

**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   //\\_/_\\_/\_ Nom et Prénom: GRIN Brian
5   \_\_/_/  /___/ Enseignant : Monsieur Antoine Schmid
6   /* * \  /^{^} ] Classe : I.DA-P4B
7   \_\0/_/ [ ] Date : 18.05.2022
8   / \_ [ / Nom du projet : GrinEat-API
9   \_\_/_ / / Version du projet : 1.0
10  [ [ / \_/_ Cours : TPI
11  _[ [ \ /_/
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 .
".json?access_token=pk.eyJ1IjoibWl5b2tpIiwiYSI6ImNsMW96ejJ5NTAzMjQza3B0NHB3bHMxYncifQ.03AlBHWNc
21
22     $curl = curl_init($url);
23     curl_setopt($curl, CURLOPT_URL, $url);
24     curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
25
26     $resp = curl_exec($curl);
27     curl_close($curl);
28     $data = json_decode($resp, true);
29
30     if (count($data["features"]) != 0) {
31         $addressFound = $latitudeAddr = $data["features"][0]["place_name"];
32         $latitudeAddr = $data["features"][0]["geometry"]["coordinates"][1];
33         $longitudeAddr = $data["features"][0]["geometry"]["coordinates"][0];
34
35         $result["street"] = $addressFound;
36         $result["coordinates"]["latitude"] = $latitudeAddr;
37         $result["coordinates"]["longitude"] = $longitudeAddr;
38     } else {
39         $result = null;
40     }
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    //\\_/_\\_  / \_  Nom et Prénom: GRIN Brian
5    \_\_/_/  / \_  Enseignant : Monsieur Antoine Schmid
6    /* * \  / ^^\_ Classe : I.DA-P4B
7    \_\0/_/ [ ] Date : 18.05.2022
8    / \_ [ / \_ Nom du projet : GrinEat-API
9    \_\_/_/ / \_ Version du projet : 1.0
10   [ [ / \_/_ Cours : TPI
11   _[ [ \ /_/
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    //\\_/_\\_  /  _\  Nom et Prénom: GRIN Brian
5    \_\_/_/  /  ^\_\_ Enseignant : Monsieur Antoine Schmid
6    /* * \  /  ^\_\_ Classe : I.DA-P4B
7    \_\_0/_/ [ ] Date : 18.05.2022
8    /  \_ [ ] / Nom du projet : GrinEat-API
9    \_\_ \_ / / Version du projet : 1.0
10   [ [ / \ \_/ Cours : TPI
11   _[ [ \ /_/
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 ajoutes à la propriété
27             // categories de l'objet Restaurant
28             $restaurant->addCategory($c->getNameFrench());
29         }
30
31         // Récupérer les menus et éléments relative au restaurant
32         $myMenuItems = new MenuItems();
33         $myMenuItems->setRestaurantId($id);
34         $menuItems = $myMenuItems->findItemsByRestaurantId();
35
36         array_push($message, "success");
37         http_response_code(200);
38         echo json_encode(["result" => ["info" => $restaurant, "menu_items" =>
39             $menuItems], "message" => $message]);
40     }
41     break;
42 default:
43     http_response_code(405);
44     break;
45 }
```

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

```
1 <?php
2 /*
3     /\   /\
4    //\\_/_\\_/\_ Nom et Prénom: GRIN Brian
5     \_   _/   /   / Enseignant : Monsieur Antoine Schmid
6     /* * \  /^^^] Classe : I.DA-P4B
7     \_\0/_/ [   ] Date : 18.05.2022
8     /   \_ [   / Nom du projet : GrinEat-API
9     \_   \_ /   / Version du projet : 1.0
10    [ [ /   \_/ Cours : TPI
11    _[ [ \_ /_
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
        limite autour de l'adresse du client par défaut 5 si rien spécifié.
26         $name = $data["name"] ?? null; // filtre par nom si le client souhaite faire une
        recherche
27         $categories = $data["categories"] ?? array(); // filtre par catégorie si le
        client en sélectionne
28
29         if (isset($coordinates["latitude"]) && isset($coordinates["longitude"]) &&
        !isset($info)) {
30             $info["coordinates"] = $coordinates;
31
32             if ($address != null) {
33                 $info["street"] = $address;
34             }
35         }
36
37         if ($address != null && !isset($info)) {
38             $info = convertAdressToCoordinate($address);
39         }
40
41         if ($info != null) {
42             $latFrom = $info["coordinates"]["latitude"]; // latitude de l'adresse du
        client
43             $lonFrom = $info["coordinates"]["longitude"]; // longitude de l'adresse du
        client
44
45             $restaurants = Restaurant::findAll();
46
47             foreach ($restaurants as $r) {
```

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

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

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

```
1 <?php
2 /*
3  /\  /
4  //\\_/_\\  /__/ Nom et Prénom: GRIN Brian
5  \_ _/_/  /____ Enseignant : Monsieur Antoine Schmid
6  /* * \  /^^^] Classe : I.DA-P4B
7  \_\0/_/ [ ] Date : 18.05.2022
8  /  \_ [ / Nom du projet : GrinEat-API
9  \  \_ / / Version du projet : 1.0
10 [ [ / \_/_/ Cours : TPI
11 _[ [ \ /_/
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
39         return $this;
40     }
41
42     /**
43      * Get the value of restaurantId
44      */
45     public function getRestaurantId()
46     {
47         return $this->restaurantId;
48     }
49
50     /**
51      * Set the value of restaurantId
52      */
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 = :id');
150         $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'MenuItems');
151         $sql->bindParam(':id', $this->getRestaurantId());
152         $sql->execute();
153         $result = $sql->fetchAll();
154
155         return $result;
156     }
157 }
```

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

```
1 <?php
2 /*
3     /\   /\
4    //\\_/_\\_  / \_  Nom et Prénom: GRIN Brian
5    \_\_/_/  / \_  Enseignant : Monsieur Antoine Schmid
6    /* * \  / ^^\_ Classe : I.DA-P4B
7    \_\0/_/ [ ] Date : 18.05.2022
8    / \_ [ / \_ / Nom du projet : GrinEat-API
9    \_\_\_ / / Version du projet : 1.0
10   [ [ / \_/_ Cours : TPI
11   _[ [ \ /_/
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 : $instanceMyPdo = MyPdo::getInstance();
41 * @return l'unique objet de la classe MyPdo
42 */
43     public static function getInstance()
44 {
45         if(self::$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    //\\_/_\\_  _____ Nom et Prénom: GRIN Brian
5     \_   _/  /_____| Enseignant : Monsieur Antoine Schmid
6      * * \  /^{^} ] Classe : I.DA-P4B
7      \_0/_/ [   ] Date : 18.05.2022
8      /  \_ [   / Nom du projet : GrinEat-API
9      \  \_ /  / Version du projet : 1.0
10     [ [ / \_ / Cours : TPI
11     _[ [ \_ /_
12
13 */
14
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      */
```

```
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
370 restaurants_categories as rc ON categories.id = rc.categoryId LEFT JOIN restaurants as r ON
371 rc.restaurantId = r.id WHERE r.id = :id;');
372         $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'Category');
373         $sql->bindParam(':id', $this->getId());
374         $sql->execute();
375         $result = $sql->fetchAll();
376
377         return $result;
378     }
379 }
```

## c:\grin-eat-tpi2022\grineat-client\index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>GrinEat - Home</title>
8     <script src='https://api.mapbox.com/mapbox-gl-js/v2.8.2/mapbox-gl.js'></script>
9     <link href='https://api.mapbox.com/mapbox-gl-js/v2.8.2/mapbox-gl.css' rel='stylesheet' />
10    <script
11        src="https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-geocoder/v5.0.0/mapbox-gl-
12        geocoder.min.js"></script>
13    <link rel="stylesheet"
14        href="https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-geocoder/v5.0.0/mapbox-
15        gl-geocoder.css"
16        type="text/css">
17    <link rel="stylesheet" href=".//assets/css/style.css">
18 </head>
19 <body>
20     <header>
21         <h3 class="text-white">Grin<strong>Eat</strong></h3>
22     </header>
23     <div class="main">
24         <div class="search-container text-white text-center">
25             <h3>Faites vous livrer vos plats de votre restaurant favoris directement chez
26             vous grâce à <i>GrinEat</i>.</h3>
27             <div class="search-inputs-container">
28                 <div id="idSearchBarContainer" class=""></div>
29                 <button id="idBtnSearch" class="btn-search text-white">Chercher</button>
30             </div>
31         </div>
32         <footer>
33             <p>&copy; 2022 GrinEat, developped by <a href="https://github.com/miyokidev"
34             class="text-white">@miyokidev</a>.</p>
35 </body>
36 </html>
```

## c:\grin-eat-tpi2022\grineat-client\menu.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8     <title>GrinEat - Menu</title>
9     <script src='https://api.mapbox.com/mapbox-gl-js/v2.8.2/mapbox-gl.js'></script>
10    <link href='https://api.mapbox.com/mapbox-gl-js/v2.8.2/mapbox-gl.css' rel='stylesheet'
11 />
12    <script
13        src="https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-geocoder/v5.0.0/mapbox-
14 gl-geocoder.min.js"></script>
15    <link rel="stylesheet"
16        href="https://api.mapbox.com/mapbox-gl-js/plugins/mapbox-gl-geocoder/v5.0.0/mapbox-
17 gl-geocoder.css"
18        type="text/css">
19    <link rel="stylesheet" href=".//assets/css/style.css">
20 </head>
21
22 <body>
23     <header>
24         <h3 id="idLogo" class="text-white">Grin<strong>Eat</strong></h3>
25     </header>
26     <div class="main-menus">
27         <div class="restaurant-info-container text-white">
28             <img id="idImage">
29             <div class="restaurant-name-city">
30                 <h2 id="idName"></h2>
31                 <p><span id="idCp"></span> - <span id="idCity"></span></p>
32             </div>
33             <div class="restaurant-street-number">
34                 <div>
35                     <h1>Adresse</h1>
36                     <p id="idStreet"></p>
37                 </div>
38                 <div>
39                     <h1>Numéro</h1>
40                     <p id="idPhone"></p>
41                 </div>
42                 <div id="idCategories">
43                     <h1>Catégories</h1>
44                 </div>
45             </div>
46         </div>
47         <div class="menu-items-container">
48             <div id="idListMenuItems">
49                 <div class="card-menu">
50                     
52
53     <div class="card-menu">
54         
56         <div class="card-menu-body">
57             <p>6 Chicken McNuggets</p>
58         </div>
59     </div>
60
61     <div class="card-menu">
62         
64         <div class="card-menu-body">
65             <p>6 Chicken McNuggets</p>
66             <p class="card-menu-description">Le burger le plus légendaire du
67 monde est désormais encore meilleur. De la viande de bœuf bien juteuse de la boucherie Bell,
68 de la salade croquante et du fromage fondu, dans un petit pain fabriqué avec de la farine de
69 qualité IP-Suisse. Le tout rehaussé de la mystérieuse sauce Big Mac. Es-tu à la hauteur de
70 son goût ?</p>
71         </div>
72     </div>
73
74
75     <div class="card-menu">
76         
78         <div class="card-menu-body">
79             <p>6 Chicken McNuggets</p>
80         </div>
81     </div>
82
83     <div class="card-menu">
84         
86         <div class="card-menu-body">
87             <p>6 Chicken McNuggets</p>
88         </div>
89
90     <div class="card-menu">
91         
93         <div class="card-menu-body">
94             <p>6 Chicken McNuggets</p>
95         </div>
96     <div class="card-menu">
```

```
97             
99         <div class="card-menu-body">
100            <p>6 Chicken McNuggets</p>
101        </div>
102    </div>
103
104    <div class="card-menu">
105        
107         <div class="card-menu-body">
108            <p>6 Chicken McNuggets</p>
109        </div>
110    </div>
111
112    <div class="card-menu">
113        
115         <div class="card-menu-body">
116            <p>6 Chicken McNuggets</p>
117        </div>
118    </div>
119
120    <div class="card-menu">
121        
123         <div class="card-menu-body">
124            <p>6 Chicken McNuggets</p>
125        </div>
126    </div>
127
128    <div class="card-menu">
129        
131         <div class="card-menu-body">
132            
134         <div class="card-menu-body">
135            <p>6 Chicken McNuggets</p>
136        </div>
137    </div>
138  </div>
139 </div>
140 <footer>
141   <p>&copy; 2022 GrinEat, developped by <a href="https://github.com/miyokidev"
142   class="text-white">@miyokidev</a>.
143   </p>
144 </footer>
145 <script src=".//assets/js/menu.js"></script>
146
147 </body>
```

146 |  
147 | [`</html>`](#)

## c:\grin-eat-tpi2022\grineat-client\restaurants.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>GrinEat - Restaurants</title>
9   <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
10    integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq
11    /sMZM19scR4PsZChSR7A==">
12    crossorigin="" />
13    <!-- Make sure you put this AFTER Leaflet's CSS -->
14    <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"
15     integrity="sha512-
16    XQoYMqMTK8LvdXXYG3nZ448h0EQiglfqkJs1NOQV44cWhUrBc8PKAOcXy20w0vlaXaVUearIOBhiXZ5V3ynxwA==">
17    crossorigin=""></script>
18    <link rel="stylesheet" href=".//assets/css/style.css">
19 </head>
20
21 <body>
22   <header class="text-white shadow-header">
23     <h3 id="idLogo">Grin<strong>Eat</strong></h3>
24     <p id="idAddress" class="address-display">address</p>
25   </header>
26   <div class="tools-container">
27     <input id="idSearchName" type="text" placeholder="Restaurants">
28     <div>
29       <label for="idSelectCategory" class="text-white">Catégories: </label>
30       <select id="idSelectCategory" onchange="categoryChanged(); changed()">
31         <option value=""></option>
32       </select>
33     </div>
34     <div class="select-editable">
35       <select id="idSelectRadius"
36         onchange="this.nextElementSibling.value=this.value; radiusChanged();
37         changed();">
38         <option value="1 km">1 km</option>
39         <option value="3 km">3 km</option>
40         <option value="5 km" selected>5 km</option>
41         <option value="8 km">8 km</option>
42         <option value="10 km">10 km</option>
43         <option value="20 km">20 km</option>
44       </select>
45       <input id="idInputRadius" type="text" name="format" value="5 km"
46       onchange="radiusChanged(); changed();"/>
47     </div>
48   </div>
49   <div class="main-restaurants">
50     <div class="list-container">
51       <div id="idListDisplay"></div>
52     </div>
53
54     <div class="map-container">
```

```
51      <div id="map"></div>
52    </div>
53  </div>
54  <footer>
55    <p>&copy; 2022 GrinEat, developped by <a href="https://github.com/miyokidev"
56    class="text-white">@miyokidev</a>.
57    </p>
58  </footer>
59  <script src=".//assets/js/restaurants.js"></script>
60</body>
61</html>
```

## c:\grin-eat-tpi2022\grineat-client\assets\css\style.css

```
1 * {
2     /*padding: 0;*/
3     margin: 0;
4 }
5
6 :root {
7     --bs-font-sans-serif: system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue",
8     Arial, "Noto Sans", "Liberation Sans", sans-serif, "Apple Color Emoji", "Segoe UI Emoji",
9     "Segoe UI Symbol", "Noto Color Emoji";
10    --bs-body-font-family: var(--bs-font-sans-serif);
11    --bs-body-font-size: 1rem;
12    --bs-body-font-weight: 400;
13    --bs-body-line-height: 1.5;
14    --bs-white-rgb: 255, 255, 255;
15    --bs-greenBg-rgb: 25, 135, 84;
16 }
17
18 body {
19     height: 100%;
20     font-family: var(--bs-body-font-family);
21     font-size: var(--bs-body-font-size);
22     font-weight: var(--bs-body-font-weight);
23     line-height: var(--bs-body-line-height);
24     text-shadow: 0 .05rem .1rem rgba(0, 0, 0, .5);
25     --bs-bg-opacity: 1;
26     background-color: rgba(var(--bs-greenBg-rgb), var(--bs-bg-opacity)) !important;
27     box-shadow: inset 0 0 5rem rgba(0, 0, 0, .5);
28     overflow: hidden;
29 }
30
31 header {
32     width: 100%;
33     height: 60px;
34     display: flex;
35     /* align horizontal */
36     justify-content: center;
37     /* align vertical */
38     align-items: center;
39     user-select: none;
40     flex-wrap: wrap;
41 }
42
43 header
44
45 .header-logo-start {
46     /* align horizontal */
47     justify-content: start;
48 }
49
50 .header-logo-start h3 {
51     margin-left: 1rem;
52 }
53 #idLogo {
```

```
53     position: absolute;
54     left: 10px;
55     cursor: pointer;
56 }
57
58 .address-display {
59     cursor: pointer;
60 }
61
62 .shadow-header {
63     box-shadow: 0 4px 2px -2px rgba(0, 0, 0, .2);
64 }
65
66 h1,
67 h2,
68 h3,
69 h4,
70 h5,
71 h6 {
72     font-size: 1.75rem;
73     font-weight: 500;
74     line-height: 1.2;
75 }
76
77 .text-white {
78     --bs-text-opacity: 1;
79     color: rgba(var(--bs-white-rgb), var(--bs-text-opacity)) !important;
80 }
81
82 /* If the screen size is 601px wide or more, set the font-size of <div> to 80px */
83 @media screen and (min-width: 661px) {
84     .main {
85         height: calc(100vh - (60px + 50px));
86         display: flex;
87         justify-content: center;
88         /* align horizontal */
89         align-items: center;
90         /* align vertical */
91     }
92
93     .search-container {
94         width: 60%;
95         text-align: center !important;
96         padding-left: 1rem;
97         padding-right: 1rem;
98     }
99
100    .search-container h3 {
101        font-size: 200%;
102        margin-bottom: 1rem;
103    }
104
105    .btn-search {
106        background-color: rgb(1, 114, 52);
107        border: none;
```

```
108     font-weight: 700 !important;
109     padding: .375rem .75rem;
110     border-radius: .25rem;
111     transform: scale(1.25);
112     flex: 0;
113 }
114
115 .mapboxgl-ctrl-geocoder {
116     flex: 1;
117     transform: scale(1.25);
118     min-width: 100%;
119 }
120
121 .main-restaurants {
122     min-width: 100%;
123     height: calc(100vh - (60px + 30px + 50px));
124     display: flex;
125 }
126 }
127
128 /* If the screen size is 600px wide or less, set the font-size of <div> to 30px */
129 @media screen and (max-width: 660px) {
130     .main {
131         height: calc(100vh - (60px + 50px));
132         display: flex;
133         justify-content: center;
134         /* align horizontal */
135         align-items: center;
136         /* align vertical */
137     }
138
139     .search-container {
140         width: 60%;
141         text-align: center !important;
142         padding-left: 1rem;
143         padding-right: 1rem;
144     }
145
146     .search-container h3 {
147         font-size: 150%;
148         margin-bottom: 0.5rem;
149     }
150
151     .btn-search {
152         background-color: rgb(1, 114, 52);
153         border: none;
154         font-weight: 700 !important;
155         padding: .375rem .75rem;
156         border-radius: .25rem;
157         transform: scale(1.25);
158         flex: 0;
159     }
160
161     .mapboxgl-ctrl-geocoder {
162         flex: 1;
```

```
163     transform: scale(0.75);
164     width: 300px !important;
165 }
166
167 .main-restaurants {
168     height: calc(100vh - (60px + 30px + 50px));
169     display: flex;
170     flex-wrap: wrap-reverse;
171 }
172
173 .map-container {
174     height: 400px;
175 }
176 }
177
178 .btn-search:hover {
179     background-color: rgb(0, 189, 85);
180 }
181
182 .btn-search:enabled {
183     cursor: pointer;
184 }
185
186 .btn-search:disabled {
187     background-color: #6c757d;
188     opacity: .65;
189     pointer-events: none;
190 }
191
192 .search-inputs-container {
193     display: flex;
194     justify-content: center;
195     flex-wrap: wrap;
196     gap: 10px 40px;
197 }
198
199 footer {
200     height: 50px;
201     display: flex;
202     justify-content: center;
203     /* align horizontal */
204     align-items: center;
205     /* align vertical */
206     color: rgba(255, 255, 255, .5) !important;
207 }
208
209 .tools-container {
210     height: 40px;
211     display: flex;
212     flex-wrap: wrap;
213     gap: 10px 30px;
214     display: flex;
215     align-items: center;
216     /* align vertical */
217 }
```

```
218
219 .list-container {
220   flex: 1 1 15%;
221   background-color: white;
222   min-height: 100%;
223   border: solid 2px black;
224 }
225
226 #idListDisplay {
227   height: 100%;
228   overflow-y: scroll;
229 }
230
231 .card {
232   width: 100%;
233   height: 280px;
234   display: flex;
235   flex-direction: column;
236   align-items: center;
237   border-bottom: solid 2px black;
238 }
239
240 .card img {
241   object-fit: cover;
242   height: 70%;
243   min-width: 100%;
244 }
245
246 .card-body {
247   display: flex;
248   flex-direction: column;
249   width: 100%;
250   height: 30%;
251   justify-content: center;
252   align-items: center;
253 }
254
255 .card p {
256   margin: 5px 0px 5px 0px;
257 }
258
259 .card button {
260   width: 30%;
261   height: 40%;
262   background-color: #0d6efd;
263   border: none;
264   font-weight: 700 !important;
265   padding: .375rem .75rem;
266   border-radius: .25rem;
267   --bs-text-opacity: 1;
268   color: rgba(var(--bs-white-rgb), var(--bs-text-opacity)) !important;
269   cursor: pointer;
270 }
271
272 .card button:hover {
```

```
273     background-color: #0b5ed7;
274     border-color: #0a58ca;
275 }
276
277 .map-container {
278     flex: 1 1 85%;
279     border: solid 2px black;
280     border-left: none;
281 }
282
283 #map {
284     min-height: 100%;
285 }
286
287 #idSearchName {
288     font-size: 16px;
289     height: 20px;
290     margin-left: 1rem;
291     padding: 1px 4px 1px 35px;
292     background: transparent url("../images/search.png") no-repeat 8px center;
293     background-size: 20px;
294     background-color: white;
295     border-radius: .50rem;
296 }
297
298 #idSearchName:focus {
299     outline: none;
300     border: solid 2px;
301     border-color: rgb(1, 114, 52);
302 }
303
304 #idSelectCategory {
305     font-size: 16px;
306     height: 25px;
307 }
308
309 /* EDITABLE SELECT */
310
311 .select-editable {
312     position: relative;
313     background-color: white;
314     border: solid grey 1px;
315     width: 120px;
316     height: 18px;
317 }
318
319 .select-editable select {
320     position: absolute;
321     top: 0px;
322     left: 0px;
323     border: none;
324     font-size: 16px;
325     width: 120px;
326     margin: 0;
327 }
```

```
328
329 .select-editable input {
330   position: absolute;
331   top: 0px;
332   left: 0px;
333   width: 100px;
334   padding: 1px;
335   font-size: 14px;
336   border: none;
337 }
338
339 .select-editable select:focus,
340 .select-editable input:focus {
341   outline: none;
342 }
343
344 /* END EDITABLE SELECT */
345
346 .main-menus {
347   height: calc(100vh - (60px + 50px));
348   margin-top: 5px;
349 }
350
351 .restaurant-info-container {
352   width: 100%;
353   height: 40%;
354   display: flex;
355   gap: 100px;
356   flex-wrap: wrap;
357   border-bottom: solid 2px rgb(1, 114, 52);
358   padding: 5px;
359 }
360
361 .restaurant-info-container img {
362   object-fit: cover;
363   width: 20%;
364   height: 300px;
365 }
366
367 .restaurant-name-city {
368   padding: 1rem;
369   width: auto;
370   display: flex;
371   flex-direction: column;
372   gap: 50px;
373 }
374
375 .restaurant-name-city p {
376   font-size: 1.25rem;
377 }
378
379 .restaurant-street-number {
380   padding: 1rem;
381   width: auto;
382   display: flex;
```

```
383     gap: 20px;
384     flex-direction: column;
385 }
386
387 .menu-items-container {
388     height: 60%;
389     overflow-y: scroll;
390     border-bottom: solid 2px rgb(1, 114, 52);
391 }
392
393 #idListMenuItems {
394     display: flex;
395     justify-content: space-between !important;
396     flex-wrap: wrap;
397 }
398
399 .card-menu {
400     flex: 1 0 calc(33% - 2rem);
401     max-width: calc(33% - 2rem);
402     background-color: white;
403     z-index: 0;
404     padding: .5rem;
405     margin: 1rem .5rem;
406     display: flex;
407     flex-direction: row;
408     flex-wrap: wrap;
409 }
410
411 .card-menu-body {
412     width: 65%;
413 }
414
415 .card-menu img {
416     width: 35%;
417 }
418
419 .card-menu-description {
420     font-size: .75rem;
421 }
```

## c:\grin-eat-tpi2022\grineat-client\assets\js\main.js

```
1 /*  
2  /\_/\_/  
3  /\\\_//\\_/_  
4  \_\_/_/_/ / / Nom et Prénom: GRIN Brian  
5  /* * \ /[^^] Enseignant : Monsieur Antoine Schmid  
6  \_\_0/_/ [ ] Classe : I.DA-P4B  
7  / \_ [ / Date : 18.05.2022  
8  \_\_ / / Nom du projet : GrinEat-Client  
9  [ [ / \_ / Version du projet : 1.0  
10 _[ [ \ /_/ Cours : TPI  
11  
12 */  
13  
14 // Création de l'autocomplétion qu'on récupère grâce au token public de MapBox  
15 mapboxgl.accessToken =  
'pk.eyJ1IjoibW15b2tpIiwiYSI6ImNsMw96ejJ5NTAzMjQza3B0NHB3bHMxYncifQ.03AlBHWNd8MiauuZz_sSNQ';  
16 const geocoder = new MapboxGeocoder({  
17   accessToken: mapboxgl.accessToken,  
18   types: 'address'  
19});  
20  
21 // Récupération des éléments de la page  
22 const searchBarContainer = document.getElementById('idSearchBarContainer'); // Conteneur du  
champ de recherche  
23 const btnSearch = document.getElementById('idBtnSearch'); // Bouton pour effectuer la  
recherche  
24 let address = {};  
// objet address qui prendra en propriété le résultat de l'autocomplete ou  
la saisie de l'utilisateur dans le champ  
25  
26 // Evenement lors du chargement de la page  
27 addEventListener("DOMContentLoaded", () => {  
  btnSearch.disabled = true; // On désactive par défaut le bouton rechercher lors du  
chargement car le champ de saisi est vide  
  document.getElementsByClassName("mapboxgl-ctrl-geocoder--input")  
[0].setAttribute("placeholder", "Ex: Quai Capo D'istria 9, 1205 Genève"); // On ajoute un  
placeholder au champ créé par MapBox pour l'autocomplétion  
28});  
29  
30 geocoder.addTo('#idSearchBarContainer'); // Ajout du champ d'autocomplétion dans le conteneur  
31  
32  
33 // Evenement se lançant lorsque l'utilisateur clique sur une proposition de l'autocomplétion  
34 geocoder.on('result', (e) => {  
  address.coordinates = {latitude : e.result.center[1], longitude : e.result.center[0]};  
  address.street = e.result.place_name;  
35});  
36  
37 // Evenement lancé lorsque l'utilisateur clique sur le bouton recherché  
38 btnSearch.addEventListener("click", () => {  
  // On vérifie si l'objet address contient des informations  
  if (Object.keys(address).length !== 0) {  
    sessionStorage.setItem('address', JSON.stringify(address)); // Sauvegarde en session  
    de l'adresse de la recherche pour s'en servir sur la page suivante  
    location.href = "restaurants.html";  
  } else {  
    alert("Adresse invalide");  
  }  
});
```

```
48     }
49 });
50
51 // Evenement lancé à chaque fois que l'utilisateur relâche une touche du clavier sur le champ
52 searchBarContainer.addEventListener('keyup', () => {
53     var searchBar = document.getElementsByClassName("mapboxgl-ctrl-geocoder--input")[0];
54
55     address.coordinates = null;
56     address.street = searchBar.value;
57
58     if (searchBar.value === "") {
59         btnSearch.disabled = true;
60     } else {
61         btnSearch.disabled = false;
62     }
63});
```

## c:\grin-eat-tpi2022\grineat-client\assets\js\menu.js

```
1 /*  
2  /\_/\_/  
3  //\\_/_\\_/_\_ Nom et Prénom: GRIN Brian  
4  \_\_/_/_/ /____ Enseignant : Monsieur Antoine Schmid  
5  /* * \_ /^{^} ] Classe : I.DA-P4B  
6  \_\_0/_/_ [ ] Date : 18.05.2022  
7  / \_\_ [ / Nom du projet : GrinEat-Client  
8  \_\_ \_\_ / / Version du projet : 1.0  
9  [ [ / \_\_ / Cours : TPI  
10 _[ [ \_\_ / /  
11 */  
12  
13  
14 const logo = document.getElementById("idLogo"); // Logo du nav  
15 // Variables pour la récupération de l'id du restaurant en paramètre dans l'URL  
16 const queryString = window.location.search;  
17 const urlParams = new URLSearchParams(queryString);  
18 // Variable des différents éléments pour l'affichage des informations  
19 const image = document.getElementById("idImage");  
20 const nameRestaurant = document.getElementById("idName");  
21 const cp = document.getElementById("idCp");  
22 const city = document.getElementById("idCity");  
23 const street = document.getElementById("idStreet");  
24 const phone = document.getElementById("idPhone");  
25 const categories = document.getElementById("idCategories");  
26  
27 // Rediriger vers l'index si on clique sur le logo.  
28 logo.addEventListener("click", () => {  
29     sessionStorage.removeItem('address');  
30     location.href = "index.html";  
31 });  
32  
33 addEventListener("DOMContentLoaded", () => {  
34     if (sessionStorage.getItem('address') == null) {  
35         location.href = "index.html";  
36     } else {  
37         let idRestaurant = urlParams.get('idRestaurant'); // Récupération de la valeur donné  
dans l'URL  
38         fetch(`http://localhost/grin-eat-tpi2022/grineat-api/restaurants/${idRestaurant}  
/menus`)  
39             .then(function (response) {  
40                 return response.json()  
41             }).then(function (json) {  
42                 displayInfoMenuItems(json.result);  
43             }).catch(function (ex) {  
44                 console.log(ex);  
45             });  
46     }  
47 });  
48  
49 // Fonction appelée en callback après la récupération des données avec fetch  
50 function displayInfoMenuItems(restaurant) {  
51     // On assigne pour tout les éléments de la description les données du restaurant  
52     image.setAttribute("src", restaurant.info.image);
```

```
53     nameRestaurant.innerText = restaurant.info.name;
54     cp.innerText = restaurant.info.cp;
55     city.innerText = restaurant.info.city;
56     street.innerText = restaurant.info.street;
57     phone.innerText = restaurant.info.phone;
58     // On affiche que les 2 premières catégories du restaurant
59     for (let j = 0; j < restaurant.info.categories.length; j++) {
60         if (j < 2) {
61             let category = document.createElement('p');
62             category.innerText = restaurant.info.categories[j];
63             categories.appendChild(category);
64         } else {
65             break;
66         }
67     }
68
69     // On remplit la liste des menus
70     document.getElementById("idListMenuItems").innerHTML = "";
71     for (let i = 0; i < restaurant.menu_items.length; i++) {
72         let card = document.createElement('div');
73         card.setAttribute('id', `item${restaurant.menu_items[i].id}`);
74         card.classList.add("card-menu");
75         card.innerHTML = `
76             <div class="card-menu-body">
77                 <p>${restaurant.menu_items[i].name}</p>
78                 <p class="card-menu-description">${restaurant.menu_items[i].description}</p>
79                 <p>${restaurant.menu_items[i].price} CHF</p>
80             </div>`;
81         document.getElementById("idListMenuItems").appendChild(card);
82     }
83 }
```

## c:\grin-eat-tpi2022\grineat-client\assets\js\restaurants.js

```
1 /*  
2  /\_/\_/  
3  //\\_/_/\\_/_  
4  \_\_ _/_ / / Nom et Prénom: GRIN Brian  
5  / * * \ / ^^^] Enseignant : Monsieur Antoine Schmid  
6  \_\0/_/ [ ] Classe : I.DA-P4B  
7  / \_ [ / Date : 18.05.2022  
8  \_\_\_ / / Nom du projet : GrinEat-Client  
9  [ [ / \_/_ Version du projet : 1.0  
10 _[ [ \ /_/_ Cours : TPI  
11  
12 */  
13  
14 const search = {}; // objet qu'on envoit dans le body lors des appelles au serveur  
15 search.categories = [];  
16 const addressDisplay = document.getElementById("idAddress"); // Adresse affichée dans le nav  
17 const searchName = document.getElementById("idSearchName");  
18 const logo = document.getElementById("idLogo"); // Logo du nav  
19 let address = {}; // objet address contenant la rue et les coordonnées de l'adresse sélectionnée par l'utilisateur  
20 let restaurants = []; // Tableau des restaurants récupérés du serveur  
21 let categories = []; // Tableau des catégories récupérés du serveur  
22 let map;  
23 let markers;  
24  
25 var iconHome = L.icon({  
26   iconUrl: './assets/images/home.png',  
27   iconSize: [41, 41], // size of the icon  
28   iconAnchor: [20, 41]  
29 });  
30  
31 var iconRestaurant = L.icon({  
32   iconUrl: './assets/images/restaurant.png',  
33   iconSize: [41, 41], // size of the icon  
34   iconAnchor: [20, 41]  
35 });  
36  
37  
38 addEventListener("DOMContentLoaded", () => {  
39   if (sessionStorage.getItem('address') == null) {  
40     location.href = "index.html";  
41   } else {  
42     address = JSON.parse(sessionStorage.getItem('address'));  
43  
44     // On fait une promesse pour d'abord récupérer les catégories données par l'API pour  
45     select quand c'est terminé on resolve pour effectuer la suite  
46     new Promise((resolve, reject) => {  
47       fetch('http://localhost/grin-eat-tpi2022/grineat-api/categories')  
48         .then(function (response) {  
49           return response.json()  
50         }).then(function (json) {  
51           categories = json.result;  
52           var select = document.getElementById("idSelectCategory");  
53         })  
54     })  
55   }  
56 })
```

```
53 // Remplir la liste déroulante de catégories.  
54 for (let i = 0; i < categories.length; i++) {  
55     var option = document.createElement('option');  
56     option.setAttribute('value', categories[i].id);  
57     option.appendChild(document.createTextNode(categories[i].nameFrench));  
58     select.appendChild(option);  
59 }  
60 resolve();  
61 }).catch(function (ex) {  
62     reject(ex);  
63 })  
64 ).then(() => {  
65     search.address = address.street;  
66     if (address.coordinates != null) {  
67         if (address.coordinates.latitude != null && address.coordinates.longitude != null)  
68             search.coordinates = address.coordinates;  
69     }  
70 }  
71 radiusChanged();  
72 sendData(displayMapList, "http://localhost/grin-eat-tpi2022/grineat-api/restaurant");  
73});  
74}  
75});  
76  
77 // Afficher un modal pour modifier son adresse quand on clique sur l'adresse.  
78 addressDisplay.addEventListener("click", () => {  
79     flyToMarker(address.coordinates);  
80});  
81  
82 // Rediriger vers l'index si on clique sur le logo.  
83 logo.addEventListener("click", () => {  
84     sessionStorage.removeItem('address');  
85     location.href = "index.html";  
86});  
87  
88 // A chaque fois que l'utilisateur relâche une touche la variable search prend en valeur la saisie et effectue un appel au serveur  
89 searchName.addEventListener("keyup", () => {  
90     search.name = searchName.value;  
91     changed();  
92});  
93  
94 // Fonction pour appeler le serveur API cette fonction est appelé à chaque fois qu'on filtre ou à chaque fois quand on arrive sur la page  
95 async function sendData(successCallBack, link, obj) {  
96     fetch(link, { method: 'POST', body: JSON.stringify(obj) }).then(function (response) {  
97         response.json().then(function (myJson) {  
98             // On vérifie si les données qu'on a déjà sont similaires à ceux qu'on reçoit pour recharger toute la page si le contenu est le même  
99             if (JSON.stringify(restaurants) != JSON.stringify(myJson.result.restaurants) || restaurants.length == 0) {  
100                 restaurants = myJson.result.restaurants;  
101                 if (address != myJson.result.address) {  
102                     address = myJson.result.address;  
103                     sessionStorage.setItem('address', JSON.stringify(address));  
104                 }  
105             }  
106             successCallBack();  
107         });  
108     });  
109 };  
110  
111 // Fonction qui permet de faire une recherche dans la liste des restaurants  
112 function search() {  
113     let searchValue = searchName.value;  
114     if (searchValue != '') {  
115         let filteredRestaurants = restaurants.filter(function (restaurant) {  
116             return restaurant.nameFrench.toLowerCase().includes(searchValue.toLowerCase());  
117         });  
118         displayList(filteredRestaurants);  
119     }  
120 }  
121  
122 // Fonction qui permet de faire une recherche dans la liste des restaurants  
123 function searchName() {  
124     let searchValue = searchName.value;  
125     if (searchValue != '') {  
126         let filteredRestaurants = restaurants.filter(function (restaurant) {  
127             return restaurant.nameFrench.toLowerCase().includes(searchValue.toLowerCase());  
128         });  
129         displayList(filteredRestaurants);  
130     }  
131 }  
132  
133 // Fonction qui permet de faire une recherche dans la liste des restaurants  
134 function searchAddress() {  
135     let searchValue = address.value;  
136     if (searchValue != '') {  
137         let filteredRestaurants = restaurants.filter(function (restaurant) {  
138             return restaurant.address.toLowerCase().includes(searchValue.toLowerCase());  
139         });  
140         displayList(filteredRestaurants);  
141     }  
142 }  
143  
144 // Fonction qui permet de faire une recherche dans la liste des restaurants  
145 function searchCategory() {  
146     let searchValue = category.value;  
147     if (searchValue != '') {  
148         let filteredRestaurants = restaurants.filter(function (restaurant) {  
149             return restaurant.category.toLowerCase().includes(searchValue.toLowerCase());  
150         });  
151         displayList(filteredRestaurants);  
152     }  
153 }  
154  
155 // Fonction qui permet de faire une recherche dans la liste des restaurants  
156 function searchRadius() {  
157     let searchValue = radius.value;  
158     if (searchValue != '') {  
159         let filteredRestaurants = restaurants.filter(function (restaurant) {  
160             return restaurant.radius == searchValue;  
161         });  
162         displayList(filteredRestaurants);  
163     }  
164 }  
165  
166 // Fonction qui permet de faire une recherche dans la liste des restaurants  
167 function searchPrice() {  
168     let searchValue = price.value;  
169     if (searchValue != '') {  
170         let filteredRestaurants = restaurants.filter(function (restaurant) {  
171             return restaurant.price == searchValue;  
172         });  
173         displayList(filteredRestaurants);  
174     }  
175 }  
176  
177 // Fonction qui permet de faire une recherche dans la liste des restaurants  
178 function searchRating() {  
179     let searchValue = rating.value;  
180     if (searchValue != '') {  
181         let filteredRestaurants = restaurants.filter(function (restaurant) {  
182             return restaurant.rating == searchValue;  
183         });  
184         displayList(filteredRestaurants);  
185     }  
186 }  
187  
188 // Fonction qui permet de faire une recherche dans la liste des restaurants  
189 function searchDistance() {  
190     let searchValue = distance.value;  
191     if (searchValue != '') {  
192         let filteredRestaurants = restaurants.filter(function (restaurant) {  
193             return restaurant.distance == searchValue;  
194         });  
195         displayList(filteredRestaurants);  
196     }  
197 }  
198  
199 // Fonction qui permet de faire une recherche dans la liste des restaurants  
200 function searchCuisines() {  
201     let searchValue = cuisines.value;  
202     if (searchValue != '') {  
203         let filteredRestaurants = restaurants.filter(function (restaurant) {  
204             return restaurant.cuisines == searchValue;  
205         });  
206         displayList(filteredRestaurants);  
207     }  
208 }  
209  
210 // Fonction qui permet de faire une recherche dans la liste des restaurants  
211 function searchReviews() {  
212     let searchValue = reviews.value;  
213     if (searchValue != '') {  
214         let filteredRestaurants = restaurants.filter(function (restaurant) {  
215             return restaurant.reviews == searchValue;  
216         });  
217         displayList(filteredRestaurants);  
218     }  
219 }  
220  
221 // Fonction qui permet de faire une recherche dans la liste des restaurants  
222 function searchRating() {  
223     let searchValue = rating.value;  
224     if (searchValue != '') {  
225         let filteredRestaurants = restaurants.filter(function (restaurant) {  
226             return restaurant.rating == searchValue;  
227         });  
228         displayList(filteredRestaurants);  
229     }  
230 }  
231  
232 // Fonction qui permet de faire une recherche dans la liste des restaurants  
233 function searchDistance() {  
234     let searchValue = distance.value;  
235     if (searchValue != '') {  
236         let filteredRestaurants = restaurants.filter(function (restaurant) {  
237             return restaurant.distance == searchValue;  
238         });  
239         displayList(filteredRestaurants);  
240     }  
241 }  
242  
243 // Fonction qui permet de faire une recherche dans la liste des restaurants  
244 function searchCuisines() {  
245     let searchValue = cuisines.value;  
246     if (searchValue != '') {  
247         let filteredRestaurants = restaurants.filter(function (restaurant) {  
248             return restaurant.cuisines == searchValue;  
249         });  
250         displayList(filteredRestaurants);  
251     }  
252 }  
253  
254 // Fonction qui permet de faire une recherche dans la liste des restaurants  
255 function searchReviews() {  
256     let searchValue = reviews.value;  
257     if (searchValue != '') {  
258         let filteredRestaurants = restaurants.filter(function (restaurant) {  
259             return restaurant.reviews == searchValue;  
260         });  
261         displayList(filteredRestaurants);  
262     }  
263 }  
264  
265 // Fonction qui permet de faire une recherche dans la liste des restaurants  
266 function searchRating() {  
267     let searchValue = rating.value;  
268     if (searchValue != '') {  
269         let filteredRestaurants = restaurants.filter(function (restaurant) {  
270             return restaurant.rating == searchValue;  
271         });  
272         displayList(filteredRestaurants);  
273     }  
274 }  
275  
276 // Fonction qui permet de faire une recherche dans la liste des restaurants  
277 function searchDistance() {  
278     let searchValue = distance.value;  
279     if (searchValue != '') {  
280         let filteredRestaurants = restaurants.filter(function (restaurant) {  
281             return restaurant.distance == searchValue;  
282         });  
283         displayList(filteredRestaurants);  
284     }  
285 }  
286  
287 // Fonction qui permet de faire une recherche dans la liste des restaurants  
288 function searchCuisines() {  
289     let searchValue = cuisines.value;  
290     if (searchValue != '') {  
291         let filteredRestaurants = restaurants.filter(function (restaurant) {  
292             return restaurant.cuisines == searchValue;  
293         });  
294         displayList(filteredRestaurants);  
295     }  
296 }  
297  
298 // Fonction qui permet de faire une recherche dans la liste des restaurants  
299 function searchReviews() {  
300     let searchValue = reviews.value;  
301     if (searchValue != '') {  
302         let filteredRestaurants = restaurants.filter(function (restaurant) {  
303             return restaurant.reviews == searchValue;  
304         });  
305         displayList(filteredRestaurants);  
306     }  
307 }  
308  
309 // Fonction qui permet de faire une recherche dans la liste des restaurants  
310 function searchRating() {  
311     let searchValue = rating.value;  
312     if (searchValue != '') {  
313         let filteredRestaurants = restaurants.filter(function (restaurant) {  
314             return restaurant.rating == searchValue;  
315         });  
316         displayList(filteredRestaurants);  
317     }  
318 }  
319  
320 // Fonction qui permet de faire une recherche dans la liste des restaurants  
321 function searchDistance() {  
322     let searchValue = distance.value;  
323     if (searchValue != '') {  
324         let filteredRestaurants = restaurants.filter(function (restaurant) {  
325             return restaurant.distance == searchValue;  
326         });  
327         displayList(filteredRestaurants);  
328     }  
329 }  
330  
331 // Fonction qui permet de faire une recherche dans la liste des restaurants  
332 function searchCuisines() {  
333     let searchValue = cuisines.value;  
334     if (searchValue != '') {  
335         let filteredRestaurants = restaurants.filter(function (restaurant) {  
336             return restaurant.cuisines == searchValue;  
337         });  
338         displayList(filteredRestaurants);  
339     }  
340 }  
341  
342 // Fonction qui permet de faire une recherche dans la liste des restaurants  
343 function searchReviews() {  
344     let searchValue = reviews.value;  
345     if (searchValue != '') {  
346         let filteredRestaurants = restaurants.filter(function (restaurant) {  
347             return restaurant.reviews == searchValue;  
348         });  
349         displayList(filteredRestaurants);  
350     }  
351 }  
352  
353 // Fonction qui permet de faire une recherche dans la liste des restaurants  
354 function searchRating() {  
355     let searchValue = rating.value;  
356     if (searchValue != '') {  
357         let filteredRestaurants = restaurants.filter(function (restaurant) {  
358             return restaurant.rating == searchValue;  
359         });  
360         displayList(filteredRestaurants);  
361     }  
362 }  
363  
364 // Fonction qui permet de faire une recherche dans la liste des restaurants  
365 function searchDistance() {  
366     let searchValue = distance.value;  
367     if (searchValue != '') {  
368         let filteredRestaurants = restaurants.filter(function (restaurant) {  
369             return restaurant.distance == searchValue;  
370         });  
371         displayList(filteredRestaurants);  
372     }  
373 }  
374  
375 // Fonction qui permet de faire une recherche dans la liste des restaurants  
376 function searchCuisines() {  
377     let searchValue = cuisines.value;  
378     if (searchValue != '') {  
379         let filteredRestaurants = restaurants.filter(function (restaurant) {  
380             return restaurant.cuisines == searchValue;  
381         });  
382         displayList(filteredRestaurants);  
383     }  
384 }  
385  
386 // Fonction qui permet de faire une recherche dans la liste des restaurants  
387 function searchReviews() {  
388     let searchValue = reviews.value;  
389     if (searchValue != '') {  
390         let filteredRestaurants = restaurants.filter(function (restaurant) {  
391             return restaurant.reviews == searchValue;  
392         });  
393         displayList(filteredRestaurants);  
394     }  
395 }  
396  
397 // Fonction qui permet de faire une recherche dans la liste des restaurants  
398 function searchRating() {  
399     let searchValue = rating.value;  
400     if (searchValue != '') {  
401         let filteredRestaurants = restaurants.filter(function (restaurant) {  
402             return restaurant.rating == searchValue;  
403         });  
404         displayList(filteredRestaurants);  
405     }  
406 }  
407  
408 // Fonction qui permet de faire une recherche dans la liste des restaurants  
409 function searchDistance() {  
410     let searchValue = distance.value;  
411     if (searchValue != '') {  
412         let filteredRestaurants = restaurants.filter(function (restaurant) {  
413             return restaurant.distance == searchValue;  
414         });  
415         displayList(filteredRestaurants);  
416     }  
417 }  
418  
419 // Fonction qui permet de faire une recherche dans la liste des restaurants  
420 function searchCuisines() {  
421     let searchValue = cuisines.value;  
422     if (searchValue != '') {  
423         let filteredRestaurants = restaurants.filter(function (restaurant) {  
424             return restaurant.cuisines == searchValue;  
425         });  
426         displayList(filteredRestaurants);  
427     }  
428 }  
429  
430 // Fonction qui permet de faire une recherche dans la liste des restaurants  
431 function searchReviews() {  
432     let searchValue = reviews.value;  
433     if (searchValue != '') {  
434         let filteredRestaurants = restaurants.filter(function (restaurant) {  
435             return restaurant.reviews == searchValue;  
436         });  
437         displayList(filteredRestaurants);  
438     }  
439 }  
440  
441 // Fonction qui permet de faire une recherche dans la liste des restaurants  
442 function searchRating() {  
443     let searchValue = rating.value;  
444     if (searchValue != '') {  
445         let filteredRestaurants = restaurants.filter(function (restaurant) {  
446             return restaurant.rating == searchValue;  
447         });  
448         displayList(filteredRestaurants);  
449     }  
450 }  
451  
452 // Fonction qui permet de faire une recherche dans la liste des restaurants  
453 function searchDistance() {  
454     let searchValue = distance.value;  
455     if (searchValue != '') {  
456         let filteredRestaurants = restaurants.filter(function (restaurant) {  
457             return restaurant.distance == searchValue;  
458         });  
459         displayList(filteredRestaurants);  
460     }  
461 }  
462  
463 // Fonction qui permet de faire une recherche dans la liste des restaurants  
464 function searchCuisines() {  
465     let searchValue = cuisines.value;  
466     if (searchValue != '') {  
467         let filteredRestaurants = restaurants.filter(function (restaurant) {  
468             return restaurant.cuisines == searchValue;  
469         });  
470         displayList(filteredRestaurants);  
471     }  
472 }  
473  
474 // Fonction qui permet de faire une recherche dans la liste des restaurants  
475 function searchReviews() {  
476     let searchValue = reviews.value;  
477     if (searchValue != '') {  
478         let filteredRestaurants = restaurants.filter(function (restaurant) {  
479             return restaurant.reviews == searchValue;  
480         });  
481         displayList(filteredRestaurants);  
482     }  
483 }  
484  
485 // Fonction qui permet de faire une recherche dans la liste des restaurants  
486 function searchRating() {  
487     let searchValue = rating.value;  
488     if (searchValue != '') {  
489         let filteredRestaurants = restaurants.filter(function (restaurant) {  
490             return restaurant.rating == searchValue;  
491         });  
492         displayList(filteredRestaurants);  
493     }  
494 }  
495  
496 // Fonction qui permet de faire une recherche dans la liste des restaurants  
497 function searchDistance() {  
498     let searchValue = distance.value;  
499     if (searchValue != '') {  
500         let filteredRestaurants = restaurants.filter(function (restaurant) {  
501             return restaurant.distance == searchValue;  
502         });  
503         displayList(filteredRestaurants);  
504     }  
505 }  
506  
507 // Fonction qui permet de faire une recherche dans la liste des restaurants  
508 function searchCuisines() {  
509     let searchValue = cuisines.value;  
510     if (searchValue != '') {  
511         let filteredRestaurants = restaurants.filter(function (restaurant) {  
512             return restaurant.cuisines == searchValue;  
513         });  
514         displayList(filteredRestaurants);  
515     }  
516 }  
517  
518 // Fonction qui permet de faire une recherche dans la liste des restaurants  
519 function searchReviews() {  
520     let searchValue = reviews.value;  
521     if (searchValue != '') {  
522         let filteredRestaurants = restaurants.filter(function (restaurant) {  
523             return restaurant.reviews == searchValue;  
524         });  
525         displayList(filteredRestaurants);  
526     }  
527 }  
528  
529 // Fonction qui permet de faire une recherche dans la liste des restaurants  
530 function searchRating() {  
531     let searchValue = rating.value;  
532     if (searchValue != '') {  
533         let filteredRestaurants = restaurants.filter(function (restaurant) {  
534             return restaurant.rating == searchValue;  
535         });  
536         displayList(filteredRestaurants);  
537     }  
538 }  
539  
540 // Fonction qui permet de faire une recherche dans la liste des restaurants  
541 function searchDistance() {  
542     let searchValue = distance.value;  
543     if (searchValue != '') {  
544         let filteredRestaurants = restaurants.filter(function (restaurant) {  
545             return restaurant.distance == searchValue;  
546         });  
547         displayList(filteredRestaurants);  
548     }  
549 }  
550  
551 // Fonction qui permet de faire une recherche dans la liste des restaurants  
552 function searchCuisines() {  
553     let searchValue = cuisines.value;  
554     if (searchValue != '') {  
555         let filteredRestaurants = restaurants.filter(function (restaurant) {  
556             return restaurant.cuisines == searchValue;  
557         });  
558         displayList(filteredRestaurants);  
559     }  
560 }  
561  
562 // Fonction qui permet de faire une recherche dans la liste des restaurants  
563 function searchReviews() {  
564     let searchValue = reviews.value;  
565     if (searchValue != '') {  
566         let filteredRestaurants = restaurants.filter(function (restaurant) {  
567             return restaurant.reviews == searchValue;  
568         });  
569         displayList(filteredRestaurants);  
570     }  
571 }  
572  
573 // Fonction qui permet de faire une recherche dans la liste des restaurants  
574 function searchRating() {  
575     let searchValue = rating.value;  
576     if (searchValue != '') {  
577         let filteredRestaurants = restaurants.filter(function (restaurant) {  
578             return restaurant.rating == searchValue;  
579         });  
580         displayList(filteredRestaurants);  
581     }  
582 }  
583  
584 // Fonction qui permet de faire une recherche dans la liste des restaurants  
585 function searchDistance() {  
586     let searchValue = distance.value;  
587     if (searchValue != '') {  
588         let filteredRestaurants = restaurants.filter(function (restaurant) {  
589             return restaurant.distance == searchValue;  
590         });  
591         displayList(filteredRestaurants);  
592     }  
593 }  
594  
595 // Fonction qui permet de faire une recherche dans la liste des restaurants  
596 function searchCuisines() {  
597     let searchValue = cuisines.value;  
598     if (searchValue != '') {  
599         let filteredRestaurants = restaurants.filter(function (restaurant) {  
600             return restaurant.cuisines == searchValue;  
601         });  
602         displayList(filteredRestaurants);  
603     }  
604 }  
605  
606 // Fonction qui permet de faire une recherche dans la liste des restaurants  
607 function searchReviews() {  
608     let searchValue = reviews.value;  
609     if (searchValue != '') {  
610         let filteredRestaurants = restaurants.filter(function (restaurant) {  
611             return restaurant.reviews == searchValue;  
612         });  
613         displayList(filteredRestaurants);  
614     }  
615 }  
616  
617 // Fonction qui permet de faire une recherche dans la liste des restaurants  
618 function searchRating() {  
619     let searchValue = rating.value;  
620     if (searchValue != '') {  
621         let filteredRestaurants = restaurants.filter(function (restaurant) {  
622             return restaurant.rating == searchValue;  
623         });  
624         displayList(filteredRestaurants);  
625     }  
626 }  
627  
628 // Fonction qui permet de faire une recherche dans la liste des restaurants  
629 function searchDistance() {  
630     let searchValue = distance.value;  
631     if (searchValue != '') {  
632         let filteredRestaurants = restaurants.filter(function (restaurant) {  
633             return restaurant.distance == searchValue;  
634         });  
635         displayList(filteredRestaurants);  
636     }  
637 }  
638  
639 // Fonction qui permet de faire une recherche dans la liste des restaurants  
640 function searchCuisines() {  
641     let searchValue = cuisines.value;  
642     if (searchValue != '') {  
643         let filteredRestaurants = restaurants.filter(function (restaurant) {  
644             return restaurant.cuisines == searchValue;  
645         });  
646         displayList(filteredRestaurants);  
647     }  
648 }  
649  
650 // Fonction qui permet de faire une recherche dans la liste des restaurants  
651 function searchReviews() {  
652     let searchValue = reviews.value;  
653     if (searchValue != '') {  
654         let filteredRestaurants = restaurants.filter(function (restaurant) {  
655             return restaurant.reviews == searchValue;  
656         });  
657         displayList(filteredRestaurants);  
658     }  
659 }  
660  
661 // Fonction qui permet de faire une recherche dans la liste des restaurants  
662 function searchRating() {  
663     let searchValue = rating.value;  
664     if (searchValue != '') {  
665         let filteredRestaurants = restaurants.filter(function (restaurant) {  
666             return restaurant.rating == searchValue;  
667         });  
668         displayList(filteredRestaurants);  
669     }  
670 }  
671  
672 // Fonction qui permet de faire une recherche dans la liste des restaurants  
673 function searchDistance() {  
674     let searchValue = distance.value;  
675     if (searchValue != '') {  
676         let filteredRestaurants = restaurants.filter(function (restaurant) {  
677             return restaurant.distance == searchValue;  
678         });  
679         displayList(filteredRestaurants);  
680     }  
681 }  
682  
683 // Fonction qui permet de faire une recherche dans la liste des restaurants  
684 function searchCuisines() {  
685     let searchValue = cuisines.value;  
686     if (searchValue != '') {  
687         let filteredRestaurants = restaurants.filter(function (restaurant) {  
688             return restaurant.cuisines == searchValue;  
689         });  
690         displayList(filteredRestaurants);  
691     }  
692 }  
693  
694 // Fonction qui permet de faire une recherche dans la liste des restaurants  
695 function searchReviews() {  
696     let searchValue = reviews.value;  
697     if (searchValue != '') {  
698         let filteredRestaurants = restaurants.filter(function (restaurant) {  
699             return restaurant.reviews == searchValue;  
700         });  
701         displayList(filteredRestaurants);  
702     }  
703 }  
704  
705 // Fonction qui permet de faire une recherche dans la liste des restaurants  
706 function searchRating() {  
707     let searchValue = rating.value;  
708     if (searchValue != '') {  
709         let filteredRestaurants = restaurants.filter(function (restaurant) {  
710             return restaurant.rating == searchValue;  
711         });  
712         displayList(filteredRestaurants);  
713     }  
714 }  
715  
716 // Fonction qui permet de faire une recherche dans la liste des restaurants  
717 function searchDistance() {  
718     let searchValue = distance.value;  
719     if (searchValue != '') {  
720         let filteredRestaurants = restaurants.filter(function (restaurant) {  
721             return restaurant.distance == searchValue;  
722         });  
723         displayList(filteredRestaurants);  
724     }  
725 }  
726  
727 // Fonction qui permet de faire une recherche dans la liste des restaurants  
728 function searchCuisines() {  
729     let searchValue = cuisines.value;  
730     if (searchValue != '') {  
731         let filteredRestaurants = restaurants.filter(function (restaurant) {  
732             return restaurant.cuisines == searchValue;  
733         });  
734         displayList(filteredRestaurants);  
735     }  
736 }  
737  
738 // Fonction qui permet de faire une recherche dans la liste des restaurants  
739 function searchReviews() {  
740     let searchValue = reviews.value;  
741     if (searchValue != '') {  
742         let filteredRestaurants = restaurants.filter(function (restaurant) {  
743             return restaurant.reviews == searchValue;  
744         });  
745         displayList(filteredRestaurants);  
746     }  
747 }  
748  
749 // Fonction qui permet de faire une recherche dans la liste des restaurants  
750 function searchRating() {  
751     let searchValue = rating.value;  
752     if (searchValue != '') {  
753         let filteredRestaurants = restaurants.filter(function (restaurant) {  
754             return restaurant.rating == searchValue;  
755         });  
756         displayList(filteredRestaurants);  
757     }  
758 }  
759  
760 // Fonction qui permet de faire une recherche dans la liste des restaurants  
761 function searchDistance() {  
762     let searchValue = distance.value;  
763     if (searchValue != '') {  
764         let filteredRestaurants = restaurants.filter(function (restaurant) {  
765             return restaurant.distance == searchValue;  
766         });  
767         displayList(filteredRestaurants);  
768     }  
769 }  
770  
771 // Fonction qui permet de faire une recherche dans la liste des restaurants  
772 function searchCuisines() {  
773     let searchValue = cuisines.value;  
774     if (searchValue != '') {  
775         let filteredRestaurants = restaurants.filter(function (restaurant) {  
776             return restaurant.cuisines == searchValue;  
777         });  
778         displayList(filteredRestaurants);  
779     }  
780 }  
781  
782 // Fonction qui permet de faire une recherche dans la liste des restaurants  
783 function searchReviews() {  
784     let searchValue = reviews.value;  
785     if (searchValue != '') {  
786         let filteredRestaurants = restaurants.filter(function (restaurant) {  
787             return restaurant.reviews == searchValue;  
788         });  
789         displayList(filteredRestaurants);  
790     }  
791 }  
792  
793 // Fonction qui permet de faire une recherche dans la liste des restaurants  
794 function searchRating() {  
795     let searchValue = rating.value;  
796     if (searchValue != '') {  
797         let filteredRestaurants = restaurants.filter(function (restaurant) {  
798             return restaurant.rating == searchValue;  
799         });  
800         displayList(filteredRestaurants);  
801     }  
802 }  
803  
804 // Fonction qui permet de faire une recherche dans la liste des restaurants  
805 function searchDistance() {  
806     let searchValue = distance.value;  
807     if (searchValue != '') {  
808         let filteredRestaurants = restaurants.filter(function (restaurant) {  
809             return restaurant.distance == searchValue;  
810         });  
811         displayList(filteredRestaurants);  
812     }  
813 }  
814  
815 // Fonction qui permet de faire une recherche dans la liste des restaurants  
816 function searchCuisines() {  
817     let searchValue = cuisines.value;  
818     if (searchValue != '') {  
819         let filteredRestaurants = restaurants.filter(function (restaurant) {  
820             return restaurant.cuisines == searchValue;  
821         });  
822         displayList(filteredRestaurants);  
823     }  
824 }  
825  
826 // Fonction qui permet de faire une recherche dans la liste des restaurants  
827 function searchReviews() {  
828     let searchValue = reviews.value;  
829     if (searchValue != '') {  
830         let filteredRestaurants = restaurants.filter(function (restaurant) {  
831             return restaurant.reviews == searchValue;  
832         });  
833         displayList(filteredRestaurants);  
834     }  
835 }  
836  
837 // Fonction qui permet de faire une recherche dans la liste des restaurants  
838 function searchRating() {  
839     let searchValue = rating.value;  
840     if (searchValue != '') {  
841         let filteredRestaurants = restaurants.filter(function (restaurant) {  
842             return restaurant.rating == searchValue;  
843         });  
844         displayList(filteredRestaurants);  
845     }  
846 }  
847  
848 // Fonction qui permet de faire une recherche dans la liste des restaurants  
849 function searchDistance() {  
850     let searchValue = distance.value;  
851     if (searchValue != '') {  
852         let filteredRestaurants = restaurants.filter(function (restaurant) {  
853             return restaurant.distance == searchValue;  
854         });  
855         displayList(filteredRestaurants);  
856     }  
857 }  
858  
859 // Fonction qui permet de faire une recherche dans la liste des restaurants  
860 function searchCuisines() {  
861     let searchValue = cuisines.value;  
862     if (searchValue != '') {  
863         let filteredRestaurants = restaurants.filter(function (restaurant) {  
864             return restaurant.cuisines == searchValue;  
865         });  
866         displayList(filteredRestaurants);  
867     }  
868 }  
869  
870 // Fonction qui permet de faire une recherche dans la liste des restaurants  
871 function searchReviews() {  
872     let searchValue = reviews.value;  
873     if (searchValue != '') {  
874         let filteredRestaurants = restaurants.filter(function (restaurant) {  
875             return restaurant.reviews == searchValue;  
876         });  
877         displayList(filteredRestaurants);  
878     }  
879 }  
880  
881 // Fonction qui permet de faire une recherche dans la liste des restaurants  
882 function searchRating() {  
883     let searchValue = rating.value;  
884     if (searchValue != '') {  
885         let filteredRestaurants = restaurants.filter(function (restaurant) {  
886             return restaurant.rating == searchValue;  
887         });  
888         displayList(filteredRestaurants);  
889     }  
890 }  
891  
892 // Fonction qui permet de faire une recherche dans la liste des restaurants  
893 function searchDistance() {  
894     let searchValue = distance.value;  
895     if (searchValue != '') {  
896         let filteredRestaurants = restaurants.filter(function (restaurant) {  
897             return restaurant.distance == searchValue;  
898         });  
899         displayList(filteredRestaurants);  
900     }  
901 }  
902  
903 // Fonction qui permet de faire une recherche dans la liste des restaurants  
904 function searchCuisines() {  
905     let searchValue = cuisines.value;  
906     if (searchValue != '') {  
907         let filteredRestaurants = restaurants.filter(function (restaurant) {  
908             return restaurant.cuisines == searchValue;  
909         });  
910         displayList(filteredRestaurants);  
911     }  
912 }  
913  
914 // Fonction qui permet de faire une recherche dans la liste des restaurants  
915 function searchReviews() {  
916     let searchValue = reviews.value;  
917     if (searchValue != '') {  
918         let filteredRestaurants = restaurants.filter(function (restaurant) {  
919             return restaurant.reviews == searchValue;  
920         });  
921         displayList(filteredRestaurants);  
922     }  
923 }  
924  
925 // Fonction qui permet de faire une recherche dans la liste des restaurants  
926 function searchRating() {  
927     let searchValue = rating.value;  
928     if (searchValue != '') {  
929         let filteredRestaurants = restaurants.filter(function (restaurant) {  
930             return restaurant.rating == searchValue;  
931         });  
932         displayList(filteredRestaurants);  
933     }  
934 }  
935  
936 // Fonction qui permet de faire une recherche dans la liste des restaurants  
937 function searchDistance() {  
938     let searchValue = distance.value;  
939     if (searchValue != '') {  
940         let filteredRestaurants = restaurants.filter(function (restaurant) {  
941             return restaurant.distance == searchValue;  
942         });  
943         displayList(filteredRestaurants);  
944     }  
945 }  
946  
947 // Fonction qui permet de faire une recherche dans la liste des restaurants  
948 function searchCuisines() {  
949     let searchValue = cuisines.value;  
950     if (searchValue != '') {  
951         let filteredRestaurants = restaurants.filter(function (restaurant) {  
952             return restaurant.cuisines == searchValue;  
953         });  
954         displayList(filteredRestaurants);  
955     }  
956 }  
957  
958 // Fonction qui permet de faire une recherche dans la liste des restaurants  
959 function searchReviews() {  
960     let searchValue = reviews.value;  
961     if (searchValue != '') {  
962         let filteredRestaurants = restaurants.filter(function (restaurant) {  
963             return restaurant.reviews == searchValue;  
964         });  
965         displayList(filteredRestaurants);  
966     }  
967 }  
968  
969 // Fonction qui permet de faire une recherche dans la liste des restaurants  
970 function searchRating() {  
971     let searchValue = rating.value;  
972     if (searchValue != '') {  
973         let filteredRestaurants = restaurants.filter(function (restaurant) {  
974             return restaurant.rating == searchValue;  
975         });  
976         displayList(filteredRestaurants);  
977     }  
978 }  
979  
980 // Fonction qui permet de faire une recherche dans la liste des restaurants  
981 function searchDistance() {  
982     let searchValue = distance.value;  
983     if (searchValue != '') {  
984         let filteredRestaurants = restaurants.filter(function (restaurant) {  
985             return restaurant.distance == searchValue;  
986         });  
987         displayList(filteredRestaurants);  
988     }  

```

```
105         // Méthode similaire à un pattern Singleton si c'est la première fois qu'on ajoute une carte interactive
106         if (map == null) {
107             map = L.map('map').setView([address.coordinates.latitude,
108             address.coordinates.longitude], 15); // Map centrée sur l'adresse de l'utilisateur
109             L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}
110             /{y}?access_token=pk.eyJ1IjoibWl5b2tpIiwiYSI6ImNsMW96ejJ5NTAzMjQza3B0NHB3bHMxYncifQ.03A1BHWNd{
111                 maxZoom: 18,
112                 id: 'mapbox/streets-v11',
113                 tileSize: 512,
114                 zoomOffset: -1,
115             }).addTo(map);
116         } else {
117             // Pour supprimer les marqueurs
118             map.removeLayer(markers);
119         }
120         // Pour supprimer les restaurants de la liste
121         document.getElementById("idListDisplay").innerHTML = "";
122         addressDisplay.innerText = address.street != null ? address.street : "";
123         successCallBack();
124     });
125 });
126 }
127
128 // Méthode pour ajouter les marqueurs à la carte interactive et ajouter les restaurants dans : dynamique
129 function displayMapList() {
130     // Placer les marqueurs et remplir la liste de restaurants affichée.
131     markers = L.featureGroup().addTo(map); // Groupe de marqueurs
132     // On ajoute un marqueur à l'adresse de l'utilisateur
133     L.marker([address.coordinates.latitude, address.coordinates.longitude], { icon: iconHome })
134     .addTo(markers);
135     for (let j = 0; j < restaurants.length; j++) {
136         // Remplir la liste
137         addToListDisplay(restaurants[j]);
138
139         // Créer le marqueur et l'ajouter au groupe de marqueurs
140         L.marker([restaurants[j].latitude, restaurants[j].longitude], { icon: iconRestaurant })
141         .addTo(markers).on('click', function(e) {
142             let list = document.getElementById("idListDisplay");
143             let card = document.getElementById(`restaurant${restaurants[j].id}`);
144
145             list.scrollTop = card.offsetTop;
146         });
147     }
148
149     map.flyToBounds(markers.getBounds().pad(0.1), { duration: 1 }); // On recentre la carte pour le marqueur le plus éloigné visible
150 }
151
152 // Fonction pour ajouter dans la liste dynamique les restaurants
153 function addToListDisplay(restaurant) {
154     let card = document.createElement('div');
```

```
153 card.setAttribute('id', `restaurant${restaurant.id}`);
154 card.classList.add("card");
155 card.onclick = (event) => {
156     const isButton = event.target.nodeName === 'BUTTON';
157     if (isButton) {
158         location.href = `menu.html?idRestaurant=${restaurant.id}`;
159     } else {
160         flyToMarker(restaurant);
161     }
162 };
163 card.innerHTML = `
164 <div class="card-body">
165     <p>${restaurant.name}</p>
166     <button >Commander</button>
167 </div>`;
168
169 document.getElementById("idListDisplay").appendChild(card);
170 }
171
172 // Fonction pour se déplacer vers un point sur la carte
173 function flyToMarker(location) {
174     map.flyTo([location.latitude, location.longitude], 18, {
175         animate: true,
176         duration: 2 // en seconde
177     });
178 }
179
180 // Fonction appelée quand on change la catégorie dans le select
181 function categoryChanged() {
182     var select = document.getElementById("idSelectCategory");
183     search.categories = [];
184
185     if (select.value != "") {
186         search.categories.push(select.value);
187     }
188 }
189
190 // Fonction appelée quand on change le rayon
191 function radiusChanged() {
192     var input = document.getElementById("idInputRadius");
193
194     search.radius = input.value;
195 }
196
197 // Fonction appelée quand un des champs de filtre à changer pour appeler le serveur
198 function changed() {
199     sendData(displayMapList, "http://localhost/grin-eat-tpi2022/grineat-api/restaurants", sear
200 }
```

## c:\grin-eat-tpi2022\server-node-old\src\functions.js

```
1 /*  
2  /\_/_/  
3  /__\_\_/\_ Nom et Prénom: GRIN Brian  
4  \_\_/_/_/ /___ Enseignant : Monsieur Antoine Schmid  
5  /* * \_ /____ Classe : I.DA-P4B  
6  \_\_0/_/_ [ ] Date : 18.05.2022  
7  / \_/_ [ / Nom du projet : GrinEat-Server-Node  
8  \_\_ \_/_ / / Version du projet : 0.1  
9  [ [ / \_/_ Cours : TPI  
10 _[ [ \_/_/  
11 */  
12  
13  
14 const axios = require('axios').default;  
15  
16 async function convertAdressToCoordoninate(adress) {  
17     return new Promise((resolve, reject) => {  
18         encoded = encodeURIComponent(adress);  
19  
20         axios.get(`https://api.mapbox.com/geocoding/v5/mapbox.places  
/${encoded}.json?access_token=${process.env.MAPBOX_TOKEN}`)  
21             .then(function (response) {  
22                 let adressFound = response.data.features[0].place_name;  
23                 let latitudeAdrr = response.data.features[0].geometry.coordinates[1];  
24                 let longitudeAdrr = response.data.features[0].geometry.coordinates[0];  
25  
26                 resolve({ adress: adressFound, latitude: latitudeAdrr, longitude:  
longitudeAdrr });  
27             })  
28             .catch(function (error) {  
29                 console.table(error);  
30                 reject("Invalid adress");  
31             });  
32     });  
33 }  
34  
35 function haversineGreatCircleDistance(  
36 , latitudeFrom, longitudeFrom, latitudeTo, longitudeTo, earthRadius = 6371) {  
37 // convert from degress to radians  
38 let latFrom = deg2rad(latitudeFrom);  
39 let lonFrom = deg2rad(longitudeFrom);  
40 let latTo = deg2rad(latitudeTo);  
41 let lonTo = deg2rad(longitudeTo);  
42  
43 let latDelta = latTo - latFrom;  
44 let lonDelta = lonTo - lonFrom;  
45  
46 let angle = 2 * Math.asin(Math.sqrt(Math.pow(Math.sin(latDelta / 2), 2) +  
Math.cos(latFrom) * Math.cos(latTo) * Math.pow(Math.sin(lonDelta / 2), 2)));  
47  
48 return angle * earthRadius;  
49 }  
50 }
```

```
51
52 function deg2rad(degrees) {
53     var pi = Math.PI;
54     return degrees * pi / 180;
55 }
56
57 module.exports = { convertAddressToCoordinate, haversineGreatCircleDistance };
```

## c:\grin-eat-tpi2022\server-node-old\src\index.js

```
1 /*
2    /\   /
3   //\\_//\\  /____/ Nom et Prénom: GRIN Brian
4   \_\_/_/  /____/ Enseignant : Monsieur Antoine Schmid
5   /* * \  /^^^] Classe : I.DA-P4B
6   \_\0/_/ [ ] Date : 18.05.2022
7   / \_ [ / Nom du projet : GrinEat-Server-Node
8   \ \_ / / Version du projet : 0.1
9   [ [ / \_ / Cours : TPI
10  _[ [ \ / /
11
12 */
13
14 const express = require('express');
15 const categories = require('./controllers/categoriesController.js');
16 const restaurants = require('./controllers/restaurantsController.js');
17 const menu_items = require('./controllers/menu_itemsController.js');
18
19 const app = express()
20 const port = process.env.PORT;
21
22 app.use(function (req, res, next) {
23     res.setHeader('Access-Control-Allow-Origin', '*')
24     res.setHeader('Access-Control-Allow-Methods', 'GET, PUT, POST, DELETE')
25     res.setHeader('Access-Control-Allow-Headers', 'Content-Type')
26     res.setHeader('Content-Type', 'application/json')
27     next();
28 });
29 app.use(express.json());
30
31 app.get('/', (req, res) => { res.status(200).json("Welcome to the API") });
32 app.use('/categories', categories);
33 app.use('/restaurants', restaurants);
34 app.use('/menu-items', menu_items);
35
36 app.listen(port, () => {
37     console.log("App listening at http://localhost:" + port)
38 });
```

**c:\grin-eat-tpi2022\server-node-old\src\controllers  
\categoriesController.js**

```
1 /*  
2  /\_/\_/  
3  //\\_/_/\\_/_ Nom et Prénom: GRIN Brian  
4  \_\_/_/_/ /____ Enseignant : Monsieur Antoine Schmid  
5  / * * \_ /[^^\_][^^\_][^^\_][^^\_] Classe : I.DA-P4B  
6  \_\_0/_/_ [ ] Date : 18.05.2022  
7  / \_/_ [ / ] Nom du projet : GrinEat-Server-Node  
8  \_\_/_/_ / / Version du projet : 0.1  
9  [ [ / \_/_/ Version du projet : 0.1  
10 _[ [ \_/_/ Cours : TPI  
11  
12 */  
13  
14 const express = require('express');  
15 const router = express.Router();  
16 const Category = require('../models/Category.js');  
17  
18 router.get('/', (req, res) => {  
19   Category.getCategories().then(categories => {  
20     res.status(200).json(categories);  
21   })  
22 });  
23  
24 module.exports = router;
```

**c:\grin-eat-tpi2022\server-node-old\src\controllers  
\menu\_itemsController.js**

```
1 /*  
2      /\   /\  
3      //\\_/_/\\_  /  _/    Nom et Prénom: GRIN Brian  
4      \_  _/  /  /  /  Enseignant : Monsieur Antoine Schmid  
5      / * * \  /  ^^^]  Classe : I.DA-P4B  
6      \_ \0/_/  [  ]  Date : 18.05.2022  
7      /  \_  [  /  Nom du projet : GrinEat-Server-Node  
8      \_  \_  /  /  Version du projet : 0.1  
9      [  [ /  \_ /  Cours : TPI  
10     _[ [ \  / /  
11  
12 */  
13  
14 const express = require('express');  
15 const router = express.Router();  
16 const MenuItems = require('../models/MenuItems.js');  
17  
18 router.get('/:id', (req, res) => {  
19     let myMenuItems = new MenuItems();  
20     let idRestaurant = req.params.id;  
21  
22     myMenuItems.setRestaurantId(idRestaurant);  
23  
24     myMenuItems.findById().then(result => {  
25         res.status.json(result);  
26     });  
27});  
28  
29 module.exports = router;
```

## c:\grin-eat-tpi2022\server-node-old\src\controllers \restaurantsController.js

```
1 /*  
2  /\_/\_/  
3  //\\_/_/\\_/_ Nom et Prénom: GRIN Brian  
4  \_\_/_/_/ / \_ Enseignant : Monsieur Antoine Schmid  
5  / * * \_ / ^^^ Classe : I.DA-P4B  
6  \_\_0/_/_ [ ] Date : 18.05.2022  
7  / \_/_ [ / Nom du projet : GrinEat-Server-Node  
8  \_\_ \_/_ / / Version du projet : 0.1  
9  [ [ / \_/_ Cours : TPI  
10 _[ [ \_/_/  
11 */  
12  
13  
14 const { response } = require('express');  
15 const express = require('express');  
16 const router = express.Router();  
17 const { convertAdressToCoordoninate, haversineGreatCircleDistance } =  
require('../functions.js');  
18 const Restaurant = require('../models/Restaurant.js');  
19  
20 router.post('/', (req, res) => {  
21     let adress = req.body.adress; // adresse du client.  
22     let radius = req.body.radius || 5; // rayon limite autour de l'adresse du client par  
défaut 5 si rien spécifié.  
23     let name = req.body.name || null; // filtre par nom si le client souhaite faire une  
recherche.  
24     let categories = req.body.categories || []; // filtre par catégorie si le client en  
sélectionne.  
25  
26     let restaurants = [];  
27     let message = [];  
28  
29     // On vérifie que le client nous a bien renseigné une adresse  
30     if (adress != null) {  
31         convertAdressToCoordoninate(adress).then(response => {  
32             let latFrom = response.latitude;  
33             let lonFrom = response.longitude;  
34  
35             Restaurant.getRestaurants().then(restaurants => {  
36                 // On crée une promesse pour récupérer les catégories de tout les restaurants  
37                 new Promise((resolve) => {  
38                     for (let i = 0; i < restaurants.length; i++) {  
39                         restaurants[i].categories = [];  
40                         let restaurant = new Restaurant();  
41  
42                         restaurant.setIdRestaurant(restaurants[i].id);  
43  
44                         restaurant.getRestaurantCategoriesFR().then(categories => {  
45                             for (let j = 0; j < categories.length; j++) {  
46                                 restaurants[i].categories.push(categories[j].nameFrench ||  
null);  
47                         }  
48                 })  
49             })  
50         })  
51     })  
52 })  
53 })  
54 })  
55 })  
56 })  
57 })  
58 })  
59 })  
60 })  
61 })  
62 })  
63 })  
64 })  
65 })  
66 })  
67 })  
68 })  
69 })  
70 })  
71 })  
72 })  
73 })  
74 })  
75 })  
76 })  
77 })  
78 })  
79 })  
80 })  
81 })  
82 })  
83 })  
84 })  
85 })  
86 })  
87 })  
88 })  
89 })  
90 })  
91 })  
92 })  
93 })  
94 })  
95 })  
96 })  
97 })  
98 })  
99 })  
100 })  
101 })  
102 })  
103 })  
104 })  
105 })  
106 })  
107 })  
108 })  
109 })  
110 })  
111 })  
112 })  
113 })  
114 })  
115 })  
116 })  
117 })  
118 })  
119 })  
120 })  
121 })  
122 })  
123 })  
124 })  
125 })  
126 })  
127 })  
128 })  
129 })  
130 })  
131 })  
132 })  
133 })  
134 })  
135 })  
136 })  
137 })  
138 })  
139 })  
140 })  
141 })  
142 })  
143 })  
144 })  
145 })  
146 })  
147 })  
148 })  
149 })  
150 })  
151 })  
152 })  
153 })  
154 })  
155 })  
156 })  
157 })  
158 })  
159 })  
160 })  
161 })  
162 })  
163 })  
164 })  
165 })  
166 })  
167 })  
168 })  
169 })  
170 })  
171 })  
172 })  
173 })  
174 })  
175 })  
176 })  
177 })  
178 })  
179 })  
180 })  
181 })  
182 })  
183 })  
184 })  
185 })  
186 })  
187 })  
188 })  
189 })  
190 })  
191 })  
192 })  
193 })  
194 })  
195 })  
196 })  
197 })  
198 })  
199 })  
200 })  
201 })  
202 })  
203 })  
204 })  
205 })  
206 })  
207 })  
208 })  
209 })  
210 })  
211 })  
212 })  
213 })  
214 })  
215 })  
216 })  
217 })  
218 })  
219 })  
220 })  
221 })  
222 })  
223 })  
224 })  
225 })  
226 })  
227 })  
228 })  
229 })  
230 })  
231 })  
232 })  
233 })  
234 })  
235 })  
236 })  
237 })  
238 })  
239 })  
240 })  
241 })  
242 })  
243 })  
244 })  
245 })  
246 })  
247 })  
248 })  
249 })  
250 })  
251 })  
252 })  
253 })  
254 })  
255 })  
256 })  
257 })  
258 })  
259 })  
260 })  
261 })  
262 })  
263 })  
264 })  
265 })  
266 })  
267 })  
268 })  
269 })  
270 })  
271 })  
272 })  
273 })  
274 })  
275 })  
276 })  
277 })  
278 })  
279 })  
280 })  
281 })  
282 })  
283 })  
284 })  
285 })  
286 })  
287 })  
288 })  
289 })  
290 })  
291 })  
292 })  
293 })  
294 })  
295 })  
296 })  
297 })  
298 })  
299 })  
300 })  
301 })  
302 })  
303 })  
304 })  
305 })  
306 })  
307 })  
308 })  
309 })  
310 })  
311 })  
312 })  
313 })  
314 })  
315 })  
316 })  
317 })  
318 })  
319 })  
320 })  
321 })  
322 })  
323 })  
324 })  
325 })  
326 })  
327 })  
328 })  
329 })  
330 })  
331 })  
332 })  
333 })  
334 })  
335 })  
336 })  
337 })  
338 })  
339 })  
340 })  
341 })  
342 })  
343 })  
344 })  
345 })  
346 })  
347 })  
348 })  
349 })  
350 })  
351 })  
352 })  
353 })  
354 })  
355 })  
356 })  
357 })  
358 })  
359 })  
360 })  
361 })  
362 })  
363 })  
364 })  
365 })  
366 })  
367 })  
368 })  
369 })  
370 })  
371 })  
372 })  
373 })  
374 })  
375 })  
376 })  
377 })  
378 })  
379 })  
380 })  
381 })  
382 })  
383 })  
384 })  
385 })  
386 })  
387 })  
388 })  
389 })  
390 })  
391 })  
392 })  
393 })  
394 })  
395 })  
396 })  
397 })  
398 })  
399 })  
400 })  
401 })  
402 })  
403 })  
404 })  
405 })  
406 })  
407 })  
408 })  
409 })  
410 })  
411 })  
412 })  
413 })  
414 })  
415 })  
416 })  
417 })  
418 })  
419 })  
420 })  
421 })  
422 })  
423 })  
424 })  
425 })  
426 })  
427 })  
428 })  
429 })  
430 })  
431 })  
432 })  
433 })  
434 })  
435 })  
436 })  
437 })  
438 })  
439 })  
440 })  
441 })  
442 })  
443 })  
444 })  
445 })  
446 })  
447 })  
448 })  
449 })  
450 })  
451 })  
452 })  
453 })  
454 })  
455 })  
456 })  
457 })  
458 })  
459 })  
460 })  
461 })  
462 })  
463 })  
464 })  
465 })  
466 })  
467 })  
468 })  
469 })  
470 })  
471 })  
472 })  
473 })  
474 })  
475 })  
476 })  
477 })  
478 })  
479 })  
480 })  
481 })  
482 })  
483 })  
484 })  
485 })  
486 })  
487 })  
488 })  
489 })  
490 })  
491 })  
492 })  
493 })  
494 })  
495 })  
496 })  
497 })  
498 })  
499 })  
500 })  
501 })  
502 })  
503 })  
504 })  
505 })  
506 })  
507 })  
508 })  
509 })  
510 })  
511 })  
512 })  
513 })  
514 })  
515 })  
516 })  
517 })  
518 })  
519 })  
520 })  
521 })  
522 })  
523 })  
524 })  
525 })  
526 })  
527 })  
528 })  
529 })  
530 })  
531 })  
532 })  
533 })  
534 })  
535 })  
536 })  
537 })  
538 })  
539 })  
540 })  
541 })  
542 })  
543 })  
544 })  
545 })  
546 })  
547 })  
548 })  
549 })  
550 })  
551 })  
552 })  
553 })  
554 })  
555 })  
556 })  
557 })  
558 })  
559 })  
560 })  
561 })  
562 })  
563 })  
564 })  
565 })  
566 })  
567 })  
568 })  
569 })  
570 })  
571 })  
572 })  
573 })  
574 })  
575 })  
576 })  
577 })  
578 })  
579 })  
580 })  
581 })  
582 })  
583 })  
584 })  
585 })  
586 })  
587 })  
588 })  
589 })  
590 })  
591 })  
592 })  
593 })  
594 })  
595 })  
596 })  
597 })  
598 })  
599 })  
600 })  
601 })  
602 })  
603 })  
604 })  
605 })  
606 })  
607 })  
608 })  
609 })  
610 })  
611 })  
612 })  
613 })  
614 })  
615 })  
616 })  
617 })  
618 })  
619 })  
620 })  
621 })  
622 })  
623 })  
624 })  
625 })  
626 })  
627 })  
628 })  
629 })  
630 })  
631 })  
632 })  
633 })  
634 })  
635 })  
636 })  
637 })  
638 })  
639 })  
640 })  
641 })  
642 })  
643 })  
644 })  
645 })  
646 })  
647 })  
648 })  
649 })  
650 })  
651 })  
652 })  
653 })  
654 })  
655 })  
656 })  
657 })  
658 })  
659 })  
660 })  
661 })  
662 })  
663 })  
664 })  
665 })  
666 })  
667 })  
668 })  
669 })  
670 })  
671 })  
672 })  
673 })  
674 })  
675 })  
676 })  
677 })  
678 })  
679 })  
680 })  
681 })  
682 })  
683 })  
684 })  
685 })  
686 })  
687 })  
688 })  
689 })  
690 })  
691 })  
692 })  
693 })  
694 })  
695 })  
696 })  
697 })  
698 })  
699 })  
700 })  
701 })  
702 })  
703 })  
704 })  
705 })  
706 })  
707 })  
708 })  
709 })  
710 })  
711 })  
712 })  
713 })  
714 })  
715 })  
716 })  
717 })  
718 })  
719 })  
720 })  
721 })  
722 })  
723 })  
724 })  
725 })  
726 })  
727 })  
728 })  
729 })  
730 })  
731 })  
732 })  
733 })  
734 })  
735 })  
736 })  
737 })  
738 })  
739 })  
740 })  
741 })  
742 })  
743 })  
744 })  
745 })  
746 })  
747 })  
748 })  
749 })  
750 })  
751 })  
752 })  
753 })  
754 })  
755 })  
756 })  
757 })  
758 })  
759 })  
760 })  
761 })  
762 })  
763 })  
764 })  
765 })  
766 })  
767 })  
768 })  
769 })  
770 })  
771 })  
772 })  
773 })  
774 })  
775 })  
776 })  
777 })  
778 })  
779 })  
780 })  
781 })  
782 })  
783 })  
784 })  
785 })  
786 })  
787 })  
788 })  
789 })  
790 })  
791 })  
792 })  
793 })  
794 })  
795 })  
796 })  
797 })  
798 })  
799 })  
800 })  
801 })  
802 })  
803 })  
804 })  
805 })  
806 })  
807 })  
808 })  
809 })  
810 })  
811 })  
812 })  
813 })  
814 })  
815 })  
816 })  
817 })  
818 })  
819 })  
820 })  
821 })  
822 })  
823 })  
824 })  
825 })  
826 })  
827 })  
828 })  
829 })  
830 })  
831 })  
832 })  
833 })  
834 })  
835 })  
836 })  
837 })  
838 })  
839 })  
840 })  
841 })  
842 })  
843 })  
844 })  
845 })  
846 })  
847 })  
848 })  
849 })  
850 })  
851 })  
852 })  
853 })  
854 })  
855 })  
856 })  
857 })  
858 })  
859 })  
860 })  
861 })  
862 })  
863 })  
864 })  
865 })  
866 })  
867 })  
868 })  
869 })  
870 })  
871 })  
872 })  
873 })  
874 })  
875 })  
876 })  
877 })  
878 })  
879 })  
880 })  
881 })  
882 })  
883 })  
884 })  
885 })  
886 })  
887 })  
888 })  
889 })  
890 })  
891 })  
892 })  
893 })  
894 })  
895 })  
896 })  
897 })  
898 })  
899 })  
900 })  
901 })  
902 })  
903 })  
904 })  
905 })  
906 })  
907 })  
908 })  
909 })  
910 })  
911 })  
912 })  
913 })  
914 })  
915 })  
916 })  
917 })  
918 })  
919 })  
920 })  
921 })  
922 })  
923 })  
924 })  
925 })  
926 })  
927 })  
928 })  
929 })  
930 })  
931 })  
932 })  
933 })  
934 })  
935 })  
936 })  
937 })  
938 })  
939 })  
940 })  
941 })  
942 })  
943 })  
944 })  
945 })  
946 })  
947 })  
948 })  
949 })  
950 })  
951 })  
952 })  
953 })  
954 })  
955 })  
956 })  
957 })  
958 })  
959 })  
960 })  
961 })  
962 })  
963 })  
964 })  
965 })  
966 })  
967 })  
968 })  
969 })  
970 })  
971 })  
972 })  
973 })  
974 })  
975 })  
976 })  
977 })  
978 })  
979 })  
980 })  
981 })  
982 })  
983 })  
984 })  
985 })  
986 })  
987 })  
988 })  
989 })  
990 })  
991 })  
992 })  
993 })  
994 })  
995 })  
996 })  
997 })  
998 })  
999 })  
1000 })  
1001 })  
1002 })  
1003 })  
1004 })  
1005 })  
1006 })  
1007 })  
1008 })  
1009 })  
1010 })  
1011 })  
1012 })  
1013 })  
1014 })  
1015 })  
1016 })  
1017 })  
1018 })  
1019 })  
1020 })  
1021 })  
1022 })  
1023 })  
1024 })  
1025 })  
1026 })  
1027 })  
1028 })  
1029 })  
1030 })  
1031 })  
1032 })  
1033 })  
1034 })  
1035 })  
1036 })  
1037 })  
1038 })  
1039 })  
1040 })  
1041 })  
1042 })  
1043 })  
1044 })  
1045 })  
1046 })  
1047 })  
1048 })  
1049 })  
1050 })  
1051 })  
1052 })  
1053 })  
1054 })  
1055 })  
1056 })  
1057 })  
1058 })  
1059 })  
1060 })  
1061 })  
1062 })  
1063 })  
1064 })  
1065 })  
1066 })  
1067 })  
1068 })  
1069 })  
1070 })  
1071 })  
1072 })  
1073 })  
1074 })  
1075 })  
1076 })  
1077 })  
1078 })  
1079 })  
1080 })  
1081 })  
1082 })  
1083 })  
1084 })  
1085 })  
1086 })  
1087 })  
1088 })  
1089 })  
1090 })  
1091 })  
1092 })  
1093 })  
1094 })  
1095 })  
1096 })  
1097 })  
1098 })  
1099 })  
1100 })  
1101 })  
1102 })  
1103 })  
1104 })  
1105 })  
1106 })  
1107 })  
1108 })  
1109 })  
1110 })  
1111 })  
1112 })  
1113 })  
1114 })  
1115 })  
1116 })  
1117 })  
1118 })  
1119 })  
1120 })  
1121 })  
1122 })  
1123 })  
1124 })  
1125 })  
1126 })  
1127 })  
1128 })  
1129 })  
1130 })  
1131 })  
1132 })  
1133 })  
1134 })  
1135 })  
1136 })  
1137 })  
1138 })  
1139 })  
1140 })  
1141 })  
1142 })  
1143 })  
1144 })  
1145 })  
1146 })  
1147 })  
1148 })  
1149 })  
1150 })  
1151 })  
1152 })  
1153 })  
1154 })  
1155 })  
1156 })  
1157 })  
1158 })  
1159 })  
1160 })  
1161 })  
1162 })  
1163 })  
1164 })  
1165 })  
1166 })  
1167 })  
1168 })  
1169 })  
1170 })  
1171 })  
1172 })  
1173 })  
1174 })  
1175 })  
1176 })  
1177 })  
1178 })  
1179 })  
1180 })  
1181 })  
1182 })  
1183 })  
1184 })  
1185 })  
1186 })  
1187 })  
1188 })  
1189 })  
1190 })  
1191 })  
1192 })  
1193 })  
1194 })  
1195 })  
1196 })  
1197 })  
1198 })  
1199 })  
1200 })  
1201 })  
1202 })  
1203 })  
1204 })  
1205 })  
1206 })  
1207 })  
1208 })  
1209 })  
1210 })  
1211 })  
1212 })  
1213 })  
1214 })  
1215 })  
1216 })  
1217 })  
1218 })  
1219 })  
1220 })  
1221 })  
1222 })  
1223 })  
1224 })  
1225 })  
1226 })  
1227 })  
1228 })  
1229 })  
1230 })  
1231 })  
1232 })  
1233 })  
1234 })  
1235 })  
1236 })  
1237 })  
1238 })  
1239 })  
1240 })  
1241 })  
1242 })  
1243 })  
1244 })  
1245 })  
1246 })  
1247 })  
1248 })  
1249 })  
1250 })  
1251 })  
1252 })  
1253 })  
1254 })  
1255 })  
1256 })  
1257 })  
1258 })  
1259 })  
1260 })  
1261 })  
1262 })  
1263 })  
1264 })  
1265 })  
1266 })  
1267 })  
1268 })  
1269 })  
1270 })  
1271 })  
1272 })  
1273 })  
1274 })  
1275 })  
1276 })  
1277 })  
1278 })  
1279 })  
1280 })  
1281 })  
1282 })  
1283 })  
1284 })  
1285 })  
1286 })  
1287 })  
1288 })  
1289 })  
1290 })  
1291 })  
1292 })  
1293 })  
1294 })  
1295 })  
1296 })  
1297 })  
1298 })  
1299 })  
1300 })  
1301 })  
1302 })  
1303 })  
1304 })  
1305 })  
1306 })  
1307 })  
1308 })  
1309 })  
1310 })  
1311 })  
1312 })  
1313 })  
1314 })  
1315 })  
1316 })  
1317 })  
1318 })  
1319 })  
1320 })  
1321 })  
1322 })  
1323 })  
1324 })  
1325 })  
1326 })  
1327 })  
1328 })  
1329 })  
1330 })  
1331 })  
1332 })  
1333 })  
1334 })  
1335 })  
1336 })  
1337 })  
1338 })  
1339 })  
1340 })  
1341 })  
1342 })  
1343 })  
1344 })  
1345 })  
1346 })  
1347 })  
1348 })  
1349 })  
1350 })  
1351 })  
1352 })  
1353 })  
1354 })  
1355 })  
1356 })  
1357 })  
1358 })  
1359 })  
1360 })  
1361 })  
1362 })  
1363 })  
1364 })  
1365 })  
1366 })  
1367 })  
1368 })  
1369 })  
1370 })  
1371 })  
1372 })  
1373 })  
1374 })  
1375 })  
1376 })  
1377 })  
1378 })  
1379 })  
1380 })  
1381 })  
1382 })  
1383 })  
1384 })  
1385 })  
1386 })  
1387 })  
1388 })  
1389 })  
1390 })  
1391 })  
1392 })  
1393 })  
1394 })  
1395 })  
1396 })  
1397 })  
1398 })  
1399 })  
1400 })  
1401 })  
1402 })  
1403 })  
1404 })  
1405 })  
1406 })  
1407 })  
1408 })  
1409 })  
1410 })  
1411 })  
1412 })  
1413 })  
1414 })  
1415 })  
1416 })  
1417 })  
1418 })  
1419 })  
1420 })  
1421 })  
1422 })  
1423 })  
1424 })  
1425 })  
1426 })  
1427 })  
1428 })  
1429 })  
1430 })  
1431 })  
1432 })  
1433 })  
1434 })  
1435 })  
1436 })  
1437 })  
1438 })  
1439 })  
1440 })  
1441 })  
1442 })  
1443 })  
1444 })  
1445 })  
1446 })  
1447 })  
1448 })  
1449 })  
1450 })  
1451 })  
1452 })  
1453 })  
1454 })  
1455 })  
1456 })  
1457 })  
1458 })  
1459 })  
1460 })  
1461 })  
1462 })  
1463 })  
1464 })  
1465 })  
1466 })  
1467 })  
1468 })  
1469 })  
1470 })  
1471 })  
1472 })  
1473 })  
1474 })  
1475 })  
1476 })  
1477 })  
1478 })  
1479 })  
1480 })  
1481 })  
1482 })  
1483 })  
1484 })  
1485 })  
1486 })  

```

```
49             if (i == restaurants.length - 1) {
50                 resolve(restaurants);
51             }
52         });
53     }
54 ).then(restaurants => {
55     let body = [];
56
57     for (let i = 0; i < restaurants.length; i++) {
58         let latTo = restaurants[i].latitude;
59         let lonTo = restaurants[i].longitude;
60
61         // On vérifie si le restaurant actuel (restaurants[i]) est dans le
62         // rayon demandé
63         if (haversineGreatCircleDistance(latFrom, lonFrom, latTo, lonTo) <
64             radius) {
65             // On vérifie ensuite si le client souhaite filtrer par des
66             // catégories
67             if (categories.length > 0) {
68                 for (let j = 0; j < categories.length; j++) {
69                     // On vérifie si le restaurant actuel contient une des
70                     // catégories par lesquels on souhaite filtrer et si le restaurant
71                     // n'est pas déjà ajouté dans le corps de notre réponse.
72                     if (restaurants[i].categories.includes(categories[j]) &&
73                         !body.includes(restaurants[i])) {
74                         body.push(restaurants[i]);
75                     }
76                 }
77             }
78
79             res.status(200).json(body);
80         });
81     });
82 }).catch(error => {
83     message.push("Adresse invalide");
84     res.status(400).json({
85         result: false,
86         message: message
87     });
88 });
89 } else {
90     message.push("Aucune adresse n'est renseignée");
91     res.status(400).json({
92         result: false,
93         message: message
94     });
95 }
96 });
97
98 module.exports = router;
```

## c:\grin-eat-tpi2022\server-node-old\src\models\Category.js

```
1 /*  
2  /\_/\_/  
3  //\\_/_\\_/_ Nom et Prénom: GRIN Brian  
4  \_\_/_/_/ /____ Enseignant : Monsieur Antoine Schmid  
5  /* * \_ /^{^} Classe : I.DA-P4B  
6  \_\_0/_/_ [ ] Date : 18.05.2022  
7  / \_/_ [ / ] Nom du projet : GrinEat-Server-Node  
8  \_\_/_/_ / / Version du projet : 0.1  
9  [ [ / \_/_ Cours : TPI  
10 _[ [ \_/_/  
11 */  
12  
13  
14 const Db = require('./Db.js');  
15  
16 class Category {  
17  
18     getIdCategory() {  
19         return this.idCategory;  
20     }  
21  
22     setIdCategory(idCategory) {  
23         this.idCategory = idCategory;  
24     }  
25  
26     getNameEnglish() {  
27         return this.nameEnglish;  
28     }  
29  
30     setNameEnglish(nameEnglish) {  
31         this.nameEnglish = nameEnglish;  
32     }  
33  
34     getNameFrench() {  
35         return this.nameFrench;  
36     }  
37  
38     setNameFrench(nameFrench) {  
39         this.nameFrench = nameFrench;  
40     }  
41  
42     static async getCategories() {  
43         let sql = "SELECT * FROM categories;"  
44         const [rows, fields] = await Db.query(sql);  
45         return rows;  
46     }  
47 }  
48 module.exports = Category;
```

## c:\grin-eat-tpi2022\server-node-old\src\models\Db.js

```
1 /*  
2  /\_ /\_/  
3  //\\_//\\_/_ / Nom et Prénom: GRIN Brian  
4  \_\_/_/_/ /____ Enseignant : Monsieur Antoine Schmid  
5  /* * \_ /^{^} ] Classe : I.DA-P4B  
6  \_\_0/_/_ [ ] Date : 18.05.2022  
7  / \_/_ [ / Nom du projet : GrinEat-Server-Node  
8  \_\_/_/_ / / Version du projet : 0.1  
9  [ [ / \_/_/ Cours : TPI  
10 _[ [ \_/_/  
11 */  
12  
13  
14 const mysql = require('mysql2/promise');  
15 const dotenv = require('dotenv').config();  
16  
17 const Db = mysql.createPool({  
18   host: process.env.DB_HOST,  
19   user: process.env.DB_USER,  
20   password: process.env.DB_PASS,  
21   database: process.env.DB_NAME  
22 });  
23  
24 module.exports = Db;
```

## c:\grin-eat-tpi2022\server-node-old\src\models\MenuItems.js

```
1 /*  
2  /\_ /\_/  
3  //\\_//\\_/_ / Nom et Prénom: GRIN Brian  
4  \_\_/_/_/ /____ Enseignant : Monsieur Antoine Schmid  
5  /* * \_ /^{^} ] Classe : I.DA-P4B  
6  \_\_0/_/_ [ ] Date : 18.05.2022  
7  / \_/_ [ / Nom du projet : GrinEat-Server-Node  
8  \_\_/_/_ / / Version du projet : 0.1  
9  [ [ / \_/_/ Cours : TPI  
10 _[ [ \_/_/  
11 */  
12  
13  
14 class MenuItems {  
15  
16 }  
17 module.exports = MenuItems;
```

## c:\grin-eat-tpi2022\server-node-old\src\models\Restaurant.js

```
1 /*  
2  /\_/\_  
3  //\\_/_/\\_/_/  
4  \_\_ _/_ /_ /_ Nom et Prénom: GRIN Brian  
5  / * * \_ /_ /^\^\] Enseignant : Monsieur Antoine Schmid  
6  \_\_0/_/_ [ ] Classe : I.DA-P4B  
7  / \_/_ [ / Date : 18.05.2022  
8  \_\_ \_ / / Nom du projet : GrinEat-Server-Node  
9  [ [ / \_/_ Version du projet : 0.1  
10 _[ [ \_/_/ Cours : TPI  
11  
12 */  
13  
14 const Db = require('./Db.js');  
15  
16 class Restaurant {  
17  
18     getIdRestaurant() {  
19         return this.idRestaurant;  
20     }  
21  
22     setIdRestaurant(idRestaurant) {  
23         this.idRestaurant = idRestaurant;  
24     }  
25  
26     getCreatedOn() {  
27         return this.createdOn;  
28     }  
29  
30     setCreatedOn(createdOn) {  
31         this.createdOn = createdOn;  
32     }  
33  
34     getEmail() {  
35         return this.email;  
36     }  
37  
38     setEmail(email) {  
39         this.email = email;  
40     }  
41  
42     getName() {  
43         return this.name;  
44     }  
45  
46     setName(name) {  
47         this.name = name;  
48     }  
49  
50     getPhone() {  
51         return this.phone;  
52     }  
53 
```

```
54     setPhone(phone) {
55         this.phone = phone;
56     }
57
58     getWebsite() {
59         return this.website;
60     }
61
62     setWebsite(website) {
63         this.website = website;
64     }
65
66     getImage() {
67         return this.image;
68     }
69
70     setImage(image) {
71         this.image = image;
72     }
73
74     getStreet() {
75         return this.street;
76     }
77
78     setStreet(street) {
79         this.street = street;
80     }
81
82     getCp() {
83         return this.cp;
84     }
85
86     setCp(cp) {
87         this.cp = cp;
88     }
89
90     getCity() {
91         return this.city;
92     }
93
94     setCity(city) {
95         this.city = city;
96     }
97
98     getCountryId() {
99         return this.countryId;
100    }
101
102    setCountryId(countryId) {
103        this.countryId = countryId;
104    }
105
106    getLatitude() {
107        return this.latitude;
108    }
```

```
109
110     setLatitude(latitude) {
111         this.latitude = latitude;
112     }
113
114     getLongitude() {
115         return this.longitude;
116     }
117
118     setLongitude(longitude) {
119         this.longitude = longitude;
120     }
121
122     static async getRestaurants() {
123         let sql = "SELECT * FROM restaurants;"
124         const [rows, fields] = await Db.query(sql);
125         return rows;
126     }
127
128     async getRestaurantCategoriesFR() {
129         let sql = "SELECT nameFrench FROM categories LEFT JOIN restaurants_categories as rc
130         ON categories.id = rc.categoryId LEFT JOIN restaurants as r ON rc.restaurantId = r.id WHERE
131         r.id = ?;"  

132         const [rows, fields] = await Db.query(sql, [this.getIdRestaurant()]);
133         return rows;
134     }
135     async getRestaurantCountry() {
136         let sql = "SELECT * FROM `countries` WHERE id = ?;"  

137         const [rows, fields] = await Db.query(sql, [this.getCountryId()]);
138         return rows;
139     }
140 module.exports = Restaurant;
```