API REST con Node.js y Express

Problema: **Contexto del Problema**

Usted es un desarrollador de software en una empresa que necesita una API para gestionar el inventario de productos en su almacén. La API debe permitir a los usuarios agregar nuevos productos, consultar la lista de productos disponibles y actualizar la información de productos existentes. Su tarea es modificar el ejemplo de API proporcionado anteriormente para adaptarlo a este nuevo caso de uso.

Requisitos

1. **Estructura de Datos del Producto**:
    - Cada producto debe tener los siguientes atributos:
        - `id` (string): Identificador único del producto.
        - `name` (string): Nombre del producto.
        - `price` (number): Precio del producto.
2. **Endpoints de la API**:

    GET /products: Devuelve la lista de productos en el inventario.
    POST /products: Agrega un nuevo producto al inventario.
    PUT /products/:id: Actualiza la información de un producto existente en el inventario.

Información previa

¿Qué es API REST?

Una API REST es un estilo arquitectónico que aprovecha el protocolo HTTP para facilitar el intercambio de datos entre aplicaciones cliente y servidor. Se basa en un conjunto de restricciones y principios, que incluyen una interfaz uniforme, comunicación sin estado y el uso de métodos HTTP estándar (GET, POST, PUT, DELETE) para realizar operaciones en recursos.

¿Qué es Node.js?

Node.js, un entorno de ejecución de JavaScript del lado del servidor, y Express, un framework popular de Node.js, entre los dos forman una excelente combinación para crear API RESTful. En esta práctica, exploraremos el proceso de creación de una API REST usando Node.js y Express, paso a paso.

Paso 1. Configurar entorno e instalar dependencias

Crear el Proyecto:

Abre tu terminal y crea un nuevo directorio para tu proyecto.

```
mkdir user-api0
cd user-api
```

Inicializar el Proyecto:

Inicializa un nuevo proyecto de Node.js.

```
npm init -y
```

Instalar dependencias:

Necesitamos un par de paquetes para que nuestra API REST esté en funcionamiento:

- **Express:** el marco de aplicación web para Node.js.
- **Body-parser**: un middleware para analizar los cuerpos de las solicitudes entrantes.
- **Nodemon (opcional):** una herramienta que ayuda a reiniciar automáticamente el servidor durante el desarrollo.

```
npm install express body-parser
```

Paso 2: Configurar el Servidor Express

Vamos crear una aplicación Express y configuremos un servidor básico. Crea un archivo llamado `app.js` en el directorio raíz del proyecto. Usa el siguiente código

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.json());

const port = process.env.PORT || 3000;
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Aquí, creamos una aplicación Express básica, agregamos el middleware del analizador corporal para analizar datos JSON e iniciamos un servidor en el puerto 3000.

Definir rutas: En una API REST, las rutas definen los puntos finales para diferentes métodos HTTP (GET, POST, PUT, DELETE). Creemos un ejemplo simple con una solicitud GET. Agregar el siguiente código al archivo `app.js`

```
app.get('/api/hello', (req, res) => {
  res.json({ message: 'Hola, somos Universidad ICESI!' });
});
```

Ejecute su API: Puedes ejecutar tu API usando Node.js. En el power Shell ejecutar:

```
node app.js
```

Ahora, en tu navegador accede a http://localhost:3000/api/hello, debería ver el mensaje "`Hola, somos Universidad ICESI!`

Ejercicio 1: implementa la siguiente api. Crear una nueva carpeta por fuera del proyecto actual. Llamarla `user-api1`. Crear el archivo `app.js` con el siguiente código

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const port = 3000;

// Middleware
app.use(bodyParser.json());

// In-memory user list
let users = [];

// Start the server
app.listen(port, () => {
  console.log(`Server is running on
http://localhost:${port}`);
});
```

Implementar Métodos GET, POST y PUT

GET: Consultar la Lista de Usuarios:

```
// Endpoint to get the list of users
app.get('/users', (req, res) => {
  res.json(users);
});
```

POST: Agregar un Usuario a la Lista:

```
// Endpoint to add a new user

app.post('/users', (req, res) => {
  const user = req.body;

  if (user && user.name && user.email) {
    users.push(user);
    res.status(201).json({ message: 'User added successfully', user });
  } else {
    res.status(400).json({ message: 'Invalid user data' });
  }
});
```

PUT: Actualizar un Usuario Existente:

```
// Endpoint to update an existing user
app.put('/users/:email', (req, res) => {
  const email = req.params.email;
  const updatedUser = req.body;

  let userIndex = users.findIndex(user => user.email === email);

  if (userIndex !== -1) {
    users[userIndex] = { ...users[userIndex], ...updatedUser };
    res.json({ message: 'User updated successfully', user:
users[userIndex] });
  } else {
    res.status(404).json({ message: 'User not found' });
  }
});
```

Probar la API: Iniciar el Servidor, En la terminal, inicia el servidor.

```
node app.js
```

Probar los Endpoints:

- Utiliza herramientas como **Postman** o **el navegador web** para probar los endpoints.

Instalar Postman

1. **Descargar e Instalar Postman:**
   - Ve a la página oficial de Postman y descarga la aplicación para tu sistema operativo.
   - Sigue las instrucciones de instalación.

Probar los Endpoints de la API

Usar el siguiente video de referencia: https://www.youtube.com/watch?v=FQAQO90LoQU

1. GET: Consultar la Lista de Usuarios

Crear una Nueva Solicitud:

Haz clic en "New" y selecciona "Request".
Asigna un nombre a la solicitud, por ejemplo, "Get Users".
Guarda la solicitud en la colección que creaste anteriormente.

Configurar la Solicitud GET:

Selecciona el método `GET` en el desplegable.
Ingresa la URL de la API: `http://localhost:3000/users`.
Haz clic en "Send".

Verificar la Respuesta:

En el panel de respuesta, deberías ver una lista vacía (si no hay usuarios) o la lista de usuarios que hayas agregado.

2. POST: Agregar un Usuario a la Lista

1. **Crear una Nueva Solicitud:**
   Haz clic en "New" y selecciona "Request".
   Asigna un nombre a la solicitud, por ejemplo, "Add User".
   Guarda la solicitud en la colección que creaste anteriormente.
2. **Configurar la Solicitud POST:**
   Selecciona el método `POST` en el desplegable.
   Ingresa la URL de la API: `http://localhost:3000/users`.
   Ve a la pestaña "Body", selecciona "raw" y luego "JSON" en el desplegable.
   Ingresa el siguiente JSON en el cuerpo de la solicitud:

```
{
  "name": "John Doe",
  "email": "john@example.com"
}
```

   Haz clic en "Send".
2. **Verificar la Respuesta:**
   Deberías ver una respuesta con el mensaje "User added successfully" y los detalles del usuario agregado.

3. PUT: Actualizar un Usuario Existente

1. **Crear una Nueva Solicitud:**
   Haz clic en "New" y selecciona "Request".
   Asigna un nombre a la solicitud, por ejemplo, "Update User".
   Guarda la solicitud en la colección que creaste anteriormente.
2. **Configurar la Solicitud PUT:**
   Selecciona el método `PUT` en el desplegable.
   Ingresa la URL de la API:
   `http://localhost:3000/users/john@example.com`.
   Ve a la pestaña "Body", selecciona "raw" y luego "JSON" en el desplegable.
   Ingresa el siguiente JSON en el cuerpo de la solicitud para actualizar el usuario:
   `json`

```
{
  "name": "Johnathan Doe"
}
```

   Haz clic en "Send".

Verificar la Respuesta: Deberías ver una respuesta con el mensaje "User updated successfully" y los detalles del usuario actualizado.