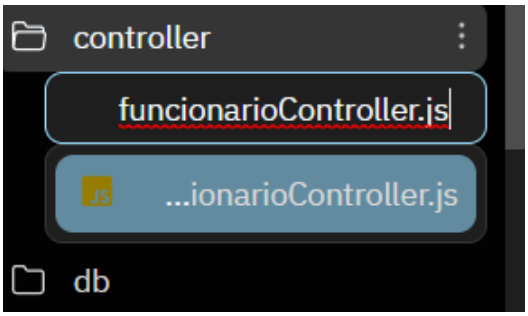
Criando o controller, o create e o listar da API

Vamos criar nesta aula o controller da API.

Crie uma pasta chamada controller e dentro dela um arquivo chamado **funcionarioController.js.**



Dentro do arquivo **funcionarioController.js** vamos codificar nosso create.

Vamos começar colocando o cabeçalho do nosso controller funcionário.

```
const Funcionario = require("../model/funcionarioModel");
```

O próximo passo é criar uma classe chamada funcionarioController.

```
module.exports = class funcionarioController {
```

A classe funcionarioController será responsável por guardar todas as funções referente ao CRUD funcionário.

Criamos o nosso module.exports recebendo a classe ou seja será exportado todo o conteúdo.

Vamos criar agora o o create com o nome de **FuncionarioCreate.**

```
  static async FuncionarioCreate(req,res) {
    let nome = req.body.nome;
    let endereco =  req.body.endereco;
    let telefone =  req.body.telefone;
    let email =  req.body.email;
    let nascimento =  req.body.nascimento;
```

```
  const funcionario = {
    nome: nome,
    endereco: endereco,
    telefone: telefone,
    email: email,
    nascimento: nascimento
  }
  await Funcionario.create(funcionario);
```

Recebemos os valores passados via body para declarações do tipo let. Relembrando que declarações **let** somente são usadas nos blocos que são declaradas.

Criamos uma constante para receber todas as declarações dos valores recebidos e logo após solicitamos a criação no banco com **Funcionario.create.**

Para terminar, enviamos uma mensagem via JSON com a mensagem de cadastro criado com sucesso.

```
res.json({message: "Cadastro realizado com sucesso!"});
```

Vamos agora no nosso arquivo **index.js** e incluir nosso módulo funcionarioController.js.

Arquivo **index.js**

```
//BIBLIOTECAS/MODULOS UTILIZADOS
const database = require("./db/db");
const Funcionario = require("./model/funcionarioModel");
const funcionarioController = require("./controller/funcionarioController");
```

No arquivo index.js vamos incluir nosso app.post para acessar nossa função. **FuncionarioCreate.**

```
app.post("/Cadastrar",funcionarioController.FuncionarioCreate);
```

Vamos incluir a codificação para JSON antes de utilização da constante app.

```
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
```

Nosso index.js ficará a codificação abaixo:

```
const express = require("express");
const app = express();
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

//BIBLIOTECAS/MODULOS UTILIZADOS
const database = require("./db/db");
const Funcionario = require("./model/funcionarioModel");
const funcionarioController = require("./controller/funcionarioController");

//SINCRONISMO COM O BANCO DE DADOS
try {
    database.sync().then(() => {

    })
}
catch(erro) {
    console.log("Houve uma falha ao sincronizar com o banco de dados. ", erro);
};
app.get("/",(req, res) =>{
  return res.json({message: "Olá Mundo!"});
})
app.post("/Cadastrar",funcionarioController.FuncionarioCreate);

app.listen(3000);
```

O caminho da requisição será a URL de execução do seu projeto mais /Cadastrar.

Vamos criar agora o *read* da API.

Dentro do arquivo funcionarioController.js, crie a função **FuncionarioListar**

```
  static async FuncionarioListar(req,res) {
    const funcionario = await Funcionario.findAll({ raw:true });
    res.json(funcionario);
 }
```

Agora vamos no arquivo **index.js** para incluir nossa rota listar.

A rota listar vai ser uma requisição get.

```
//GET - LISTAR
app.get("/Funcionarios",funcionarioController.FuncionarioListar);
```

Esse código basicamente pega todas as requisições GET para a URL e retorna todos os funcionários cadastrados no banco no formato JSON, diretamente no corpo da resposta, que é o esperado para uma API.

Finalizamos a realização do serviço de create e listar da nossa API.

Na próxima aula, você verá os objetos update e delete.

Até lá...