API RESTful com Node.js, Express e MongoDB (CRUD)

Este tutorial mostra como criar uma API RESTful utilizando Node.js, Express e MongoDB. Com ele, você aprenderá a estruturar seu projeto com pastas separadas e a implementar as operações de Create, Read, Update e Delete (CRUD).

# 1. Estrutura de Pastas

seu-projeto/  
├── models/  
│ └── Person.js  
├── controllers/  
│ └── personController.js  
├── routes/  
│ └── personRoutes.js  
├── index.js  
├── package.json

# 2. models/Person.js

const mongoose = require('mongoose')  
  
const Person = mongoose.model('Person', {  
 name: String,  
 salary: Number,  
 approved: Boolean  
})  
  
module.exports = Person

# 3. controllers/personController.js

const Person = require('../models/Person')  
  
// CREATE  
exports.createPerson = async (req, res) => {  
 const { name, salary, approved } = req.body  
  
 try {  
 const person = await Person.create({ name, salary, approved })  
 res.status(201).json({ message: 'Pessoa criada com sucesso', person })  
 } catch (error) {  
 res.status(500).json({ erro: error.message })  
 }  
}  
  
// READ ALL  
exports.getAllPeople = async (req, res) => {  
 try {  
 const people = await Person.find()  
 res.status(200).json(people)  
 } catch (error) {  
 res.status(500).json({ erro: error.message })  
 }  
}  
  
// READ BY ID  
exports.getPersonById = async (req, res) => {  
 const { id } = req.params  
 try {  
 const person = await Person.findById(id)  
 if (!person) return res.status(404).json({ message: 'Pessoa não encontrada' })  
 res.status(200).json(person)  
 } catch (error) {  
 res.status(500).json({ erro: error.message })  
 }  
}  
  
// UPDATE  
exports.updatePerson = async (req, res) => {  
 const { id } = req.params  
 const { name, salary, approved } = req.body  
  
 try {  
 const updated = await Person.updateOne({ \_id: id }, { name, salary, approved })  
 if (updated.matchedCount === 0) return res.status(404).json({ message: 'Pessoa não encontrada' })  
 res.status(200).json({ message: 'Pessoa atualizada com sucesso' })  
 } catch (error) {  
 res.status(500).json({ erro: error.message })  
 }  
}  
  
// DELETE  
exports.deletePerson = async (req, res) => {  
 const { id } = req.params  
 try {  
 const deleted = await Person.deleteOne({ \_id: id })  
 if (deleted.deletedCount === 0) return res.status(404).json({ message: 'Pessoa não encontrada' })  
 res.status(200).json({ message: 'Pessoa removida com sucesso' })  
 } catch (error) {  
 res.status(500).json({ erro: error.message })  
 }  
}

# 4. routes/personRoutes.js

const express = require('express')  
const router = express.Router()  
const controller = require('../controllers/personController')  
  
// Rotas  
router.post('/', controller.createPerson)  
router.get('/', controller.getAllPeople)  
router.get('/:id', controller.getPersonById)  
router.put('/:id', controller.updatePerson)  
router.delete('/:id', controller.deletePerson)  
  
module.exports = router

# 5. index.js

const express = require('express')  
const mongoose = require('mongoose')  
const app = express()  
  
// Middlewares  
app.use(express.urlencoded({ extended: true }))  
app.use(express.json())  
  
// Rotas  
const personRoutes = require('./routes/personRoutes')  
app.use('/person', personRoutes)  
  
// Rota inicial  
app.get('/', (req, res) => {  
 res.json({ message: 'API rodando com sucesso!' })  
})  
  
// Conexão com o Mongo  
mongoose.connect('mongodb://localhost:27017/ARQUIVO')  
 .then(() => {  
 console.log('Conectado ao Banco!')  
 app.listen(3000)  
 })  
 .catch((err) => console.log(err))