Capítulo 4
La plataforma Node.js

1. Presentación de Node.js

Node.js es un entorno que permite ejecutar código JavaScript, fuera de un navegador. Cuando se está redactando este libro, se basa en el motor JavaScript V8 desarrollado por Google para sus navegadores Chrome y Chromium.

Su arquitectura es modular y orientada a eventos. Está fuertemente orientado a la red. Tiene numerosos módulos de red (de los que se muestra a continuación los principales en orden alfabético: DNS, HTTP, TCP, TLS/SSL, UDP), para los principales sistemas operativos (Unix/Linux, Windows, Mac OS). De esta manera, sustituye ventajosamente, en el marco que nos interesa aquí (es decir, la creación y la gestión de aplicaciones web), a un servidor web como Apache.

Creado por Ryan Lienhart Dahl en 2009, este entorno se ha hecho muy popular rápidamente, por sus dos cualidades principales:

– Su ligereza (resultado de su modularidad).
– Su eficacia, provocada por su arquitectura monothread (resultado de la gestión orientada a eventos que ofrece de manera nativa el entorno JavaScript).

Por lo tanto, integrar Node.js en el desarrollo de aplicaciones web forma parte de la lógica actual, que consiste en hacer las operaciones de acceso a los datos lo menos bloqueantes posible (para solventar la problemática llamada «bound I/O» según la que, antes que cualquier otra causa, la latencia global de una aplicación se debe al tiempo de latencia de los accesos a los datos).

Por lo tanto, Node.js permite a las aplicaciones web crear servidores extremadamente reactivos.

En lo sucesivo, va a:

- Instalar y probar Node.js en Linux, Windows o macOS.
- Crear un servidor HTTP que devuelva una cadena de caracteres.
- Implementar un módulo.
- Crear un servidor HTTP utilizando el módulo express, invocado en una ruta REST y devolviendo los datos formateados en JSON, inicialmente en su totalidad y después, filtrados por una propiedad.

En este capítulo, solo introduciremos algunos módulos (y funciones) de Node.js.

La documentación completa de los módulos, está disponible en esta dirección: https://nodejs.org/api/

2. Instalación y prueba de Node.js

2.1 Creación del archivo de prueba

Para probar Node.js, en primer lugar va a crear un código JavaScript que va a ser lo más sencillo posible y lo va a ejecutar con Node.js.

Por lo tanto, cree el archivo pruebaDeNode.js, que solo comprende una línea de código:

```
console.log("Prueba de Node");
```

2.2 Instalación y prueba de Node.js en Ubuntu

- Para instalar Node.js en Ubuntu, lo más sencillo es utilizar el comando `curl` y el administrador de paquetes en línea de comandos (`apt-get`):

```
curl -sL https://deb.nodesource.com/setup_11.x | sudo -E bash
-sudo apt-get install nodejs
```

Observe que en el momento en que se escribe este libro, la versión actual de Node.js es la 11. Para una versión posterior, es suficiente con cambiar su número de versión en el comando de instalación.

- Se puede crear un enlace simbólico llamado `node`, para lanzar de manera más natural sus servidores:

```
sudo ln -s /usr/bin/nodejs /usr/bin/node
```

- Abra un terminal (shell) y ejecute el archivo de prueba:

```
node pruebaDeNode.js
```

Se muestra la cadena de caracteres «Prueba de Node».

Un procedimiento completo está en línea en la dirección:
https://wiki.ubuntu.com/SpanishDocumentation

2.3 Instalación y prueba de Node.js en Windows

La instalación de Node.js y su prueba en Windows, se van a desarrollar en cuatro etapas:

- Descargue el instalador Windows Installer, yendo al sitio oficial de Node.js: https://nodejs.org/en/download
- Ejecute el instalador (el archivo .msi anteriormente descargado), aceptando las condiciones de utilización y la configuración por defecto.
- Vuelva a arrancar su ordenador.
- Abra la línea de comandos y ejecute el archivo de prueba:

```
node pruebaDeNode.js
```

Se muestra la cadena de caracteres «Prueba de Node».

2.4 Instalación y prueba de Node.js en Mac OS

La instalación de Node.js y su prueba en Mac Os, se van a hacer en tres etapas:

- Descargue el paquete de instalación para Mac Os (Macintosh Installer), yendo al sitio oficial de Node.js: https://nodejs.org/en/download/
- Abra un terminal e instale el paquete:

```
pkg install nombrePaquete.pkg
```

- Ejecute el archivo de prueba:

```
node pruebaDeNode.js
```

Se muestra la cadena de caracteres «Test de Node».

3. La modularidad de Node.js

3.1 Los módulos y los paquetes

Una de las principales fortalezas de Node.js es ser modular (y principalmente ofrecer numerosos módulos de red). Si algunos de estos módulos se instalan directamente al mismo tiempo que Node.js, la mayor parte se debe instalar bajo demanda.

Durante la creación de una aplicación que exige la instalación de módulos, dos métodos son posibles para realizarla aquí:

– Directamente con el administrador de módulos npm (y su opción `install`).
– Indirectamente (pero siempre con npm), a través de la especificación de las dependencias de la aplicación (es decir, los módulos necesarios para esto), en un archivo llamado package.json.

Un módulo se utiliza en una aplicación, con la función `require()`:

```
var moduloEnsuAplicacion = require('<nombreDelModulo>');
```

Su aplicación Node.js se puede reutilizar a su vez como módulo en determinadas condiciones, que se presentarán más adelante.

Detengámonos en un punto un poco sutil: la distinción entre módulos y paquetes.

Los módulos son las piezas conceptuales de una aplicación Node.js. Un módulo se puede organizar en diferentes códigos JavaScript y depender de otros módulos. De esta manera, todos los recursos necesarios para un módulo (código, archivo package.json que especifica sus dependencias, etc.) se agrupan en un paquete que, de hecho, es una carpeta.

Por lo tanto, si los dos términos son casi intercambiables, el término «módulo» hace pensar más en la funcionalidad global, y «paquete» en la carpeta y la organización de los archivos de código que se encuentran en ella.

3.1.1 El administrador de módulos de Node.js: npm

npm (*Node.js Package Manager*), es el administrador de módulos de Node.js (se instala con éste).

Los módulos se instalan globalmente en la carpeta node_modules, situada a nivel de los directorios de sistema, si se utiliza la opción `-g`:

```
npm install -g <module>
```

o en caso contrario (sin la opción `g`) en el directorio actual (pero también en una carpeta llamada node_modules).

3.1.2 Especificación de las dependencias: el archivo package.json

Para especificar las dependencias necesarias para la creación de una aplicación Node.js (es decir los módulos asociados a los paquetes necesarios), se recomienda crear un archivo llamado package.json.

En el contexto de un archivo package.json, solo hablamos más de paquetes (y de módulos).

A continuación se muestra un esquema mínimo de este archivo:

```
{
   "name":         "<nombre de la aplicación>",
   "version":      "<versión de la aplicación>",
   "description": "<descripción de la aplicación>",
   "author":       "<nombre del autor de la aplicación>",
   "main":         "<código a ejecutar como punto de entrada>",
   "scripts": {
       "start": "node <código a ejecutar>"
   },
   "dependencies": {
     "<nombre del paquete>": "<versión mínima del paquete a instalar>",
     ...
   }
}
```

Para instalar los módulos necesarios, se debe ejecutar el siguiente comando:

```
npm install
```

Y a continuación se muestra el que va a lanzar el servidor:

```
npm start
```

Para crear un esqueleto de archivo *package.json*, utilice el siguiente comando:

```
npm init --yes
```

En este caso, el valor de la propiedad `main` se inicializa a index.js. Explicaremos un poco el interés de esta propiedad:

Si su aplicación se convierte en un paquete (incluyendo uno o varios códigos reutilizables), la propiedad `main` hace referencia al código, que es el punto de entrada en el paquete durante la ejecución de la instrucción `require()`.

3.2 Creación de un primer servidor Node.js de prueba

Va a escribir su primer servidor (el clásico «Hello World»), utilizando el módulo HTTP que ofrece Node.js.

Escriba el código de este servidor en el archivo helloConNode.js:

```
var http = require('http');

var server = http.createServer((request, => response){
    response.end('Hello World de Node.js');
});

server.listen(8888);
```

Dos observaciones:

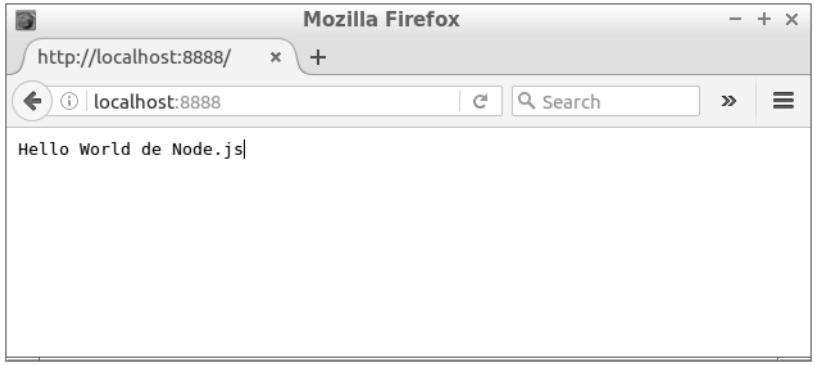
– La función de callback se ejecuta una vez que el servidor HTTP se ha creado y devuelve al cliente (es decir, al navegador) una cadena de caracteres bruta («Hello world de Node.js»).
– El servidor HTTP funciona aquí en el puerto 8888 (elegido arbitralmente).

Para ejecutar este servidor, escriba el siguiente comando en un terminal (en Linux o macOS) o en una línea de comandos (en Windows):

```
node helloConNode.js
```

▶ Para probar este servidor, abra su navegador y en la barra de URL, invóquelo con la siguiente URL: http://localhost:8888 o simplemente con: localhost:8888.

Y Node.js le saluda:



localhost hace referencia a la dirección IP de su ordenador (es un alias para 127.0.0.1).