

Desarrollo de API con Node y Express

Valeria Ivaniuk

En esta parte de código llamo la función que he definido en el archivo "api" y guardo los datos que me trae en una variable dataApi.

Es un proceso asíncrono y node.js espera una promesa, por lo tanto se hace .then y .catch.

POSIBLE MEJORA: probar otras maneras de devolver la promesa.

```
JS router.js x JS api.js {} package.json
src > routes > JS router.js > app.post('/products/index') callback > newProduct > id
1 const express = require('express');
2 const productRouter = express.Router();
3 const getDataAPI = require('../api');
4 const mongoose = require('mongoose');
5 require('dotenv').config();
6 const axios = require('axios');
7 let dataAPI;
8
9 //En esta parte de código llamo la función que he definido en el archivo "api" y guardo los datos que me trae en una variable dataApi.
10 //Es un proceso asincrono y node.js espera una promesa, por lo tanto se hace .then y .catch.
11 // POSIBLE MEJORA: probar otras maneras de devolver la promesa.
12 getDataAPI()
13 .then(data => {
14   dataAPI = data;
15 })
16 .catch(error => {
17   console.error(error.message);
18 })
```

En este archivo llamo a la API externa para obtener todos los datos que trae.

Uso axios por el tipo de la versión de node.js que tengo instalada.

Esta separado y guardado en otro archivo en el intento de mejorar el directorio.

```
JS router.js JS api.js x {} package.json
src > JS api.js > ...
1 //En este archivo llamo a la API externa para obtener todos los datos que trae.
2 //Uso axios por el tipo de la version de node.js que tengo instalada.
3 //Esta separado y guardado en otro archivo en el intento de mejorar el directorio.
4 const axios = require('axios');
5
6 const getDataAPI = async () => {
7   try {
8     const response = await axios.get('https://dummyjson.com/products');
9     const data = response.data;
10    return data;
11  } catch (error) {
12    console.log('Error:', error);
13  }
14 };
15
16
17
18 module.exports = getDataAPI;
```

```

const app = express();
const port = 3000;
app.use(express.json())

//Conecto mi base de datos precreada en Mongo Atlas.
//POSIBLE MEJORA: Mejoraria que no se viera la contrasena. Aunque haya creado .env no me habia funcionado
//y no pude ocultar mi contrasena.
mongoose.connect("mongodb+srv://valeria1994:1234@curso-mongo-ue.os3hzb.mongodb.net/mydatabase?retryWrites=true&w=majority")
    .then(() => console.log("Connected to MongoDB"))
    .catch((error) => console.log(error));

//En esta parte del codigo creo el schema que representa el esquema de mi base de datos.
//POSIBLE MEJORA: Mejoraria el directorio de los archivos con carpetas determinadas para hacer mas limpio
// y organizado el codigo (ej:una carpeta para todos los modelos)
const productsSchema = mongoose.Schema ({
  id: { type: Number },
  title: { type: String },
  price: { type: Number },
  category: { type: String }
})

const Products = mongoose.model ('Products', productsSchema)

```

```

//Esta parte del codigo sirve para hacer un post de los datos que me invento a la base de datos. Agregamos el control de errores.
app.post('/products', async (req,res) => {
  try{
    newProduct = Products(req.body)
    await newProduct.save();
    res.status(201).json(newProduct);
  } catch(error){
    res.status(500).json({ error: err.message });
  }
});

```

aquí elegimos un elemento de la API externa y lo guardamos en nuestra base de datos. Agregamos el control de errores.

Esto me ayuda a controlar que el número del producto en los params no sea ni menor que 0 ni mayor que el número de los productos que nos de la API.

Inicializo el Products accediendo a los atributos del elemento del dataAPI.

```

// Aquí elegimos un elemento de la API externa y lo guardamos en nuestra base de datos.Agregamos el control de errores.
app.post('/products/:index', async (req, res) => {
  try {
    let {index} = req.params;

    index = parseInt(index);

    //Esto me ayuda a controlar que el numero del producto en los params no sea ni menor que 0 ni mayor que el numero de los productos que nos de la API
    if (index < 0 || index >= dataAPI.products.length) {
      return res.status(400).json({ error: 'Invalid array index' });
    }

    //Inicializo el Products accediendo a los atributos del elemento del dataAPI.
    const newProduct = new Products({ id: dataAPI.products[index].id,title:dataAPI.products[index].title,price: dataAPI.products[index].price, category: dataAPI.products[index].category });
    await newProduct.save();
    res.status(201).json(newProduct);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

```

```
//Esta parte del codigo sirve para traerme todos los datos que estan en mi base de datos. Agregamos el control de errores.
app.get('/products', async (request,response) => {
  try{
    const products = await Products.find();
    res.status(200).json(products);
  } catch(error){
    res.status(500).json({ error: err.message });
  }
});
```

```
//Esta parte del codigo sirve para modificar todos los datos que estan en mi base de datos dependiendo del id del objeto.
//Agregamos el control de errores.
app.put('/products/:id', async (req,res) => {
  try{
    const { id } = req.params;
    const {title, price, category} = req.body
    const products = await Products.updateOne({id: id}, { $set: {title: title, price:price, category:category}});
    res.status(200).json(products);
  } catch(error){
    res.status(500).json({ error: err.message });
  }
});
```

```
//Esta parte del codigo sirve para borrar datos que estan en mi base de datos. Agregamos el control de errores.
app.delete('/products/:id', async (req,res) => {
  try{
    const { id } = req.params;
    const products = await Products.deleteOne({id: id});
    res.status(200).json(products);
  } catch(error){
    res.status(500).json({ error: err.message });
  }
});
```