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 15:

Peticiones HTTP con la API en Node.js

Peticiones HTTP con la API en Node.js

Introducción

En una API RESTful construida con **Node.js y Express**, las **peticiones HTTP** son la forma estándar de interacción entre el cliente y el servidor. Estas peticiones permiten crear, leer, actualizar y eliminar recursos (operaciones CRUD). Cada tipo de petición HTTP tiene un propósito específico y debe implementarse correctamente para cumplir con las buenas prácticas del diseño REST.

Desarrollo

Métodos HTTP principales

Método	**Función**	**Ejemplo en API REST**
GET	Obtener datos	`GET /api/usuarios`
POST	Crear un nuevo recurso	`POST /api/usuarios`
PUT	Actualizar un recurso completo	`PUT /api/usuarios/1`
PATCH	Actualizar parcialmente	`PATCH /api/usuarios/1`
DELETE	Eliminar un recurso	`DELETE /api/usuarios/1`

Códigos de estado HTTP comunes

Código	Significado
200	OK
201	Recurso creado
204	Sin contenido
400	Solicitud incorrecta
401	No autorizado
403	Prohibido
404	No encontrado
500	Error interno del servidor

Ejemplo de implementación

Supongamos una API que gestiona usuarios:

1. `routes/usuarios.js`

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

const router = express.Router();

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


router.get('/', controlador.obtenerUsuarios);          // GET

router.get('/:id', controlador.obtenerUsuario);         //
GET por ID

router.post('/', controlador.crearUsuario);             //
POST

router.put('/:id', controlador.actualizarUsuario);      //
PUT

router.delete('/:id', controlador.eliminarUsuario);     //
DELETE


module.exports = router;
```

2. controllers/usuarioController.js

```
let usuarios = [{ id: 1, nombre: 'Ana' }, { id: 2, nombre:
'Luis' }];
```

```
exports.obtenerUsuarios = (req, res) => {
  res.status(200).json(usuarios);
};

exports.obtenerUsuario = (req, res) => {
  const usuario = usuarios.find(u => u.id == req.params.id);
  if (!usuario) return res.status(404).json({ error: 'No
encontrado' });
  res.status(200).json(usuario);
};

exports.crearUsuario = (req, res) => {
  const nuevo = { id: Date.now(), ...req.body };
  usuarios.push(nuevo);
  res.status(201).json(nuevo);
};

exports.actualizarUsuario = (req, res) => {
  const index = usuarios.findIndex(u => u.id ==
req.params.id);
  if (index === -1) return res.status(404).json({ error: 'No
encontrado' });
  usuarios[index] = { ...usuarios[index], ...req.body };
```

```
  res.status(200).json(usuarios[index]);
};


exports.eliminarUsuario = (req, res) => {
  usuarios = usuarios.filter(u => u.id != req.params.id);
  res.status(204).send(); // Sin contenido
};
```

Conclusión

El uso adecuado de las **peticiones HTTP** y sus respectivos códigos de estado es esencial para construir APIs RESTful coherentes, comprensibles y mantenibles. Node.js junto con Express facilita la definición de rutas y controladores para cada tipo de petición, permitiendo desarrollar servicios robustos y eficientes. Esta estructura básica puede ampliarse para incluir validaciones, autenticación, y conexión a bases de datos.

Bibliografía

1. Mozilla MDN Web Docs – *HTTP Methods*. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

2. Express.js Documentation. https://expressjs.com

3. Node.js Official Docs. https://nodejs.org

4. Richardson, L. & Ruby, S. (2007). *RESTful Web Services*. O'Reilly Media.