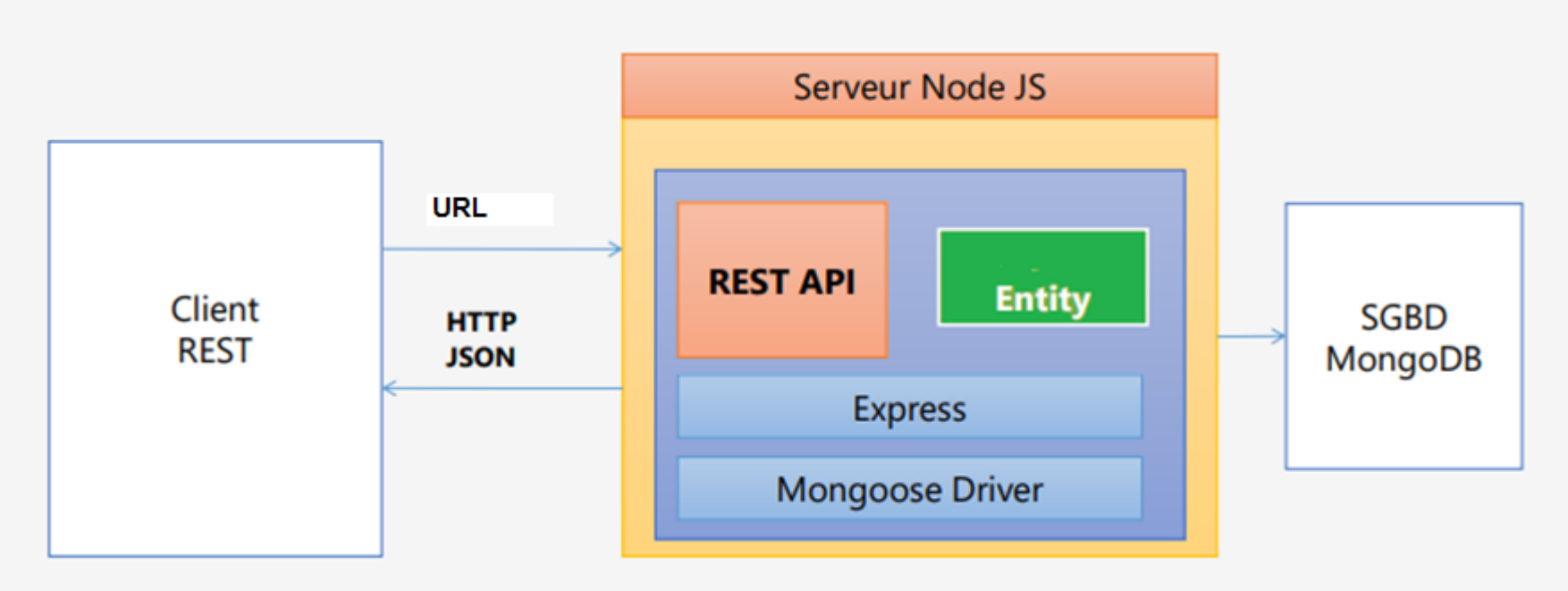
# Créer une REST API



- Le protocole HTTP utilise des verbes pour décrire les opérations à effectuer. Puisque on passe par un site web (c'est tout le principe d'une REST API), on va utiliser les verbes pour représenter les opérations CRUD :

  - Lire : On utilise le verbe **get** , c'est celui qui est utilisé de base lorsque l'on accède à un site web. Dans la conception des REST API, on a décidé d'assigner le verbe get à l'opération de lecture car c'est l'opération la plus courante.
  - Ecrire : Pour écrire des données, on utilise le verbe **post**. Par écrire, on entend créer des données. Le verbe post est initialement utilisé dans les site web pour soumettre des formulaires. Il était donc logique de l'utiliser pour la création de données.
  - Modifier : Pour modifier des données, on utilise utiliser le verbe **put**.
  - Supprimer : Lorsque l'on veut supprimer des données, on utilise le verbe **delete**. Il permet de supprimer définitivement un élément de la base de donnée.

Code de connexion à la base de donnée et au serveur

# Fichier app.js

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
// connexion à la base de donnée
const db = "mongodb://localhost:27017/bibliotheque"
const mongoose = require('mongoose');
mongoose.connect(db,{
    useNewUrlParser: true,
    useUnifiedTopology: true
  }).then(()=>{
        console.log("Connexion a la base de donnee reussie");
        }).catch(err=>{
                console.log('Connexion impossible a la base de donnée',err);
                process.exit();
});
// appel aux routes
require('./app/routes/livre.routes.js')(app);
app.listen(3000, () => {
    console.log("Serveur port : 3000"); });
```

Créer dans le dossier app/models un fichier livre.js pour la classe livre

```
const mongoose=require("mongoose")
const livreSchema=mongoose.Schema({
    code : Number,
    titre : String,
    auteur : String,
    anedition : Number,
    maisedition : String,
    nbexmplaire : Number,
    prix : Number
});
module.exports=mongoose.model('Livre',livreSchema)
```

# Définition des routes

Créer un nouveau dossier appelé routes dans le dossier de l'application.

créer un nouveau fichier appelé livre.routes.js dans le dossier app/routes avec le contenu suivant :

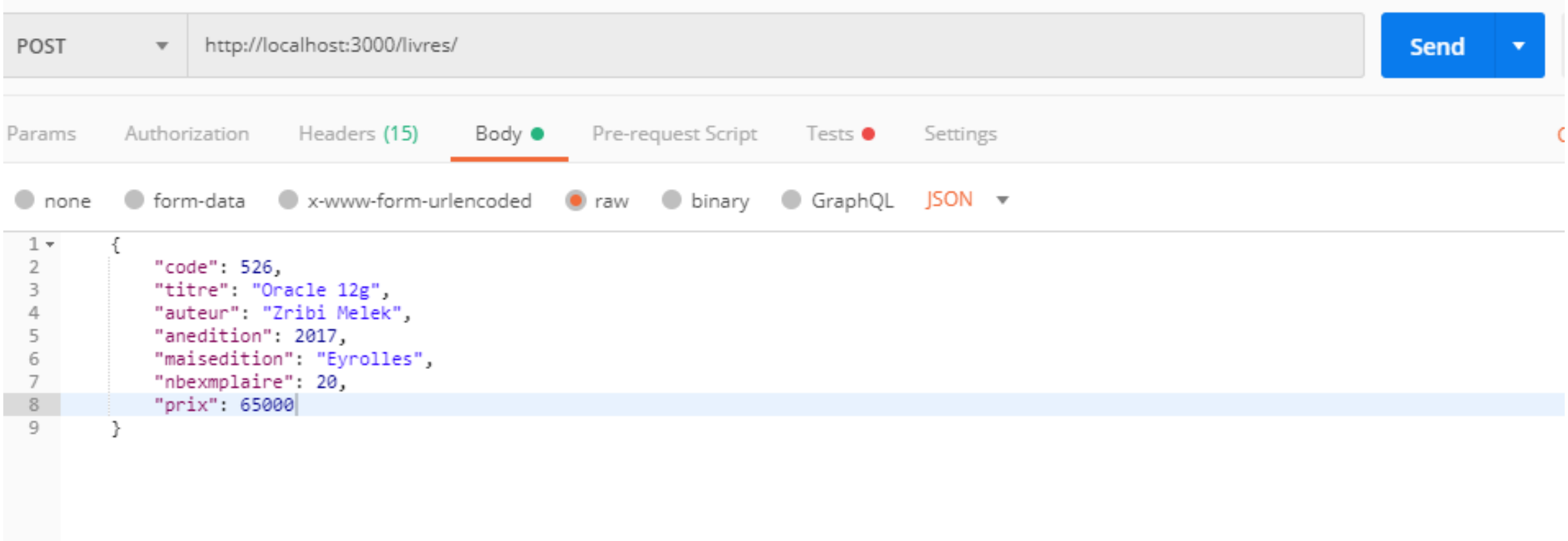
```
const liv = require('../controllers/livre.controllers.js');
module.exports = (app) => {

   // afficher tous les livres
   app.get('/livres', liv.afficherTout);

   // créer un livre
   app.post('/livres', liv.creer);

  // afficher un livre de Id donné
   app.get('/livres/:livreId', liv.afficherUn);

  // modifier un livre
   app.put('/livres/:livreId', liv.modifier);

  // supprimer un livre
   app.delete('/livres/:livreId', liv.supprimer);
    }
```

# Fichier livre.controller.js dans le répertoire app/controllers
# Méthodes get, post, update, delete

```
const Livre = require('../models/livre');

// créer un nouveau livre
exports.creer =  async(req, res) =>{

};

// afficher la liste des livres.
exports.afficherTout =  async(req, res) =>  {

};

// chercher un livre
exports.afficherUn =  async(req, res) =>  {

};

// modifier un livre
exports.modifier =  async(req, res) =>  {

};

// Supprimer un livre
exports.supprimer =  async(req, res) =>  {

};
```

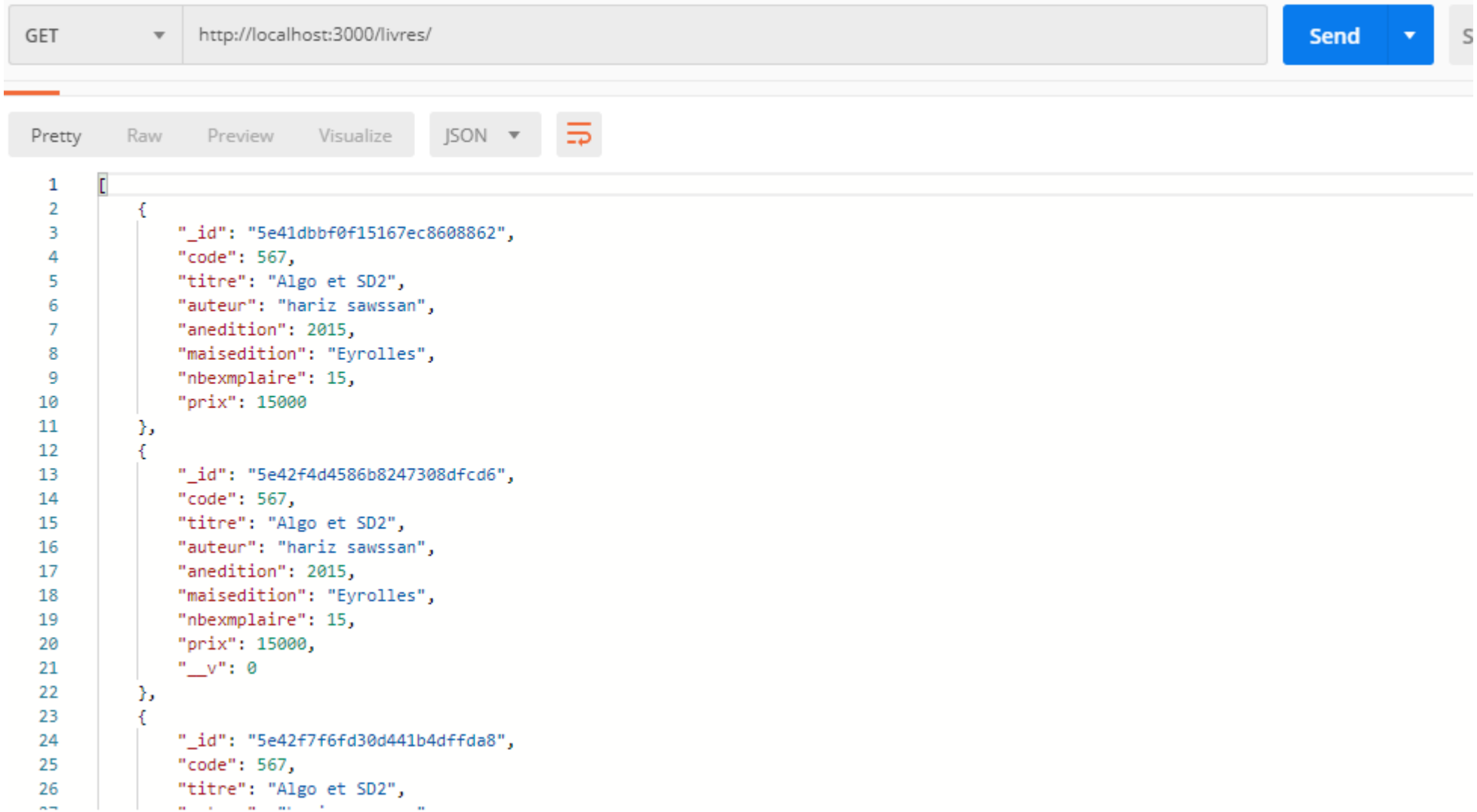
## Fichier livre.controller.js dans le répertoire controllers
## Méthode creer

```
// créer un nouveau livre
exports.creer = async(req, res) => {
    try {
        var livre = new Livre(req.body);
        var result = await livre.save();
        res.send(result);
    } catch (error) {
        res.status(500).send(error);
    }
};
```


POST
http://localhost:3000/livres/
Send
Params
Authorization
Headers (15)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"code": 526,
"titre": "Oracle 12g",
"auteur": "Zribi Melek",
"anedition": 2017,
"maisedition": "Eyrolles",
"nbexmplaire": 20,
"prix": 65000
}

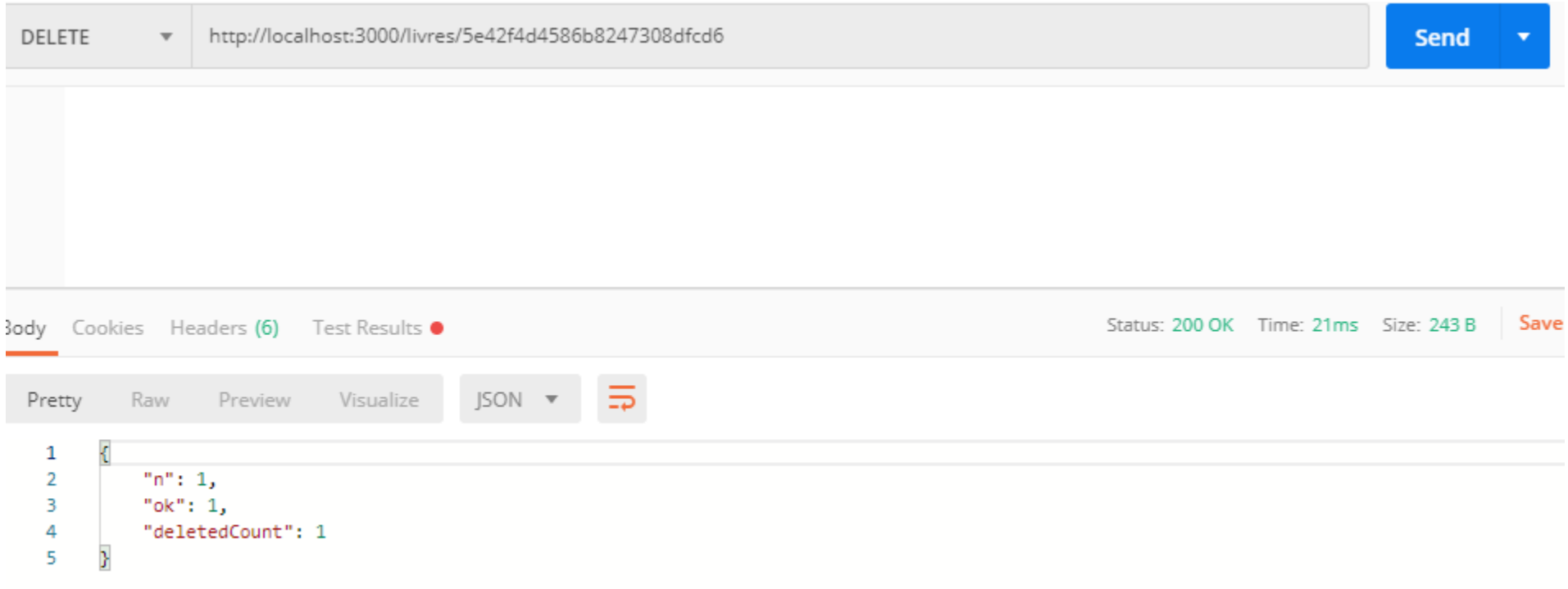
## Fichier livre.controller.js dans le répertoire controllers
## Méthode afficherTout

```
// afficher la liste des livres.
exports.afficherTout = async(req, res) => {
    try {
        var result = await Livre.find().exec();
        res.send(result);
        }
    catch (error) {
        res.status(500).send(error);
    }
};
```


GET
http://localhost:3000/livres/
Send
Pretty
Raw
Preview
Visualize
JSON
[
{
"_id": "5e41dbbf0f15167ec8608862",
"code": 567,
"titre": "Algo et SD2",
"auteur": "hariz sawssan",
"anedition": 2015,
"maisedition": "Eyrolles",
"nbexmplaire": 15,
"prix": 15000
},
{
"_id": "5e42f4d4586b8247308dfcd6",
"code": 567,
"titre": "Algo et SD2",
"auteur": "hariz sawssan",
"anedition": 2015,
"maisedition": "Eyrolles",
"nbexmplaire": 15,
"prix": 15000,
"__v": 0
},
{
"_id": "5e42f7f6fd30d441b4dffda8",
"code": 567,
"titre": "Algo et SD2",

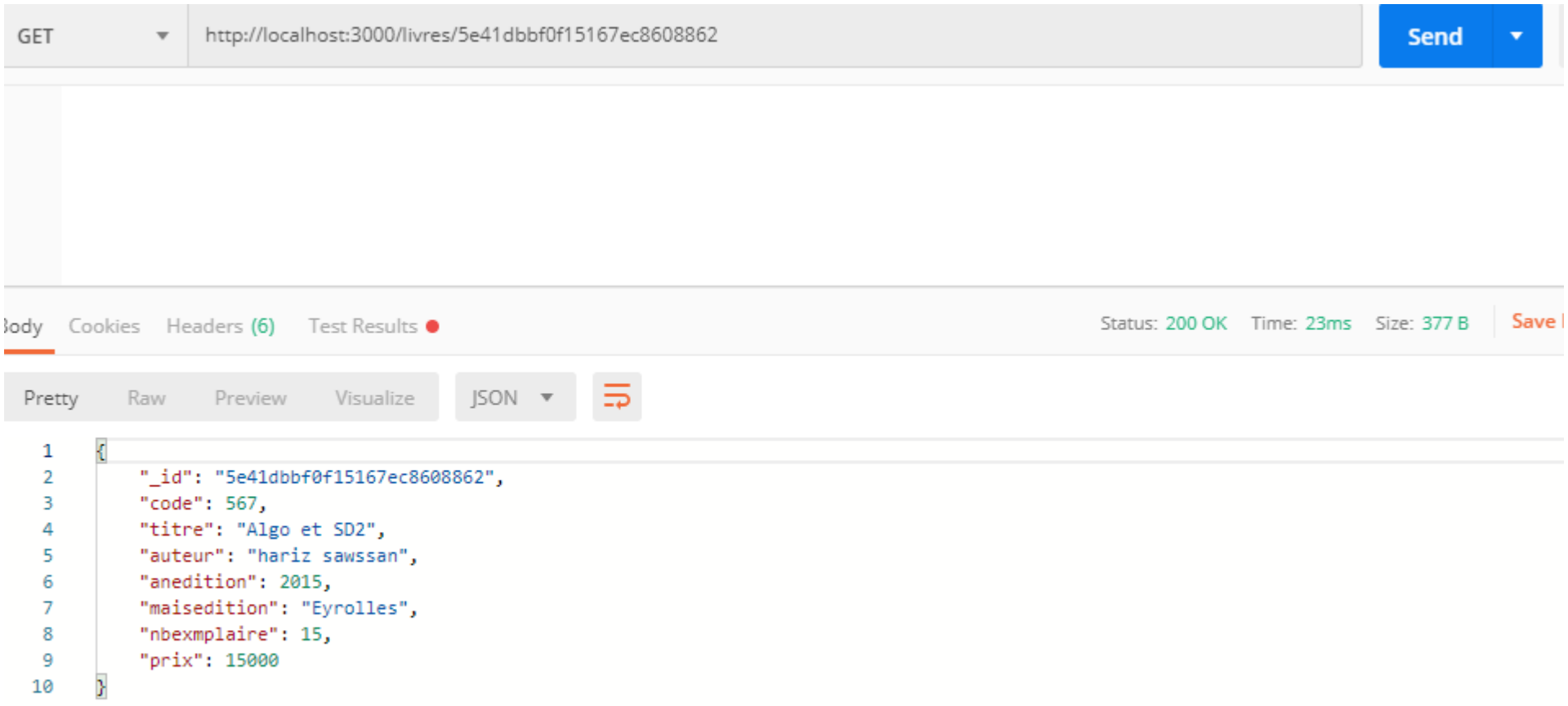
# Fichier livre.controller.js dans le répertoire controllers
# Méthode supprimer

```
// Supprimer un livre

exports.supprimer = async(req, res) => {
try {
 var result = await Livre.deleteOne({ _id: req.params.livreId }).exec();
 res.send(result);
        }
catch (error)
    {
        res.status(500).send(error);
    }
};
```


DELETE
http://localhost:3000/livres/5e42f4d4586b8247308dfcd6
Send
Body
Cookies
Headers (6)
Test Results
Status: 200 OK
Time: 21ms
Size: 243 B
Save
Pretty
Raw
Preview
Visualize
JSON
{
"n": 1,
"ok": 1,
"deletedCount": 1
}

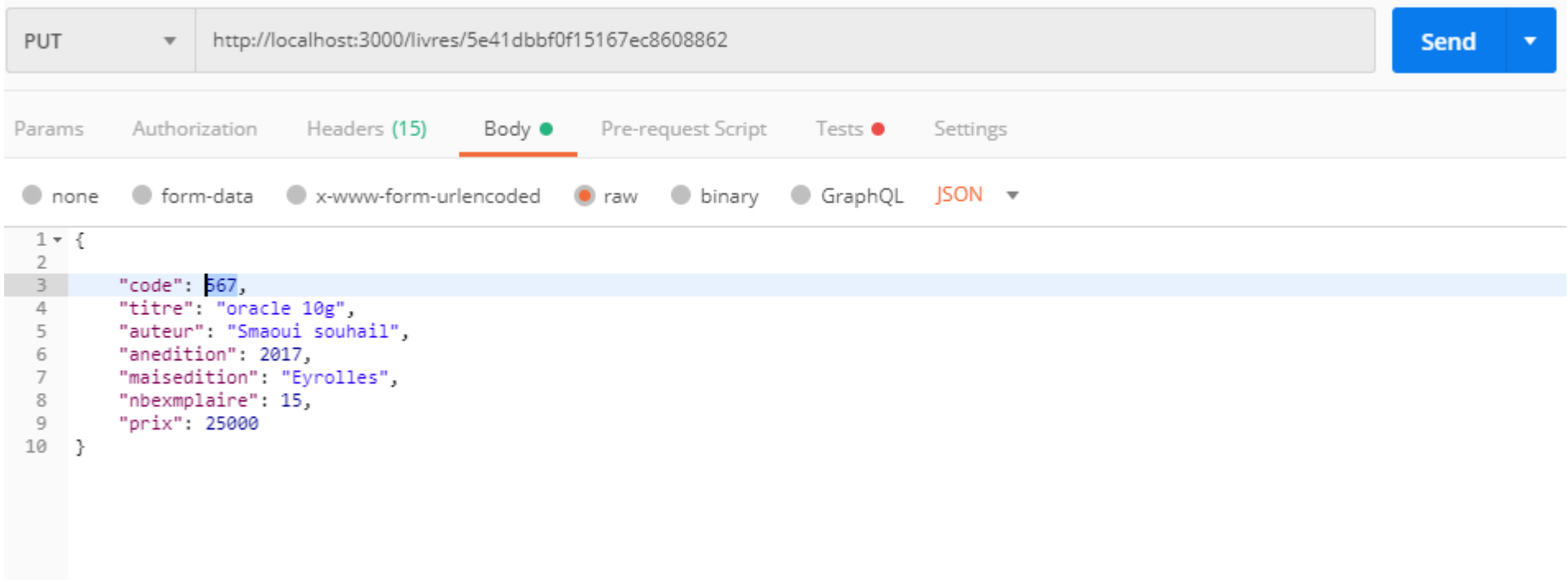
# Fichier livre.controller.js dans le répertoire controllers
# Méthode afficherUn

```
// chercher un livre


exports.afficherUn = async(req, res) => {
    try {
        var result = await Livre.findById({ _id: req.params.livreId }).exec();
        res.send(result);
    } catch (error) {
        res.status(500).send(error);
    }
};
```


GET
http://localhost:3000/livres/5e41dbbf0f15167ec8608862
Send
Body
Cookies
Headers (6)
Test Results
Status: 200 OK
Time: 23ms
Size: 377 B
Save
Pretty
Raw
Preview
Visualize
JSON
{
"_id": "5e41dbbf0f15167ec8608862",
"code": 567,
"titre": "Algo et SD2",
"auteur": "hariz sawssan",
"anedition": 2015,
"maisedition": "Eyrolles",
"nbexmplaire": 15,
"prix": 15000
}

# Fichier livre.controller.js dans le répertoire controllers
# Méthode modifier

```
// modifier un livre
exports.modifier = async(req, res) => {
    try
    {
    var anclivre =  await Livre.findById({ _id: req.params.livreId }).exec();

    anclivre.titre = req.body.titre;
    anclivre.auteur = req.body.auteur;
    anclivre.anedition = req.body.anedition;
    anclivre.maisedition = req.body.maisedition;
    anclivre.nbexmplaire = req.body.nbexmplaire;
    anclivre.prix = req.body.prix;
        var result = await anclivre.save();
        res.send(result);
      }
    catch (error){

            res.status(400).send("impossible de modifier la base de données");
      }
};
```


PUT
http://localhost:3000/livres/5e41dbbf0f15167ec8608862
Send
Params
Authorization
Headers (15)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"code": 667,
"titre": "oracle 10g",
"auteur": "Smaoui souhail",
"anedition": 2017,
"maisedition": "Eyrolles",
"nbexmplaire": 15,
"prix": 25000
}