

**c:\grin-eat-tpi2022\grineat-api\.htaccess**

```
1 #Turn Rewrite Engine On
2 RewriteEngine On
3
4 #Rewrite URL
5 RewriteRule ^(.*)/(.*)/(.*)$ index.php?endpoint=$1&id=$2&context=$3 [QSA,L]
6 RewriteRule ^(.*)/(.*)$ index.php?endpoint=$1&id=$2 [QSA,L]
7 RewriteRule ^(.*)$ index.php?endpoint=$1 [QSA,L]
```

## c:\grin-eat-tpi2022\grineat-api\functions.inc.php

```

1 <?php
2 /*
3     /\    /\
4     /\ \  /\ \
5     \_   _/   /   /   Nom et Prénom: GRIN Brian
6     / * * \   /^^^]   Enseignant : Monsieur Antoine Schmid
7     \_ \O/_/   [   ]   Classe : I.DA-P4B
8     /   \_   [   /   Date : 18.05.2022
9     \   \_   /   /   Nom du projet : GrinEat-API
10    [ [ /   \/_/   Version du projet : 1.0
11    _[ [ \   /_/   Cours : TPI
12
13 */
14
15 function convertAdressToCoordinate($adress)
16 {
17     $encoded = rawurlencode($adress);
18     $result = array();
19
20     $url = "https://api.mapbox.com/geocoding/v5/mapbox.places/" . $encoded .
21 ".json?access_token=pk.eyJ1IjoibWl5b2tpIiwiaSI6ImNsMW96ejJ5NTAzMjZa3B0NHB3bHMxYncifQ.03AlBHWnc
22
23     $curl = curl_init($url);
24     curl_setopt($curl, CURLOPT_URL, $url);
25     curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
26
27     $resp = curl_exec($curl);
28     curl_close($curl);
29     $data = json_decode($resp, true);
30
31     if (count($data["features"]) != 0) {
32         $addressFound = $latitudeAddr = $data["features"][0]["place_name"];
33         $latitudeAddr = $data["features"][0]["geometry"]["coordinates"][1];
34         $longitudeAddr = $data["features"][0]["geometry"]["coordinates"][0];
35
36         $result["street"] = $addressFound;
37         $result["coordinates"]["latitude"] = $latitudeAddr;
38         $result["coordinates"]["longitude"] = $longitudeAddr;
39     } else {
40         $result = null;
41     }
42
43     return $result;
44     /*
45     $result = array();
46
47     $result["adress"] = "Quai Capo D'istria 9, 1205 Genève, Switzerland";
48     $result["latitude"] = 46.187687;
49     $result["longitude"] = 6.143453;
50     return $result;
51     */
52 }

```

```
53 |
54 | function haversineGreatCircleDistance(
55 |   $latitudeFrom,
56 |   $longitudeFrom,
57 |   $latitudeTo,
58 |   $longitudeTo,
59 |   $earthRadius = 6371
60 | ,) {
61 |   // Convertir de degrés en radian
62 |   $latFrom = deg2rad($latitudeFrom);
63 |   $lonFrom = deg2rad($longitudeFrom);
64 |   $latTo = deg2rad($latitudeTo);
65 |   $lonTo = deg2rad($longitudeTo);
66 |
67 |   $latDelta = $latTo - $latFrom;
68 |   $lonDelta = $lonTo - $lonFrom;
69 |
70 |   $angle = 2 * asin(sqrt(pow(sin($latDelta / 2), 2) +
71 |     cos($latFrom) * cos($latTo) * pow(sin($lonDelta / 2), 2)));
72 |   return $angle * $earthRadius;
73 | }
74 |
75 | function compareDistance($a, $b) {
76 |   $distanceA = $a->getDistanceFrom();
77 |   $distanceB = $b->getDistanceFrom();
78 |   return ($distanceA < $distanceB) ? -1 : 1;
79 | }
```

## c:\grin-eat-tpi2022\grineat-api\index.php

```

1 <?php
2 /*
3     /\    /\
4     /\ \  /\ \    ____
5     \_   _/    /   /    Nom et Prénom: GRIN Brian
6     / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
7     \_ \0/_/    [   ]    Classe : I.DA-P4B
8     /   \_   [   /    Date : 18.05.2022
9     \   \_   /   /    Nom du projet : GrinEat-API
10    [ [ /   \/_/    Version du projet : 1.0
11    _[ [ \   /_/_    Cours : TPI
12
13 */
14
15 // Include CORS headers
16 header('Access-Control-Allow-Origin: *');
17 header('Access-Control-Allow-Methods: GET, POST, PUT, DELETE');
18 header('Access-Control-Allow-Headers: X-Requested-With');
19 header('Content-Type: application/json');
20
21 // Include Models & functions
22 include "functions.inc.php";
23 include "models/MyPdo.php";
24 include "models/Category.php";
25 include "models/Restaurant.php";
26 include "models/MenuItems.php";
27
28 // create a api variable to get HTTP method dynamically
29 $api = $_SERVER['REQUEST_METHOD'];
30 // get headers
31 $headers = getallheaders();
32 // get endpoint from url
33 $endpoint = $_GET['endpoint'] ?? '';
34 // get id from url
35 $id = intval($_GET['id'] ?? '');
36 // get context
37 $context = $_GET['context'] ?? '';
38
39 switch ($endpoint) {
40     case '':
41         echo json_encode("Welcome to the API");
42         break;
43     case 'categories':
44         include "controllers/categoriesController.php";
45         break;
46     case 'restaurants':
47         switch($context) {
48             case '':
49                 include "controllers/restaurantsController.php";
50                 break;
51             case 'menus':
52                 include "controllers/menu_itemsController.php";
53                 break;

```

```

54         default:
55             http_response_code(400);
56             break;
57     }
58     break;
59 default:
60     http_response_code(404);
61     break;
62 }

```

## c:\grin-eat-tpi2022\grineat-api\controllers\categoriesController.php

```

1  <?php
2  /*
3      /\      /\
4      /\ \_/\ /\
5      \_ \_/_/  /___/  Nom et Prénom: GRIN Brian
6      / * * \_ /^^^]  Enseignant : Monsieur Antoine Schmid
7      \_ \0/_/  [ ]  Classe : I.DA-P4B
8      / \_ \_ [ /  Date : 18.05.2022
9      \ \_ \_ / /  Nom du projet : GrinEat-API
10     [ [ / \_/_/  Version du projet : 1.0
11     _[ [ \ /_/_/  Cours : TPI
12
13  */
14
15  switch ($api) {
16      case 'GET':
17          $message = array();
18
19          if ($id != 0) {
20              $myCategory = new Category();
21
22              $myCategory->setId($id);
23
24              $body = $myCategory->findById();
25          } else {
26              $categories = Category::findAll();
27
28              $body = $categories;
29          }
30
31          array_push($message, "success");
32          http_response_code(200);
33          echo json_encode(["result" => $body, "message" => $message]);
34          break;
35      default:
36          http_response_code(405);
37          break;
38  }

```

## c:\grin-eat-tpi2022\grineat-api\controllers\menu\_itemsController.php

```
1 <?php
2 /*
3     /\    /\
4     /\ \  /\ \
5     \_   _/  /   /   Nom et Prénom: GRIN Brian
6     / * * \   /^^^]   Enseignant : Monsieur Antoine Schmid
7     \_ \0/_/   [   ]   Classe : I.DA-P4B
8     /   \_   [   /   Date : 18.05.2022
9     \   \_   /   /   Nom du projet : GrinEat-API
10    [ [ /   \/_/   Version du projet : 1.0
11    _[ [ \   /_/_   Cours : TPI
12
13 */
14
15 switch ($api) {
16     case 'GET':
17         $message = array();
18
19         if ($id != 0) {
20             // Récupérer les informations du restaurant à l'ID donné
21             $myRestaurant = new Restaurant();
22             $myRestaurant->setId($id);
23             $restaurant = $myRestaurant->findById();
24
25             foreach ($restaurant->findCategoriesById() as $c) {
26                 // On cherche les catégories du restaurant et les ajoute à la propriété
categories de l'objet Restaurant
27                 $restaurant->addCategory($c->getNameFrench());
28             }
29
30             // Récupérer les menus et éléments relative au restaurant
31             $myMenuItems = new MenuItem();
32             $myMenuItems->setRestaurantId($id);
33             $menuItems = $myMenuItems->findItemsByRestaurantId();
34
35             array_push($message, "success");
36             http_response_code(200);
37             echo json_encode(["result" => ["info" => $restaurant, "menu_items" =>
$menuItems], "message" => $message]);
38         }
39         break;
40     default:
41         http_response_code(405);
42         break;
43 }
```

## c:\grin-eat-tpi2022\grineat-api\controllers\restaurantsController.php

```

1 <?php
2 /*
3     /\    /\
4     /\ \  /\ \
5     \_   _/   /   /   Nom et Prénom: GRIN Brian
6     / * * \   /^^^]   Enseignant : Monsieur Antoine Schmid
7     \_ \0/_/   [   ]   Classe : I.DA-P4B
8     /   \_   [   /   Date : 18.05.2022
9     \   \_   /   /   Nom du projet : GrinEat-API
10    [ [ /   \/_/   Version du projet : 1.0
11    _[ [ \   /_/_   Cours : TPI
12
13 */
14
15 switch ($api) {
16     case 'POST':
17         $jsonString = file_get_contents("php://input");
18         $data = json_decode($jsonString, true) ?? array();
19         $message = array();
20         $body = array();
21
22         if ($data != null) {
23             $address = $data["address"] ?? null; // adresse du client.
24             $coordinates = $data["coordinates"] ?? array();
25             $radius = intval($data["radius"]) != null ? intval($data["radius"]) : 5; // rayon
26             // limite autour de l'adresse du client par défaut 5 si rien spécifié.
27             $name = $data["name"] ?? null; // filtre par nom si le client souhaite faire une
28             // recherche
29             $categories = $data["categories"] ?? array(); // filtre par catégorie si le
30             // client en sélectionne
31
32             if (isset($coordinates["latitude"]) && isset($coordinates["longitude"]) &&
33                 !isset($info)) {
34                 $info["coordinates"] = $coordinates;
35
36                 if ($address != null) {
37                     $info["street"] = $address;
38                 }
39             }
40
41             if ($address != null && !isset($info)) {
42                 $info = convertAdressToCoordinate($address);
43             }
44
45             if ($info != null) {
46                 $latFrom = $info["coordinates"]["latitude"]; // latitude de l'adresse du
47                 // client
48                 $lonFrom = $info["coordinates"]["longitude"]; // longitude de l'adresse du
49                 // client
50
51                 $restaurants = Restaurant::findAll();
52
53                 foreach ($restaurants as $r) {

```

```
48         // On cherche les catégories de chaque restaurant et les ajoutes à la
propriété categories de l'objet Restaurant
49         foreach ($r->findCategoriesById() as $c) {
50             $r->addCategory($c->getId());
51         }
52
53         $latTo = $r->getLatitude(); // latitude du restaurant actuel
54         $lonTo = $r->getLongitude(); // longitude du restaurant actuel
55
56         // On vérifie si le restaurant actuel est dans le rayon demandé
57         $r->setDistanceFrom(haversineGreatCircleDistance($latFrom, $lonFrom,
$latTo, $lonTo));
58         if ($r->getDistanceFrom() < $radius) {
59             // On vérifie ensuite si le client souhaite filtrer par des
catégories
60             if (count($categories) > 0) {
61                 // On vérifie si le restaurant actuel contient au moins une des
catégories demandés par le client
62                 if (count(array_intersect($r->getCategories(), $categories)) < 1)
{
63                     continue; // On passe au restaurant suivant
64                 }
65             }
66
67             // On vérifie si le client souhaite filtrer par un nom
68             if ($name != null) {
69                 // On vérifie si le restaurant actuel commence par la saisie du
client
70                 if (strpos(strtolower($r->getName()), strtolower($name)) !== 0) {
71                     continue; // On passe au restaurant suivant
72                 }
73             }
74
75             array_push($body, $r);
76         }
77     }
78     // Trier les restaurants par distance croissanteQu
79     usort($body, "compareDistance");
80
81     array_push($message, "success");
82     http_response_code(200);
83     echo json_encode(["result" => ["address" => $info, "restaurants" => $body],
"message" => $message]);
84 } else {
85     array_push($message, "Adresse invalide");
86     http_response_code(400);
87     echo json_encode(["result" => false, "message" => $message]);
88 }
89 } else {
90     array_push($message, "Il faut au minimum une adresse ou des coordonnées latitude,
longitude");
91     http_response_code(400);
92     echo json_encode(["result" => false, "message" => $message]);
93 }
94 break;
95 default:
```



```
96         http_response_code(405);  
97         break;  
98     }
```

## c:\grin-eat-tpi2022\grineat-api\models\Category.php

```

1 <?php
2 /*
3     /\      /\
4     /\_\_/\_\_
5     \_      _/  /  /  Nom et Prénom: GRIN Brian
6     / * * \    /^^^]  Enseignant : Monsieur Antoine Schmid
7     \_0/_/    [  ]    Classe : I.DA-P4B
8     /   \_    [  /    Date : 18.05.2022
9     \   \_    /  /    Nom du projet : GrinEat-API
10    [ [ /  \_/_/    Version du projet : 1.0
11    _[ [ \  /_/_/    Cours : TPI
12
13 */
14
15 class Category implements JsonSerializable {
16     private $id;
17     private $nameEnglish;
18     private $nameFrench;
19
20     /**
21      * Get the value of id
22      */
23     public function getId()
24     {
25         return $this->id;
26     }
27
28     /**
29      * Set the value of id
30      *
31      * @return self
32      */
33     public function setId($id)
34     {
35         $this->id = $id;
36
37         return $this;
38     }
39
40     /**
41      * Get the value of nameEnglish
42      */
43     public function getNameEnglish()
44     {
45         return $this->nameEnglish;
46     }
47
48     /**
49      * Set the value of nameEnglish
50      *
51      * @return self
52      */
53     public function setNameEnglish($nameEnglish)

```

```
54     {
55         $this->nameEnglish = $nameEnglish;
56
57         return $this;
58     }
59
60     /**
61      * Get the value of nameFrench
62      */
63     public function getNameFrench()
64     {
65         return $this->nameFrench;
66     }
67
68     /**
69      * Set the value of nameFrench
70      *
71      * @return self
72      */
73     public function setNameFrench($nameFrench)
74     {
75         $this->nameFrench = $nameFrench;
76
77         return $this;
78     }
79
80     public function jsonSerialize()
81     {
82         return get_object_vars($this);
83     }
84
85     public static function findAll() {
86         $sql = MyPdo::getInstance()->prepare('SELECT * FROM categories');
87         $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'Category');
88         $sql->execute();
89         $result = $sql->fetchAll();
90
91         return $result;
92     }
93
94     public function findById() {
95         $sql = MyPdo::getInstance()->prepare('SELECT * FROM categories WHERE id = :id');
96         $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'Category');
97         $sql->bindParam(':id', $this->getId());
98         $sql->execute();
99         $result = $sql->fetch();
100
101         return $result;
102     }
103 }
```

## c:\grin-eat-tpi2022\grineat-api\models\MenuItems.php

```

1 <?php
2 /*
3     /\      /\
4     /\ \_/\ /\
5     \_      _/  /  /  Nom et Prénom: GRIN Brian
6     / * * \    /^^^] Enseignant : Monsieur Antoine Schmid
7     \_0/_/    [   ] Classe : I.DA-P4B
8     /   \_    [   / Date : 18.05.2022
9     \   \_    /   / Nom du projet : GrinEat-API
10    [ [ / \_/_/ Version du projet : 1.0
11    _[ [ \ \_/_/ Cours : TPI
12
13 */
14
15 class MenuItems implements JsonSerializable {
16     private $id;
17     private $restaurantId;
18     private $name;
19     private $description;
20     private $image;
21     private $price;
22
23     /**
24      * Get the value of id
25      */
26     public function getId()
27     {
28         return $this->id;
29     }
30
31     /**
32      * Set the value of id
33      *
34      * @return self
35      */
36     public function setId($id)
37     {
38         $this->id = $id;
39
40         return $this;
41     }
42
43     /**
44      * Get the value of restaurantId
45      */
46     public function getRestaurantId()
47     {
48         return $this->restaurantId;
49     }
50
51     /**
52      * Set the value of restaurantId
53      */

```

```
54     * @return self
55     */
56     public function setRestaurantId($restaurantId)
57     {
58         $this->restaurantId = $restaurantId;
59
60         return $this;
61     }
62
63     /**
64      * Get the value of name
65      */
66     public function getName()
67     {
68         return $this->name;
69     }
70
71     /**
72      * Set the value of name
73      *
74      * @return self
75      */
76     public function setName($name)
77     {
78         $this->name = $name;
79
80         return $this;
81     }
82
83     /**
84      * Get the value of description
85      */
86     public function getDescription()
87     {
88         return $this->description;
89     }
90
91     /**
92      * Set the value of description
93      *
94      * @return self
95      */
96     public function setDescription($description)
97     {
98         $this->description = $description;
99
100        return $this;
101    }
102
103    /**
104     * Get the value of image
105     */
106    public function getImage()
107    {
108        return $this->image;
```

```
109     }
110
111     /**
112      * Set the value of image
113      *
114      * @return self
115      */
116     public function setImage($image)
117     {
118         $this->image = $image;
119
120         return $this;
121     }
122
123     /**
124      * Get the value of price
125      */
126     public function getPrice()
127     {
128         return $this->price;
129     }
130
131     /**
132      * Set the value of price
133      *
134      * @return self
135      */
136     public function setPrice($price)
137     {
138         $this->price = $price;
139
140         return $this;
141     }
142
143     public function jsonSerialize()
144     {
145         return get_object_vars($this);
146     }
147
148     public function findItemsByRestaurantId() {
149         $sql = MyPdo::getInstance()->prepare('SELECT * FROM menu_items WHERE restaurantId =
150 :id');
151         $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'MenuItems');
152         $sql->bindParam(':id', $this->getRestaurantId());
153         $sql->execute();
154         $result = $sql->fetchAll();
155
156         return $result;
157     }
158 }
```

c:\grin-eat-tpi2022\grineat-api\models\MyPdo.php

```

1 <?php
2 /*
3     /\      /\
4     //\\_//\\_   _
5     \_    _/   /___/   Nom et Prénom: GRIN Brian
6     / * * \   /^^^]   Enseignant : Monsieur Antoine Schmid
7     \_0/_/_/   [ ]   Classe : I.DA-P4B
8     /    \_   [ ]   Date : 18.05.2022
9     \      \_ /_/   Nom du projet : GrinEat-API
10    [ [ / \ \_/_/   Version du projet : 1.0
11    _[ [ \ \/_/_/   Cours : TPI
12
13 */
14
15 class MyPdo
16 {
17     private static $dbhost = 'localhost';
18     private static $dbname = 'db_grineat';
19     private static $dbuser = 'admin_grineat';
20     private static $dbpasswd = 'ND)BO/4S/dr[_6H6';
21
22     private static $MyPdo;
23     private static $unPdo = null;
24
25     // Constructeur privé, crée l'instance de PDO qui sera sollicitée
26     // pour toutes les méthodes de la classe
27     private function __construct()
28     {
29         MyPdo::$unPdo = new PDO("mysql:host=".MyPdo::$dbhost.";dbname=".MyPdo::$dbname,
30             MyPdo::$dbuser, MyPdo::$dbpasswd);
31         MyPdo::$unPdo->query("SET CHARACTER SET utf8");
32         MyPdo::$unPdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
33     }
34     public function __destruct()
35     {
36         MyPdo::$unPdo = null;
37     }
38 /**
39  * Fonction statique qui cree l'unique instance de la classe
40  * Appel : $instanceMyPdo = MyPdo::getMyPdo();
41  * @return l'unique objet de la classe MyPdo
42  */
43     public static function getInstance()
44     {
45         if(self::$unPdo == null)
46         {
47             self::$MyPdo= new MyPdo();
48         }
49         return self::$unPdo;
50     }
51 }

```

```
c:\grin-eat-tpi2022\grineat-api\models\Restaurant.php
```

```

1 <?php
2 /*
3     /\      /\
4    /\ \_/\ /\ \
5   \_  _\ /  _/_/
6  / * * \  /^^^]
7  \_0/_/ [ ]
8   /   \_ [ /
9   \       \_ / /
10  [ [ / \_/_/
11  _[ [ \ /_/_/
12
13 */

```

Nom et Prénom: GRIN Brian  
Enseignant : Monsieur Antoine Schmid  
Classe : I.DA-P4B  
Date : 18.05.2022  
Nom du projet : GrinEat-API  
Version du projet : 1.0  
Cours : TPI

```

15 class Restaurant implements JsonSerializable {
16     private $id;
17     private $createdOn;
18     private $email;
19     private $name;
20     private $phone;
21     private $website;
22     private $image;
23     private $street;
24     private $cp;
25     private $city;
26     private $countryId;
27     private $latitude;
28     private $longitude;
29     private $categories = array();
30     private $distanceFrom;
31
32     /**
33      * Get the value of id
34      */
35     public function getId()
36     {
37         return $this->id;
38     }
39
40     /**
41      * Set the value of id
42      *
43      * @return self
44      */
45     public function setId($id)
46     {
47         $this->id = $id;
48
49         return $this;
50     }
51
52     /**
53      * Get the value of createdOn

```



```
54     */
55     public function getCreatedOn()
56     {
57         return $this->createdOn;
58     }
59
60     /**
61      * Set the value of createdOn
62      *
63      * @return self
64      */
65     public function setCreatedOn($createdOn)
66     {
67         $this->createdOn = $createdOn;
68
69         return $this;
70     }
71
72     /**
73      * Get the value of email
74      */
75     public function getEmail()
76     {
77         return $this->email;
78     }
79
80     /**
81      * Set the value of email
82      *
83      * @return self
84      */
85     public function setEmail($email)
86     {
87         $this->email = $email;
88
89         return $this;
90     }
91
92     /**
93      * Get the value of name
94      */
95     public function getName()
96     {
97         return $this->name;
98     }
99
100    /**
101     * Set the value of name
102     *
103     * @return self
104     */
105    public function setName($name)
106    {
107        $this->name = $name;
108    }
```

```
109         return $this;
110     }
111
112     /**
113      * Get the value of phone
114      */
115     public function getPhone()
116     {
117         return $this->phone;
118     }
119
120     /**
121      * Set the value of phone
122      *
123      * @return self
124      */
125     public function setPhone($phone)
126     {
127         $this->phone = $phone;
128
129         return $this;
130     }
131
132     /**
133      * Get the value of website
134      */
135     public function getWebsite()
136     {
137         return $this->website;
138     }
139
140     /**
141      * Set the value of website
142      *
143      * @return self
144      */
145     public function setWebsite($website)
146     {
147         $this->website = $website;
148
149         return $this;
150     }
151
152     /**
153      * Get the value of image
154      */
155     public function getImage()
156     {
157         return $this->image;
158     }
159
160     /**
161      * Set the value of image
162      *
163      * @return self
```

```
164     */
165     public function setImage($image)
166     {
167         $this->image = $image;
168
169         return $this;
170     }
171
172     /**
173      * Get the value of street
174      */
175     public function getStreet()
176     {
177         return $this->street;
178     }
179
180     /**
181      * Set the value of street
182      *
183      * @return self
184      */
185     public function setStreet($street)
186     {
187         $this->street = $street;
188
189         return $this;
190     }
191
192     /**
193      * Get the value of cp
194      */
195     public function getCp()
196     {
197         return $this->cp;
198     }
199
200     /**
201      * Set the value of cp
202      *
203      * @return self
204      */
205     public function setCp($cp)
206     {
207         $this->cp = $cp;
208
209         return $this;
210     }
211
212     /**
213      * Get the value of city
214      */
215     public function getCity()
216     {
217         return $this->city;
218     }
```

```
219
220 /**
221  * Set the value of city
222  *
223  * @return self
224  */
225 public function setCity($city)
226 {
227     $this->city = $city;
228
229     return $this;
230 }
231
232 /**
233  * Get the value of countryId
234  */
235 public function getCountryId()
236 {
237     return $this->countryId;
238 }
239
240 /**
241  * Set the value of countryId
242  *
243  * @return self
244  */
245 public function setCountryId($countryId)
246 {
247     $this->countryId = $countryId;
248
249     return $this;
250 }
251
252 /**
253  * Get the value of latitude
254  */
255 public function getLatitude()
256 {
257     return $this->latitude;
258 }
259
260 /**
261  * Set the value of latitude
262  *
263  * @return self
264  */
265 public function setLatitude($latitude)
266 {
267     $this->latitude = $latitude;
268
269     return $this;
270 }
271
272 /**
273  * Get the value of longitude
```

```
274     */
275     public function getLongitude()
276     {
277         return $this->longitude;
278     }
279
280     /**
281      * Set the value of longitude
282      *
283      * @return self
284      */
285     public function setLongitude($longitude)
286     {
287         $this->longitude = $longitude;
288
289         return $this;
290     }
291
292     /**
293      * Get the value of categories
294      */
295     public function getCategories()
296     {
297         return $this->categories;
298     }
299
300     /**
301      * Set the value of longitude
302      *
303      * @return self
304      */
305     public function setCategories($categories)
306     {
307         $this->categories = $categories;
308
309         return $this;
310     }
311
312     /**
313      * add the value of categories
314      *
315      * @return self
316      */
317     public function addCategory($category)
318     {
319         array_push($this->categories, $category);
320
321         return $this;
322     }
323
324     /**
325      * Get the value of distanceFrom
326      */
327     public function getDistanceFrom()
328     {
```

```
329         return $this->distanceFrom;
330     }
331
332     /**
333      * Set the value of distanceFrom
334      *
335      * @return self
336      */
337     public function setDistanceFrom($distanceFrom)
338     {
339         $this->distanceFrom = $distanceFrom;
340
341         return $this;
342     }
343
344     public function jsonSerialize()
345     {
346         return get_object_vars($this);
347     }
348
349     public static function findAll() {
350         $sql = MyPdo::getInstance()->prepare('SELECT * FROM restaurants;');
351         $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'Restaurant');
352         $sql->execute();
353         $result = $sql->fetchAll();
354
355         return $result;
356     }
357
358     public function findById() {
359         $sql = MyPdo::getInstance()->prepare('SELECT * FROM restaurants WHERE id = :id;');
360         $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'Restaurant');
361         $sql->bindParam(':id', $this->getId());
362         $sql->execute();
363         $result = $sql->fetch();
364
365         return $result;
366     }
367
368     public function findCategoriesById() {
369         $sql = MyPdo::getInstance()->prepare('SELECT categories.* FROM categories LEFT JOIN
restaurants_categories as rc ON categories.id = rc.categoryId LEFT JOIN restaurants as r ON
rc.restaurantId = r.id WHERE r.id = :id;');
370         $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'Category');
371         $sql->bindParam(':id', $this->getId());
372         $sql->execute();
373         $result = $sql->fetchAll();
374
375         return $result;
376     }
377 }
```