Materia:

DISEÑO ELECTRÓNICO BASADO EN SISTEMAS EMBEBIDOS

Alumno:

Posadas Pérez Isaac Sayeg

Paniagua Rico Juan Julian

García Azzúa Jorge Roberto

Grado y grupo:

8°G

Profesor:

Garcia Ruiz Alejandro Humberto

Unidad 4 -Tarea 14:

Estructura base de una API Restful con Node JS

Estructura base de una API RESTful con Node.js

Introducción

Una **API RESTful** es una interfaz que permite la comunicación entre sistemas utilizando el protocolo HTTP y siguiendo los principios de la arquitectura REST. Node.js, junto con el framework **Express**, es una combinación ideal para construir este tipo de APIs por su rendimiento, escalabilidad y simplicidad. Definir una estructura base clara y modular es fundamental para asegurar la mantenibilidad y evolución del proyecto.

Desarrollo

Principios básicos de una API RESTful

- **Uso de métodos HTTP** (GET, POST, PUT, DELETE).
- **Recursos identificados por URLs** (`/usuarios`, `/productos`).
- **Formato estándar de intercambio** (generalmente JSON).
- **Stateless** (el servidor no guarda estado entre peticiones).
- **Respuestas con códigos HTTP** adecuados.

Componentes comunes de una estructura base

1. **`app.js`** o `server.js`: Punto de entrada principal.
2. **`routes/`**: Archivos donde se definen las rutas (endpoints).
3. **`controllers/`**: Lógica de negocio asociada a cada ruta.
4. **`models/`**: Modelos de datos y conexión a la base de datos.
5. **`middlewares/`**: Funciones de validación, autenticación, manejo de errores, etc.
6. **`config/`**: Variables de entorno y configuración general.

7. **`package.json`**: Archivo de configuración del proyecto y dependencias.

Ejemplo de estructura base mínima

```
mi-api-rest/
|
├── app.js
├── routes/
|   └── usuarios.js
├── controllers/
|   └── usuarioController.js
├── models/
|   └── usuario.js
├── config/
|   └── db.js
├── middlewares/
|   └── auth.js
└── package.json
```

1. app.js

```
const express = require('express');

const app = express();

const usuariosRoutes = require('./routes/usuarios');
```

```
app.use(express.json());

app.use('/api/usuarios', usuariosRoutes);


const PORT = process.env.PORT || 3000;

app.listen(PORT, () => console.log(`Servidor en puerto
${PORT}`));
```

2. routes/usuarios.js

```
const express = require('express');

const router = express.Router();

const usuarioController =
require('../controllers/usuarioController');


router.get('/', usuarioController.listarUsuarios);

router.post('/', usuarioController.crearUsuario);


module.exports = router;
```

3. controllers/usuarioController.js

```
exports.listarUsuarios = (req, res) => {

  res.json([{ id: 1, nombre: 'Ana' }, { id: 2, nombre: 'Luis' }]);

};


exports.crearUsuario = (req, res) => {

  const { nombre } = req.body;

  res.status(201).json({ mensaje: `Usuario ${nombre} creado.` });

};
```

4. models/usuario.js

(Este puede contener lógica para conexión con una base de datos como MongoDB o SQL. Aquí se omite para simplificar.)

5. middlewares/auth.js

```
module.exports = function (req, res, next) {

  const token = req.headers['authorization'];

  if (!token) return res.status(403).json({ error: 'No autorizado' });
```

```
  next();

};
```

Conclusión

Diseñar una **estructura base clara y modular** para una API RESTful en Node.js permite desarrollar sistemas mantenibles, escalables y organizados. Separar la lógica en rutas, controladores y modelos promueve buenas prácticas y facilita la integración con bases de datos, validaciones, seguridad y servicios externos. Esta base puede crecer hacia una arquitectura más compleja con microservicios, autenticación JWT, tests automatizados y despliegue en la nube.

Bibliografía

1. Express.js Documentation. https://expressjs.com
2. Node.js Official Docs. https://nodejs.org
3. Postman Blog. *Best practices for REST API design*
4. Mozilla MDN Web Docs – *HTTP overview*. https://developer.mozilla.org
5. Richardson, L. & Ruby, S. (2007). *RESTful Web Services*. O'Reilly Media.