# Creación de un servidor web REST y un cliente REST con Node

K. Peña, E. Simbaña, MJ. Vizuete

**Resumen**
Fielding trabajó junto con Tim Berners-Lee y otros para aumentar la escalabilidad de la Web. Para estandarizar sus diseños, escribieron una especificación para la nueva versión del Protocolo de transferencia de hipertexto, HTTP. La adopción de estos estándares se extendió rápidamente por la Web y allanó el camino para su crecimiento continuo. "Representational State Transfer" (REST) es el nombre que Fielding asignó a sus descripciones del estilo arquitectónico de la Web. Los servicios web REST APIS son servidores web especialmente diseñados que respaldan las necesidades de un sitio o cualquier otra aplicación.(Masse, M.2011)

## I.INTRODUCCIÓN

El objeto de investigación de este artículo es comprender el modo en que se desarrolla el diseño de un servidor web Rest y un cliente rest en el cual está basado en un sistema basado en cliente-servidor, mismo en el que clientes y los servidores tienen partes distintas que desempeñar y todo esto se logrará través de la herramienta Node-Red con nodos http.Esto fin de que el usuario forme parte activa de esto.

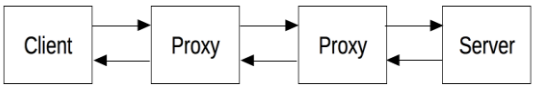
## II. MARCO TEÓRICO

### A) PROTOCOLO HTTP

HTTP (Protocolo de Transferencia de Hipertexto), es un conjunto de reglas acordadas para transferir texto con atributos propios de Internet, que permite a dos máquinas comunicarse entre sí.
Se trata de un protocolo que opera a través de solicitudes entre un cliente y un servidor. Este término es utilizado para describir el lenguaje basado en texto. No importa cómo se desarrolle, el objetivo del servidor será siempre entender y devolver respuestas de texto sencillo[1].
Los clientes y servidores se comunican intercambiando mensajes individuales. Los mensajes que envía el cliente, se llaman *peticiones*, y los mensajes enviados por el servidor se llaman *respuestas*.

**Arquitectura de los sistemas basados en HTTP**

Cada petición individual se envía a un servidor, el cual la gestiona y responde. Entre cada *petición* y *respuesta*, hay varios intermediarios, normalmente denominados proxies, los cuales realizan distintas funciones, como: gateways o caches[2].




**Cliente: el agente del usuario**

Cada conversación en la web comienza con una petición, la cual es enviada por el cliente que espera la respuesta.
Ejemplo de petición en lenguaje HTTP:

```
GET / HTTP/1.1
Host: google.com
Accept: text/html
User-Agent: Chrome/31.0.1650.57 (Macintosh; Intel Mac OS X 10_9_0)
```

Este mensaje comunica todo lo necesario acerca de qué recurso exactamente está solicitando el cliente.
La primera línea de una petición HTTP es la más importante y contiene dos cosas:

- el **URI** (Identificador de recursos uniforme)
- el **método HTTP**

El URI (e.g. /, /contact, etc) es la única dirección o ubicación que identifica el recurso que el cliente quiere.
El método HTTP (e.g. GET) define lo que quieres hacer con el recurso[2]. Los métodos HTTP son los verbos de la petición y definen las pocas maneras comunes que pueden actuar sobre el recurso:

- GET Recuperar el recurso del servidor
- POST Crear un recurso en el servidor
- PUT Actualizar el recurso en el servidor

- DELETE Borrar el recurso del servidor

**El servidor Web**

Una vez que el servidor ha recibido la petición, sabe exactamente qué recurso necesita el cliente (vía URI) y qué es lo que el cliente quiere hacer con ese recurso (vía método HTTP).

Ejemplo de respuesta enviada al navegador:

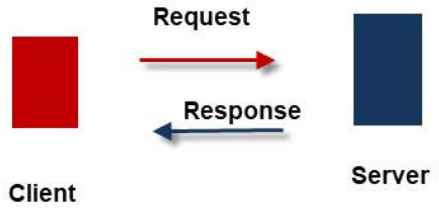
```
HTTP/1.1 200 OK
Date: Sun, 01 Dec 2013 18:17:45 GMT
Server: Apache/2.2.22 (Ubuntu)
Content-Type: text/html

<html>
    <!-- ... -->
</html>
```

La respuesta HTTP contiene el recurso solicitado, así como otra información acerca de la respuesta. En la primera línea se encuentra el código de estado de respuesta HTTP (200 en este caso). El código de estado comunica el resultado global de la solicitud [3].

Existen diferentes códigos de estado que indican éxito, error o que el cliente necesita hacer algo (e.g. redirigir a otra página). Los códigos de estado más comunes son:

- 200 OK. Indica éxito.
- 304 Not Modified. Esto muestra que el recurso en cuestión no ha cambiado y que el navegador debe cargarlo desde su caché.
- 404 Not Found. Esto sugiere que el recurso no se encuentra en el servidor.
- 401 Authorization Required. Esto indica que el recurso está protegido y requiere credenciales válidos antes de que el servidor pueda conceder el acceso.
- 500 Internal Error. Esto significa que el servidor ha tenido un problema procesando la petición.

Al igual que la petición, una respuesta HTTP contiene piezas de información adicionales conocidas como cabeceras HTTP o *headers*. El cuerpo o *body* de un mismo recurso puede ser devuelto en múltiples formatos diferentes como HTML, XML o JSON(Rau.SS ,2004).




HTTP Protocol Basics

**SERVIDOR WEB REST**

REST (Transferencia de Estado Representacional) se trata de un estilo de arquitectura de software que se utiliza para describir cualquier interfaz entre diferentes sistemas que utilicen HTTP para comunicarse [4]. Dicho estilo arquitectónico se caracteriza por seguir los siguientes principios:

- Cliente-servidor: las aplicaciones REST tienen un servidor que administra los datos y el estado de la aplicación. El servidor se comunica con un cliente que maneja las interacciones del usuario.
- Sin estado: los servidores no mantienen ningún estado de cliente. Los clientes gestionan el estado de su aplicación. Sus solicitudes a los servidores contienen toda la información necesaria para procesarlas.
- En caché: los servidores deben marcar sus respuestas como almacenables en caché o no. Por lo tanto, las infraestructuras y los clientes pueden almacenarse en caché cuando sea posible para mejorar el rendimiento. Pueden deshacerse de información que no se puede almacenar en caché, por lo que ningún cliente utiliza datos obsoletos.
- Interfaz uniforme: esta es la característica central que distingue el estilo arquitectónico REST de otros estilos basados en red. Los servicios REST proporcionan datos como recursos, con un espacio de nombres coherente.
- Sistema en capas: los componentes del sistema no pueden "ver" más

allá de su capa. Por lo tanto, puede agregar fácilmente equilibradores de carga y proxies para mejorar la seguridad o el rendimiento.
- Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son Get, Post, Put, Delete.

**Qué es una API REST**
Es una interfaz para conectar varios sistemas basados en el protocolo HTTP, el cual sirve para obtener y generar datos y operaciones sobre esos mismos datos en formatos muy específicos, como XML y JSON, siendo el segundo el más usado en la actualidad ya que es más ligero y legible en comparación al formato XML[5].

**Nodo http**


-Para utilizar el servidor y el cliente HTTP, es necesario require('http').
Las interfaces HTTP en Node.js están diseñadas para admitir muchas características del protocolo que tradicionalmente han sido difíciles de usar. En particular, mensajes grandes, posiblemente codificados por fragmentos. La interfaz tiene cuidado de nunca almacenar en búfer solicitudes o respuestas completas, de modo que el usuario pueda transmitir datos [6].

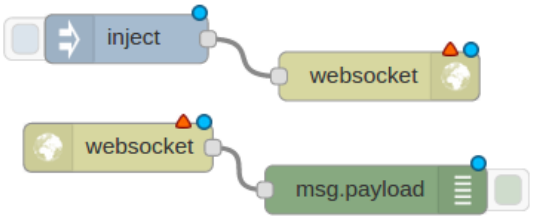
Hay tres nodos http principales.
- http-in: se utiliza para configurar un servidor web
- Respuesta http: se usa con http-in para enviar respuestas.
- Solicitud http: se utiliza para realizar solicitudes http, es decir, un cliente http.

El nodo de solicitud http se puede utilizar para.
- Recuperar páginas web de un sitio web
- Realización de una solicitud de API
- Enviar y recibir datos JSON a un sitio web o API.
- etc

El nodo enviará una solicitud y recibirá la respuesta.
El nodo maneja tanto la solicitud como la respuesta [7].

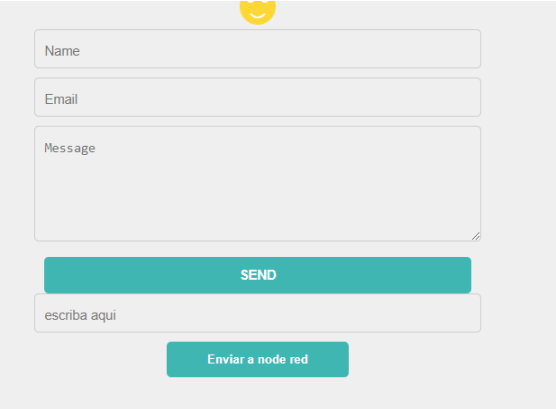
**Nodo http in y http response**

El HTTP In y el HTTP Response par de nodos son el punto de partida para todos los extremos HTTP que se crean.
Cualquier flujo que comience con un HTTP In nodo debe tener una ruta a un HTTP Response nodo, de lo contrario, las solicitudes finalmente se agotarán.
El HTTP Response nodo utiliza la payload propiedad de los mensajes que recibe como cuerpo de la respuesta. Se pueden usar otras propiedades para personalizar aún más la respuesta; se tratan en otras recetas.
El Template nodo proporciona una forma conveniente de incrustar un cuerpo de contenido en un flujo. Puede ser deseable mantener dicho contenido estático fuera del flujo.
Si ha activado la autenticación http, es posible que deba agregar su ID de usuario y contraseña al comando curl. p.ej
**Nodos websocket**




Los websockets son otra capacidad de comunicación útil (Melnikov, A.2011) que está integrada en Node-RED a través del nodo websocket . Los Websockets proporcionan una conexión TCP dúplex y fueron diseñados para permitir que los navegadores web y los servidores mantengan un 'canal de retorno' que podría usarse para aumentar el HTTP tradicional. Interacciones, lo que permite a los servidores actualizar las páginas web sin que el cliente realice una nueva solicitud de extracción [8].
El nodo websocket viene en dos versiones, entrada y salida, lo que le permite escuchar los datos entrantes (entrada) o enviar

(salida) en un websocket. La versión de salida está diseñada para verificar si la carga útil de salida se originó en un websocket en un nodo, en cuyo caso responde al remitente original. De lo contrario, transmitirá la carga útil a todos los websockets conectados. Además, los nodos websocket de entrada y salida se pueden configurar como servidor o cliente: en el modo de servidor 'escuchan' una URL y en el modo de cliente se conectan a una dirección IP específica [9].




## III. SERVIDOR WEB REST CON NODE RED

### Explicación del código fuente

Para la utilización del nodo http recurrimos a una estructura html en donde usaremos la etiqueta form para poder enrutar el host a nuestro servidor dentro de la etiqueta form implementamos un action que nos dirija al localhost generado por defecto por node red incluyendo el nombre atribuido en nuestra configuración del nodo en nuestro caso es test consecuente a esto generamos entradas de texto que nos permitan enviar información al lugar enrutado es decir a nuestro servidor

```
<!-- Form section start -->
<div class="section">
  <h1>…</h1>
  <svg class="face" height="100" width="100">…</svg>
  <form action="http://localhost:1880/test" method="get">
    <input name="name" placeholder="Name" type="text" required>
    <br>
    <input name="email" placeholder="Email" type="email" required>
    <br>
    <textarea name="message" placeholder="Message" required></textarea> == $0
    <input type="submit" value="SEND" class="submit">
  </form>
  <form action="http://localhost:1880/test" method="get">…</form>
</div>
<!-- Form section end -->
```

El resultando de esta codificación lo podemos ver a continuación, cada clase Name, Email and Message retendrá un distinto objeto el cual lo proporciona el cliente

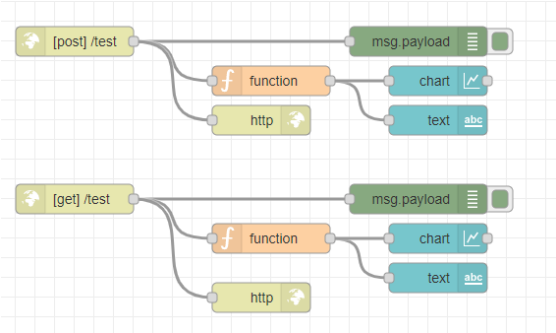
## IV.METODOLOGÍA

En el desarrollo de este estudio se empleó principalmente el método de revisión bibliográfica, ya que ayudándonos de diversas fuentes bibliográficas obtuvimos datos e información del tema respectivo los cuales fueron analizados, esto nos permitió afianzar los conceptos necesarios para el desarrollo de los ejemplos. La investigación se ejecutó por medio de herramientas como Node-Red, así como la herramienta Github y del mismo modo la información recolectada permitió reconocer los principales métodos a utilizar en una arquitectura web rest de la misma forma se encontró la herramienta XAMPP para la creación de un servidor local en el cual se pueda trabajar

## V. RESULTADOS

A partir del análisis y revisión de información se logró desarrollar y ejecutar a través de la plataforma de Node-Red,un servidor web Rest y un cliente Rest; en dichos programas se visualizan nodos http in, http request, http response. Estos servirán a partir de métodos sean POST, GET, PUT, DELETE, PATCH. Se utilizó 2 de estos métodos los cuales son post y get los que nos permiten dar una idea práctica del cómo funcionan y cuáles son sus utilidades también se generó una página web estructurada en HTML.

Se implementó la conexión de la página y el servidor a partir de la etiqueta form y la sentencia acción que logra la correcta recepción de información por parte del servidor en forma de objeto




## VI. CONCLUSIONES

-El protocolo HTTP es amplio y fácil de usar. Su estructura cliente-servidor, junto con la capacidad para usar cabeceras, le permite evolucionar con las nuevas y futuras aplicaciones en Internet.

-La principal ventaja del uso de una API REST reside en la independencia que proporciona frente a cualquier consumidor, sin importar el lenguaje o plataforma con el que se acceda a ella. Esto permite que una misma API REST sea consumida por varios clientes y que el cambio a cualquier otro tipo de consumidor no provoque impacto alguno en ella.

-La separación entre el cliente y el servidor hace que se pueda migrar a otros servidores o bases de datos de manera transparente, siempre y cuando los datos se sigan enviado de manera correcta. APIs REST es una de las arquitecturas web más utilizadas actualmente por la flexibilidad que aporta a cualquier entorno de trabajo sea cual sea su naturaleza.

## VII. RECOMENDACIONES

- Al realizar operaciones que modifican el estado de un objeto, se debe utilizar preferentemente los métodos POST, PUT y DELETE. El método GET se usa para retornar solamente una versión de lectura del recurso.
- Se deben asegurar las Apis que se desarrollen, ya que cuando se trata de aplicaciones web es muy sencillo visualizar lo que está detrás del código.

## VIII. BIBLIOGRAFÍA

[1]Vega Jiménez, D. (2016). Estudio de prestaciones del protocolo HTTP versión 2 (Bachelor's thesis).

[2]Souza Silva, A. L., & Silva, R. C. M. F. (2009). Protocolo http x protocolo https. Nucleus, 6(1), 1-8.

[3] Mozilla and individual contributors. (7 de Agosto de 2020). Generalidades del protocolo HTTP. Obtenido de https://developer.mozilla.org/es/docs/Web/HTTP/Overview

[4] Innovation & Entrepreneurship Business School. (2019). Qué es Api Rest y por qué debes de integrarla en tu negocio. Obtenido de https://www.iebschool.com/blog/que-es-api-rest-integrar-negocio-business-tech/

[5] BBVAOpen4U. (23 de Marzo de 2016). API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos. Obtenido de https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

[6] Desarrollo de una aplicación web usando Node-RED. (2019). IBM Developer. https://developer.ibm.com/es/articles/desarrollo-de-una-aplicacion-web-usando-node-red/

[7] Lea, V. A. P. B. R. (2018, 22 noviembre). Node-RED: Lecture 3 – Example 3.7 Using http nodes with Node-RED – Node RED Programming Guide. Using Websockets with Node-RED. http://noderedguide.com/node-red-lecture-3-example-3-7-using-httpsnodes-with-node-red/

[8] How do I use WebSockets in Node-RED to control IO on a BB-400? - Brainboxes - Industrial Ethernet IO and Serial. (2020). Brainboxes. http://www.brainboxes.com/faq/items/websockets-in-node-red-control-io-bb-400

[9] Set up websocket communication using Node-RED between a Jupyter notebook on IBM Watson Studio and a web interface. (2017). IBM Developer. https://developer.ibm.com/technologies/analyt.