AGO
10

SPRINT 3 - MEETING 41

Servidor web con Express I

Grabación

Presentación

El equipo docente subió contenido extra de la meeting

Ahora sí, construyamos un servidor web

> *"Los servidores web son uno de los aspectos más importantes de Internet, ya que se trata de los encargados de despachar las páginas a los usuarios." —Santiago Borges. Escritor en Infranetworking Hosting.*

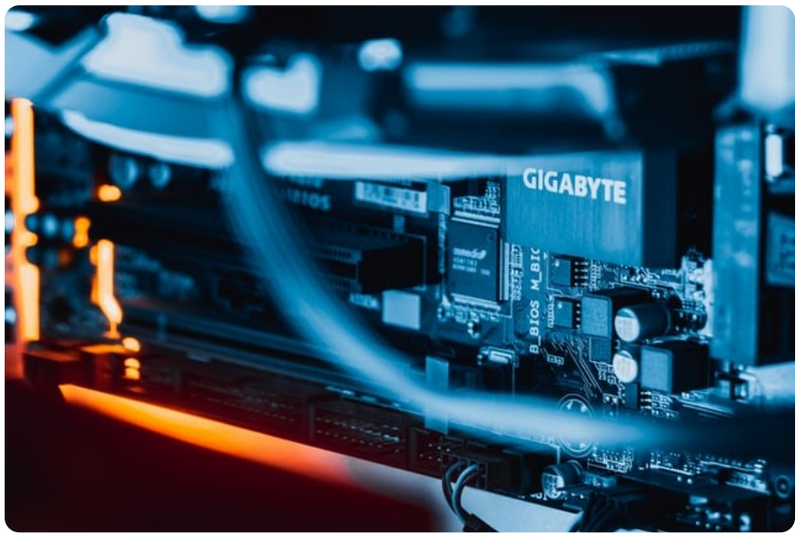


Photo by Florian Krumm on Unsplash

Nos estamos acercando al desarrollo de aplicaciones o sitios web desde el back-end, ¡pero tenemos que ir paso a paso! Ya te has familiarizado con este nuevo entorno, aprendiste a ejecutar código JavaScript en Node JS, a crear tus módulos e incluir los de otros/as en tus proyectos. Ya cuentas con conocimientos de la arquitectura cliente-servidor y cómo se comunican a través del protocolo HTTP, ¡es tiempo de comenzar a construir tus servidores a través de Express JS, un framework ágil y sencillo!

Express JS es uno de los paquetes más populares: lo utilizan compañías como IBM, Accenture, Fox Sport, Mulesoft y Myntra. Fue creado por TJ Holowaychuk y, como mencionamos en la bitácora anterior, es un framework minimalista que permite crear servidores y aplicaciones web

Las ventajas de su uso son varias:

- mantiene actualizadas sus características principales;
- tiene la capacidad de manejar muchas solicitudes a la vez;
- es simple, liviano y fácil de utilizar, por lo tanto, es una excelente opción para quienes recién comienzan a desarrollar aplicaciones;
- además, al sustentarse por los grandes aportes de la comunidad de developers, siempre encontrarás personas que puedan ayudarte ante cualquier duda que tengas.

Según MDN mozilla, proporciona los siguientes mecanismos:

- Escritura de manejadores de peticiones con diferentes verbos HTTP (GET, POST, PUT, DELETE) en diferentes caminos URL (rutas). Por ejemplo, para insertar, eliminar, extraer o actualizar información puedes utilizar el método correspondiente.
- Integración con motores de renderización de "vistas", para generar respuestas mediante la introducción de datos en plantillas. Si necesitas trabajar con una vista para retornar información con un formato específico puedes añadir un sistema de templates para no tener que escribir HTML dentro de tu endpoint.
- Establecer ajustes de aplicaciones web como, por ejemplo, qué puerto usar para conectarse.
- Añadir procesamiento de peticiones "middleware" adicionales en cualquier punto antes de que la petición llegue a su Endpoint. El próximo encuentro profundizaremos sobre este punto.

¿Cómo lo uso?

¿Te gustaría empezar a utilizar **Express JS** y conocer qué funcionalidades tiene? A continuación aprende cómo crear un servidor HTTP, y cómo dejarlo disponible para que pueda recibir pedidos.

1. **Instálalo.** Para comenzar con Express, primero tienes que instalarlo dentro de tu proyecto.

2. **Importa el módulo de Express.** Una vez instalado, importa el módulo de express y asigna a una variable la la respuesta del método express() para tener todas las funcionalidades disponibles.

```
var express = require('express');
var app = express();
```

3. **Crea las rutas.** Entre los diferentes métodos que tiene Express, tenemos el de definir rutas mediante el uso de cada verbo HTTP, como GET, POST, PUT y DELETE. Cada definición cuenta con dos argumentos. El primero es la ruta que queremos marcar, y el segundo es un callback que tiene la función de manejar una solicitud con dos parámetros: request (solicitud) y response (respuesta) Estos parámetros permiten obtener información de la solicitud y enviar la respuesta al cliente.

```
app.get('/mi_ruta', function (req, res) {
    res.send('Hello World!');
});
```

4. **Inicia el servidor.** Hasta el momento hemos instalado express, importado el módulo y definimos un ruta pero el servidor aún no escuchando peticiones, para esto debemos iniciarlo e indicar en que puerto estará escuchando peticiones de los clientes, para esto tenemos el método listen donde el primer parámetro en que puerto escuchará peticiones y el segundo es un callback opcional que se ejecutará cuando el servidor inicie.

```
app.listen(3000, function () {
    console.log('Example app listening on port 3000!');
});
```

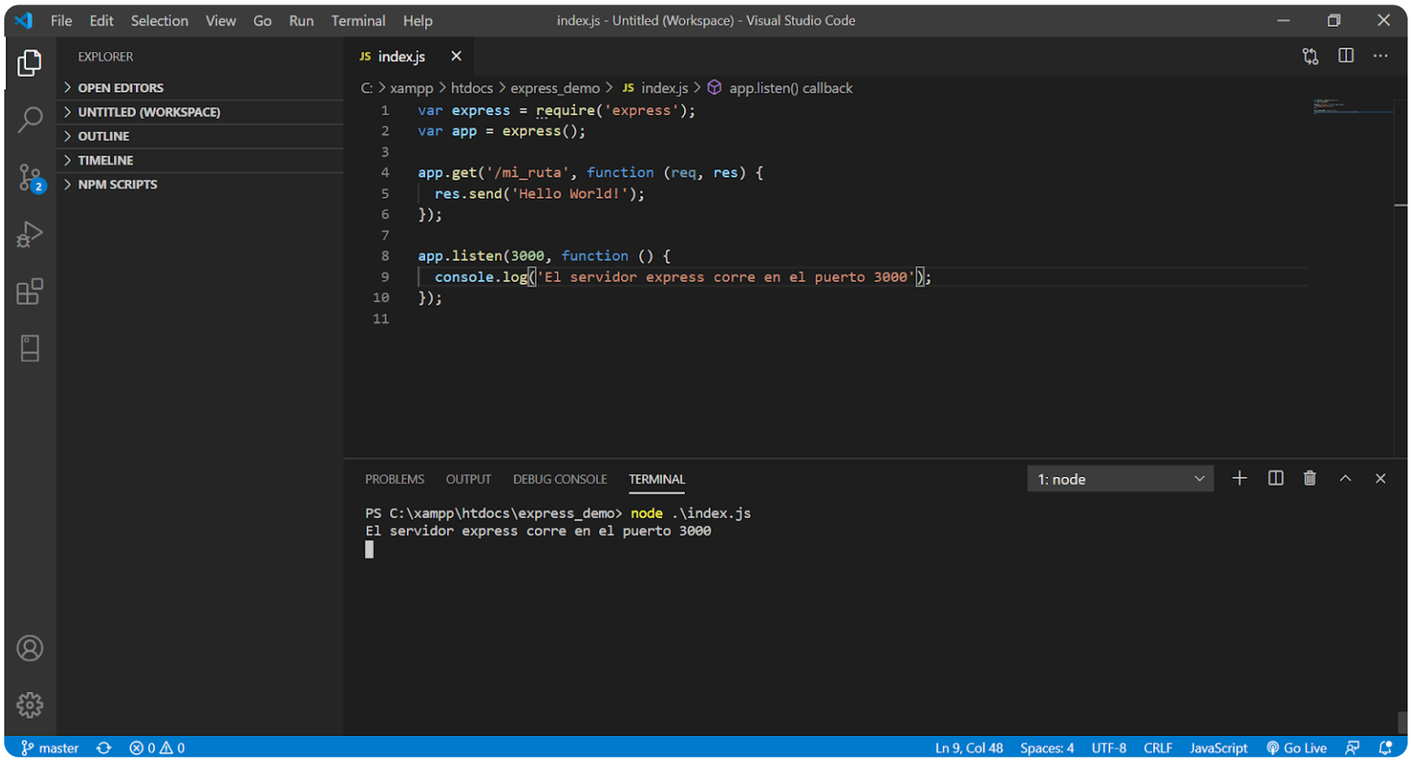
Hasta aquí hemos iniciado un servidor en el puerto 3000 y escuchará peticiones en una ruta "mi_ruta" de tipo GET. Aquí el código completo:

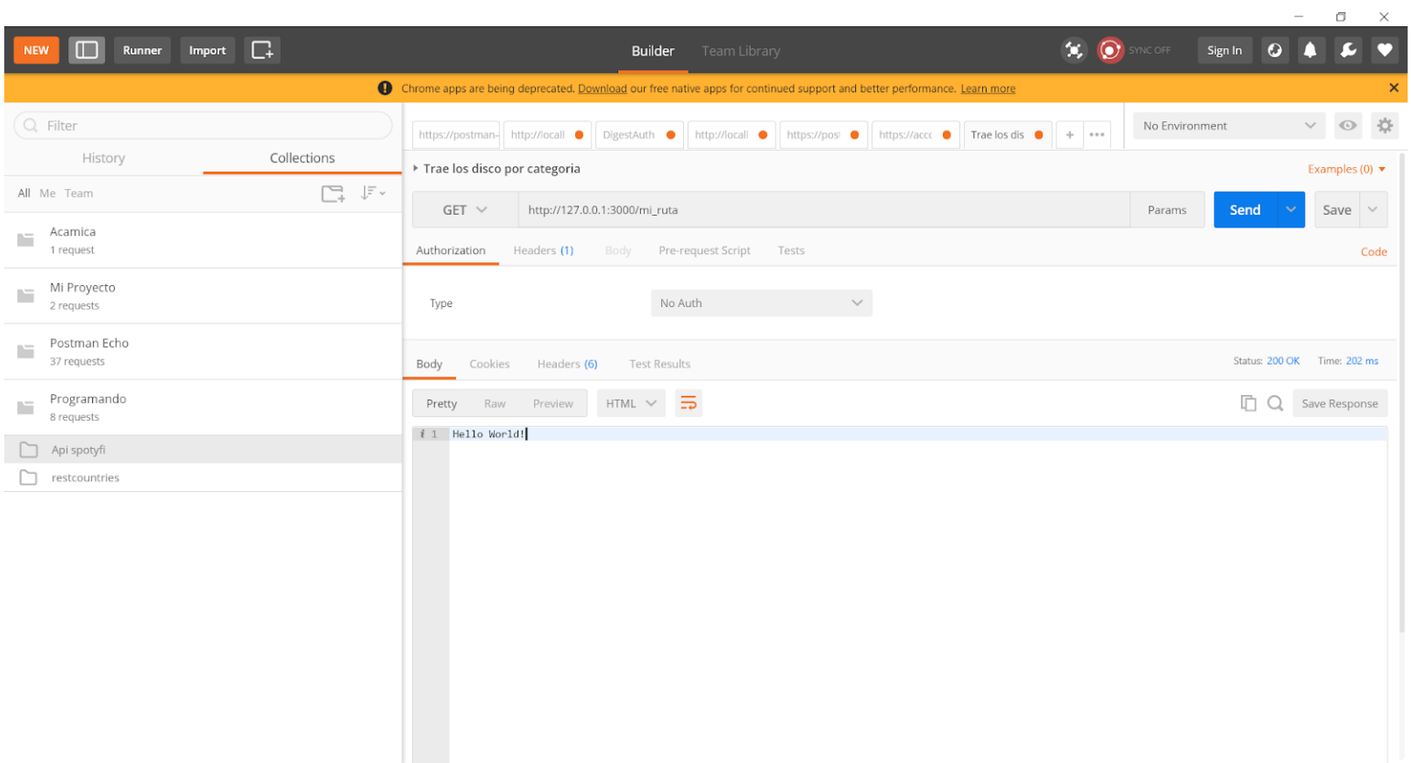
```
var express = require('express');
var app = express();
```

```
});

app.listen(3000, function () {
    console.log('El servidor express corre en el puerto 3000');
});
```

5. **Testea.** Ya has hecho lo que necesitas, ¡es momento de probarlo! Abre el Postman y hace un request tipo GET a **http://127.0.0.1:3000/mi_ruta**





En resumen

acercando cada vez más a poder diseñarlas! En las próximas bitácoras seguiremos sumando herramientas para hacerlo completamente desde Express JS.

¡Prepárate para el próximo encuentro!

Profundiza

Te invitamos a conocer más sobre el tema de esta bitácora.

Herramientas

Programas necesarios para facilitar tu experiencia.

Challenge

Te proponemos el siguiente desafío, ¿te animas?

Potencia tu Talento - Entrevistas con RRHH

Tips para construir tu perfil profesional.

Contenido extra de la meeting

Recursos

Recuerden que aqui estan los archivos de la clase