

Architecture MVC en PHP - Modèle

Partie 1 : généralités

Partie 2 : contrôleur

Partie 3 : vue

Partie 4 : modèle

Partie 5 : contrôleur principal et routage

Fonctionnement du modèle dans le patron de conception MVC

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.

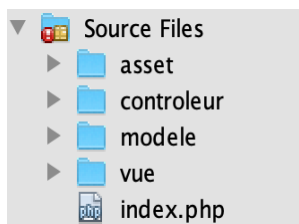
Ressources à utiliser téléchargeable

- 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 **MVCTP4** à la racine de votre serveur web. Copier le contenu du dossier « **site** » dans ce dossier. Importer le dossier **MVCTP4** 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"; //notre SGBRD est exécuté en local
```

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](#).

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.

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()`.

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.

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 propriétés 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.

1.5. Rappeler comment accéder au nom du restaurant dans la variable retournée par la fonction `getRestoById()`.

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.

données <- fetch()	idR	nomR	numAdrR	voieAdrR	cpR	villeR
données <- fetch()	1	l'entrepote	2	rue Maurice Ravel	33000	Bordeaux
données <- fetch()	2	le bar du charcutier	30	rue Parlement Sainte-Catherine	33000	Bordeaux
booléen <- fetch()	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()` ?

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()` ?

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

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.

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
```

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

Exécuter la requête en donnant la valeur 2 à la propriété `idR`.

3.3. Comment est nommée la colonne affichée dans le résultat

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

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

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.

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;  
}
```

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:

<https://dev.sio56.org/kercode>

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

- 4.1.** Expliquer le rôle de la requête présente dans la fonction addAimer().
- 4.2.** Expliquer pourquoi les deux appels à bindValue() n'utilisent pas la même constante PDO en 3ème paramètre.
- 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.
- 4.4.** Dédire de la question précédente le type de données retourné par la fonction addAimer().
- 4.5.** Si un utilisateur tente d'aimer un restaurant qu'il aime déjà, quelle sera la valeur retournée par la fonction addAimer() ?

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.

```
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 à 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.

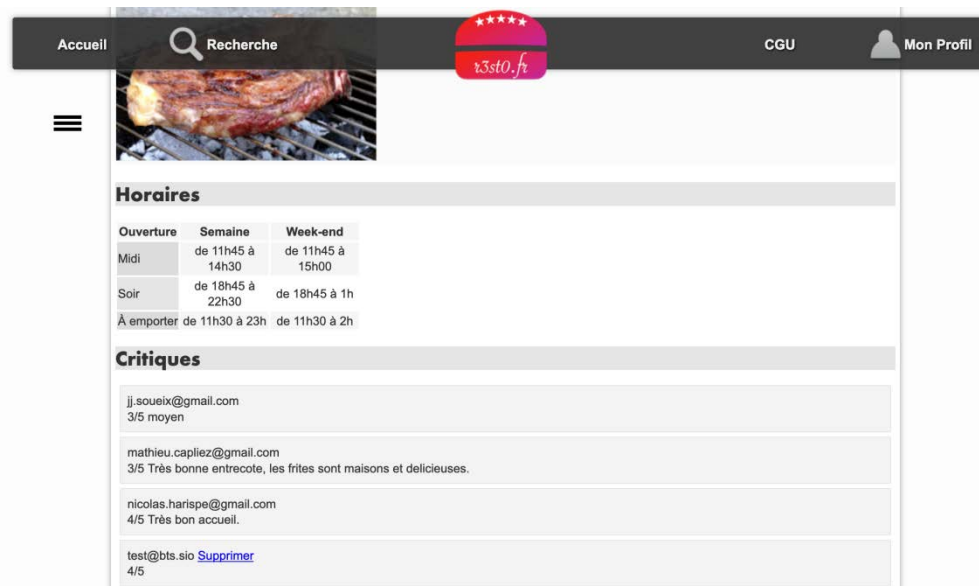
- 5.1.** Quelle est la clé primaire de la table aimer ?
- 5.2.** Rappeler la syntaxe de la requête de suppression en SQL.
- 5.3.** Écrire la requête SQL permettant de supprimer une occurrence précise de la table aimer.
- 5.4.** Se connecter à phpmyadmin et tester la requête à l'aide d'un exemple.
- 5.5.** Dans quel script du modèle doit être placée la fonction delAimer() ? Justifier.
- 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.
- 5.7.** Écrire le code de la fonction, puis ajouter un appel à cette fonction dans la section de test du script modèle approprié.

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";
            }
          </?php>
        </span>
      </div>
    </li>
  }
}</ul>
```



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

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

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

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

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()`.

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.

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
<input type="checkbox"/> Modifier <input type="checkbox"/> Copier <input type="checkbox"/> Effacer	1	l'entrepote	2	rue Maurice Ravel	33000	Bordeaux	44.7948	-0.58754	description	<table> <thead> <tr> <td>...

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
<input type="checkbox"/> Modifier <input type="checkbox"/> Copier <input type="checkbox"/> Effacer	2	le bar du charcutier	30	rue Parlement Sainte-Catherine	33000	Bordeaux	NULL	NULL	description	<table> <thead> <tr> <td>...

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, PHP 8, 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] =>
        )
)
```