**Métodos HTTP usando postman**

Andrea R. González B.

Facultad de Ingeniería, Universidad Valle del Momboy

Unidad II: Servidores Backend y Protocolo HTTP

Prof. Edgar Valero

Octubre 11, 2021

Contenido

[Introducción 3](#_Toc84923694)

[Desarrollo de un servidor Backend 4](#_Toc84923695)

[Inicialización 4](#_Toc84923696)

[Middleware 5](#_Toc84923697)

[Instalación de nojde.js 7](#_Toc84923698)

[Protocolo HTTP 15](#_Toc84923699)

[Códigos de estado 16](#_Toc84923700)

[Actividad 2.1 16](#_Toc84923701)

[Desarrollo 17](#_Toc84923702)

[Rutas 17](#_Toc84923703)

[Conclusión 19](#_Toc84923704)

[Bibliografía 20](#_Toc84923705)

# Introducción

La programación es una disciplina que cada día toma más auge, y siendo el desarrollo web una rama importante de la misma, nace la necesidad de conocer los métodos y códigos de estado HTTP ya que permite la comunicación y entre un servidor y un cliente. En muchos casos es importante también conocer las terminologías y usos profesionales del desarrollo web, una herramienta importante que facilita la creación de servidores es el entorno de ejecución multiplataforma node.js, ya que gracias a la instalación de sus módulos se puede crear un servidor en pocas líneas de código y configurar el mismo de manera muy sencilla. Dentro del presente informe se encuentra la explicación para la instalación de node.js.

Un framework importante hecho para la creación de servidores es express, es una aplicación web de back-end para Node.js, diseñado para crear aplicaciones web y API y se encuentra como el framework estándar para trabajar con desarrollo web con node.js. es importante saber de sus características como sus usos y middleware para aprovechar al máximo este recurso. Su instalación en un proyecto se puede ejecutar de distintas formas, una opción es la instalación global.

# Desarrollo de un servidor Backend

Según colaboradores de Wikipedia (2021d) “La programación del lado del servidor es una tecnología que consiste en el procesamiento de una petición de un usuario mediante la interpretación de un script en el servidor web para generar dinámicamente páginas HTML como respuesta”. Para crear un servidor backend existen muchas alternativas, una de ellas es usando módulos de node.js.

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. (colaboradores de Wikipedia, 2021)

## Inicialización

Una forma básica de crear un servidor con node.js es usando el modulo ‘http’ de la siguiente forma;

**const** http = require('http');

**const** hostname = '127.0.0.1';

**const** port = 1337;

http.createServer((req, res) **=>** {

  res.writeHead(200, { 'Content-Type': 'text/plain' });

  res.end('Hello World\n');

}).listen(port, hostname, () **=>** {

  console.log(`Server running at http://${hostname}:${port}/`);

});

Este módulo http de node.js no necesita instalación, viene por defecto con node, pero existen otros módulos, como tal es el caso de express, que es disponible a través del registro $ npm install express. Una forma rápida para iniciar, es utilizar el ejecutable express para generar una aplicación como se muestra a continuación (Npm: Express, 2019):

Instalar el ejecutable

$ npm install -g express-generator@4

Crear la app

$ express /tmp/foo **&&** cd /tmp/foo

Instalar las dependencias

$ npm install

Empezar el servidor

$ npm start

## Middleware

Según los colaboradores de Wikipedia (2021a) el middleware se puede definir como la lógica de intercambio de información entre aplicaciones, o como un software que asiste a una aplicación para interactuar con otras aplicaciones, o paquetes de programas, redes, hardware o sistemas operativos.

Body-parser por ejemplo, en Npm: Body-Parser (2019), se presenta como un middleware de análisis del cuerpo de Node.js ya que es capaz de analizar los cuerpos de las solicitudes entrantes antes que sus controladores, disponible en la propiedad req.body. Este middleware es bastante conocido, sin embargo, si se usa express ya no hace falta instalar Body-parser en un proyecto, ya que express en sus últimas versiones ahora lo muestra como obsoleto, y como remplazo recomienda usar la función express.urlencoded() que es “una función de middleware incorporada en Express. Analiza las solicitudes entrantes con cargas útiles codificadas en urlencoded y se basa en body-parser” (GeeksforGeeks, 2021).

Otra función de middleware incorporada en Express es express.json(), esta sirve para “analizar las solicitudes entrantes con cargas útiles JSON y se basa en body-parser” (*Express 4.x - Referencia de API*, s. f.).

Morgan, es un Middleware que funciona con el fin de registrar de solicitudes HTTP para node.js, y entre sus distintas utilidades, una de ellas es la capacidad de mostrar por consola el código de estado del servidor.

# Instalación de nojde.js

Para instalar node.js hay que ir a su página principal de descarga <https://nodejs.org/es/download/>, elegir el sistema operativo deseado y empezar la descarga.

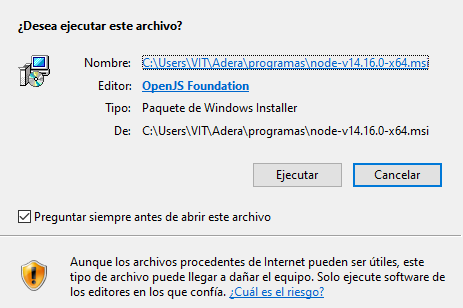
*Figura 1. Página principal de descargas.*

Fuente: OpenJS Foundation.

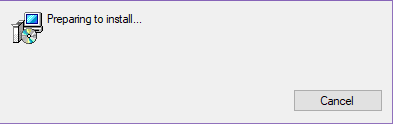
*Figura 2. Indicador de tiempo de descarga.*



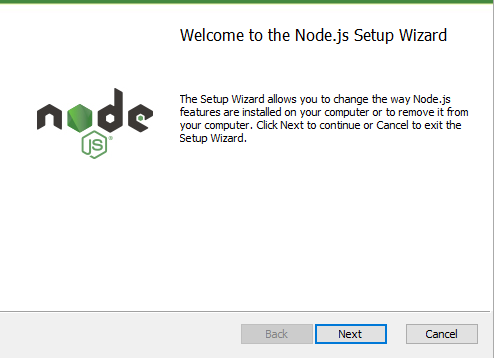
Esperar la descarga de internet del instalador .msi de node.js.

*Figura 4.ventana emergente de ejecución* .

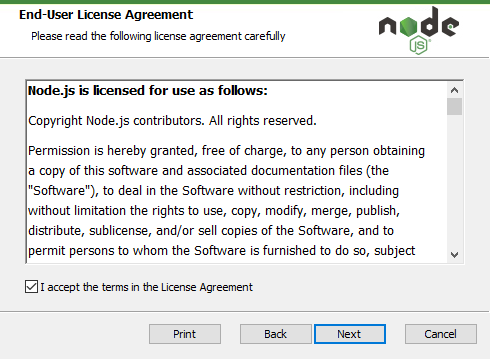
Una vez descargado el instalador, lo ejecutamos y empiezan los pasos de instalación.

*Figura 5. Aviso de preparación de la instalación*

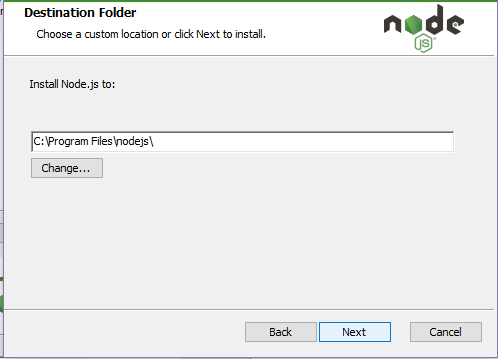
Se espera unos segundos por la preparacion de la instalación.

*Figura 6. Bienvenida de node.js para la instalación*

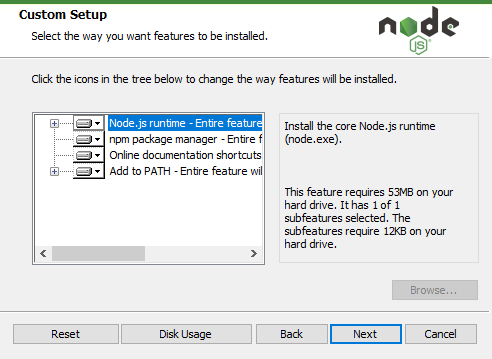
Ya una vez iniciado el setup wizard se procede a elegir las configuraciones para la instalación.

*Figura 7. Acuerdos de licencia de node.js*

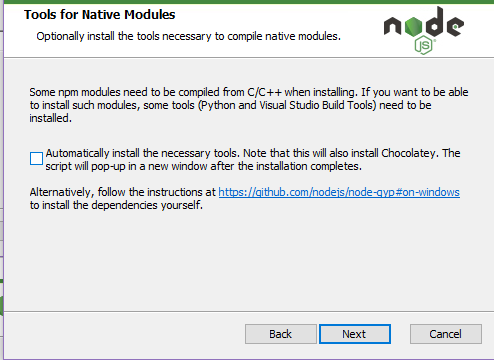
Hay que aceptar los acuerdos de licensia que es un contrato entre los creadores y el usuario donde se otorga el derecho de utilizar node.js.

*Figura 8. Selección de la carpeta de instalación*

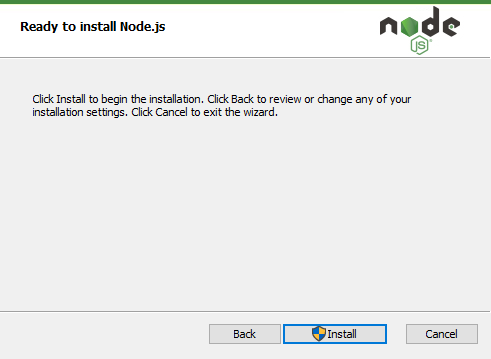
Se debe elegir la ruta de instalacion de node.js, donde estaran todas sus carpetas y archivos necesarios para su perfecto funcionamiento.

Figura 9.Seleccion de las características de instalación 

Se elige la forma en que desea que se instalen las funciones.

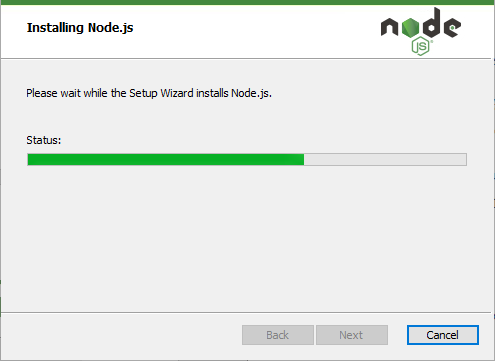
Figura 10. Opción de instalar automáticamente los módulos necesarios de node.js

Se elige o no la automatización de descargas de las herramientas necesarias de node.js

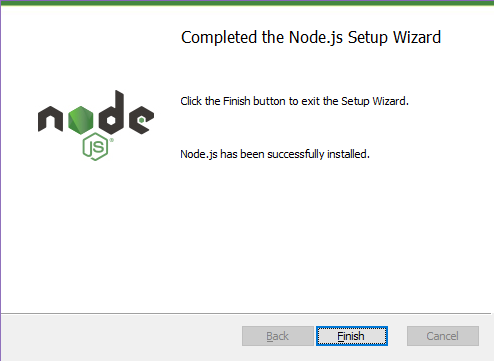
Figura 11. Comenzar la instalación finalmente

Finalmente se inicia la instalación con todas las configuraciones establecidas anteriormente.

Figura 12. Barra del estatus de instalación de node.js



Se espera hasta que finalice la instalación.

Figura 13. Finalizacion de instalacion de node.js

Ya terminada la espera, se da clic en finalizar y tendremos node.js instalado en el ordenador.

# Protocolo HTTP

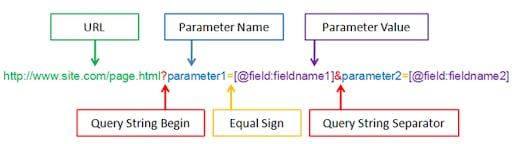
Al momento de navegar por la web, este protocolo se hace presente facilitando la comunicación entre la World Wide Web y los usuarios. Cuando se habla de HTTPS, se hace referencia un protocolo HTTP con Secure Sockets Layer, con el objetivo de medidos de seguridad por métodos de encriptación y asi proteger de ataques de terceros. Para poder tener esta quinta letra “s” en nuestro HTTP necesitamos adquirir un certificado SSL, el cual está hecho para certificar como segura la página. Para hacer posible esta comunicación entre servidor y usuario, se usan métodos para poder especificar estas peticiones, entre los cuales los más importantes son; POST, GET, PUT, DELETE y HEAD.

Get es un comando de petición, Las peticiones que usan el método GET sólo deben recuperar datos.

Post se utiliza para enviar información a un recurso en específico causando así un cambio de estado.

Put se utiliza para cambiar o actualizar un recurso.

Delete para borrar un recurso

*Figura 14. Una URL típica que contiene un Query String es la siguiente:*

Fuente: (GitBook, s. f.)

# Códigos de estado

Según la página MDN (2021). Estos códigos de respuesta HTTP demuestran si se ha completado o no una solicitud HTTP específica, de estas respuestas existen 5 clases respuestas informativas (100–199), respuestas satisfactorias (200–299), redirecciones (300–399), errores de los clientes (400–499), y errores de los servidores (500–599)

# Actividad 2.1

El servidor debe administrar internamente la estructura de una ciudad. Los controladores deben tener para administrar las edificaciones de una ciudad en donde se incluyen: Casas, Edificios, Apartamentos, Carreteras, Puentes, Parques y Zonas Comerciales. Los controladores deben tener la posibilidad de contabilizar la cantidad de estructuras creadas, tanto para crear, destruir, agrandar o disminuir el tamaño de las edificaciones si se desea.

Para guardar la información de la estructura de la ciudad se requiere de la utilización de arreglos. Estos arreglos deben estar estructurados en formato JSON con toda la información pertinente a las especificaciones de las edificaciones.

## Desarrollo

Se declaró un arreglo de objetos "ciudad" que guarda por ID la edificación que se deseé, como casas, edificios, parques... Y las estructuras de las mismas, que es un nuevo arreglo de objetos para cada estructura individual, estás pueden guardar las características que se deseé al crearlas.

### Rutas

|  |  |  |
| --- | --- | --- |
| Método | Rutas | Descripción |
| get | / | Lleva al Home Page, una vista ejs con las indicaciones de uso, la cantidad total de edificaciones y todas las rutas. |
| get | /ciudad/:id | Filtra el parámetro :id en el arreglo de objetos ciudad para mostrar la estructura con el id que coincidencia. |
| get | /ciudad | Muestra por completo el arreglo de objetos ciudad. |
| post | /query/item/:id | Esta ruta guarda por medio de query en ciudad la edificación con el id que coincidencia con el parámetro :id. Las query pueden ser tantas quiera el usuario y puede devolver dos mensajes, uno de éxito y otro que indica que la edificación no tiene una id propia en el caso de que está no exista. |
| post | /body/item/:id | Está ruta recorre el arreglo ciudad para finalmente agregar con un push el objeto con la información de la nueva estructura. Está también puede dar dos mensajes, uno de éxito y otro que indica que la edificación no tiene una id propia en el caso de que está no exista. |
| POST | /ciudad/:id | Está ruta es capaz de crear un nuevo tipo de edificación en el arreglo de objetos principal ciudad, recibe el parámetro :id para guardar el nombre de este nuevo tipo de estructura y a su vez guarda un elemento estructuras como un array vacío en espera de llenar con objetos. |
| DELETE | /delete/ítem/  :id/:itemid | Aquí se puede eliminar un elemento en específico de un tipo de edificación de la ciudad, es necesario el params del id del tipo de edificación y el id de la estructura. Está ruta llama a una función eliminar, esta función recibe 3 datos, el primero es el arreglo ciudad, el segundo es el id de aquella edificación que coincidencia con el re.params.id, y en tercer lugar recibe el id de la estructura que coincida con el req.params.itemid. |
| DELETE | /delete/ciudad/:id | Está es capaz de eliminar un tipo de edificación de la ciudad por completo. |
| PUT | /ciudad/:id/:cambio | En esta ruta se puede cambiar el id de un elemento de ciudad. |
| PUT | /estructura/:id/  :itemid/:cambio | Aquí se puede cambiar los datos de una estructura dentro de un objeto de ciudad. |

# Conclusión

En esta actualidad virtualizada, es muy beneficiosa la variedad de páginas web que se pueden encontrar, pero como contraparte también existen muchos peligros, por lo que es importante el protocolo de transferencia que cifra los flujos de datos. Como también es importante conocer los conceptos básicos de node.js para poder trabajar con sus framework y módulos que facilitan la ejecución de servidores y configuración de los mismos.

# Bibliografía

colaboradores de Wikipedia. (2021d, junio 6). *Script del lado del servidor*. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Script_del_lado_del_servidor>

*npm: express*. (2019, 26 mayo). Npm. <https://www.npmjs.com/package/express>

colaboradores de Wikipedia. (2021e, septiembre 27). *Node.js*. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Node.js>

colaboradores de Wikipedia. (2021a, agosto 8). *Middleware*. Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Middleware>

*npm: body-parser*. (2019, 26 abril). Npm. <https://www.npmjs.com/package/body-parser>

GeeksforGeeks. (2021, 5 mayo). *Express.js express.urlencoded() Function*. <https://www.geeksforgeeks.org/express-js-express-urlencoded-function/>

*Express 4.x - Referencia de API*. (s. f.). Express 4.x - Referencia de API. Recuperado 11 de octubre de 2021, de <https://expressjs.com/es/api.html#express>

*npm: morgan*. (2020, 20 marzo). Npm. <https://www.npmjs.com/package/morgan>

Mateos, S. M. U. (2017, 15 febrero). *Importancia del HTTPS*. Actualidad eCommerce. <https://www.actualidadecommerce.com/importancia-del-https/>

OpenJs Foundation. (s. f.). *Descarga Node.js* [Captura]. Descarga Node.js. <https://nodejs.org/es/download/>

*Códigos de estado de respuesta HTTP - HTTP | MDN*. (2021, 11 octubre). Codigos HTTP. <https://developer.mozilla.org/es/docs/Web/HTTP/Status>

*GitBook. (s. f.). Capítulo 5: Query String [Imagen]. Capítulo 5: Query String.* [*https://ull-esit-dsi-1617.github.io/estudiar-cookies-y-sessions-en-expressjs-angel-alexandra-nicolangelo-p4-35l2/cookies/chapter5/chapter5.html*](https://ull-esit-dsi-1617.github.io/estudiar-cookies-y-sessions-en-expressjs-angel-alexandra-nicolangelo-p4-35l2/cookies/chapter5/chapter5.html)