

# Memoria de práctica

# Ejemplos

# (DSD P4S1)



---

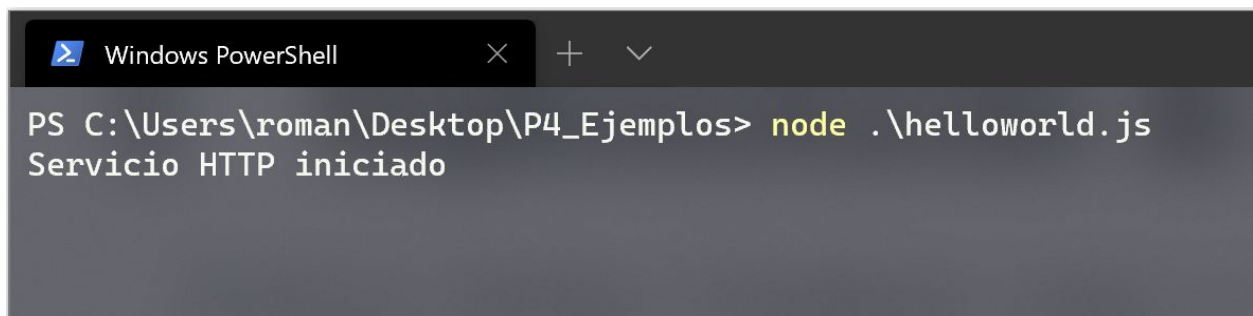
Román Larrosa Lewandowska  
mayo 2020

## 1. Introducción

En el siguiente documento se detalla el proceso de prueba de las herramientas Node.js, Socket.io y MongoDB necesarias para la ejecución de la práctica 4.

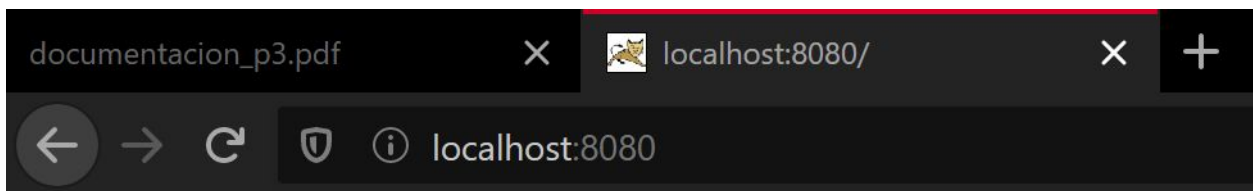
## 2. Implementación de servicios con Node.js

El primer ejemplo es un sencillo “Hola Mundo” mediante Node.js. Tras la instalación de la herramienta ejecuto en el terminal:



```
Windows PowerShell
PS C:\Users\roman\Desktop\P4_Ejemplos> node .\helloworld.js
Servicio HTTP iniciado
```

El servicio ha quedado iniciado, lo probamos desde un navegador en <http://localhost:8080/>

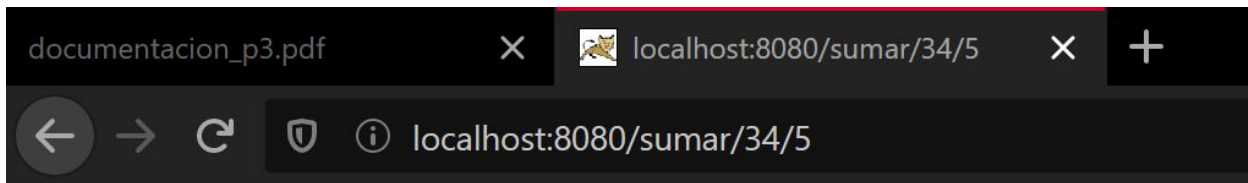


Hola mundo

Para el siguiente ejemplo el servicio implementa una calculadora a la que se accede mediante <http://localhost:8080/<operacion>/<op1>/<op2>> donde operación puede ser sumar, restar, producto o dividir, y op1 y op2 los operadores. Al igual que en el ejemplo anterior, ejecutamos en el terminal:

```
Windows PowerShell
PS C:\Users\roman\Desktop\P4_Ejemplos> node .\calculadora.js
Servicio HTTP iniciado
```

Y probamos el servicio desde el navegador ( <http://localhost:8080/sumar/34/5> ) :

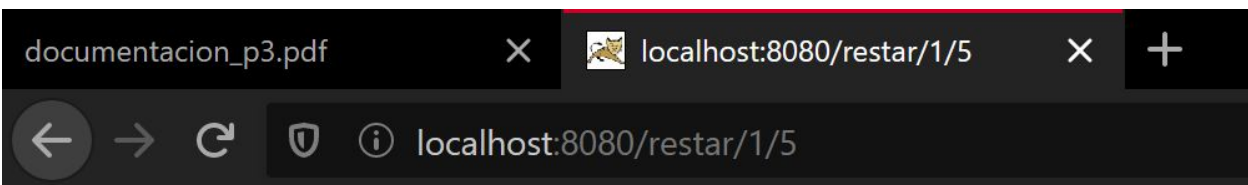


39

Se puede realizar una mejora del servicio añadiendo una interfaz para la calculadora en caso de que el usuario acceda a través de <http://localhost:8080/> manteniendo la funcionalidad si se introducen el operador y los operandos:

```
Windows PowerShell
PS C:\Users\roman\Desktop\P4_Ejemplos> node .\calculadora-web.js
Servicio HTTP iniciado
```

El servicio sigue respondiendo correctamente a las peticiones anteriores:



-4

Pero ahora, además, añade una interfaz gráfica

documentacion\_p3.pdf X Calculadora X +

← → ↻ 🔒 ⓘ localhost:8080

Valor1:

Valor2:

Operación:

Calcular

Que al pulsar “Calcular” producirá:

documentacion\_p3.pdf X Calculadora X +

← → ↻ 🔒 ⓘ localhost:8080

Valor1:

Valor2:

Operación:

Calcular

18

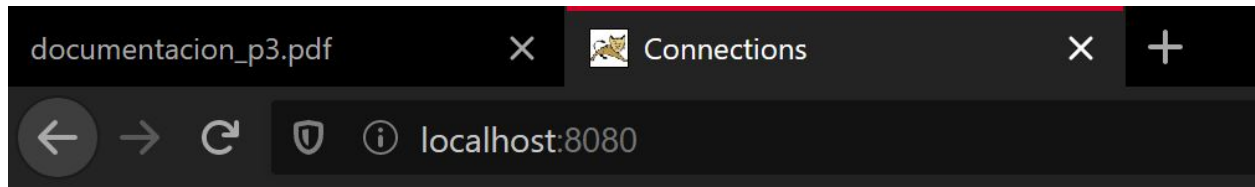
### 3. Aplicaciones en tiempo real con Socket.io

El servicio para probar Socket.io muestra en el navegador del cliente cuántos clientes hay conectados al servicio. Se ejecuta como los anteriores:

Windows PowerShell X + ▾

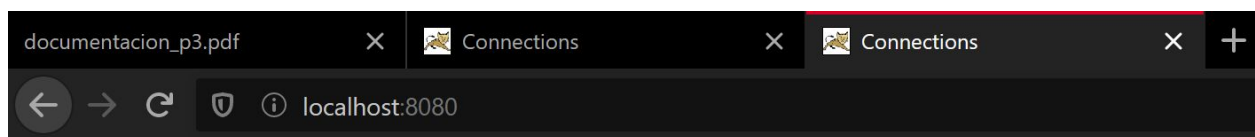
```
PS C:\Users\roman\Desktop\P4_Ejemplos> node .\connections.js
Servicio Socket.io iniciado
```

Y en un navegador quedaría así:



Mensaje de servicio: Hola Cliente!

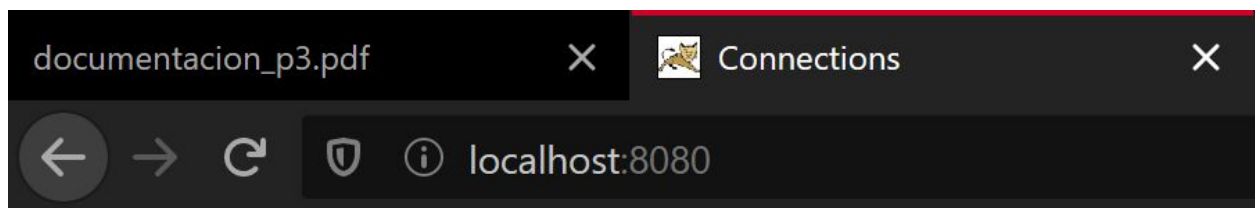
- ::1:51001



Mensaje de servicio: Hola Cliente!

- ::1:51001
- ::1:51007

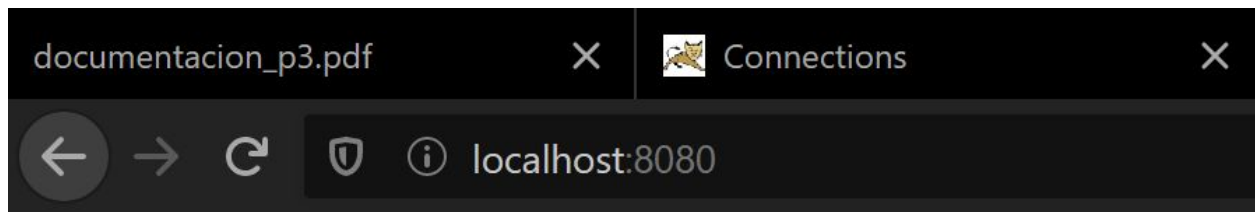
Y en caso de cerrar el servicio (Ctrl + C), en el navegador se mostraría:



El servicio ha dejado de funcionar!!

- ::1:51001
- ::1:51007

Sin embargo, cuando el servicio vuelve a estar operativo el navegador automáticamente se actualiza con el indicador de que el servicio está activo:



Mensaje de servicio: Hola Cliente!

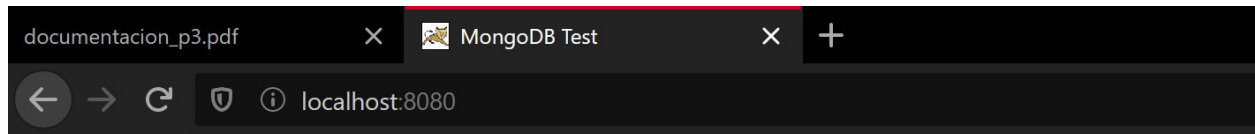
- ::1:51051
- ::1:51051

## 4. Uso de MongoDB desde Node.js

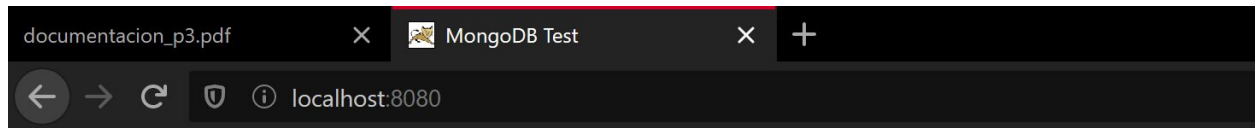
Para probar este ejemplo es necesario tener una base de datos MongoDB e instalar las dependencias. El servicio que ejecuta es parecido al ejemplo anterior. Cada vez que un cliente se conecte al servicio se guarda en la base de datos su host, puerto y el momento en el que se conecta. A continuación se recupera desde la BD el historial de conexiones realizadas desde ese host:

```
Windows PowerShell
PS C:\Users\roman\Desktop\P4_Ejemplos> node .\mongo-test.js
Servicio MongoDB iniciado
```

Y al conectarse desde el navegador se mostrará la siguiente información, que seguirá ampliándose si recargamos la página:



- {"\_id":"5eb171ff0df64d389c16ddc5","host":"","port":51145,"time":"2020-05-05T14:02:39.991Z"}



- {"\_id":"5eb171ff0df64d389c16ddc5","host":"","port":51145,"time":"2020-05-05T14:02:39.991Z"}
- {"\_id":"5eb1720f0df64d389c16ddc6","host":"","port":51148,"time":"2020-05-05T14:02:55.939Z"}
- {"\_id":"5eb172100df64d389c16ddc7","host":"","port":51148,"time":"2020-05-05T14:02:56.698Z"}