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

```
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-Server-Node
      [ [ /  \/ _/      Version du projet : 0.1
     _[ [ \  /_/        Cours : TPI

*/

const axios = require('axios').default;

async function convertAdressToCoordoninate(adress) {
    return new Promise((resolve, reject) => {
        encoded = encodeURIComponent(adress);

        axios.get(`https://api.mapbox.com/geocoding/v5/mapbox.places
/${encoded}.json?access_token=${process.env.MAPBOX_TOKEN}`)
            .then(function (response) {
                let adressFound = response.data.features[0].place_name;
                let latitudeAdrr = response.data.features[0].geometry.coordinates[1];
                let longitudeAdrr = response.data.features[0].geometry.coordinates[0];

                resolve({ adress: adressFound, latitude: latitudeAdrr, longitude:
longitudeAdrr });
            })
            .catch(function (error) {
                console.table(error);
                reject("Invalid adress");
            });
    });
}

function haversineGreatCircleDistance(
,    latitudeFrom, longitudeFrom, latitudeTo, longitudeTo, earthRadius = 6371) {
    // convert from degress to radians
    let latFrom = deg2rad(latitudeFrom);
    let lonFrom = deg2rad(longitudeFrom);
    let latTo = deg2rad(latitudeTo);
    let lonTo = deg2rad(longitudeTo);

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

    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)));

    return angle * earthRadius;
}
```

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

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

```
1  /*
2       /\   /\
3     //\\_//\\    ____
4     \_     _/   /   /   Nom et Prénom: GRIN Brian
5     / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
6     \_\O/_/    [   ]    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 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

```
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-Server-Node
      [ [ /  \/ _/      Version du projet : 0.1
     _[ [ \  /_/        Cours : TPI

*/

const express = require('express');
const router = express.Router();
const Category = require('../models/Category.js');

router.get('/', (req, res) => {
    Category.getCategories().then(categories => {
        res.status(200).json(categories);
    })
});

module.exports = router;
```

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

```
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-Server-Node
      [ [ /  \/ _/      Version du projet : 0.1
     _[ [ \  /_/        Cours : TPI

*/

const express = require('express');
const router = express.Router();
const MenuItems = require('../models/MenuItems.js');

router.get('/:id', (req, res) => {
    let myMenuItems = new MenuItems();
    let idRestaurant = req.params.id;

    myMenuItems.setRestaurantId(idRestaurant);

    myMenuItems.findById().then(result => {
        res.status.json(result);
    });
});

module.exports = router;
```

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

```
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-Server-Node
      [ [ /  \/ _/      Version du projet : 0.1
     _[ [ \  /_/        Cours : TPI

*/

const { response } = require('express');
const express = require('express');
const router = express.Router();
const { convertAdressToCoordoninate, haversineGreatCircleDistance } = require('../functions.js');
const Restaurant = require('../models/Restaurant.js');

router.post('/', (req, res) => {
    let adress = req.body.adress; // adresse du client.
    let radius = req.body.radius || 5; // rayon limite autour de l'adresse du client par défaut 5 si rien spécifié.
    let name = req.body.name || null; // filtre par nom si le client souhaite faire une recherche.
    let categories = req.body.categories || []; // filtre par catégorie si le client en séléctionne.

    let restaurants = [];
    let message = [];

    // On vérifie que le client nous a bien renseigné une adresse
    if (adress != null) {
        convertAdressToCoordoninate(adress).then(response => {
            let latFrom = response.latitude;
            let lonFrom = response.longitude;

            Restaurant.getRestaurants().then(restaurants => {
                // On crée une promesse pour récupérer les catégories de tout les restaurants
                new Promise((resolve) => {
                    for (let i = 0; i < restaurants.length; i++) {
                        restaurants[i].categories = [];
                        let restaurant = new Restaurant();

                        restaurant.setIdRestaurant(restaurants[i].id);

                        restaurant.getRestaurantCategoriesFR().then(categories => {
                            for (let j = 0; j < categories.length; j++) {
                                restaurants[i].categories.push(categories[j].nameFrench || null);
                            }

```

```
                            if (i == restaurants.length - 1) {
                                resolve(restaurants);
                            }
                        });
                    }
                }).then(restaurants => {
                    let body = [];

                    for (let i = 0; i < restaurants.length; i++) {
                        let latTo = restaurants[i].latitude;
                        let lonTo = restaurants[i].longitude;

                        // On vérifie si le restaurant actuel (restaurants[i]) est dans le rayon demandé
                        if (haversineGreatCircleDistance(latFrom, lonFrom, latTo, lonTo) < radius) {
                            // On vérifie ensuite si le client souhaite filtrer par des catégories
                            if (categories.length > 0) {
                                for (let j = 0; j < categories.length; j++) {
                                    // On vérifie si le restaurant actuel contient une des catégories par lesquels on souhaite filtrer et si le restaurant
                                    // n'est pas déjà ajouté dans le corps de notre réponse.
                                    if (restaurants[i].categories.includes(categories[j]) && !body.includes(restaurants[i])) {
                                        body.push(restaurants[i]);
                                    }
                                }
                            }
                            else {
                                body.push(restaurants[i]);
                            }
                        }
                    }

                    res.status(200).json(body);
                });
            });
        }).catch(error => {
            message.push("Adresse invalide");
            res.status(400).json({
                result: false,
                message: message
            });
        });
    } else {
        message.push("Aucune adresse n'est renseignée");
        res.status(400).json({
            result: false,
            message: message
        });
    }
});

module.exports = router;
```

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

```
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-Server-Node
      [ [ /  \/ _/      Version du projet : 0.1
     _[ [ \  /_/        Cours : TPI

*/

const Db = require('./Db.js');

class Category {

    getIdCategory() {
        return this.idCategory;
    }

    setIdCategory(idCategory) {
        this.idCategory = idCategory;
    }

    getNameEnglish() {
        return this.nameEnglish;
    }

    setNameEnglish(nameEnglish) {
        this.nameEnglish = nameEnglish;
    }

    getNameFrench() {
        return this.nameFrench;
    }

    setNameFrench(nameFrench) {
        this.nameFrench = nameFrench;
    }

    static async getCategories() {
        let sql = "SELECT * FROM categories;"
        const [rows, fields] = await Db.query(sql);
        return rows;
    }
}
module.exports = Category;
```

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

```
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-Server-Node
      [ [ /  \/ _/      Version du projet : 0.1
     _[ [ \  /_/        Cours : TPI

*/

const mysql = require('mysql2/promise');
const dotenv = require('dotenv').config();

const Db = mysql.createPool({
    host: process.env.DB_HOST,
    user: process.env.DB_USER,
    password: process.env.DB_PASS,
    database: process.env.DB_NAME
});

module.exports = Db;
```

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

```
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-Server-Node
      [ [ /  \/ _/      Version du projet : 0.1
     _[ [ \  /_/        Cours : TPI

*/

class MenuItems {

}
module.exports = MenuItems;
```

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

```
/*
     /\   /\
    //\\_//\\    ____
    \_     _/   /   /   Nom et Prénom: GRIN Brian
    / * * \    /^^^]    Enseignant : Monsieur Antoine Schmid
    \_\O/_/    [   ]    Classe : I.DA-P4B
     /   \_    [   /    Date : 18.05.2022
     \     \_  /  /     Nom du projet : GrinEat-Server-Node
      [ [ /  \/ _/      Version du projet : 0.1
     _[ [ \  /_/        Cours : TPI

*/

const Db = require('./Db.js');

class Restaurant {

    getIdRestaurant() {
        return this.idRestaurant;
    }

    setIdRestaurant(idRestaurant) {
        this.idRestaurant = idRestaurant;
    }

    getCreatedOn() {
        return this.createdOn;
    }

    setCreatedOn(createdOn) {
        this.createdOn = createdOn;
    }

    getEmail() {
        return this.email;
    }

    setEmail(email) {
        this.email = email;
    }

    getName() {
        return this.name;
    }

    setName(name) {
        this.name = name;
    }

    getPhone() {
        return this.phone;
    }

```

```
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
    ON categories.id = rc.categoryId LEFT JOIN restaurants as r ON rc.restaurantId = r.id WHERE
    r.id = ?;"
130         const [rows, fields] = await Db.query(sql, [this.getIdRestaurant()]);
131         return rows;
132     }
133
134     async getRestaurantCountry() {
135         let sql = "SELECT * FROM `countries` WHERE id = ?;"
136         const [rows, fields] = await Db.query(sql, [this.getCountryId()]);
137         return rows;
138     }
139 }
140 module.exports = Restaurant;
```