

Agencia de
Aprendizaje
a lo largo
de la vida

Desarrollo Fullstack



Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 23

Patrones de Arquitectura

- ▶ ¿Qué son?
- ▶ Tipos de patrones
- ▶ MVC
- ▶ REST

Clase 24

Node JS

- ▶ ¿Qué es?
- ▶ ¿Cómo funciona?
- ▶ Single Thread vs Multi Thread
- ▶ Instalación

Clase 25

Node JS

- ▶ Módulos
- ▶ Node Package Manager
- ▶ Servidor Web Node Nativo
- ▶ Enviar Texto
- ▶ Enviar Archivos

Llegó el momento más esperado.



NODE JS

Introducción



¿Qué es Node JS?

Es un **entorno de ejecución** para JavaScript orientado a eventos asíncronos diseñado para **crear aplicaciones web escalables**, construido con **V8**, el motor de JavaScript de Chrome, escrito en C, C++ y JavaScript.

Nos permite **desarrollar con el lenguaje Javascript** más allá del navegador.



¿Es **NODE JS** un lenguaje de programación?

En una palabra: **NO**.

NODE es un entorno de ejecución que se utiliza para ejecutar JavaScript fuera del navegador.

Tampoco es un framework

El tiempo de ejecución de **NODE** se construye sobre un lenguaje de programación -en este caso, **JavaScript**- y ayuda a la **ejecución de los propios frameworks**.

En resumen, **NODE** no es un lenguaje de programación ni un marco de trabajo, es un entorno para ellos.

Arquitectura de Node

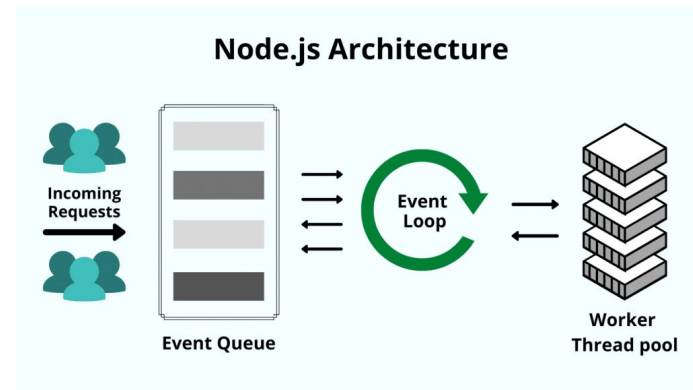
NODE utiliza la arquitectura “*Single Threaded Event Loop*” para manejar múltiples clientes al mismo tiempo, a diferencia de los clientes concurrentes multihilo en lenguajes como **Java**.



Single Thread VS Multi Thread

SINGLE THREAD

1. Mantiene un pool de hilos limitado para atender las peticiones.
2. Cada vez que llega una solicitud, la coloca en una cola.
3. El Event Loop espera las peticiones indefinidamente.
4. Cuando llega una solicitud, el bucle la recoge de la cola y comprueba si requiere una operación de entrada/salida (E/S) de bloqueo. Si no es así, procesa la solicitud y envía una respuesta.
5. Si la solicitud tiene una operación de bloqueo que realizar, el bucle de eventos asigna un hilo del pool de hilos internos para procesar la solicitud. Los hilos internos disponibles son limitados.
6. El Event Loop rastrea las solicitudes que se bloquean y las coloca en la cola una vez que se procesa la tarea que se bloquea. Así es como mantiene su naturaleza no bloqueante.

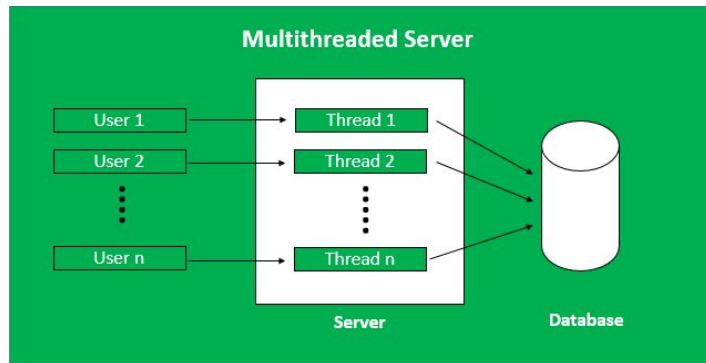


Single Thread VS Multi Thread

MULTITHREAD

En un **modelo de solicitud-respuesta multihilo**, varios clientes envían una solicitud y el **servidor** **procesa cada una de ellas** antes de devolver la respuesta.

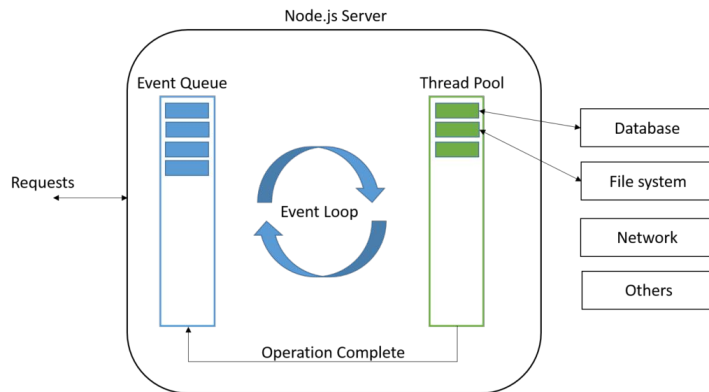
Sin embargo, se utilizan múltiples hilos para procesar las **llamadas concurrentes**. Estos hilos se definen en un pool de hilos, y cada vez que llega una petición, **se asigna un hilo individual** para manejarla.



Arquitectura Node JS

Dado que **NODE** utiliza menos hilos, utiliza **menos recursos/memoria**, lo que resulta en una ejecución más rápida de las tareas. Así que **para nuestros propósitos**, esta arquitectura de un solo hilo **es** equivalente a la arquitectura multihilo.

Cuando uno **necesita procesar** tareas con muchos datos, entonces **tiene mucho más sentido** utilizar **lenguajes multihilo como Java**. Pero para aplicaciones en tiempo real, **Node.js es la opción obvia**.



USOS COMUNES DE **NODE JS**

Aplicaciones de una sola página (SPA)

El Event Loop de Node.js viene al rescate aquí, ya que procesa las solicitudes de forma no bloqueante permitiendo tener peticiones para componentes específicos.

Aplicaciones basadas en REST API

JavaScript se utiliza tanto en el frontend como en el backend de los sitios. Así, un servidor puede comunicarse fácilmente con el frontend a través de APIs REST utilizando Node.js.

Chats en tiempo real

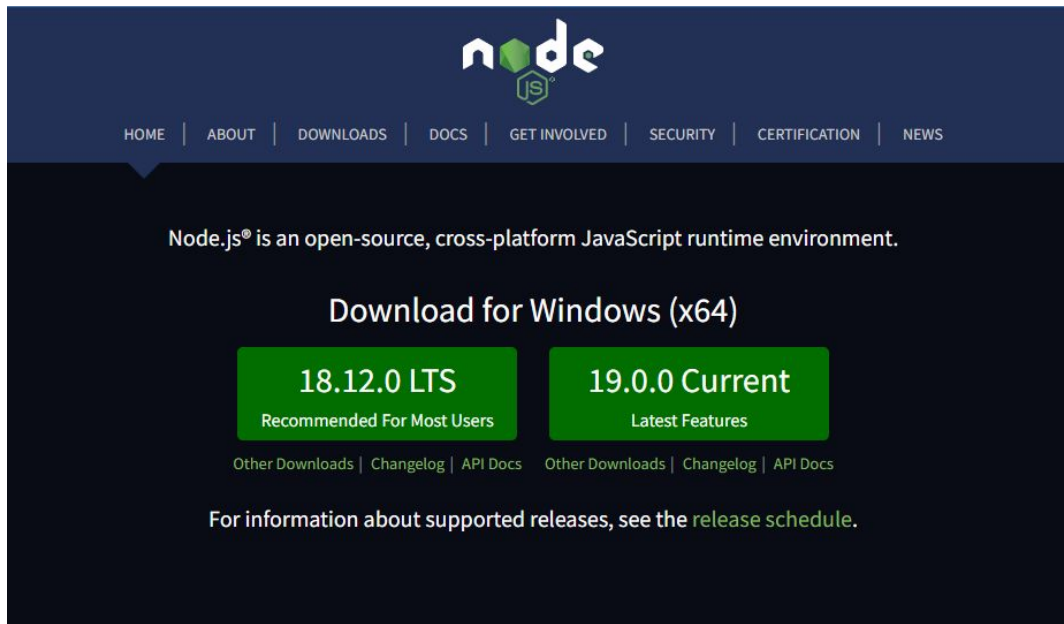
Debido a su naturaleza asíncrona de un solo hilo, es muy adecuado para procesar la comunicación en tiempo real. Se puede escalar fácilmente y se utiliza a menudo en la construcción de chatbots.

Streaming de datos

Empresas como Netflix utilizan NODE para el streaming, esto se debe principalmente a que NODE es ligero y rápido.

Ahora que conocemos cómo funciona y para qué sirve, vamos a instalarlo.

Instalación



Ingresamos a <https://nodejs.org/en/> y descargamos la versión **LTS (long term support)** ya que es la **más reciente y con soporte oficial** recomendada para proyectos “reales” o productivos.

Primeros pasos con NodeJS

Ahora que tenemos **NODE** instalado en nuestra PC podemos trabajar con él del mismo modo que lo hacíamos con **Javascript**.

En esta ocasión para ejecutar nuestro código en lugar de usar la consola del navegador, vamos a usar la terminal de **VS CODE** o de nuestra PC.



```
index.js
1 console.log("Hola Mundo");
```

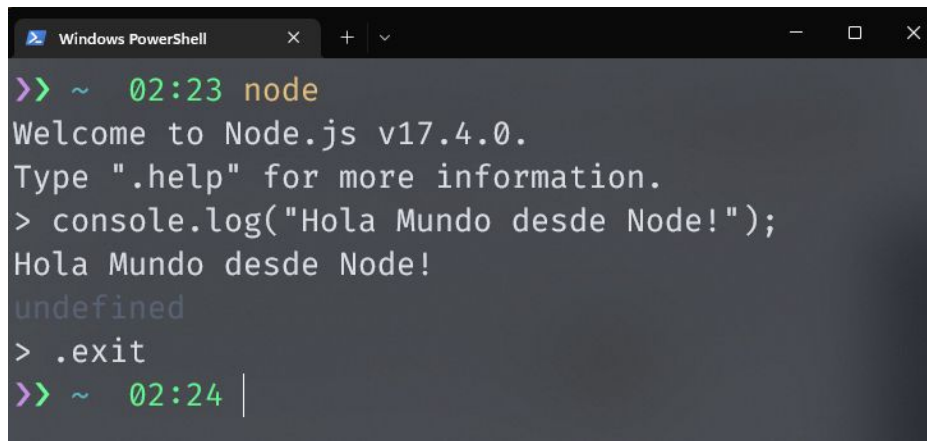
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
● >> node 02:26 node index.js
    Hola Mundo
○ >> node 02:26
```

Primeros pasos con NodeJS

También podemos escribir y ejecutar nuestro código **NODE** a través de la consola mediante el comando `node`.

Para salir de este modo, usamos el comando `.exit`



```
Windows PowerShell
>> ~ 02:23 node
Welcome to Node.js v17.4.0.
Type ".help" for more information.
> console.log("Hola Mundo desde Node!");
Hola Mundo desde Node!
undefined
> .exit
>> ~ 02:24 |
```


Como podemos ver cambia el entorno de ejecución pero mantenemos la misma sintaxis.



No te olvides de dar el presente

Recordá:

- **Revisar la Cartelera de Novedades.**
- **Hacer tus consultas en el Foro.**

Todo en el Aula Virtual.

Gracias