0 =>
    object(modele\metier\Critique)[6]
      private ?int 'note' => int 2
      private ?string 'commentaire' => string 'bof.' (length=4)
      private modele\metier\Utilisateur 'leUtilisateur' =>
        object(modele\metier\Utilisateur)[5]
          private int 'idu' => int 2
          private string 'mailU' => string 'jj.soueix@gmail.com' (length=19)
          private ?string 'mdpU' => string '$1$zVNSHYHISDFG5DGJqJSDJF.' (length=28)
          private ?string 'pseudoU' => string 'drskott' (length=7)
          private array 'lesRestosAimes' =>
            array (size=0)
              empty
  1 =>
    object(modele\metier\Critique)[8]
      private ?int 'note' => int 1
      private ?string 'commentaire' => string 'À éviter...' (length=13)
      private modele\metier\Utilisateur 'leUtilisateur' =>
        object(modele\metier\Utilisateur)[7]
          private int 'idu' => int 3
          private string 'mailU' => string 'mathieu.caplier@gmail.com' (length=25)
          private ?string 'mdpU' => string 'seS2poUAQgIl.' (length=13)
          private ?string 'pseudoU' => string 'pich' (length=4)
          private array 'lesRestosAimes' =>
            array (size=0)
              empty
  2 =>
    object(modele\metier\Critique)[10]
      private ?int 'note' => int 1
      private ?string 'commentaire' => string 'Cuisine tres moyenne.' (length=21)
      private modele\metier\Utilisateur 'leUtilisateur' =>
        object(modele\metier\Utilisateur)[9]
          private int 'idu' => int 5
          private string 'mailU' => string 'nicolas.harispe@gmail.com' (length=25)
          private ?string 'mdpU' => string '$1$zVND5FQ5dfqsDFQsdfst.' (length=24)
          private ?string 'pseudoU' => string 'Nico40' (length=6)
          private array 'lesRestosAimes' =>
            array (size=0)
              empty
  3 =>
    object(modele\metier\Critique)[12]
      private ?int 'note' => int 5
      private ?string 'commentaire' => null
      private modele\metier\Utilisateur 'leUtilisateur' =>
        object(modele\metier\Utilisateur)[11]
          private int 'idu' => int 6
          private string 'mailU' => string 'test@bts.sio' (length=12)
          private ?string 'mdpU' => string 'seS2poUAQgIl.' (length=13)
          private ?string 'pseudoU' => string 'testeur SIO' (length=11)
          private array 'lesRestosAimes' =>
            array (size=0)
              empty
```

#### 4- insert

Réussite de l'ajout :

```
C:\wamp64\www\PHPProjectSiteRestoInitial\tests\dao\testCritiqueDAO.php:65:boolean true
```

Après ajout, voici la nouvelle critique pour le restaurant d'id 1 ?

```
C:\wamp64\www\PHPProjectSiteRestoInitial\tests\dao\testCritiqueDAO.php:68:
object(modele\metier\Critique)[15]
  private ?int 'note' => int 4
  private ?string 'commentaire' => string 'Supercalifragilistic' (length=20)
  private modele\metier\Utilisateur 'leUtilisateur' =>
    object(modele\metier\Utilisateur)[3]
      private int 'idu' => int 1
      private string 'mailU' => string 'alex.garat@gmail.com' (length=20)
      private ?string 'mdpU' => string '$1$zVNSHYSQ5QDFUIQSDuFUQSDfznHFSosT.' (length=36)
      private ?string 'pseudoU' => string '@lex' (length=4)
      private array 'lesRestosAimes' =>
        array (size=0)
          empty
```



## 5- update

Réussite de la modification :

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testCritiqueDAO.php:79:boolean true
```

Après modification, voici la critique pour le restaurant d'id 1 ?

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testCritiqueDAO.php:82:
object(modele\metier\Critique)[4]
  private ?int 'note' => int 3
  private ?string 'commentaire' => string 'expidelilicious' (length=15)
  private modele\metier\Utilisateur 'leUtilisateur' =>
    object(modele\metier\Utilisateur)[3]
      private int 'idU' => int 1
      private string 'mailU' => string 'alex.garat@gmail.com' (length=20)
      private ?string 'mdpU' => string '$1$zvN5hYSQSQDFUIQSDufUQSDfZnHF5osT.' (length=36)
      private ?string 'pseudoU' => string '@lex' (length=4)
      private array 'lesRestosAimes' =>
        array (size=0)
          empty
```

## 6- delete

Réussite de la suppression :

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testCritiqueDAO.php:92:boolean true
```

Après suppression, la critique subsiste-t-elle pour le restaurant d'id 1 ?

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testCritiqueDAO.php:95:null
```

Le teste critique permet de renvoyer la note moyenne d'un restaurant, Le teste donne aussi la critique d'un utilisateur sur un restaurants, toutes les critiques qu'a reçu un restaurant et teste l'ajout, modification et suppression d'une critique.

```
<?php

use modele\dao\PhotoDAO;
use modele\dao\Bdd;

require_once '../includes/autoload.inc.php';

try {
    Bdd::connecter();
    ?>
    <h2>Test PhotoDAO</h2>

    <h3>1- getOneById</h3>
    <?php $id = 10; ?>
    <p>La photo n° <?=$id?></p>
    <?php
    $sunePhoto = PhotoDAO::getOneById($id);
    var_dump($sunePhoto);

    ?>
    <h3>2- getAllByResto</h3>
    <?php $idR = 4; ?>
    <p>Les photos du restaurant n° <?=$idR?></p>
    <?php
    $lesPhotos = PhotoDAO::getAllByResto($idR);
    var_dump($lesPhotos);

    Bdd::deconnecter();
} catch (Exception $ex) {
    ?>
    <h4>*** Erreur récupérée : <br/> <?=$ex->getMessage() ?> <br/>***</h4>
    <?php
}
?>
```

## Test PhotoDAO

### 1- getOneById

La photo n° 10

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testPhotoDAO.php:27:
object(modele\metier\Photo)[3]
  private int 'idP' => int 10
  private string 'cheminP' => string 'laTableDePottoka.jpg' (length=20)
```

### 2- getAllByResto

Les photos du restaurant n° 4

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testPhotoDAO.php:35:
array (size=3)
  0 =>
    object(modele\metier\Photo)[4]
      private int 'idP' => int 6
      private string 'cheminP' => string 'cidrerieDuFronton.jpg' (length=21)
  1 =>
    object(modele\metier\Photo)[5]
      private int 'idP' => int 14
      private string 'cheminP' => string 'cidrerieDuFronton2.jpg' (length=22)
  2 =>
    object(modele\metier\Photo)[6]
      private int 'idP' => int 15
      private string 'cheminP' => string 'cidrerieDuFronton3.jpg' (length=22)
```

Le teste photo renvoie le nom de la photo choisi et renvoie les photos du restaurant choisi.

```
try {
    Bdd::connecter();
    ?>
    <h2>Test RestoDAO</h2>

    <h3>1- getOneById</h3>
    <?php $idR = 6; ?>
    <p>Le restaurant n° <?= $idR ?></p>
    <?php
    $leResto = RestoDAO::getOneById($idR);
    var_dump($leResto);

    ?>

    <h3>2- getAll</h3>
    <p>Tous les restaurants</p>
    <?php
    $lesRestos = RestoDAO::getAll();
    var_dump($lesRestos);

    ?>

    <h3>3- getTop4</h3>
    <p>Les quatre restaurants les mieux notés</p>
    <?php
    $lesRestos = RestoDAO::getTop4();
    var_dump($lesRestos);

    ?>

    ?>

    <h3>5- getAllByNom</h3>
    <p>Filtrage des restaurants sur le nom</p>
    <?php
    $lesRestos = RestoDAO::getAllByNomR("pot");
    var_dump($lesRestos);

    ?>

    <h3>6- getAllByAdresse</h3>
    <p>Filtrage des restaurants sur l'adresse</p>
    <?php
    $lesRestos = RestoDAO::getAllByAdresse("du", "", "Bay");
    var_dump($lesRestos);

    ?>
}
```

```
?>
<h3>7- getRestosByMultiCriteres</h3>
<p>Filtrage des restaurants multicritères</p>
<?php
$lesRestos = RestoDAO::getAllMultiCriteres("pot", "du", "", "Bay", array(3, 6));
var_dump($lesRestos);

?>

<h3>8- getAimesByIdU</h3>
<?php $unIdU = 7; ?>
<p>Liste des restaurants aimés par l'utilisateur <?= $unIdU ?> (chargement de type "lazy")</p>
<?php
$lesRestos = RestoDAO::getAimesByIdU($unIdU);
var_dump($lesRestos);

?>

<?php
Bdd::deconnecter();
} catch (Exception $ex) {
?>
<h4>*** Erreur récupérée : <br/> <?= $ex->getMessage() ?> <br/>***</h4>
<?php
}
?>
```

## Test RestoDAO

### 1- getOneById

Le restaurant n° 6

```
C:\wamp64\www\PhpProjectSiteRestoInitial\tests\dao\testRestoDAO.php:33:
object(modele\metier\Resto)[3]
  private int 'idR' => int 6
  private ?string 'nomR' => string 'Le Bistrot Sainte Cluque' (length=24)
  private ?string 'numAdr' => string '9' (length=1)
  private ?string 'voieAdr' => string 'Rue Hugues' (length=10)
  private ?string 'cpR' => string '64100' (length=5)
  private ?string 'villeR' => string 'Bayonne' (length=7)
  private ?float 'latitudeDegR' => null
  private ?float 'longitudeDegR' => null
  private ?string 'descR' => string 'description' (length=11)
  private ?string 'horairesR' => string '<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>' (length=694)
  private array 'lesPhotos' =>
    array (size=1)
      0 =>
        object(modele\metier\Photo)[7]
          private int 'idP' => int 8
          private string 'cheminP' => string 'leBistrotSainteCluque.jpg' (length=25)
  private array 'lesCritiques' =>
    array (size=1)
      0 =>
        object(modele\metier\Critique)[6]
          private ?int 'note' => int 4
          private ?string 'commentaire' => string 'Cuisine de qualité.' (length=20)
          private modele\metier\Utilisateur 'leUtilisateur' =>
            object(modele\metier\Utilisateur)[5]
              private int 'idU' => int 5
              private string 'mailU' => string 'nicolas.harispe@gmail.com' (length=25)
              private ?string 'mdpU' => string '$1$zvNDSFQsdfqsDfQsdfsT.' (length=24)
              private ?string 'pseudoU' => string 'Nico40' (length=6)
              private array 'lesRestosAimes' =>
                array (size=0)
                  empty
```

Affiche les détails du restaurant n°6.

## 2- getAll

Tous les restaurants

Affiche les détails de tous les restaurants.

## 3- getTop4

Les quatre restaurants les mieux notés

Affiche les détails des 4 restaurants du top.

## 5- getAllByNom

Filtrage des restaurants sur le nom

Donne les restaurants avec le filtrage par nom.

## 6- getAllByAdresse

Filtrage des restaurants sur l'adresse

Filtre les restaurants par adresse et les renvoie.

```
try {
    Bdd::connecter();
    ?>
    <h2>Test UtilisateurDAO</h2>

    <h3>1- getOneByMail</h3>
    <?php $unMail = "test@bts.sio"; ?>
    <p>L'utilisateur de mail <?= $unMail ?></p>
    <?php
    $unUser = UtilisateurDAO::getOneByMail($unMail);
    var_dump($unUser);

    ?>

    <h3>2- getOneById</h3>
    <?php $unIdU = 3; ?>
    <p>L'utilisateur d'id <?= $unIdU ?></p>
    <?php
    $unUser = UtilisateurDAO::getOneById($unIdU);
    var_dump($unUser);

    ?>

    <h3>3- insert sans le mot de passe</h3>
    <?php $user = new Utilisateur(0, 'test@insert.nb', '1234', 'pseudo de test'); ?>
    <p>Ajouter un utilisateur <?= $user->__toString() ?></p>
    <?php
    $ok = UtilisateurDAO::insert($user);
    var_dump($ok);

    ?>

    <h3>4- update sans le mot de passe</h3>
    <?php $userLuAvant = UtilisateurDAO::getOneByMail("test@insert.nb"); ?>
    <p>Mettre à jour un utilisateur </p>
    <p>Utilisateur lu avant la mise à jour : <?= $userLuAvant->__toString() ?></p>
    <?php
    $userModifie = $userLuAvant;
    $userModifie->setMailU("le-mien@contact.fr");
    $userModifie->setPseudoU("pseudomodifie44");
    $userModifie->setMdpU("sio");
    $ok = UtilisateurDAO::update($userModifie);
    ?>
    Mise à jour réussie ?<br/>
```

```

mise à jour réussie :<br/>
<?php
var_dump($ok);

$userLuApres = UtilisateurDAO::getOneById($userLuAvant->getIdU());
?>
<p>Utilisateur lu après la mise à jour : <?= $userLuApres->__toString() ?></p>

?>
<h3>5- update du mot de passe</h3>
<?php $userLuAvant = $userLuApres ?>
<p>Mettre à jour le mot de passe d'un utilisateur </p>
<p>Utilisateur lu avant la mise à jour : <?= $userLuAvant->__toString() ?></p>
<?php
$mdp = "sio";
$ok = UtilisateurDAO::updateMdp($userModifie->getIdU(), $mdp);
?>
Mise à jour réussie ?<br/>
<?php
var_dump($ok);
$userLuApres = UtilisateurDAO::getOneById($userModifie->getIdU());
?>
<p>Utilisateur lu après la mise à jour : <?= $userLuApres->__toString() ?></p>

<?php

Bdd::deconnecter();
catch (Exception $ex) {
?>
<h4>*** Erreur récupérée : <br/> <?= $ex->getMessage() ?> <br/>***</h4>
<?php

```



Renvoie les détails d'un utilisateur.

## 2- getOneById

L'utilisateur d'id 3

C:\wamp64\www\PHPProjectSiteRes  
object(modele\metier\Util

Renvoie les détails de l'id utilisateur choisi.

## 3- insert sans le mot de passe

Ajouter un utilisateur

(!) Deprecated: Calling get\_class()  
C:\wamp64\www\PHPProjectSiteRes

#	Time	Memory	Fun
1	0.1825	454440	{ma
2	0.1882	615200	mod
3	0.1882	615200	get_

modele\metier\Utilisateur{id=0, mail=b

C:\wamp64\www\PHPProjectSiteRestoInitial\tr

## 4. update sans le mot de passe

Le mot de passe ne peut pas être modifier.

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

La méthode statique issue de la classe : « getAll() ».

### III.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 ?

Dans le dossier *controleur*, on trouve effectivement du HTML. En revanche, on ne trouve pas de CSS dans ces deux dossiers.

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

Le CSS est contenu dans le dossier CSS. Le HTML est incorporé aux fichiers PHP et HTML contenu dans le dossier *Vue*.

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

Actuellement, le code issu de « *vueListeRestos.php* » provoque des erreurs. Enfin plus précisément ce sont les méthodes

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

Le point commun entre toutes les méthodes de la classe DAO est que les données proviennent de la table Resto.

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

Non car c'est le rôle du dossier modele/dao de communiquer avec la BDD.

#### Synthèse

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

- modele/dao : Accès aux données de la base de données SQL. Il contient les classes/fonctions qui communiquent avec la base de données.
- modele/metier : Représentation des entités et logique métier. Il contient les classes PHP correspondant aux entités de l'application.
- vue : gère l'affichage de la page visible par l'utilisateur. Il contient les fichiers PHP/HTML qui construisent les pages accessibles par l'utilisateur.

### III.4 La couche contrôleur

#### III.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 ?  
La méthode DAO utilisé dans cette classe est `getAll()`. Elle permet de récupérer la liste de tous les restaurants avec toutes leurs données : `idR`, `nomR`, `numAdrR`, `voieAdrR`, `cpR`, `villeR`, `latitudeDegR`, `longitudeDegR`, `DescR`, `horairesR`.
2. Les données récupérées sont-elles affichées à l'écran ? L'affichage est-il fait dans le contrôleur ?  
Les données récupérées sont affichées à l'écran depuis le fichier : `vueListeRestos.php`. L'affichage n'est donc pas fait dans le controleur. Il sert juste à récupérer les données depuis la BDD.
3. Quels scripts sont inclus dans les dernières lignes du contrôleur ? Quels sont leurs rôles ?

Les dernières lignes incluses dans le contrôleur sont :

```
26 | require_once "$racine/vue/entete.html.php";  
27 | require_once "$racine/vue/vueListeRestos.php";  
28 | require_once "$racine/vue/pied.html.php";
```

Entete.html : correspond à la navbar du site web.

vueListeRestos.php : correspond au contenu de d'id : « corps » de la page. C'est le contenu principal.

Pied.html : est censé intégrer du code pour le footer mais il n'y a littéralement rien :

```
1 |  
2 |  
3 |  
4 |
```

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

#### III.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 l'identifiant du restaurant.
2. Rappeler le rôle de la méthode `getOneById`. Pourquoi cette méthode a-t-elle besoin d'un paramètre ?

Le rôle de `getOneById` permet de renvoyer les détails correspondant à l'identifiant du restaurant. Cette méthode a besoin d'un paramètre, car si aucun identifiant n'est donner, alors une erreur est renvoyée.

La variable `$unResto` reçoit le résultat de l'appel à `getOneById`.

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

La variable `$unResto` est utilisée aux lignes 40, 49 et 50.

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

TAILLE JADE  
DELAUNAY-GUITTON Benjamin  
PORTOLLEAU Anaïs

Le contrôleur ne gère pas directement l'accès aux données car c'est le modele/dao qui s'en occupe. Le contrôleur ne fait qu'utiliser les méthodes issus du modele/dao.



### III.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 sont affichées dans le code, par exemple dans le contrôleur `detailResto` les données sont affichées à la ligne 49 à 56.

2. Comment ces données sont transmises au contrôleur :
  - de l'utilisateur vers le contrôleur ? L'identifiant du restaurant est récupéré en méthode GET (`index.php?action=detail&idR=...`) et le contrôleur le récupère avec `$GET[idR]`.
  - de la base de données vers le contrôleur ? Le contrôleur appelle la méthode `daoResto > getOneById($idR)` qui exécute la requête SQL `getOneById`. Le résultat est stocké dans `$unResto`, puis ses données sont récupérées avec les méthodes POST ou GET.
3. Résumer le rôle de la couche « contrôleur »

Le rôle de la couche contrôleur est de gérer les traitements, elle va indirectement récupérer les informations présentes dans la base de données et les afficher.

## IV - 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.

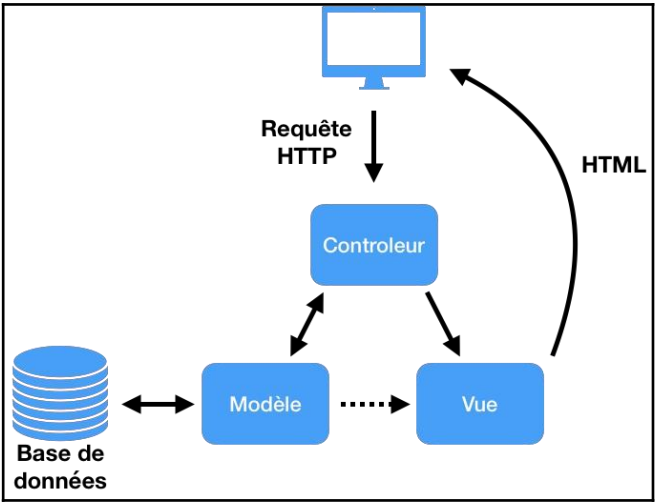
Chaque fonctionnalité, par exemple l'affichage de la liste des restaurants, fait ainsi appel aux composants :

- Modèle : fonction d'accès à la base de données ;
- Vue : affichage des données à l'utilisateur ;
- Contrôleur : composant chargé de la logique applicative : récupération 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.

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).



*Pattern MVC*