Hank Erubiel Cárdenas Zamora

2-1

CRUD articulos

Modelo de Datos (articulos.js):

```
const Sequelize = require('sequelize');

const sequelize = new Sequelize({
    dialect: 'sqlite',
    storage: './articulos.sqlite'
});
```

Este código define la estructura de la tabla de artículos:

- id: Clave primaria autoincremental
- descripción: Campo de texto para la descripción del artículo
- precio: Campo numérico decimal para el precio
- existencia: Campo numérico entero para el stock
- timestamps: false: Desactiva los campos de fecha de creación/actualización automáticos

Conexión a la Base de Datos (conexión articulos.js):

```
const Sequelize = require('sequelize');

const sequelize = new Sequelize({
    dialect: 'sqlite',
    storage: './articulos.sqlite'
});
```

Este código configura la conexión a la base de datos SQLite:

- Usa Sequelize como ORM
- Especifica que se usará SQLite como motor de base de datos
- Define la ubicación del archivo de base de datos

API REST (CRUD articulos.js):

```
const express = require('express');
const bodyParser = require('body-parser');
const articulos = require('./articulos');
const puerto = 3000;

const app = express();
app.use(bodyParser.json())
```

Configuración inicial del servidor Express:

- Importa las dependencias necesarias
- Configura el middleware para procesar JSON
- Define el puerto del servidor

Endpoints CRUD:

Crear (POST):

```
app.post('/articulos', async (req, res) => {
    const { descripcion, precio, existencia } = req.body;
    const data = await articulos.create({
        descripcion, precio, existencia
    });
    res.send(data);
});
```

Recibe los datos del artículo en el cuerpo de la petición

Crea un nuevo registro en la base de datos

Devuelve el artículo creado

Leer (GET):

```
app.get('/articulos', async (req, res) => {
    const data = await articulos.findAll();
    res.send(data);
});
```

- Obtiene todos los artículos de la base de datos
- Devuelve la lista completa

Actualizar (PUT):

```
app.put('/articulos/:id', async (req, res) => {
    const { descripcion, precio, existencia } = req.body;
    const { id } = req.params;
    const data = articulos.update({
        descripcion, precio, existencia
    }, {
        where: {
            id
        }
    })
    res.send(data);
});
```

- Recibe el ID en la URL y los nuevos datos en el cuerpo
- Actualiza el artículo específico
- Devuelve el resultado de la actualización

Eliminar (DELETE):

```
app.delete('/articulos/:id', async (req, res) => {
    const { id } = req.params;
    const data = await articulos.destroy({
        where: {
            id
        }
    })
    res.send(data);
});
```

- Recibe el ID del artículo a eliminar en la URL
- Elimina el registro de la base de datos
- Devuelve el resultado de la eliminación