

Instructor Guillo

[@guillermodelapenaruiz2345](#)

<https://www.youtube.com/@guillermodelapenaruiz2345>



INTRODUCCIÓN

¿Qué es Node.js?

Imagina que estás construyendo una casa. Tradicionalmente, para el frente de la casa (lo que los visitantes ven), usas herramientas como HTML, CSS y JavaScript. Esto es lo que se conoce como el **frontend**. Node.js, en cambio, es como si tuvieras un conjunto de herramientas superpoderosas para construir toda la estructura de la casa: la plomería, la electricidad, los cimientos, etc. Esto es el **backend**.

En términos técnicos, Node.js es un **entorno de ejecución** para JavaScript. En palabras simples, te permite usar el lenguaje de programación JavaScript para construir aplicaciones del lado del servidor (el backend). Antes de Node.js, JavaScript solo se podía ejecutar en el navegador web para hacer cosas interactivas en una página. Con Node.js, puedes usar JavaScript para crear servidores web, APIs, herramientas de línea de comandos y mucho más, ¡todo fuera del navegador!

Conceptos Clave

Para entender cómo funciona Node.js, hay dos conceptos muy importantes que debes conocer:


1. **Modelo de E/S Asíncrono no Bloqueante:** Esto suena muy técnico, pero es una de las mayores fortalezas de Node.js.
 - **E/S (Entrada/Salida):** Se refiere a cualquier operación que involucra interactuar con algo externo, como leer un archivo de tu disco duro, hacer una consulta a una base de datos o enviar una solicitud a otra página web.
 - **Asíncrono:** Esto significa que Node.js no espera a que una tarea termine para pasar a la siguiente. Imagina que le pides a alguien que te traiga un vaso de agua. En un modelo síncrono, no harías nada más hasta que te traigan el vaso. En un modelo asíncrono, le pides el vaso y mientras esperas, sigues haciendo otras cosas, como leer un libro. Cuando la persona vuelve con el agua, te avisa.
 - **No Bloqueante:** Se relaciona directamente con lo asíncrono. Significa que una operación de E/S no "bloquea" la ejecución del programa. Node.js puede manejar muchas solicitudes a la vez sin que una tenga

que esperar a que otra se complete. Esto lo hace muy eficiente, especialmente para aplicaciones que manejan muchas conexiones simultáneas, como chats en tiempo real o APIs.

2. **Motor V8 de Google:** Node.js utiliza el mismo motor V8 que usa el navegador Chrome. Este motor es el que se encarga de convertir tu código JavaScript en código de máquina, lo que permite que sea ejecutado de forma muy rápida y eficiente.

En documentos vamos a crear una carpeta de nombre

1 crear carpeta Raíz

 **multiplicar**

2 crear dentro de la carpeta raíz el archivo **index.html**







```
<> index.html > ...
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width,
6          initial-scale=1.0"/>
7      <title>Multiplicador</title>
8      <link rel="stylesheet" href="css/estilo.css" />
9  </head>
10 <body>
11     <div class="container">
12         <h1>Multiplicador</h1>
13         <input type="number" id="num1" placeholder="Primer número" />
14         <input type="number" id="num2" placeholder="Segundo número" />
15         <button id="btn">Multiplicar</button>
16         <div id="resultado"></div>
17     </div>
18     <script src="js/main.js"></script>
19 </body>
20 </html>
```

3 crear dentro de la carpeta raíz una carpeta llamada css

 **css**

Dentro de la carpeta css

Crear un archivo llamado **estilo.css**

```
css > # estilo.css >  .container
1  body {
2      font-family: Arial, sans-serif;
3      background-color:  #f1f1f1;
4      display: flex;
5      justify-content: center;
6      align-items: center;
7      height: 100vh;
8  }
9
10  .container {
11      background:  white;
12      padding: 30px;
13      border-radius: 10px;
14      box-shadow: 0 0 10px  rgba(0,0,0,0.1);
15      text-align: center;
16  }
17
18  input {
19      margin: 10px;
20      padding: 10px;
21      width: 150px;
22      border: 1px solid  #ccc;
23      border-radius: 5px;
24  }
25
26  button {
27      padding: 10px 20px;
28      background-color:  #4CAF50;
29      border: none;
30      color:  white;
31      border-radius: 5px;
32      cursor: pointer;
33  }
34
35  #resultado {
36      margin-top: 20px;
37      font-size: 1.5rem;
38  }
```

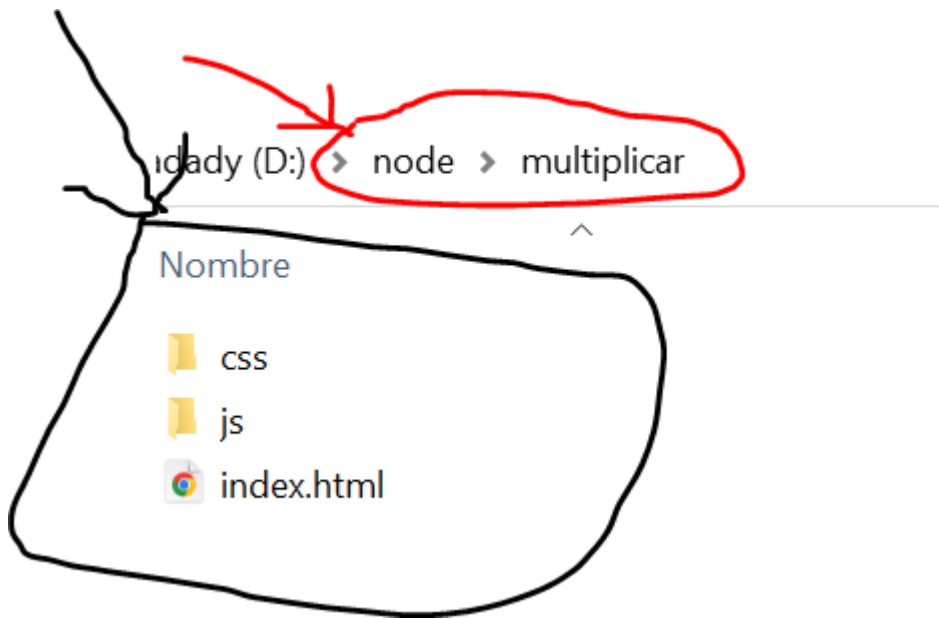
4 en la carpeta raíz vamos a crear una carpeta llamada js



Dentro de la carpeta js vamos a crear un archivo llamado main.js

```
js > JS main.js > addEventListener('click') callback > response
Tabnine | Edit | Test | Explain | Document
1 document.getElementById('btn').addEventListener('click', async ()
=> {
2   const num1 = document.getElementById('num1').value;
3   const num2 = document.getElementById('num2').value;
4
5   const response = await fetch('http://localhost:3000/
multiplicar', {
6     method: 'POST',
7     headers: {
8       'Content-Type': 'application/json'
9     },
10    body: JSON.stringify({ a: num1, b: num2 })
11  });
12
13  const data = await response.json();
14  document.getElementById('resultado').textContent = `Resultado: $
{data.resultado}`;
15  });
```

Así debe verse la carpeta raíz



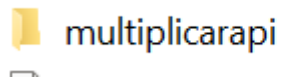
Si queremos visualizar el archivo index.html ya que es del lado dl cliente con solo darle clic el navegador debería mostrar asi.



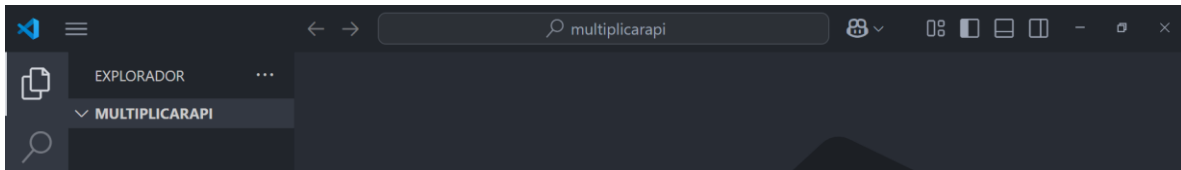
Con esto resaltar que hasta este momento ya hemos creado el consumo o Frontend, que va en una carpeta así como se indicó.

Crear Api (microservicio)

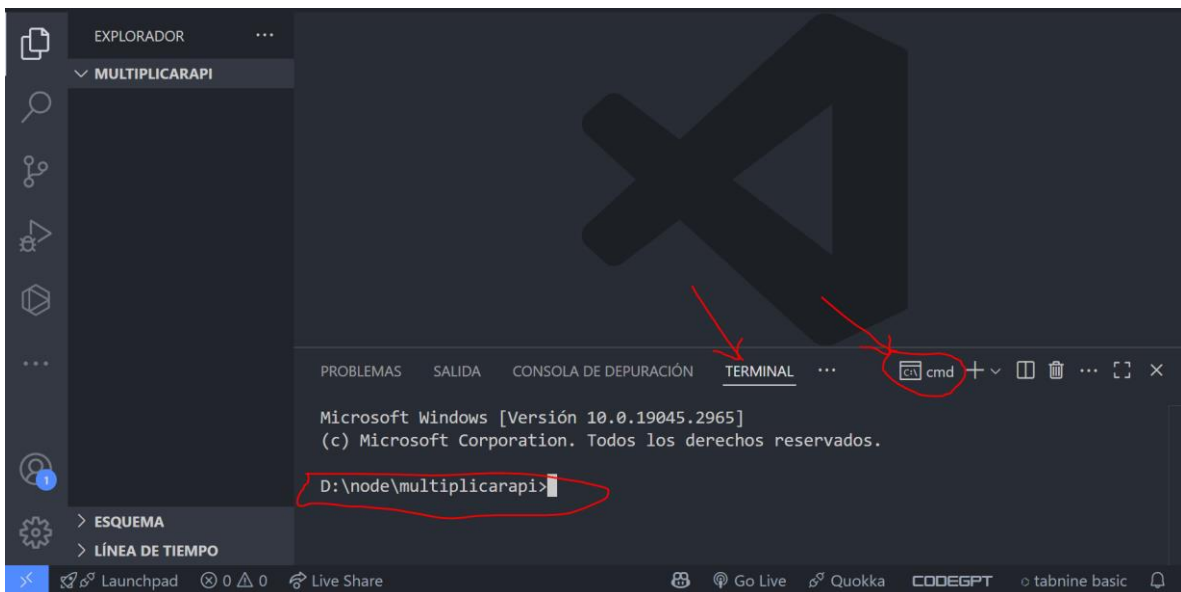
1 Vamos a crear una carpeta raíz para la api llamada



2 En esta carpeta vamos a instalar node pero para eso debemos sincronizar esta carpeta con visual code.

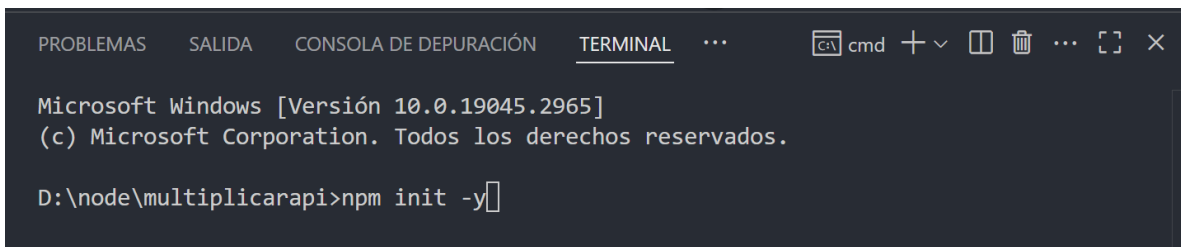


3 debemos buscar la terminal.



En esta terminal debemos instalar node con los siguientes comandos

Npm init -y



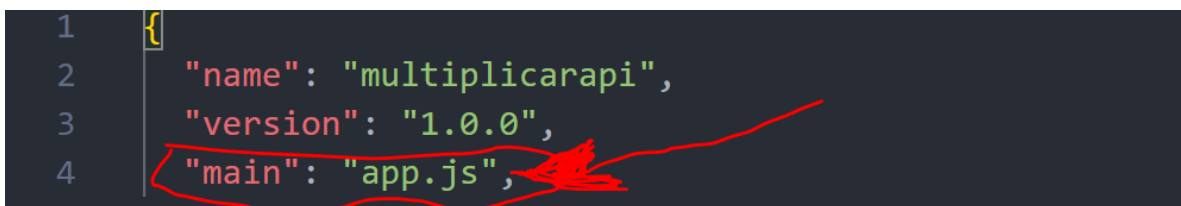
Se creara un archivo así



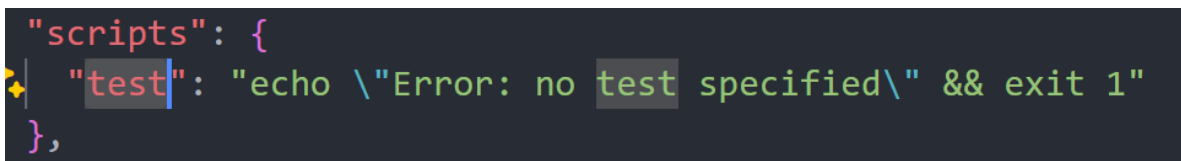
4 vamos a configurar este archivo **package.json**

Línea 4 y 6 para que cuando corramos el programa si hacemos cambio no tengamos que apagar el servidor y si volver a probar volver a prender servidor esta configuración lo hace automáticamente.

Esta línea 4 es para el nombre de mi archivo principal el nombre que llevara **.(app.js)**



La línea 6 aparece así



Ahora modificamos para que todo funcione bien.


```
5  "scripts": {  
6    "dev": "node --watch app.js"  
7  },  
  "name": "app"
```

Debe quedar como esta la imagen, dev desarrollador y node --watch espera que actualice automáticamente.

Se escribirán códigos en el package.json dependencias así.

```
6     "dev": "node --watch app.js"
7   },
8   "keywords": [],
9   "author": "",
10  "license": "ISC",
11  "description": "",
12  "dependencies": {
13    "cors": "^2.8.5",
14    "express": "^5.1.0"
15  }
```

PROBLEMAS

SALIDA

CONSOLA DE DEPURACIÓN

TERMINAL

...

14 packages are looking for funding
run `npm fund` for details

6 crear el archivo principal que fue llamado app.js

En esta parte vamos a crear el archivo principal debe llamarse obligatoriamente como le pusimos a la línea 4 del package.json yo lo llame app.js

```
{  
  "name": "multiplicarapi",  
  "version": "1.0.0",  
  "main": "app.js",  
  "scripts": {  
    "dev": "node --watch app.js"  
  },  
  "keywords": [],  
  "author": ""  
}
```

Debe quedar así

```
Abrir el chat (Ctrl+I), o seleccione un idioma (Ctrl+K M), o rellene  
con una plantilla para empezar.  
Empiece a escribir para descartar o no mostrar esto de nuevo.
```

Ahora vamos a escribir el código en app.js todo el código estará comentaría do para que sepa para que sirve cada línea.

```
JS app.js > ...  
1 //con esta linea llamamos a la libreria express  
2 const express = require('express');  
3 //con esta linea llamamos a la libreria cors  
4 const cors = require('cors'); 4.5k (gzipped: 1.9k)  
5 //con esta linea creamos una instancia de express  
6 const app = express();
```

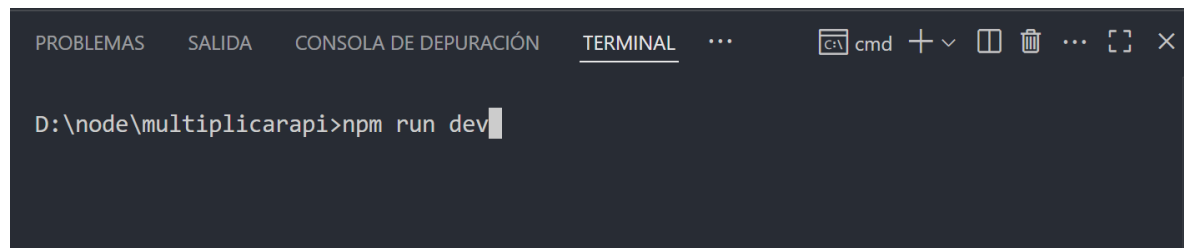
```

7 //middleware aqui se llaman
8 //con esta linea le decimos a express que vamos a usar cors
9 app.use(cors());
10 //con esta linea le decimos a express que vamos a usar json
11 app.use(express.json());
12 //esta es la funcion que se ejecuta cuando se haga
13 //la peticion a la ruta raiz
    Tabnine | Edit | Test | Explain | Document
14 app.post('/multiplicar', (req, res) => {
15     const { a, b } = req.body;
16     const resultado = Number(a) * Number(b);
17     res.json({ resultado });
18 });
19 //esta funcion es la que crea el servidor
    Tabnine | Edit | Test | Explain | Document
20 app.listen(3000, () => {
21     console.log('API ejecutándose en http://localhost:3000');
22 });

```

Con esto debemos correr la api en la terminal con la configuración que hicimos en la línea 4 del package.json

Npm run dev



```

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  ...
cmd + v  [icon] [icon] [icon] [icon] [icon] [icon] [icon]
D:\node\multiplicarapi>npm run dev

```

Dee quedar asi

