TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

DESARROLLO WEB

CONFIGURACIÓN DE SERVIDORES CON NODE.JS

CONFIGURACIÓN DE SERVIDORES CON NODE.JS: CREACIÓN Y DESPLIEGUE DE SERVIDORES WEB

La configuración de servidores con Node.js implica establecer un entorno robusto que pueda manejar solicitudes HTTP de manera eficiente, gestionar recursos del sistema adecuadamente y mantener la estabilidad bajo diferentes condiciones de carga. Esta configuración abarca desde la definición de variables de entorno hasta la implementación de clústeres para aprovechar múltiples núcleos del procesador. Node.js proporciona módulos nativos como http, https, clúster y los que facilitan la creación de servidores altamente configurables y optimizados para diferentes escenarios de uso.

El proceso de configuración incluye aspectos críticos como la gestión de puertos, el manejo de variables de entorno, la configuración de CORS (Cross-Origin Resource Sharing), y la implementación de logging para monitoreo y debugging. Cardador Cabello (2015), destaca que las aplicaciones web distribuidas requieren configuraciones específicas que permitan su correcto funcionamiento en diferentes entornos, desde desarrollo local hasta producción en la nube. La configuración adecuada también incluye la definición de límites de memoria, tiempos de espera de conexión y estrategias de manejo de errores que garanticen la resiliencia del servidor.

En el ámbito profesional de la ingeniería de software, la configuración de servidores Node.js se optimiza, según los requisitos específicos de cada aplicación. Por ejemplo, aplicaciones de streaming como Twitch requieren configuraciones especializadas para manejar conexiones WebSocket persistentes y transferencia de datos en tiempo real, mientras que APIs de e-commerce necesitan configuraciones enfocadas en seguridad, validación de datos e integración con sistemas de pago externos.

Ejercicio práctico. Configurar un servidor Node.js completo con clustering y variables de entorno.

Paso 1. Crear archivo server.js: const cluster = require('cluster'); const numCPUs = require('os').cpus().length; const PORT = process.env.PORT || 3000; if (cluster.isMaster) { console.log('Master process iniciado'); for (let i = 0; i < numCPUs; i++) { cluster.fork(); } cluster.on('exit', (worker) => { console.log('Worker terminado, reiniciando...'); cluster.fork(); }); } else { const express = require('express'); const app = express(); app.get('/', (req, res) => res.json({message: 'Servidor activo', worker: process.pid})); app.listen(PORT, () => console.log('Worker iniciado en puerto', PORT)); }.

Paso 2. Crear archivo .env con PORT=8080 y usar npm install dotenv.

Paso 3. Ejecutar con node server.js.

Resultado. Se obtiene un servidor que utiliza todos los núcleos del procesador, con configuración por variables de entorno y recuperación automática de workers fallidos.