Node JS

Curso práctico

Luciano Pucciarelli

Desde **www.ra-ma.es** podrá descargar material adicional.

Ra-Ma®

ediU®
Ediciones de la U

NODE JS

Curso práctico

Luciano Pucciarelli

Ra-Ma®

edU®

Bogotá - México, D.F.

Pucciarelli, Luciano, *et. al.*

NODE JS. Curso práctico / Luciano Pucciarelli, --. Bogotá: Ediciones de la U, 2024
270 p. ; 24 cm
ISBN 978-958-792-641-5 e-ISBN 978-958-792-642-2
1. Informática 2. Sistema de archivos 3. Servidor web 4. Base de datos I. Tít.
519.7 ed.

Edición original publicada por © Editorial Ra-ma (España)
Edición autorizada a Ediciones de la U para Colombia

Área: Sistemas e informática
Primera edición: Bogotá, Colombia, enero de 2024
ISBN. 978-958-792-641-5

© Luciano Pucciarelli
© Ra-ma Editorial. Calle Jarama, 3-A (Polígono Industrial Igarsa) 28860 Paracuellos de Jarama
www.ra-ma.es y www.ra-ma.com / E-mail: editorial @ra-ma.com
Madrid, España
© Ediciones de la U - Carrera 27 #27-43 - Tel. (+57) 601 6455049
www.edicionesdelau.com - E-mail: editor@edicionesdelau.com
Bogotá, Colombia

Ediciones de la U es una empresa editorial que, con una visión moderna y estratégica de las tecnologías, desarrolla, promueve, distribuye y comercializa contenidos, herramientas de formación, libros técnicos y profesionales, e-books, e-learning o aprendizaje en línea, realizados por autores con amplia experiencia en las diferentes áreas profesionales e investigativas, para brindar a nuestros usuarios soluciones útiles y prácticas que contribuyan al dominio de sus campos de trabajo y a su mejor desempeño en un mundo global, cambiante y cada vez más competitivo.

Coordinación editorial: Adriana Gutiérrez M.
Carátula: Ediciones de la U
Impresión: DGP Editores SAS
Calle 63 #70D-34, Pbx (+57) 601 7217756

Impreso y hecho en Colombia
Printed and made in Colombia

No está permitida la reproducción total o parcial de este libro, ni su tratamiento informático, ni la transmisión de ninguna forma o por cualquier medio, ya sea electrónico, mecánico, por fotocopia, por registro y otros medios, sin el permiso previo y por escrito de los titulares del Copyright.

ÍNDICE

ACERCA DEL AUTOR **11**

PRÓLOGO **13**

SOBRE ESTA OBRA **15**

PARTE 1 ..... **17**

CAPÍTULO 1. INTRODUCCIÓN E INSTALACIÓN **19**

1.1 ¿QUÉ ES NODE.JS? 19
- 1.1.1 Información de interés sobre Node.js 21
- 1.1.2 OpenJS Foundation 21
- 1.1.3 Node.js como servidor web 21
- 1.1.4 Frameworks y complementos para Node.js 22
- 1.1.5 Node.js en MEAN y MERN Stack 22
- 1.1.6 Callbacks 22

1.2 INSTALACIÓN 22
- 1.2.1 Instalar Node.js en Windows 23
- 1.2.2 Instalar Node.js en Linux 26

1.3 TESTEAR LA INSTALACIÓN 29
- 1.3.1 Primer programa en Node.js 29
- 1.3.2 Hola Mundo 30
- 1.3.3 Callbacks 30

CAPÍTULO 2. ARQUITECTURA **33**

2.1 COMPONENTES INTERNOS 33
- 2.1.1 Ejemplo 35

2.2 ECMASCRIPT 36

2.3 ENTORNO DE EJECUCIÓN 39

2.4 V8 41

2.5 CONSEJOS PARA EL USO DE JAVASCRIPT 43
2.5.1 Declarar todas las propiedades de un objeto en su constructor 44
2.5.2 Respetar los tipos de datos 44
2.5.3 No crear clases en el scope de una función 46
2.5.4 Cuidado con los paquetes de terceros que agregamos 46
2.5.5 Estar al día con la información y las versiones (todo cambia) 46
2.5.6 Utilizar === en lugar de == para comparar dos valores 47
2.5.7 Tener cuidado con la coerción de datos 48
2.6 PAQUETES 49
2.7 MÓDULOS 50
2.8 CREAR NUESTRO PRIMER MÓDULO 51
2.8.1 Ejemplo 51
2.8.2 Módulos incluidos en Node.js (Built-in modules) 52

CAPÍTULO 3. COMANDOS NODE Y NPM 53

3.1 PARÁMETROS DE LOS COMANDOS 53
3.2 COMANDO NODE 55
3.2.1 Verificar si un programa es sintácticamente correcto 55
3.2.2 Enviar parámetros al motor JavaScript V8 incluido en Node.js 56
3.2.3 Utilizar Node.js sin necesidad de crear un archivo .js 58
3.3 COMANDO NPM 60
3.3.1 Discusión con respecto al significado del término npm 60
3.3.2 Parámetros de configuración del comando npm 62
3.3.3 Registro de paquetes de npm 63
3.3.4 Cómo buscar un paquete en el repositorio de npm 63
3.3.5 Cómo instalar un paquete utilizando npm 63
3.3.6 Paquetes solo para el ambiente de desarrollo 65
3.3.7 Cómo verificar si existen paquetes desactualizados 65
3.3.8 Actualizar un paquete 66
3.3.9 npm Orgs y Enterprise 67
3.3.10 Tendencias de los paquetes npm 67
3.4 ESTRUCTURA DE UN PROYECTO 68
3.4.1 Inicializar un proyecto de Node.js (package.json) 69
3.4.2 Directorio node_modules 69
3.5 PRIMER PROYECTO EN NODE.JS 72

CAPÍTULO 4. BLOCKING VS. NON-BLOCKING 77

4.1 EVENT LOOP Y WORKER POOL 77
4.2 CONCEPTOS BÁSICOS DE MANEJO DE CONCURRENCIA EN UN SISTEMA OPERATIVO 78
4.3 MANEJO DE HILOS EN NODE.JS 79
4.3.1 Tipos de hilos de ejecución 80
4.3.2 No bloquear Event Loop 80

4.4 CONSEJOS PARA EL USO DE NODE.JS ... 80
4.4.1 Revisar la complejidad algorítmica de las funciones ... 80
4.4.2 Controlar los parámetros de entrada de las funciones ... 82
4.4.3 Evitar el uso de expresiones regulares que provoquen un ReDos ... 83
4.4.4 Trabajar siempre con las versiones asíncronas de las funciones ... 83
4.4.5 No procesar objetos en formato Json demasiado grandes ... 84
4.5 FUNCIONES ASÍNCRONAS EN NODE.JS ... 84
4.6 CALLBACK HELL ... 85
4.7 PROMESAS ... 87
4.7.1 Migrar una función que usa callbacks a una promesa ... 88
4.7.2 Ejecutar múltiples promesas al mismo tiempo ... 89
4.7.3 Anidamiento de promesas ... 90
4.7.4 Retornar siempre la promesa que anidamos ... 91
4.7.5 No utilizar forEach en el resultado de una promesa ... 93
4.7.6 Agregar el catch ... 94
4.7.7 No caer en el desorden ... 95
4.8 PRIMITIVAS ASYNC/AWAIT ... 95
4.8.1 Control de errores ... 96
4.8.2 Compatibilidad de async/await con Node.js ... 98

PARTE 2 ... 99

CAPÍTULO 5. SISTEMA DE ARCHIVOS ... 101
5.1 MÓDULO PARA EL MANEJO DEL SISTEMA DE ARCHIVOS ... 101
5.1.1 Retorno de promesa ... 102
5.2 WINDOWS Y UNIX/LINUX ... 102
5.2.1 Manejo de rutas de archivos ... 103
5.3 API PARA OPERAR CON ARCHIVOS ... 105
5.3.1 Abrir un archivo ... 105
5.3.2 Leer un archivo ... 109
5.3.3 Escribir en un archivo ... 110
5.3.4 Documentación de la API FileSystem de Node.js ... 113
5.4 ACTIVIDADES ... 113
5.4.1 Test de autoevaluación ... 113
5.4.2 Ejercicios prácticos ... 114

CAPÍTULO 6. CONSOLA ... 115
6.1 MÓDULO CONSOLE ... 115
6.1.1 Métodos ... 115
6.2 CLASE CONSOLE ... 128
6.2.1 Referencia al módulo fs (file system) ... 128
6.2.2 Cómo crear el archivo de log ... 128
6.2.3 Documentación de Node.js ... 131

6.3 ACTIVIDADES 131
6.3.1 Test de autoevaluación 132
6.3.2 Ejercicios prácticos 132

CAPÍTULO 7. MÓDULO HTTP2 DE NODE.JS 133
7.1 PROTOCOLO HTTP VERSIÓN 2 133
7.1.1 Conceptos básicos del protocolo HTTP 134
7.1.2 Módulo HTTP/2 de Node.js 139
7.2 SERVIDOR 139
7.2.1 Ejemplos 139
7.3 CLIENTE 145
7.3.1 Ejemplos 146
7.4 PRUEBA DE LOS EJEMPLOS 148
7.4.1 Servidor web no encriptado 149
7.4.2 Servidor web seguro 151
7.4.3 Documentación de Node.js 155
7.5 ACTIVIDADES 156
7.5.1 Test de autoevaluación 156
7.5.2 Ejercicios prácticos 156

CAPÍTULO 8. SERVIDOR WEB CON NODE.JS Y EXPRESS 157
8.1 API REST 157
8.1.1 Conceptos básicos de API REST 158
8.2 EXPRESS 160
8.2.1 Instalación 160
8.2.2 Servidor web estático 161
8.2.3 Servidor API REST 165
8.3 RUTEO 171
8.3.1 Servidor API REST con ruteo 172
8.4 ACTIVIDADES 178
8.4.1 Test de autoevaluación 178
8.4.2 Ejercicios prácticos 178

PARTE 3 179

CAPÍTULO 9. CONEXIÓN A UNA BASE DE DATOS 181
9.1 BASES DE DATOS SQL Y NOSQL 181
9.1.1 Bases de datos SQL o relacionales 181
9.1.2 Bases de datos NoSQL o no relacionales 183
9.2 OPERAR SOBRE DIFERENTES MOTORES 184
9.2.1 MSSQL o Microsoft SQL Server 184
9.2.2 PostgreSQL 192
9.2.3 MySQL 199
9.2.4 MongoDB 206

9.3 CONCLUSIÓN 212
9.4 ACTIVIDADES 213
9.4.1 Test de autoevaluación 213
9.4.2 Ejercicios prácticos 213

CAPÍTULO 10. API REST 215
10.1 ARQUITECTURA DEL PROYECTO 215
10.2 INSTALACIÓN Y CONFIGURACIÓN 216
10.2.1 Instalación 216
10.2.2 Configuración 217
10.3 BASE DE DATOS 222
10.3.1 MongoDB 222
10.3.2 MySQL 224
10.4 EXPONER LAS OPERACIONES MEDIANTE EXPRESS 228
10.4.1 Base de datos 228
10.4.2 API 228
10.5 PRUEBA DE LA API CON POSTMAN 233
10.6 CONCLUSIÓN 239
10.7 ACTIVIDADES 240
10.7.1 Test de autoevaluación 240
10.7.2 Ejercicios prácticos 240

CAPÍTULO 11. PUBLICAR UNA APLICACIÓN 241
11.1 ¿QUÉ ES PM2? 241
11.1.1 Instalación 242
11.1.2 Gestión de procesos con PM2 244
11.1.3 Archivo de configuración (Ecosystem File) 248
11.1.4 Control sobre el inicio y fin del programa 251
11.2 INTERFAZ WEB 254
11.3 PROXY INVERSO 257
11.3.1 Windows 258
11.3.2 Linux 264
11.4 ACTIVIDADES 266
11.4.1 Test de autoevaluación 266
11.4.2 Ejercicios prácticos 266

GLOSARIO 267

MATERIAL ADICIONAL 263

ACERCA DEL AUTOR

Luciano Pucciarelli es programador y arquitecto de aplicaciones, nacido en San Genaro, Santa Fe, Argentina.

Estudió Licenciatura en Ciencias de la Computación en la Universidad Nacional de Rosario, y en el año 2005 comenzó a trabajar como programador. Actualmente se desempeña como consultor enfocado en la creación, migración tecnológica y mantenimiento de aplicaciones.

PRÓLOGO

Node.js es un entorno de ejecución del lado del servidor, por lo tanto, el manejo del sistema de archivos (File System) y la salida de datos por consola son temas fundamentales si desarrollamos en esta plataforma. Para ello, Node.js nos provee, para cada uno respectivamente, los módulos llamados fs y console, que incluyen todo lo necesario para poder operar con estos dos grandes componentes del sistema operativo.

Node.js nos provee el módulo llamado http2 que implementa toda la funcionalidad necesaria para trabajar con el protocolo HTTP versión 2. Gracias a este módulo, es posible crear programas que funcionen como servidores web sin encriptación o servidores web seguros utilizando el protocolo HTTPS.

SOBRE ESTA OBRA

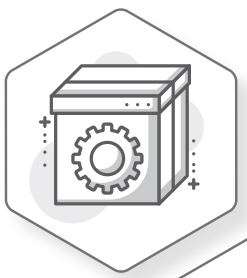
En este libro haremos un recorrido teórico y práctico por todo el ecosistema tecnológico que compone Node.js. Veremos desde su instalación en diferentes plataformas, hasta cómo crear programas, paso a paso y de manera detallada. En cada parte de este libro trataremos temas fundamentales que nos ayudarán a conocer y comprender mejor cada detalle de Node.js, sin necesidad de poseer conocimientos previos. Además, configuraremos el entorno de trabajo y, al mismo tiempo, crearemos programas de ejemplo utilizando los módulos de Node.js que estudiamos en cada capítulo.

- **Parte 1:** ¿Qué es Node.js? Indicaciones para realizar su instalación en diferentes plataformas. Análisis de su arquitectura (ECMAScript, JavaScript, motor V8). Guía paso a paso para el uso de los comandos node y npm. Diferencias entre la programación bloqueante y no bloqueante en Node.js.

- **Parte 2:** Manejo del sistema de archivos (file system) y salida por consola. Creación de un servidor web utilizando el protocolo HTTP versión 2. Uso de Express para crear una API de tipo REST utilizando el módulo de ruteo y el retorno de datos en formato JSON.

- **Parte 3:** Aplicación de Node.js en diferentes proyectos. Cómo conectarse a distintos motores de bases de datos, ejecutar operaciones CRUD sobre una API de tipo REST hecha con Node.js y Express, y publicar una aplicación en un ambiente productivo utilizando PM2.

USERS Parte 1



Instalación

Arquitectura

node y npm

1

INTRODUCCIÓN E INSTALACIÓN

El 27 de mayo de 2009 se publicó la primera versión de **Node.js**, creada por **Ryan Dahl**, entre cuyas principales características podemos mencionar que rompió con el concepto de que JavaScript solo estaba asociado al navegador, ya que permitía su ejecución del lado del servidor de manera independiente.

Si bien el concepto de ejecutar JavaScript del lado del servidor no es nuevo, con la puesta en escena de Node.js, se popularizó y se adoptó de forma masiva.

1.1 ¿QUÉ ES NODE.JS?

Node.js es un entorno de ejecución multiplataforma para el lenguaje de programación JavaScript. Es de **código abierto** y su licencia es de tipo **MIT Licence**, lo que significa que cualquier persona puede descargarlo e instalarlo en su computadora sin tener que pagar una licencia. Existe una gran comunidad alrededor de todo el mundo involucrada con Node.js. Si accedemos a su cuenta oficial del repositorio de código fuente **https://github.com/nodejs/node**, podremos ver la gran cantidad de colaboradores con que cuenta (2645 al momento de escribir este libro).

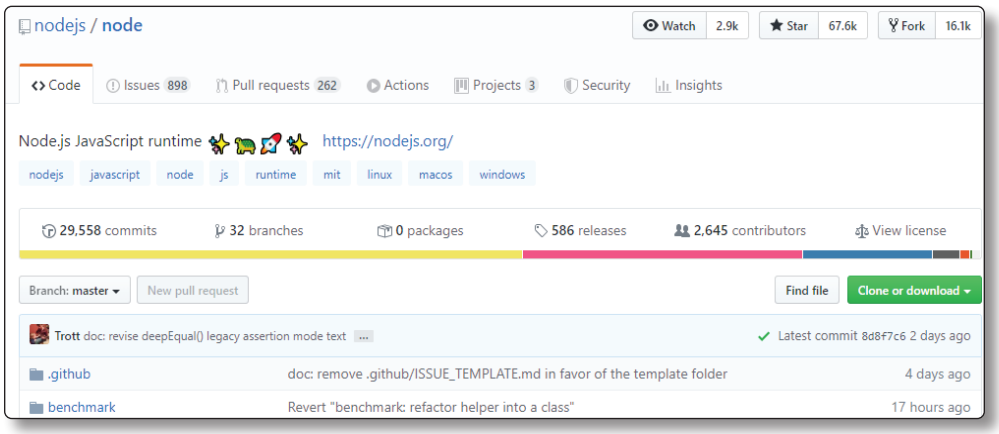


Figura 1.1.

La sintaxis de las primitivas del lenguaje que utilizamos para programar en Node.js están basadas en la especificación **ECMAScript**, publicada por la organización **ECMA International**. Su arquitectura está **orientada a eventos**, y el motor que emplea para interpretar y ejecutar el código JavaScript pertenece a Google y se llama **V8**. Originalmente, Node.js se pensó y diseñó para entornos de servidores con un alto grado de concurrencia. El modelo de evaluación que utiliza es de un único hilo de ejecución, apoyándose en un esquema asíncrono de entrada/salida llamado **EventLoop**, lo cual permite atender miles de llamadas sin recurrir a cambios de contexto, un proceso muy costoso a nivel de sistema operativo. Este tipo de modelo se debe, en gran parte, a la librería escrita en el **lenguaje C** llamada **libvu**, que provee soporte para operaciones asíncronas de entrada/salida a Node.js. Este enfoque arquitectónico tiene como consecuencia que Node.js sea uno de los candidatos principales al momento de elegir la tecnología que se va a utilizar en sistemas que requieran una respuesta rápida y eficiente bajo gran carga de trabajo. Citando algunos casos reales de uso de Node.js, podemos encontrar grandes empresas como Netflix, Linkedin, Walmart, Paypal, eBay y Uber, que lo aplican para brindar los servicios utilizados por miles de personas en todo el planeta. Inicialmente, cuando se creó Node.js en el año 2009, podía ejecutarse solo en entornos Linux y MacOS. Más tarde, en junio del año 2011, Microsoft y Joyent implementaron la primera versión nativa para entornos Windows. Uno de los componentes más importantes, no solo para Node.js sino también para muchos lenguajes de tipo JavaScript actuales, es su manejador de paquetes **NPM (Node Package Manager)**. Sin él, trabajar en entornos JavaScript que utilicen Node.js sería casi imposible, ya que es de gran ayuda para administrar, instalar y organizar todas las dependencias que utilicemos a lo largo de nuestro proyecto.

1.1.1 Información de interés sobre Node.js

A continuación, enumeraremos algunos temas de interés técnico y no técnico acerca del ecosistema tecnológico de Node.js que nos ayudarán a comprender mejor esta herramienta tecnológica y a complementar la información vista hasta el momento.

1.1.2 OpenJS Foundation

En el año 2019, **Node.js Foundation** y **JS Foundation** se fusionaron para dar lugar a **OpenJS Foundation**, cuya actividad principal es alojar, organizar y financiar proyectos de tipo JavaScript para potenciar su crecimiento. Entre las principales empresas que apoyan esta fundación podemos encontrar a Google, Microsoft, IBM y PayPal, entre otras. Si ingresamos en el sitio oficial de Node.js, **https://nodejs.org**, en la parte inferior veremos el logo de OpenJS Foundation, que aloja proyectos JavaScript sumamente importantes, como JQuery, Node.js, WebPack y Dojo. Para obtener más información, podemos visitar su página oficial, con el detalle de todos los proyectos apoyados por esta fundación: **https://openjsf.org/projects**.

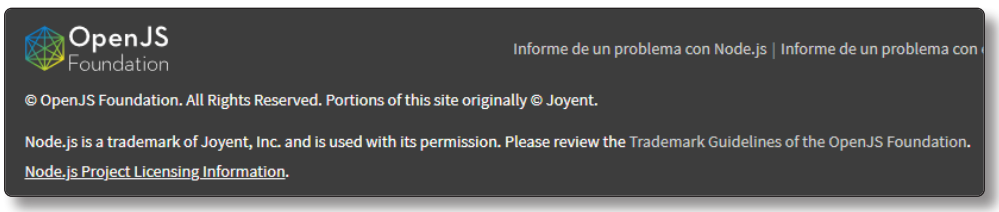


Figura 1.2.

1.1.3 Node.js como servidor web

En la actualidad, Node.js cuenta con un gran número de módulos y componentes orientados y optimizados para **networking** (redes), que sirven de soporte para el manejo de los estándares y protocolos más populares de Internet, como **DNS**, **HTTP**, **TCP**, **TLS/SSL** y **UDP**. Por todo esto, cuando escuchamos hablar sobre Node.js, es muy probable que lo asociemos con la capa de servicios de aplicaciones orientadas a Internet. Si bien esto es cierto, Node.js va mucho más allá de ser solo un servidor web.

1.1.4 Frameworks y complementos para Node.js

La sintaxis de Node.js muchas veces puede resultar de muy bajo nivel para realizar ciertas tareas típicas de una aplicación orientada a la Web, por lo que, generalmente, cuando desarrollamos sobre esta plataforma, lo complementamos con un **framework** JavaScript, como **Express.js**, **Koa**, **Meteor**, **Sails.js**, etc. Estos frameworks agregan una capa de abstracción sobre Node.js, con lo cual facilitan y añaden servicios adicionales a los ya existentes de forma nativa.

1.1.5 Node.js en MEAN y MERN Stack

Dos **stacks** de desarrollo muy conocidos y usados en la actualidad son **MEAN y MERN Stack**. El término MEAN es el acrónimo de **MongoDB**, **ExpressJS**, **Angular** y **NodeJS**; mientras que MERN es lo mismo pero reemplazando Angular por **React**. Esta combinación de tecnologías JavaScript **open source** forman parte de muchos sistemas informáticos que usamos en la actualidad, donde el rol principal de Node.js es encargarse de la capa de negocios o servicios y ser el nexo entre el cliente y la base de datos.

1.1.6 Callbacks

Como ya mencionamos, el núcleo de Node.js se basa, principalmente, en la ejecución asíncrona de código, esto es, que el proceso principal no se bloquea cuando una de las funciones del flujo de ejecución del programa lo hace. Un elemento importante en este escenario son las llamadas **callbacks**. Una callback es, simplemente, una función que suele usarse para ser invocada cuando se finaliza con una tarea que llevará un tiempo ejecutarse, como, por ejemplo, cargar una lista de clientes desde una API expuesta en Internet. Se debe tener un cuidado especial al momento de usar callbacks ya que, si se abusa de ellas, el código puede quedar ilegible, y esto se prestará a confusiones y errores. Para evitar ese desorden de código que muchas veces se asocia a las callbacks, en la versión 8 de Node.js se introdujeron dos nuevas primitivas del lenguaje llamadas **async** y **await**, que producen un código mucho más legible y ordenado.

1.2 INSTALACIÓN

En esta sección veremos una guía paso a paso de cómo instalar Node.js en los sistemas operativos Windows y Linux. Los pasos a seguir dependerán del sistema operativo; como es costumbre, en Windows será un paquete autoinstalable, mientras que en Linux haremos el proceso desde una consola de línea de comandos.

Si vamos a trabajar con Node.js en un sistema crítico o productivo, deberemos tener especial cuidado con las versiones, y no pasar de una a otra a medida que van saliendo porque podrían tener algunos bugs o no ser estables.

Antes de adoptar una nueva versión, siempre es recomendable leer las diferencias (**changelog** , por su nombre en inglés) con la versión anterior, para evitar posibles problemas. El detalle de cada versión puede obtenerse de la página oficial del repositorio de código fuente **https://github.com/nodejs/node/tree/master/doc/changelogs**. Node.js ofrece dos tipos de versiones: **LTS** (**Long Term Support**, por sus siglas en inglés) y **Actual**(la última versión estable).

¿Cuál debemos elegir? La realidad es que no hay una receta para seleccionar una. Puede ocurrir que, si es un proyecto crítico, no convenga elegir la versión LTS sino la Actual.

De todos modos, esto dependerá de la decisión del grupo de trabajo involucrado en el proyecto.

1.2.1 Instalar Node.js en Windows

A continuación, explicaremos paso a paso cómo instalar Node.js en un entorno **Windows**. Una vez finalizado el proceso, pasaremos a verificar que fue satisfactorio, y ya dispondremos de Node.js listo para usar.

PASO 1

En primer lugar, debemos ingresar en la página oficial de Node.js, **https://nodejs.org**.

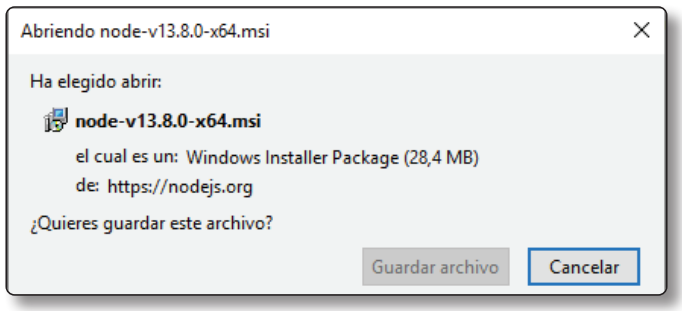
Figura 1.3.

PASO 2

Una vez allí, veremos que nos ofrece las versiones LTS y Actual; en este caso elegimos la Actual 13.8.0 haciendo clic sobre este botón.

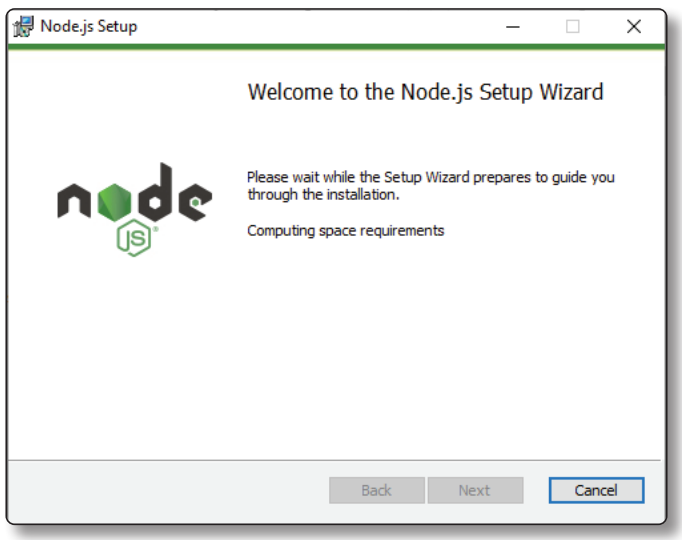
PASO 3

Se procederá a descargar el paquete con el instalador.



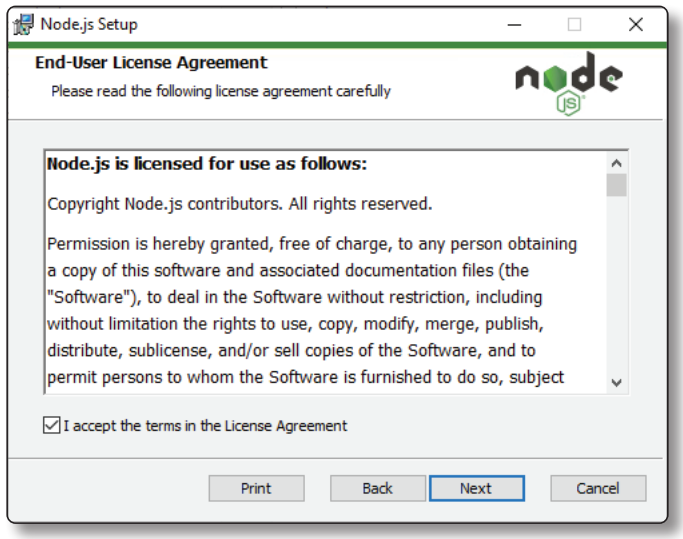
PASO 4

Cuando se completa el proceso, abrimos el archivo y pasamos a realizar la instalación haciendo clic en ***Next***.



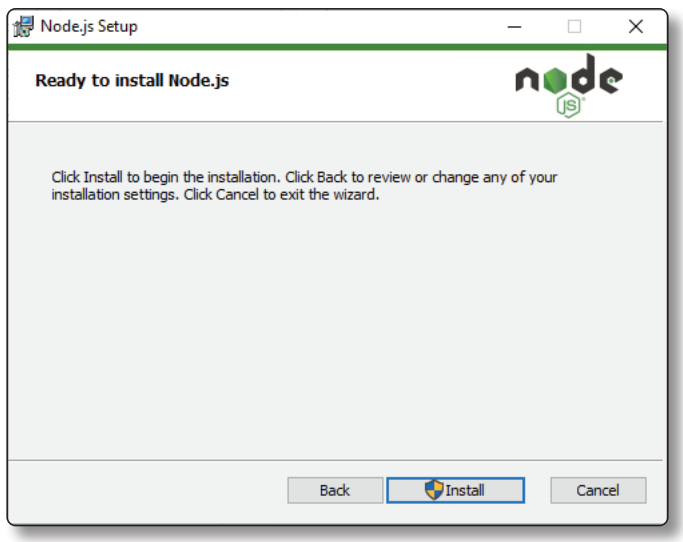
PASO 5

Luego de aceptar los términos de Licencia, continuamos con la instalación normalmente dejando las opciones por default.



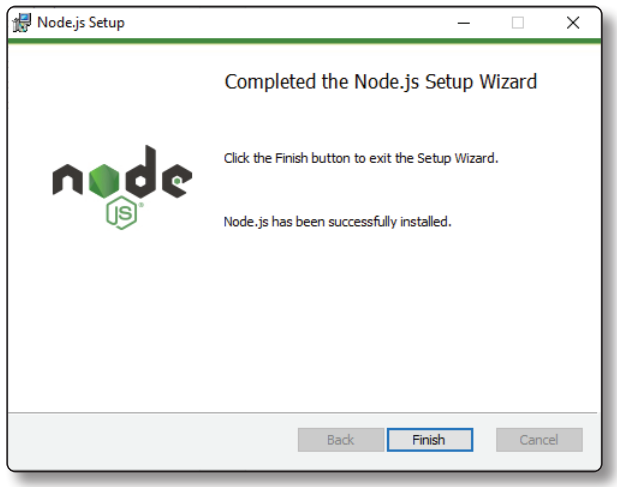
PASO 6

En el paso final, hacemos clic en el botón **`Install`**. Si observamos el icono del botón, veremos que contiene una imagen indicando que necesita permiso de administrador, por lo que el usuario que realiza la instalación deberá contar con ellos.



PASO 7

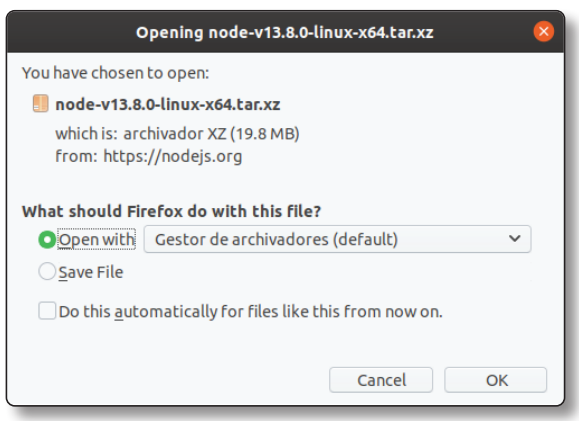
Al finalizar el proceso de instalación, pulsamos en el botón ***Finish*** y listo, ya tenemos Node.js en el sistema.



1.2.2 Instalar Node.js en Linux

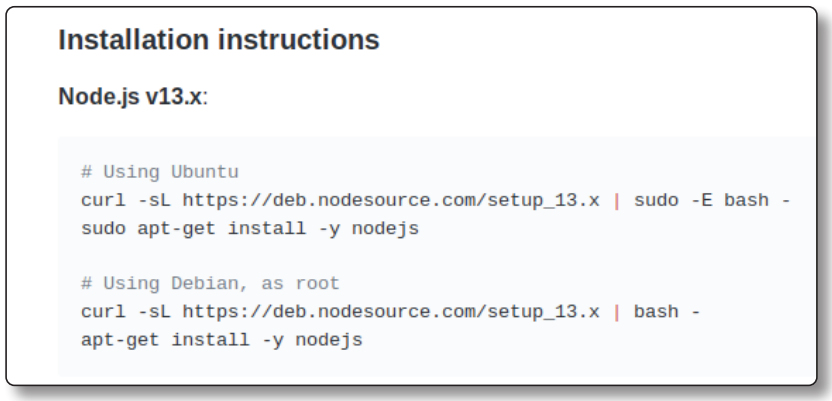
La distribución de Linux que utilizaremos será **Ubuntu 19.10**. A diferencia de Windows, en Linux la instalación se lleva a cabo mediante una terminal de línea de comandos. Si ingresamos en el sitio oficial de Node.js, veremos las opciones para instalar la versión LTS o la Actual. Al hacer clic en Actual, comienza la descarga del paquete genérico para entornos Linux, **`node-v13.8.0-linux-x64.tar.gz`**.

Si bien ese paquete es correcto, nosotros vamos a instalar el que viene preparado y compilado para Ubuntu. Por lo tanto, cancelamos la descarga y hacemos lo siguiente:



PASO 1

Ingresamos en la URL **https://github.com/nodesource/distributions/blob/master/README.md** y vamos a la sección **`Node.js v13.x # Using Ubuntu`**.

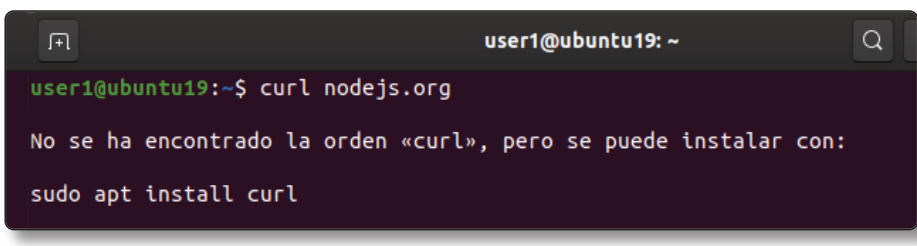


PASO 2

Procedemos a verificar si contamos con el comando ***`curl`***, que será necesario en la instalación. Para esto, abrimos una terminal de línea de comandos y ejecutamos ***`curl nodejs.org`***.

Si ya lo tenemos instalado, veremos código HTML de la página de Node.js; de lo contrario, aparecerá el mensaje

`No se ha encontrado la orden «curl»`.



PASO 3

Si este es el caso, procedemos a instalar el comando ***`curl`*** ejecutando ***`sudo apt install curl`*** en la terminal de comandos.