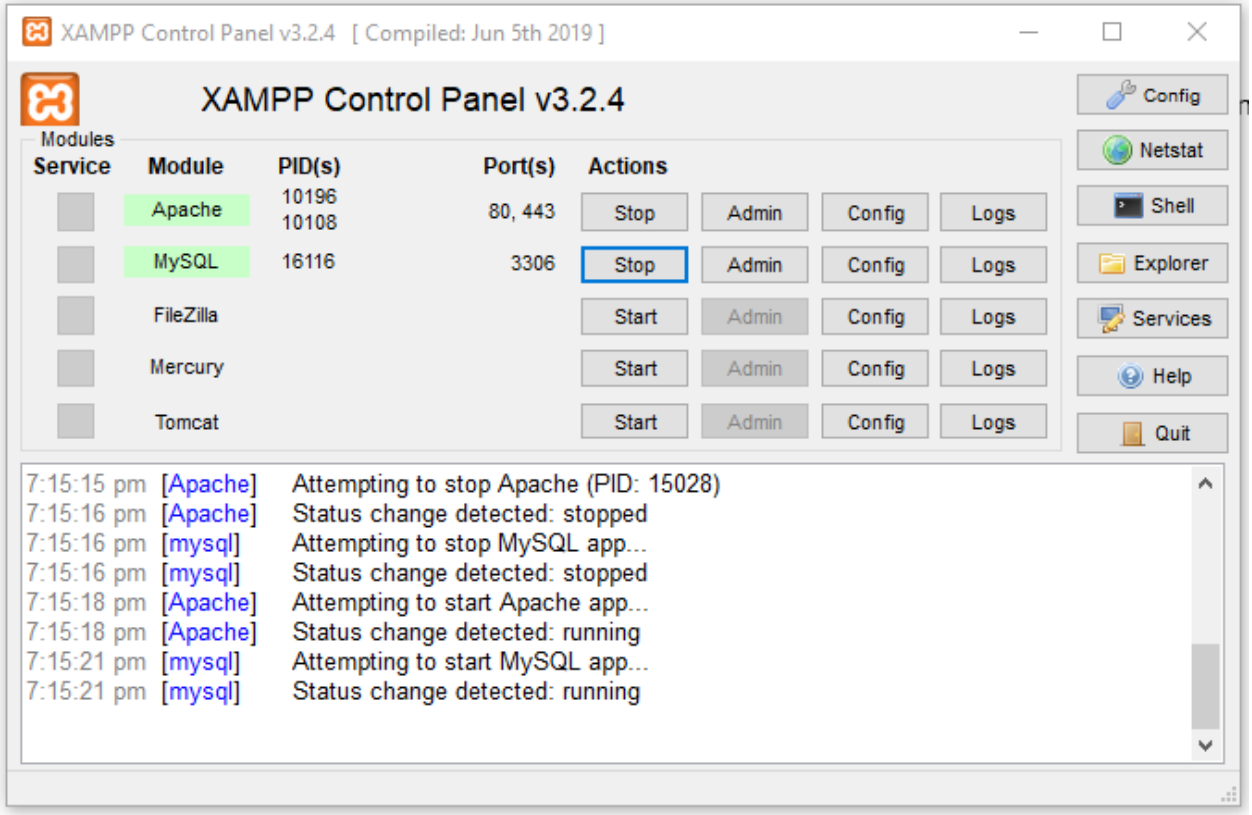
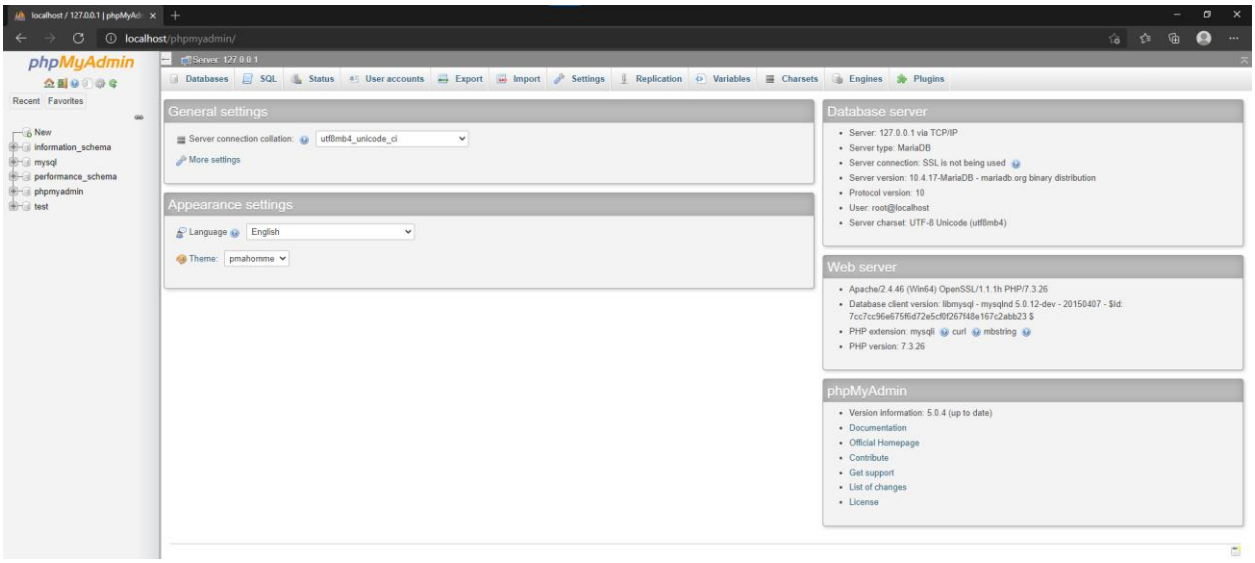
Connection to Database:

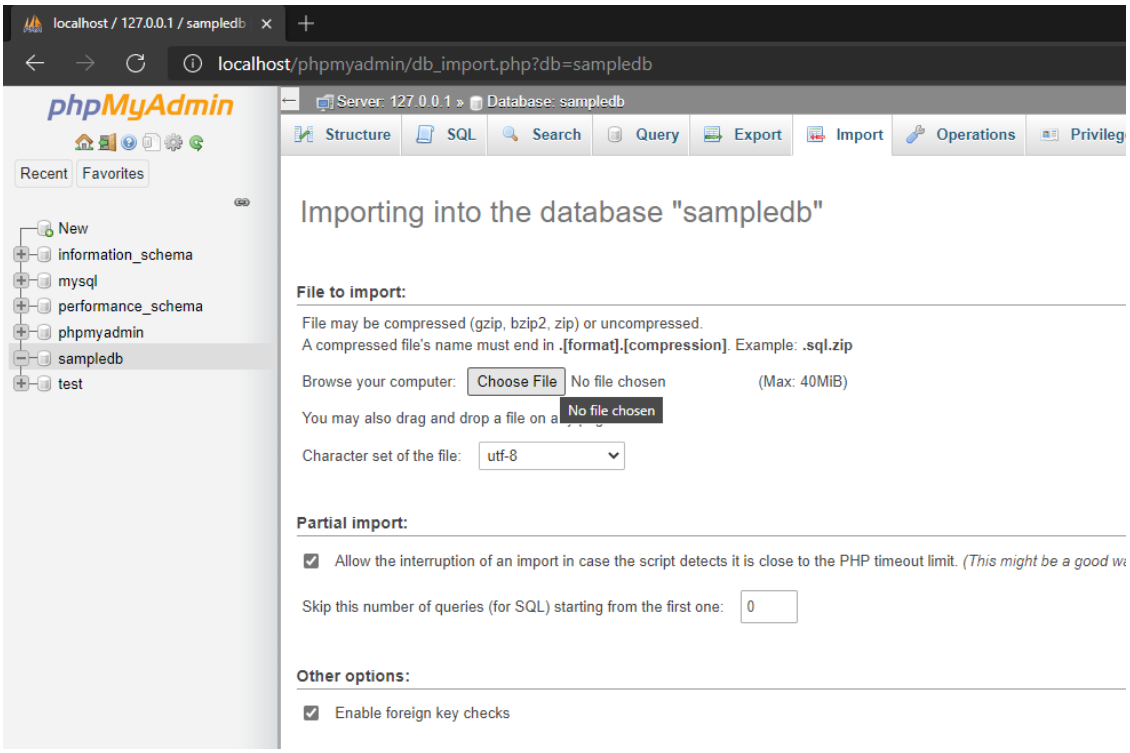
1. Go to [Download XAMPP (apachefriends.org)] and download your appropriate installer based on your computer's OS. After downloading, install it to your computer.
2. Open your XAMPP Control Panel and start Apache and MYSQL.



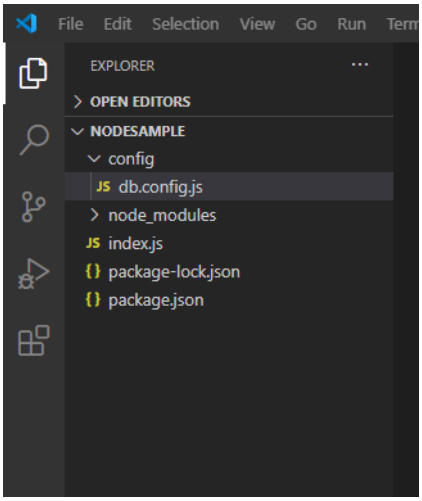
3. Go to your browser and go to this link: [http://localhost/phpmyadmin/] to open your phpMyAdmin dashboard.



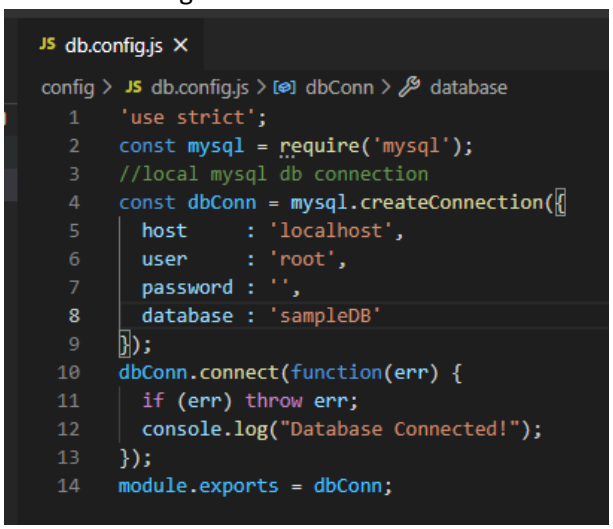
4. Go to the Import tab of your database dashboard and import the database that I included. (sampleDB.sql). Click the Go button found in the lower right of the Import tab to proceed.



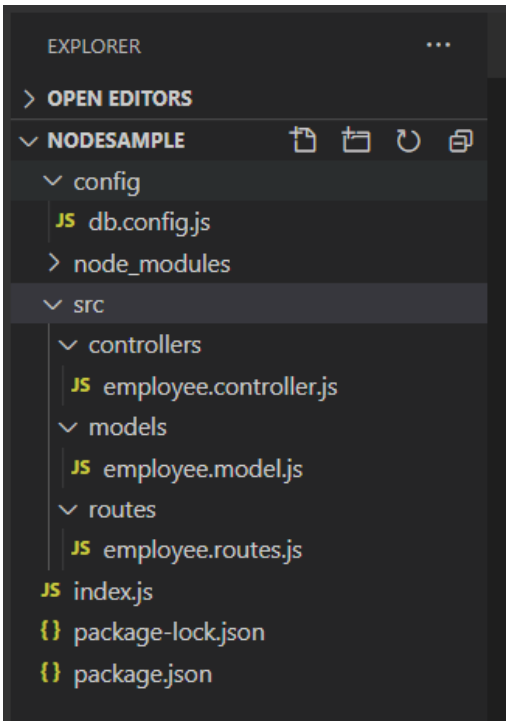
5. In your project folder, create a folder with the name of config and create a js file with the filename of db.config.



6. In your db.config.js file, add these configuration code:



7. Add a src folder for your controllers and models.



8. Add the following code for the employee.model.js file which should inside the models folder.
   employee.model.js – this manages the structure of your data, logic and db queries

   a. Establishing connection with the configuration:

models > JS employee.model.js > ...

```
'use strict';
var dbConn = require('./../../config/db.config');

//Employee object create
var Employee = function(employee){
    this.first_name     = employee.first_name;
    this.last_name      = employee.last_name;
    this.email          = employee.email;
    this.phone          = employee.phone;
    this.organization   = employee.organization;
    this.designation    = employee.designation;
    this.salary         = employee.salary;
    this.status         = employee.status ? employee.status : 1;
    this.created_at     = new Date();
    this.updated_at     = new Date();
};
```

   b. Creating/inserting an entry to the DB:

```
//create
Employee.create = function (newEmp, result) {
    dbConn.query("INSERT INTO employees set ?", newEmp, function (err, res) {
        if(err) {
            console.log("error: ", err);
            result(err, null);
        }
        else{
            console.log(res.insertId);
            result(null, res.insertId);
        }
    });
};
```

   c. Querying a single entry in the DB using the id as query parameter:

```
//query a single entry
Employee.findById = function (id, result) {
    dbConn.query("Select * from employees where id = ? ", id, function (err, res) {
        if(err) {
            console.log("error: ", err);
            result(err, null);
        }
        else{
            result(null, res);
        }
    });
};
```

d. Querying all entry from the DB:

```
//query all entry
Employee.findAll = function (result) {
    dbConn.query("Select * from employees", function (err, res) {
        if(err) {
            console.log("error: ", err);
            result(null, err);
        }
        else{
            console.log('employees : ', res);
            result(null, res);
        }
    });
};
```

e. Updating a single entry in the DB with a specific ID:

```
//update an entry
Employee.update = function(id, employee, result){
    dbConn.query("UPDATE employees SET first_name=?,last_name=?,email=?,phone=?,organization=?,designation=?,salary=? WHERE id = ?", [employee.
    first_name,employee.last_name,employee.email,employee.phone,employee.organization,employee.designation,employee.salary, id], function (err,
    res) {
        if(err) {
            console.log("error: ", err);
            result(null, err);
        }else{
            result(null, res);
        }
    });
};
```

f. Removing a specific entry in the DB:

```
//remove an entry
Employee.delete = function(id, result){
    dbConn.query("DELETE FROM employees WHERE id = ?", [id], function (err, res) {
        if(err) {
            console.log("error: ", err);
            result(null, err);
        }
        else{
            result(null, res);
        }
    });
};

module.exports= Employee;
```

9. Add the following code for the employee.controller.js file which should inside the controllers folder.
employee.controller.js – middle man between your model and the route

a. Initialization of the model:

```
'use strict';
const Employee = require('../models/employee.model');
```

b. Controller code for querying all entries:

```
exports.findAll = function(req, res) {
    Employee.findAll(function(err, employee) {
        console.log('controller')
        if (err){
            res.send(err);
        }
        console.log('res', employee);
        res.send({status: 200, data: employee});
    });
};
```

c. Controller code for adding an entry to the DB:

```
exports.create = function(req, res) {
    const new_employee = new Employee(req.body);
    //handles null error
    if(req.body.constructor === Object && Object.keys(req.body).length === 0){
        res.status(400).send({ error:true, message: 'Please provide all required field' });
    }else{
        Employee.create(new_employee, function(err, employee) {
            if (err){
                res.send(err);
            }
            res.json({error:false, status: 200, message:"Employee added successfully!", data:employee});
        });
    }
};
```

d. Controller code for querying a single entry from the DB using id

```
exports.findById = function(req, res) {
    Employee.findById(req.params.id, function(err, employee) {
        if (err){
            res.send(err);
        }
        res.json({status: 200, data: employee});
    });
};
```

e. Controller code for updating a single entry from the DB:

```
exports.update = function(req, res) {
    if(req.body.constructor === Object && Object.keys(req.body).length === 0){
        res.status(400).send({ error:true, message: 'Please provide all required field' });
    }else{
        Employee.update(req.params.id, new Employee(req.body), function(err, employee) {
            if (err){
                res.send(err);
            }
            res.json({ error:false, message: 'Employee successfully updated', status: 200 });
        });
    }
};
```

f. Controller code for deletion:

```
exports.delete = function(req, res) {
Employee.delete( req.params.id, function(err, employee) {
        if (err){
            res.send(err);
        }
        res.json({ error:false, message: 'Employee successfully deleted', status: 200 });
    });
};
```

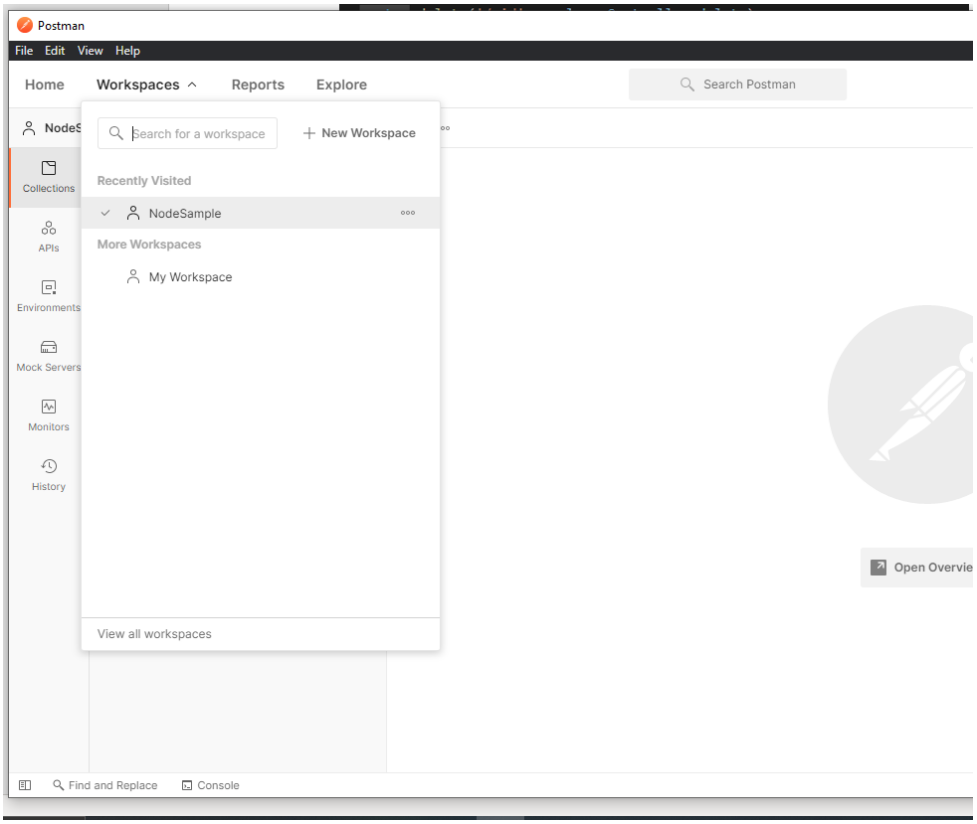
10. Add the following code for the employee.routes.js file which should inside the routes folder.
    employee.routes.js – HTTP mechanism that handles the request made from the view (front-end) to the code that handles the request.

```
const express = require('express')
const router = express.Router()
const employeeController = require('../controllers/employee.controller');
// Retrieve all employees
router.get('/', employeeController.findAll);
// Create a new employee
router.post('/', employeeController.create);
// Retrieve a single employee with id
router.get('/:id', employeeController.findById);
// Update a employee with id
router.put('/:id', employeeController.update);
// Delete a employee with id
router.delete('/:id', employeeController.delete);
module.exports = router
```

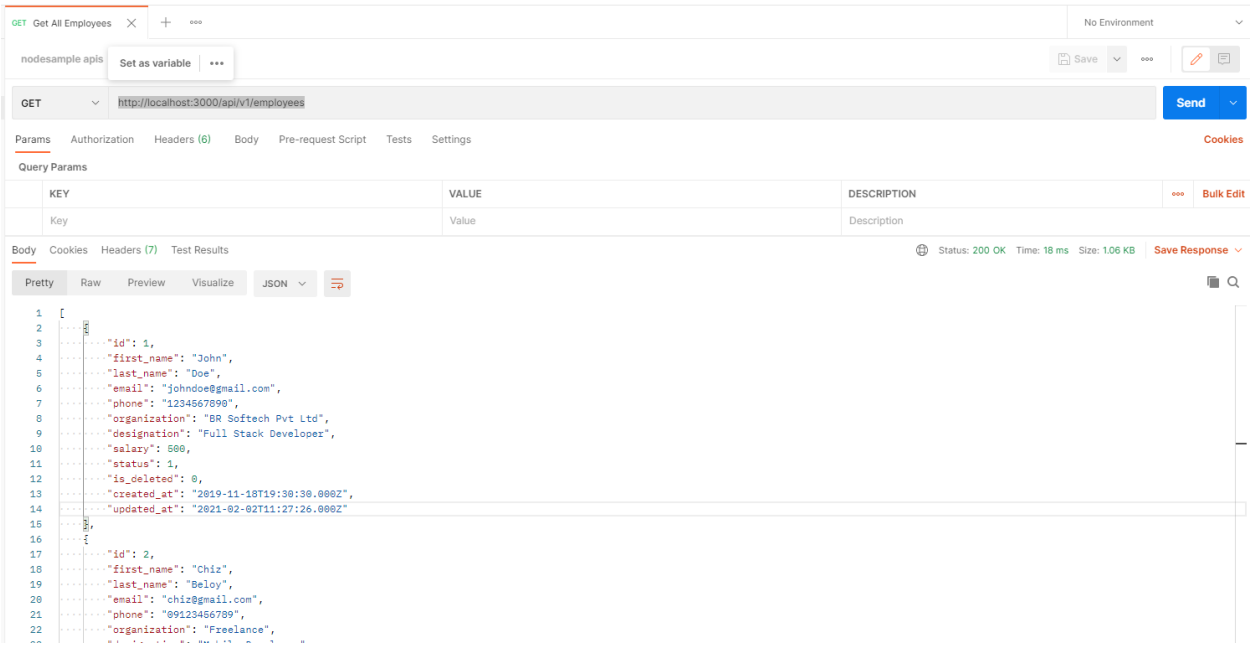
11. Then add these lines of code in your index.js

```
// Require employee routes
const employeeRoutes = require('./src/routes/employee.routes')
// using as middleware
app.use('/api/v1/employees', employeeRoutes)
// listen for requests
app.listen(port, () => {
  console.log(`Server is listening on port ${port}`);
});
```

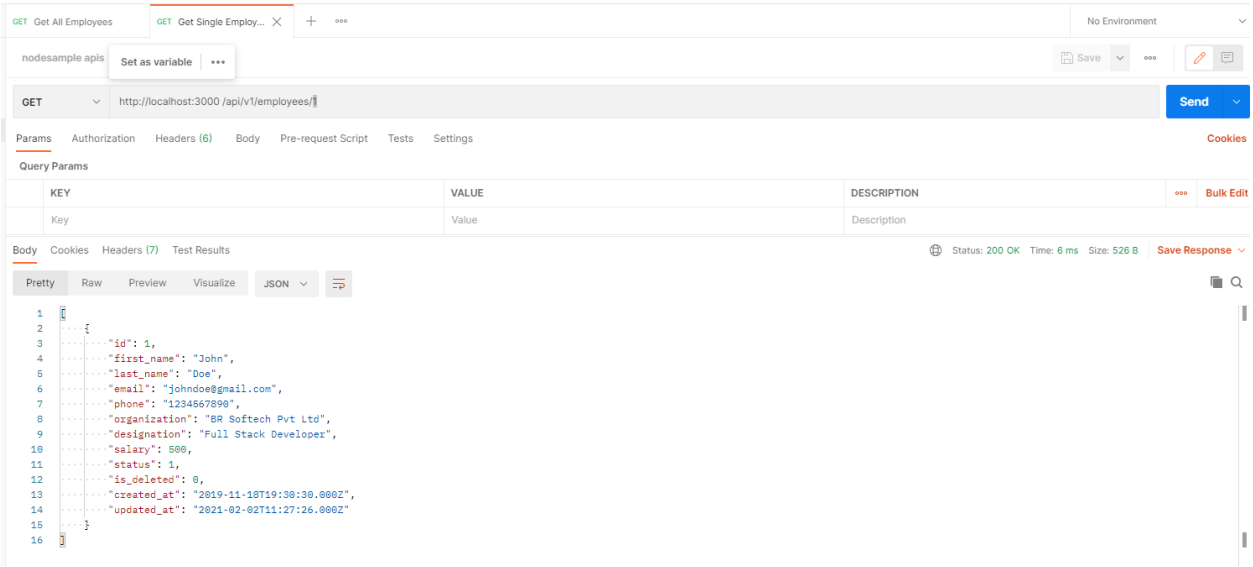
12. To test out our recently created APIs using NodeJS, you can use Postman (https://www.postman.com/downloads/).
13. After downloading and installing the tool, sign up and create an account. Once finished sign in and open the application.
14. Once in the application, go to the Workspaces tab and add a new workspace.



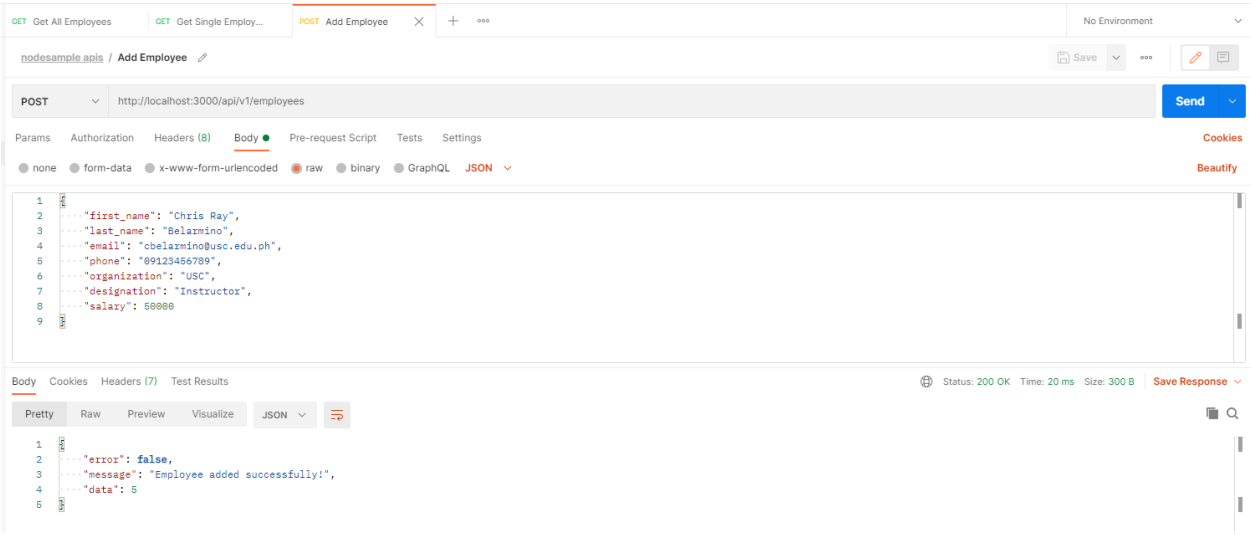
15. To test the APIs, just specify the method then input your **host:port/route**



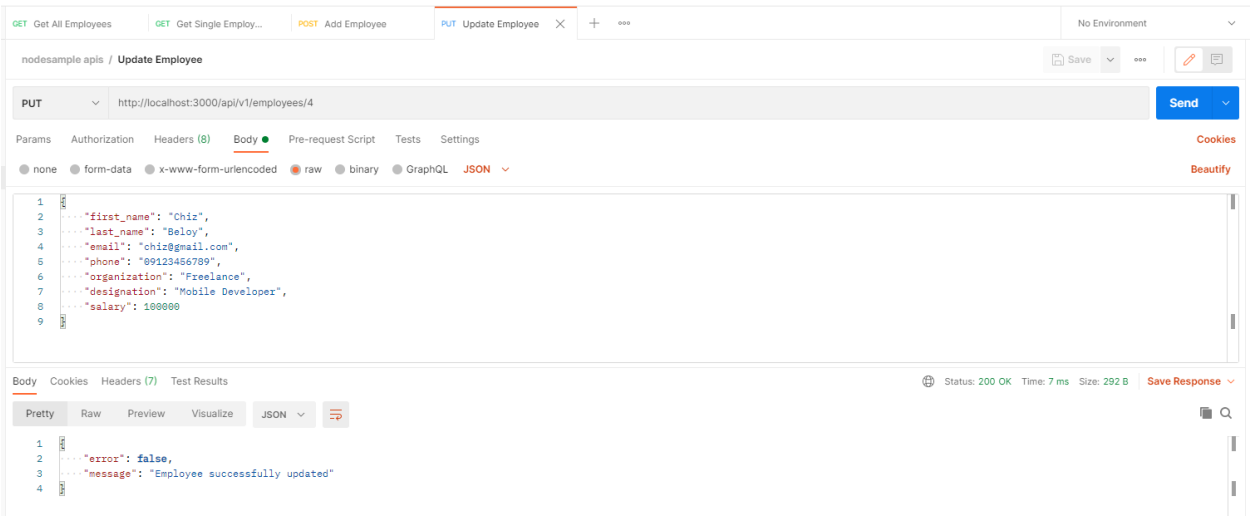
GET: QUERY ALL EMPLOYEES



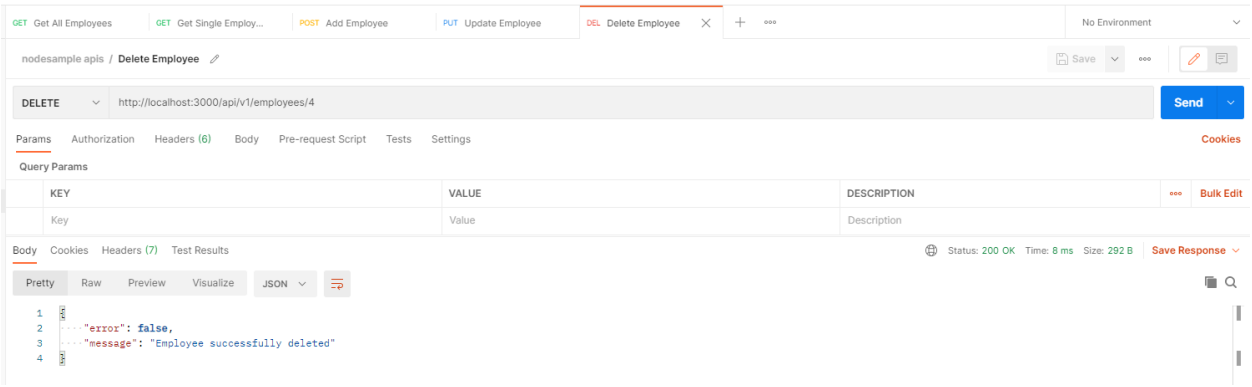
GET: QUERY SINGLE EMPLOYEE



POST: ADD EMPLOYEE



PUT: UPDATE EMPLOYEE



DELETE: REMOVE AN EMPLOYEE