Programación IV
Profesora Cynthia Estrada

Fecha: 01/09

"Node JS"

Integrantes:
- "Díaz Rossini Juan José",
- "Gallo Genaro"
- "Navarro Victor Leandro"

Node JS

entorno de ejecución de JavaScript orientado a eventos asíncronos.

Características:

- Código Open Source(código abierto)
- Multiplataforma
- Basado en Motor V8
- Es quien crea los servidores; los desarrolladores crean servidores usando Node.js (módulo http o frameworks). Node provee las APIs.
- Para Backend y Frontend

Arquitectura: Cliente <-> Servidor

Usado para: Desarrollo Backend y APIs (en su estado base)

Ventajas: Permite crear proyectos escalables y en tiempo real

Extra

Fullstack: un desarrollador web que se dedica al frontend y al backend

Protocolo: conjunto de reglas que permite que 2 dispositivos de una red se comuniquen.

Base de Datos: estructurada o no estructurada, relacional o no relacional. En SQL son tablas, donde las columnas son llamadas atributos o campos, y las filas son llamadas registros.

Páginas Web Estáticas y Dinámicas: Estática si no tiene JS, Dinámica si no tiene JS.

"Estática si no tiene JS, Dinámica si tiene JS". La clave real es si el contenido cambia en función de datos/usuario/tiempo, no solo si hay JS.

---

1. **¿Qué es Node JS?**
2. **Explique evento asincrónico y sincrónico**
3. **¿Cuáles son las características de Node JS? Y explique cada una de ellas**
4. **¿Qué tipo de arquitectura maneja Node JS? Diga cómo es el funcionamiento de dicha arquitectura.**
5. **¿Qué es el desarrollo Backend y Frontend? ¿Cómo se llama a los desarrolladores web y que se dedican tanto al frontend como al backend?**
6. **¿Qué es un protocolo?**
7. **Explique lo que es base de datos y su funcionamiento**
8. **¿Quién crea los servidores?**

9. **¿Que es una página web estática y que es una página web dinámica?. ¿Podría dar un ejemplo?**
10. **¿Para qué es usado Node JS?**
11. **¿Qué es una API?**
12. **¿Cómo funciona una API?**
13. **¿Cuáles son las ventajas de usar Node JS? Desarrolle cada una de ellas**
14. **¿Qué no es Node JS?**

---

1) ¿Qué es Node.js?

Node.js es un entorno de ejecución de JavaScript del lado del servidor, orientado a eventos y con E/S no bloqueante (asíncrona). Permite ejecutar JS fuera del navegador, generalmente para crear servidores, APIs y herramientas de backend.

---

2) Evento asíncrono y sincrónico

Sincrónico: las operaciones se ejecutan una detrás de otra. Una tarea se bloquea hasta terminar (p. ej., leer un archivo de manera síncrona detiene el hilo principal hasta obtener el resultado).

Asíncrono (Node.js): se dispara una operación de E/S y el programa sigue. Al finalizar, el sistema emite un evento y ejecuta un callback/Promise/async-await. Esto evita bloquear el event loop, mejora el rendimiento con muchas conexiones y E/S intensiva.

---

3) ¿Cuáles son las características de Node.js? Explique cada una

- **Código abierto (Open Source):** su código y ecosistema (npm) son públicos; cualquiera puede usarlos y contribuir.
- **Multiplataforma:** corre en Windows, Linux y macOS.
- **Basado en motor V8:** usa V8 (de Chrome) para compilar/ejecutar JS a código máquina de alto rendimiento.
- **Orientado a eventos y E/S no bloqueante:** modelo event-driven con libuv y un event loop que gestionan operaciones de red/archivo de forma asíncrona.
- **Single-thread (userland) con pool para E/S:** el hilo principal corre JS; tareas de E/S pesadas se delegan; existen Worker Threads si se necesita CPU intensiva.
- **Ecosistema npm:** gestor de paquetes más grande; facilita agregar librerías/frameworks (Express, NestJS, Fastify, Socket.IO, etc.).
- **Servidor embebido:** incluye el módulo http para crear servidores sin depender de Apache/Nginx (aunque pueden coexistir).

---

4) ¿Qué tipo de arquitectura maneja Node.js? ¿Cómo funciona?

Arquitectura Cliente ↔ Servidor:

1. Un cliente (navegador/app) envía una petición HTTP al servidor Node.js.
2. Node recibe la petición, la encola en el event loop y ejecuta la lógica (rutas, controladores, consultas a BD).
3. Cuando las operaciones asíncronas finalizan, Node envía la respuesta (HTML/JSON/archivo/etc.).
4. Este modelo permite manejar miles de conexiones concurrentes con bajo consumo.

---

5) Backend y Frontend. ¿Cómo se llama al que hace ambos?

- **Frontend:** parte visible/interactiva en el cliente (HTML, CSS, JS/Frameworks como React/Vue/Angular).
- **Backend:** lógica/negocio en el servidor (APIs, autenticación, acceso a BD). En Node: Express/NestJS/Fastify.
- **Fullstack:** desarrollador que trabaja tanto en frontend como en backend.

---

6) ¿Qué es un protocolo?

Conjunto de reglas que permite que dos dispositivos se comuniquen en una red. Ejemplos: HTTP/HTTPS (web), TCP/IP (transporte), WebSocket (tiempo real).

---

7) ¿Qué es una base de datos y cómo funciona?

Un sistema para almacenar y organizar datos y permitir operaciones CRUD (crear, leer, actualizar, borrar):

- **Relacionales (SQL):** datos en tablas; columnas = atributos/campos; filas = registros; usan SQL (MySQL, PostgreSQL).
- **No relacionales (NoSQL):** documentos, clave-valor, grafos, etc. (MongoDB, Redis). Node se conecta vía drivers o ORM/ODM (p. ej., Prisma, Sequelize, Mongoose).

---

8) ¿Quién crea los servidores?

Los desarrolladores los crean con Node.js. Node ofrece el módulo http y frameworks (Express/NestJS/Fastify) para levantar servidores y definir rutas, middleware y respuestas.

---

9) Páginas web estáticas y dinámicas (con ejemplo)

- **Estática:** contenido fijo pre-generado; el servidor solo sirve archivos (HTML/CSS/imágenes). **Ej.:** landing de empresa hecha con HTML puro.

- **Dinámica:** contenido se genera en tiempo real según datos o interacción (puede usar JS en cliente y/o servidor). **Ej.:** chat en tiempo real con Node + Socket.IO o una página que muestra productos desde una BD.

---

10) ¿Para qué se usa Node.js?

- **Desarrollo backend y APIs REST/GraphQL (su uso base).**
- **Aplicaciones en tiempo real (WebSocket/Socket.IO): chats, notificaciones, colaboración.**
- **Microservicios y gateways.**
- **SSR/Isomórfico (Next.js/Nuxt en ecosistemas JS).**
- **CLI y tooling, procesadores de build, automatización.**
- **Servicios de streaming y colas (streams nativos, Kafka, RabbitMQ con librerías).**

---

11) ¿Qué es una API?

Una Interfaz de Programación de Aplicaciones: un conjunto de reglas y endpoints que permiten que aplicaciones se comuniquen. En la web, una API suele exponer recursos vía HTTP y devolver JSON.

---

12) ¿Cómo funciona una API?

1. **El cliente realiza una petición** (p. ej., GET /usuarios).
2. **El servidor (Node) valida, ejecuta lógica y consulta la BD.**
3. **Devuelve una respuesta** (código de estado + cuerpo JSON/archivo). Incluye conceptos como rutas, métodos HTTP (GET/POST/PUT/DELETE), autenticación/autorización, versionado y manejo de errores.

---

13) ¿Cuáles son las ventajas de usar Node.js? Desarrolle

- **Escalabilidad y Concurrencia:** el modelo asíncrono y el event loop permiten manejar muchas conexiones simultáneas con pocos recursos.
- **Rendimiento en E/S:** V8 + E/S no bloqueante es ideal para apps con red/BD intensiva.
- **Tiempo real sencillo:** WebSocket/Socket.IO se integran naturalmente para notificaciones, chats y colaboración.
- **Un solo lenguaje (JS) en todo el stack:** facilita compartir modelos, tipos y utilidades entre frontend y backend.
- **Ecosistema enorme (npm):** miles de paquetes aceleran el desarrollo.
- **Servidor embebido y flexibilidad:** no requiere Apache/Nginx para empezar; fácil de containerizar y desplegar en microservicios.

---

14) ¿Qué no es Node.js?

- **No es un lenguaje (usa JavaScript).**
- **No es un framework (Express/NestJS/Fastify sí lo son).**

- **No es una base de datos ni un ORM.**
- **No es un navegador (no tiene DOM ni APIs del cliente).**
- **No es un servidor web completo como Apache/Nginx (aunque con http puede implementar uno).**
- **No es automáticamente multi-hilo para CPU intensiva (se puede con Worker Threads/Cluster, pero no es su caso típico).**