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

```
#Turn Rewrite Engine On
RewriteEngine On

#Rewrite URL
RewriteRule ^(.*)/(.*)/(.*)$ index.php?endpoint=$1&id=$2&context=$3 [QSA,L]
RewriteRule ^(.*)/(.*)$ index.php?endpoint=$1&id=$2 [QSA,L]
RewriteRule ^(.*)$ index.php?endpoint=$1 [QSA,L]
```

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

```php
<?php
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-API
      [ [ /  \/ _/      Version du projet : 1.0
     _[ [ \  /_/        Cours : TPI

*/

function convertAdressToCoordinate($adress)
{
    $encoded = rawurlencode($adress);
    $result = array();

    $url = "https://api.mapbox.com/geocoding/v5/mapbox.places/" . $encoded .
".json?access_token=pk.eyJ1IjoibWl5b2tpIiwiYSI6ImNsMW96ejJ5NTAzMjQza3B0NHB3bHMxYncifQ.03AlBHWNc

    $curl = curl_init($url);
    curl_setopt($curl, CURLOPT_URL, $url);
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

    $resp = curl_exec($curl);
    curl_close($curl);
    $data = json_decode($resp, true);

    if (count($data["features"]) != 0) {
        $addressFound = $latitudeAddr = $data["features"][0]["place_name"];
        $latitudeAddr = $data["features"][0]["geometry"]["coordinates"][1];
        $longitudeAddr = $data["features"][0]["geometry"]["coordinates"][0];

        $result["street"] = $addressFound;
        $result["coordinates"]["latitude"] = $latitudeAddr;
        $result["coordinates"]["longitude"] = $longitudeAddr;
    } else {
        $result = null;
    }


    return $result;
    /*
    $result = array();

    $result["adress"] = "Quai Capo D'istria 9, 1205 Genève, Switzerland";
    $result["latitude"] = 46.187687;
    $result["longitude"] = 6.143453;
    return $result;
    */
}
```

```php

function haversineGreatCircleDistance(
,    $latitudeFrom,
,    $longitudeFrom,
,    $latitudeTo,
,    $longitudeTo,
,    $earthRadius = 6371
,) {
    // Convertir de degrés en radian
    $latFrom = deg2rad($latitudeFrom);
    $lonFrom = deg2rad($longitudeFrom);
    $latTo = deg2rad($latitudeTo);
    $lonTo = deg2rad($longitudeTo);

    $latDelta = $latTo - $latFrom;
    $lonDelta = $lonTo - $lonFrom;

    $angle = 2 * asin(sqrt(pow(sin($latDelta / 2), 2) +
        cos($latFrom) * cos($latTo) * pow(sin($lonDelta / 2), 2)));
    return $angle * $earthRadius;
}

function compareDistance($a, $b) {
    $distanceA = $a->getDistanceFrom();
    $distanceB = $b->getDistanceFrom();
    return ($distanceA < $distanceB) ? -1 : 1;
}
```

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

```php
<?php
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-API
      [ [ /  \/ _/      Version du projet : 1.0
     _[ [ \  /_/        Cours : TPI

*/

// Include CORS headers
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Methods: GET, POST, PUT, DELETE');
header('Access-Control-Allow-Headers: X-Requested-With');
header('Content-Type: application/json');

// Include Models & functions
include "functions.inc.php";
include "models/MyPdo.php";
include "models/Category.php";
include "models/Restaurant.php";
include "models/MenuItems.php";

// create a api variable to get HTTP method dynamically
$api = $_SERVER['REQUEST_METHOD'];
// get headers
$headers = getallheaders();
// get endpoint from url
$endpoint = $_GET['endpoint'] ?? '';
// get id from url
$id = intval($_GET['id'] ?? '');
// get context
$context = $_GET['context'] ?? '';

switch ($endpoint) {
    case '':
        echo json_encode("Welcome to the API");
        break;
    case 'categories':
        include "controllers/categoriesController.php";
        break;
    case 'restaurants':
        switch($context) {
            case '':
                include "controllers/restaurantsController.php";
                break;
            case 'menus':
                include "controllers/menu_itemsController.php";
                break;
```

```
            default:
            http_response_code(400);
            break;
        }
        break;
    default:
        http_response_code(404);
        break;
}
```

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

```php
<?php
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-API
      [ [ /  \/ _/      Version du projet : 1.0
     _[ [ \  /_/        Cours : TPI

*/

switch ($api) {
    case 'GET':
        $message = array();

        if ($id != 0) {
            $myCategory = new Category();

            $myCategory->setId($id);

            $body = $myCategory->findById();
        } else {
            $categories = Category::findAll();

            $body = $categories;
        }

        array_push($message, "success");
        http_response_code(200);
        echo json_encode(["result" => $body, "message" => $message]);
        break;
    default:
        http_response_code(405);
        break;
}
```

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

```php
<?php
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-API
      [ [ /  \/ _/      Version du projet : 1.0
     _[ [ \  /_/        Cours : TPI

*/

switch ($api) {
    case 'GET':
        $message = array();

        if ($id != 0) {
            // Récupérer les informations du restaurant à l'ID donné
            $myRestaurant = new Restaurant();
            $myRestaurant->setId($id);
            $restaurant = $myRestaurant->findById();

            foreach ($restaurant->findCategoriesById() as $c) {
                // On cherche les catégories du restaurant et les ajoutes à la propriété categories de l'objet Restaurant
                $restaurant->addCategory($c->getNameFrench());
            }

            // Récupérer les menus et éléments relative au restaurant
            $myMenuItems = new MenuItems();
            $myMenuItems->setRestaurantId($id);
            $menuItems = $myMenuItems->findItemsByRestaurantId();

            array_push($message, "success");
            http_response_code(200);
            echo json_encode(["result" => ["info" => $restaurant, "menu_items" => $menuItems], "message" => $message]);
        }
        break;
    default:
        http_response_code(405);
        break;
}
```

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

```php
<?php
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-API
      [ [ /  \/ _/      Version du projet : 1.0
     _[ [ \  /_/        Cours : TPI

*/

switch ($api) {
    case 'POST':
        $jsonString = file_get_contents("php://input");
        $data = json_decode($jsonString, true) ?? array();
        $message = array();
        $body = array();

        if ($data != null) {
            $address = $data["address"] ?? null; // adresse du client.
            $coordinates = $data["coordinates"] ?? array();
            $radius = intval($data["radius"]) != null ? intval($data["radius"]) : 5; // rayon limite autour de l'adresse du client par défaut 5 si rien spécifié.
            $name = $data["name"] ?? null; // filtre par nom si le client souhaite faire une recherche
            $categories = $data["categories"] ?? array(); // filtre par catégorie si le client en sélectionne

            if (isset($coordinates["latitude"]) && isset($coordinates["longitude"]) && !isset($info)) {
                $info["coordinates"] = $coordinates;

                if ($address != null) {
                    $info["street"] = $address;
                }
            }

            if ($address != null && !isset($info)) {
                $info = convertAdressToCoordinate($address);
            }

            if ($info != null) {
                $latFrom = $info["coordinates"]["latitude"]; // latitude de l'adresse du client
                $lonFrom = $info["coordinates"]["longitude"]; // longitude de l'adresse du client

                $restaurants = Restaurant::findAll();

                foreach ($restaurants as $r) {
```

```php
                    // On cherche les catégories de chaque restaurant et les ajoutes à la propriété categories de l'objet Restaurant
                    foreach ($r->findCategoriesById() as $c) {
                        $r->addCategory($c->getId());
                    }

                    $latTo = $r->getLatitude(); // latitude du restaurant actuel
                    $lonTo = $r->getLongitude(); // longitude du restaurant actuel

                    // On vérifie si le restaurant actuel est dans le rayon demandé
                    $r->setDistanceFrom(haversineGreatCircleDistance($latFrom, $lonFrom, $latTo, $lonTo));
                    if ($r->getDistanceFrom() < $radius) {
                        // On vérifie ensuite si le client souhaite filtrer par des catégories
                        if (count($categories) > 0) {
                            // On vérifie si le restaurant actuel contient au moins une des catégories demandés par le client
                            if (count(array_intersect($r->getCategories(), $categories)) < 1) {
                                continue; // On passe au restaurant suivant
                            }
                        }

                        // On vérifie si le client souhaite filtrer par un nom
                        if ($name != null) {
                            // On vérifie si le restaurant actuel commence par la saisie du client
                            if (strpos(strtolower($r->getName()), strtolower($name)) !== 0) {
                                continue; // On passe au restaurant suivant
                            }
                        }

                        array_push($body, $r);
                    }
                }
                // Trier les restaurants par distance croissanteQu
                usort($body, "compareDistance");

                array_push($message, "success");
                http_response_code(200);
                echo json_encode(["result" => ["address" => $info, "restaurants" => $body], "message" => $message]);
            } else {
                array_push($message, "Adresse invalide");
                http_response_code(400);
                echo json_encode(["result" => false, "message" => $message]);
            }
        } else {
            array_push($message, "Il faut au minimum une adresse ou des coordonnées latitude, longitude");
            http_response_code(400);
            echo json_encode(["result" => false, "message" => $message]);
        }
        break;
    default:
```

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

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

```php
<?php
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-API
      [ [ /  \/ _/      Version du projet : 1.0
     _[ [ \  /_/        Cours : TPI

*/

class Category implements JsonSerializable {
    private $id;
    private $nameEnglish;
    private $nameFrench;

    /**
     * Get the value of id
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set the value of id
     *
     * @return  self
     */
    public function setId($id)
    {
        $this->id = $id;

        return $this;
    }

    /**
     * Get the value of nameEnglish
     */
    public function getNameEnglish()
    {
        return $this->nameEnglish;
    }

    /**
     * Set the value of nameEnglish
     *
     * @return  self
     */
    public function setNameEnglish($nameEnglish)
```

```php
    {
        $this->nameEnglish = $nameEnglish;

        return $this;
    }

    /**
     * Get the value of nameFrench
     */
    public function getNameFrench()
    {
        return $this->nameFrench;
    }

    /**
     * Set the value of nameFrench
     *
     * @return  self
     */
    public function setNameFrench($nameFrench)
    {
        $this->nameFrench = $nameFrench;

        return $this;
    }

    public function jsonSerialize()
    {
        return get_object_vars($this);
    }

    public static function findAll() {
        $sql = MyPdo::getInstance()->prepare('SELECT * FROM categories');
        $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'Category');
        $sql->execute();
        $result = $sql->fetchAll();

        return $result;
    }

    public function findById() {
        $sql = MyPdo::getInstance()->prepare('SELECT * FROM categories WHERE id = :id');
        $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'Category');
        $sql->bindParam(':id', $this->getId());
        $sql->execute();
        $result = $sql->fetch();

        return $result;
    }
}
```

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

```php
<?php
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-API
      [ [ /  \/ _/      Version du projet : 1.0
     _[ [ \  /_/        Cours : TPI

*/

class MenuItems implements JsonSerializable {
    private $id;
    private $restaurantId;
    private $name;
    private $description;
    private $image;
    private $price;

    /**
     * Get the value of id
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set the value of id
     *
     * @return  self
     */
    public function setId($id)
    {
        $this->id = $id;

        return $this;
    }

    /**
     * Get the value of restaurantId
     */
    public function getRestaurantId()
    {
        return $this->restaurantId;
    }

    /**
     * Set the value of restaurantId
     *
```

```php
     * @return  self
     */
    public function setRestaurantId($restaurantId)
    {
        $this->restaurantId = $restaurantId;

        return $this;
    }

    /**
     * Get the value of name
     */
    public function getName()
    {
        return $this->name;
    }

    /**
     * Set the value of name
     *
     * @return  self
     */
    public function setName($name)
    {
        $this->name = $name;

        return $this;
    }

    /**
     * Get the value of description
     */
    public function getDescription()
    {
        return $this->description;
    }

    /**
     * Set the value of description
     *
     * @return  self
     */
    public function setDescription($description)
    {
        $this->description = $description;

        return $this;
    }

    /**
     * Get the value of image
     */
    public function getImage()
    {
        return $this->image;
```

```php
    }

    /**
     * Set the value of image
     *
     * @return  self
     */
    public function setImage($image)
    {
        $this->image = $image;

        return $this;
    }

    /**
     * Get the value of price
     */
    public function getPrice()
    {
        return $this->price;
    }

    /**
     * Set the value of price
     *
     * @return  self
     */
    public function setPrice($price)
    {
        $this->price = $price;

        return $this;
    }

    public function jsonSerialize()
    {
        return get_object_vars($this);
    }

    public function findItemsByRestaurantId() {
        $sql = MyPdo::getInstance()->prepare('SELECT * FROM menu_items WHERE restaurantId = :id');
        $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'MenuItems');
        $sql->bindParam(':id', $this->getRestaurantId());
        $sql->execute();
        $result = $sql->fetchAll();

        return $result;
    }
}
```

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

```php
<?php
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-API
      [ [ /  \/ _/      Version du projet : 1.0
     _[ [ \  /_/        Cours : TPI

*/

class MyPdo
{
private static $dbhost = 'localhost';
private static $dbname = 'db_grineat';
private static $dbuser = 'admin_grineat';
private static $dbpasswd = 'ND)BO/4S/dr[_6H6';

private static $MyPdo;
private static $unPdo = null;

//  Constructeur privé, crée l'instance de PDO qui sera sollicitée
//  pour toutes les méthodes de la classe
private function __construct()
{
    MyPdo::$unPdo = new PDO("mysql:host=".MyPdo::$dbhost.';dbname='.MyPdo::$dbname, MyPdo::$dbuser, MyPdo::$dbpasswd);
    MyPdo::$unPdo->query("SET CHARACTER SET utf8");
    MyPdo::$unPdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
public function __destruct()
{
    MyPdo::$unPdo = null;
}
/**
*   Fonction statique qui cree l'unique instance de la classe
*   Appel : $instanceMyPdo = MyPdo::getMyPdo();
*   @return l'unique objet de la classe MyPdo
*/
public static function getInstance()
{
    if(self::$unPdo == null)
    {
        self::$MyPdo= new MyPdo();
    }
    return self::$unPdo;
}

}
```

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

```php
<?php
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-API
      [ [ /  \/ _/      Version du projet : 1.0
     _[ [ \  /_/        Cours : TPI

*/

class Restaurant implements JsonSerializable {
    private $id;
    private $createdOn;
    private $email;
    private $name;
    private $phone;
    private $website;
    private $image;
    private $street;
    private $cp;
    private $city;
    private $countryId;
    private $latitude;
    private $longitude;
    private $categories = array();
    private $distanceFrom;

    /**
     * Get the value of id
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set the value of id
     *
     * @return  self
     */
    public function setId($id)
    {
        $this->id = $id;

        return $this;
    }

    /**
     * Get the value of createdOn
```

```php
     */
    public function getCreatedOn()
    {
        return $this->createdOn;
    }

    /**
     * Set the value of createdOn
     *
     * @return  self
     */
    public function setCreatedOn($createdOn)
    {
        $this->createdOn = $createdOn;

        return $this;
    }

    /**
     * Get the value of email
     */
    public function getEmail()
    {
        return $this->email;
    }

    /**
     * Set the value of email
     *
     * @return  self
     */
    public function setEmail($email)
    {
        $this->email = $email;

        return $this;
    }

    /**
     * Get the value of name
     */
    public function getName()
    {
        return $this->name;
    }

    /**
     * Set the value of name
     *
     * @return  self
     */
    public function setName($name)
    {
        $this->name = $name;

```

```php
        return $this;
    }

    /**
     * Get the value of phone
     */
    public function getPhone()
    {
        return $this->phone;
    }

    /**
     * Set the value of phone
     *
     * @return  self
     */
    public function setPhone($phone)
    {
        $this->phone = $phone;

        return $this;
    }

    /**
     * Get the value of website
     */
    public function getWebsite()
    {
        return $this->website;
    }

    /**
     * Set the value of website
     *
     * @return  self
     */
    public function setWebsite($website)
    {
        $this->website = $website;

        return $this;
    }

    /**
     * Get the value of image
     */
    public function getImage()
    {
        return $this->image;
    }

    /**
     * Set the value of image
     *
     * @return  self
```

```php
     */
    public function setImage($image)
    {
        $this->image = $image;

        return $this;
    }

    /**
     * Get the value of street
     */
    public function getStreet()
    {
        return $this->street;
    }

    /**
     * Set the value of street
     *
     * @return  self
     */
    public function setStreet($street)
    {
        $this->street = $street;

        return $this;
    }

    /**
     * Get the value of cp
     */
    public function getCp()
    {
        return $this->cp;
    }

    /**
     * Set the value of cp
     *
     * @return  self
     */
    public function setCp($cp)
    {
        $this->cp = $cp;

        return $this;
    }

    /**
     * Get the value of city
     */
    public function getCity()
    {
        return $this->city;
    }
```

```php

    /**
     * Set the value of city
     *
     * @return  self
     */
    public function setCity($city)
    {
        $this->city = $city;

        return $this;
    }

    /**
     * Get the value of countryId
     */
    public function getCountryId()
    {
        return $this->countryId;
    }

    /**
     * Set the value of countryId
     *
     * @return  self
     */
    public function setCountryId($countryId)
    {
        $this->countryId = $countryId;

        return $this;
    }

    /**
     * Get the value of latitude
     */
    public function getLatitude()
    {
        return $this->latitude;
    }

    /**
     * Set the value of latitude
     *
     * @return  self
     */
    public function setLatitude($latitude)
    {
        $this->latitude = $latitude;

        return $this;
    }

    /**
     * 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     {
```

```php
        return $this->distanceFrom;
    }

    /**
     * Set the value of distanceFrom
     *
     * @return  self
     */
    public function setDistanceFrom($distanceFrom)
    {
        $this->distanceFrom = $distanceFrom;

        return $this;
    }

    public function jsonSerialize()
    {
        return get_object_vars($this);
    }

    public static function findAll() {
        $sql = MyPdo::getInstance()->prepare('SELECT * FROM restaurants;');
        $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'Restaurant');
        $sql->execute();
        $result = $sql->fetchAll();

        return $result;
    }

    public function findById() {
        $sql = MyPdo::getInstance()->prepare('SELECT * FROM restaurants WHERE id = :id;');
        $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'Restaurant');
        $sql->bindParam(':id', $this->getId());
        $sql->execute();
        $result = $sql->fetch();

        return $result;
    }

    public function findCategoriesById() {
        $sql = MyPdo::getInstance()->prepare('SELECT categories.* FROM categories LEFT JOIN restaurants_categories as rc ON categories.id = rc.categoryId LEFT JOIN restaurants as r ON rc.restaurantId = r.id WHERE r.id = :id;');
        $sql->setFetchMode(PDO::FETCH_CLASS | PDO::FETCH_PROPS_LATE, 'Category');
        $sql->bindParam(':id', $this->getId());
        $sql->execute();
        $result = $sql->fetchAll();

        return $result;
    }
}
```