

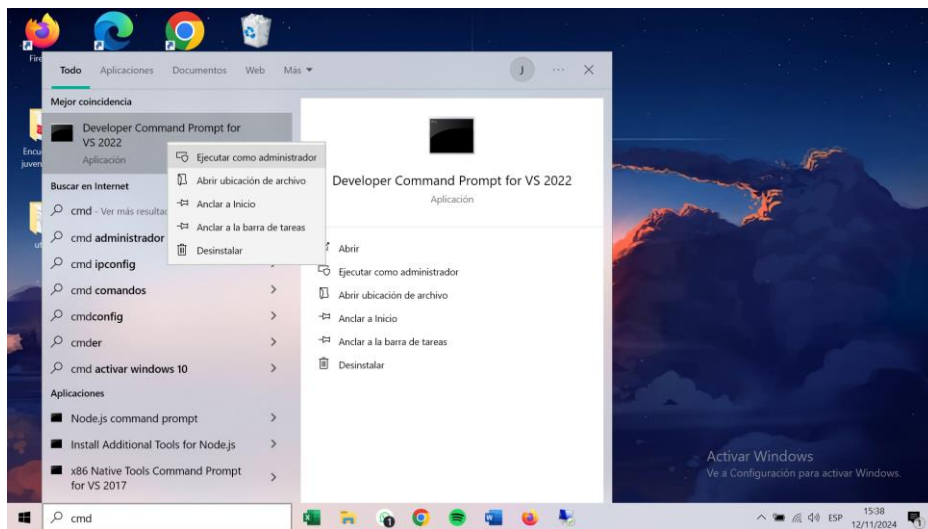
## ***Inicio de servidor y su configuración***

### ***-INSTALAR DEPENDENCIA NECESARIA (node.js)***

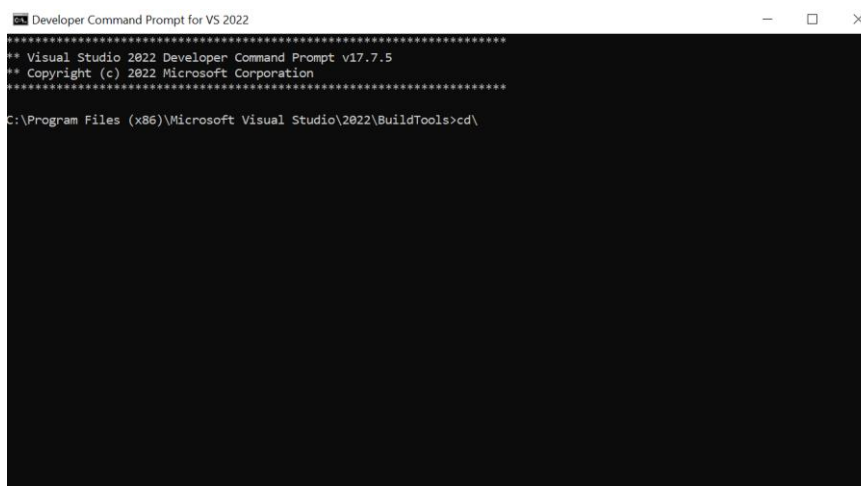
Lo primero que tenemos que hacer es bajar el entorno node.js a nuestra computadora para poder implementar todos los protocolos necesarios y levantarlo.

Para ello vamos a descargar el software desde su [sitio web](#) siempre eligiendo la configuración necesaria. Por defecto es Windows x64. Vamos a verificar que haya quedado debidamente instalado, abriendo una consola, ejecutándola como administrador.

Inicio->cmd (ejecutar como administrador, con click derecho)



Una vez dentro nos aparece algo como la siguiente imagen:



Luego limpiamos la consola donde nos coloca por defecto con el comando `cd\`

Ello nos deposita en el disco local C, la raíz de nuestra PC. Una vez ahí verificamos con el comando `node -v`

```
Developer Command Prompt for VS 2022
*****
** Visual Studio 2022 Developer Command Prompt v17.7.5
** Copyright (c) 2022 Microsoft Corporation
*****

C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools>cd\

C:\>node -v
```

```
Developer Command Prompt for VS 2022
*****
** Visual Studio 2022 Developer Command Prompt v17.7.5
** Copyright (c) 2022 Microsoft Corporation
*****

C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools>cd\

C:\>node -v
v21.4.0

C:\>
```

Vemos la versión instalada (la del ejemplo es anterior, la que bajan es valida).

Ahora navegamos hasta nuestro proyecto utilizando la siguiente secuencia de comandos: `cd desarrollo`->Enter->*nombre repositorio (ejemplo)*-> Enter

Donde veríamos lo siguiente:

```
Developer Command Prompt for VS 2022
*****
** Visual Studio 2022 Developer Command Prompt v17.7.5
** Copyright (c) 2022 Microsoft Corporation
*****

C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools>cd\

C:\>node -v
v21.4.0

C:\>cd Desarrollo

C:\Desarrollo>cd Ejemplo

C:\Desarrollo\Ejemplo>
```

Una vez posicionados en nuestro repositorio en el sistema de carpetas vamos a ejecutar los siguientes comandos: *npm init -y*

```
Developer Command Prompt for VS 2022
*****
** Visual Studio 2022 Developer Command Prompt v17.7.5
** Copyright (c) 2022 Microsoft Corporation
*****

C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools>cd\

C:\>node -v
v21.4.0

C:\>cd Desarrollo

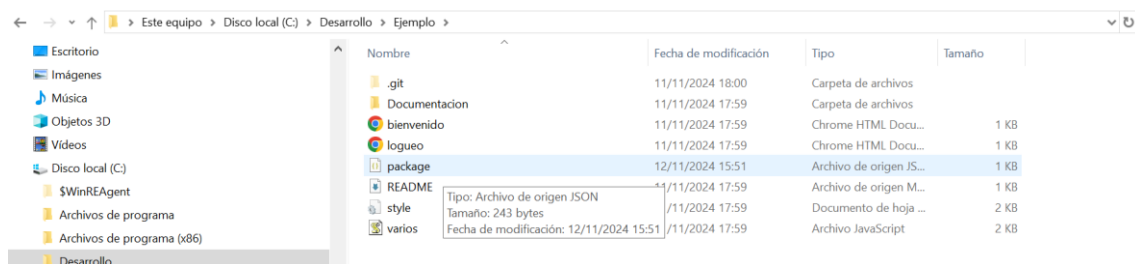
C:\Desarrollo>cd Ejemplo

C:\Desarrollo\Ejemplo>npm init -y
```

El cual nos va a permitir inicializar un proyecto sencillo que se vea con la siguiente estructura:

```
Developer Command Prompt for VS 2022
C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools>cd\
C:\>node -v
v21.4.0
C:\>cd Desarrollo
C:\Desarrollo>cd Ejemplo
C:\Desarrollo\Ejemplo>npm init -y
Wrote to C:\Desarrollo\Ejemplo\package.json:
{
  "name": "ejemplo",
  "version": "1.0.0",
  "description": "Mi primer repositorio",
  "main": "varios.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
C:\Desarrollo\Ejemplo>
```

Si vamos a nuestro explorador de archivos podemos verificar la creación de un archivo (package.json):



El mismo nos permitirá establecer una comunicación por protocolo de JSON de nuestra aplicación para manejar los datos de los formularios creados.

Volvemos a la consola (CMD) donde debemos instalar Expresso, la segunda tecnología que nos permite trabajar con nuestros datos. Una vez allí vamos a ejecutar el comando *npm install express*

```
Developer Command Prompt for VS 2022
C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools>cd\
C:\>node -v
v21.4.0
C:\>cd Desarrollo
C:\Desarrollo>cd Ejemplo
C:\Desarrollo\Ejemplo>npm init -y
Wrote to C:\Desarrollo\Ejemplo\package.json:
{
  "name": "ejemplo",
  "version": "1.0.0",
  "description": "Mi primer repositorio",
  "main": "varios.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
C:\Desarrollo\Ejemplo>npm install express_
```

Damos *Enter* lo que nos devuelve:

```
Developer Command Prompt for VS 2022
C:\Desarrollo\Ejemplo>npm init -y
Wrote to C:\Desarrollo\Ejemplo\package.json:
{
  "name": "ejemplo",
  "version": "1.0.0",
  "description": "Mi primer repositorio",
  "main": "varios.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

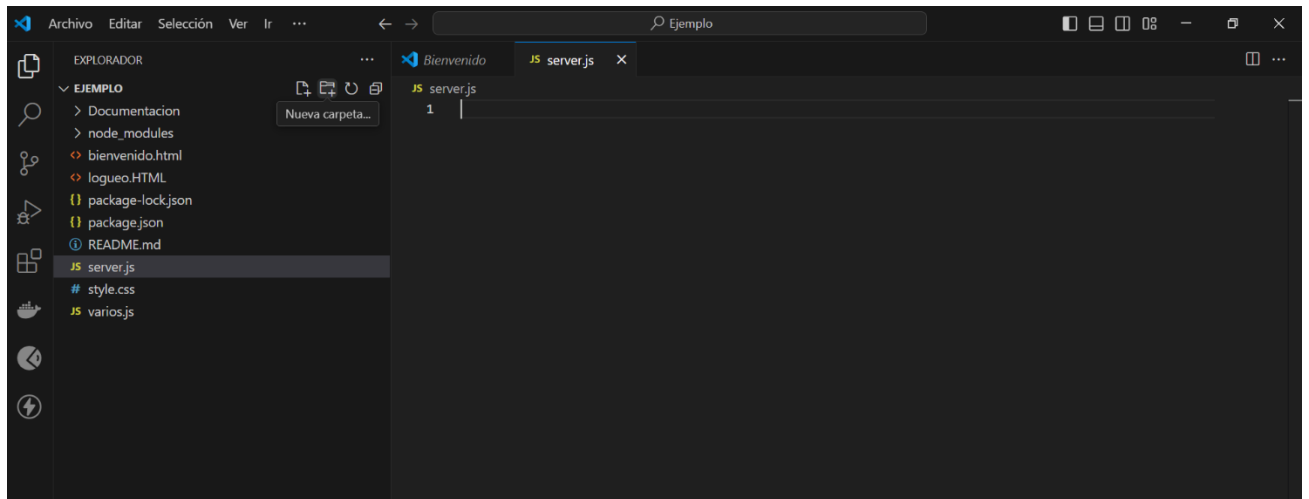
C:\Desarrollo\Ejemplo>npm install express

added 65 packages, and audited 66 packages in 2s

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

found 0 vulnerabilities
C:\Desarrollo\Ejemplo>_
```

Ahora solo nos queda crear el archivo.js que nos permita manejar las peticiones, para eso existen dos opciones. Puede crearlo desde cmd con el comando *touch server.js* o directamente desde Visual Studio Code como cualquier javascript. Recomendamos crearlo desde visual studio porque a veces por errores de las tecnologías que manejamos no reconoce dicho comando.



Una vez creado el archivo vamos a instanciar un genérico de código (vamos a ver sintaxis en clase). El mismo sirve para simplemente escuchar y poder relacionar ese servidor con un puerto de comunicación.

```
javascript Copiar código

const express = require('express');
const app = express();
const port = 3000;

// Middleware para servir archivos estáticos (opcional)
app.use(express.static('public'));

// Ruta de ejemplo
app.get('/', (req, res) => {
  res.send('¡Hola Mundo desde Express!');
});

// Iniciar el servidor
app.listen(port, () => {
  console.log(`Servidor escuchando en http://localhost:${port}`);
});
```

### **require('express'):**

Esta línea importa el paquete **Express** en tu archivo. **require()** es una función de Node.js que se usa para cargar módulos.

express es una biblioteca que simplifica la creación de servidores web y el manejo de rutas en aplicaciones Node.js.

**express():** Llamamos a **express** como si fuera una función, lo que nos devuelve una **instancia de una aplicación Express**.

La variable `app` ahora contiene esa instancia de la aplicación. A partir de aquí, podemos utilizar `app` para definir rutas, middleware, y controlar la configuración del servidor.

**port = 3000:** Aquí estamos definiendo un número de **puerto** (en este caso, el puerto 3000) en el que el servidor escuchará las solicitudes HTTP. Este valor se puede cambiar a cualquier otro número de puerto disponible (por ejemplo, 8080 o 5000).

**app.use():** Este es un **método de Express** que se utiliza para registrar **middleware**. Un middleware es una función que tiene acceso al objeto de solicitud (`req`), el objeto de respuesta (`res`), y la siguiente función en el ciclo de solicitud-respuesta.

**express.static('public'):** Aquí le estamos diciendo a Express que utilice el middleware **express.static** para servir archivos estáticos. Los archivos estáticos son aquellos que no cambian (como imágenes, archivos CSS, JavaScript, etc.).

- El directorio **public** es donde se buscan esos archivos estáticos. Si alguien accede a un archivo en la ruta `/images/imagen.jpg`, Express buscará ese archivo en la carpeta `public/images/imagen.jpg`.

**app.get():** Este método se utiliza para **definir rutas** en tu servidor. En este caso, estamos manejando solicitudes **GET** (cuando alguien accede a una URL usando el navegador, por ejemplo) en la ruta `/` (la raíz).

- El primer parámetro de `app.get()` es la **ruta**: `/` significa la ruta raíz del servidor.
- El segundo parámetro es una **función de controlador** que se ejecuta cuando el servidor recibe una solicitud a esa ruta. Esta función toma dos parámetros:
  - **req:** El objeto de solicitud, que contiene información sobre la petición del cliente (por ejemplo, parámetros de la URL, encabezados, cuerpo de la solicitud, etc.).
  - **res:** El objeto de respuesta, que se usa para enviar una respuesta al cliente.
- **res.send('¡Hola Mundo desde Express!');** Aquí estamos enviando una respuesta de texto simple (¡Hola Mundo desde Express!) al cliente cuando visita la ruta `/`.

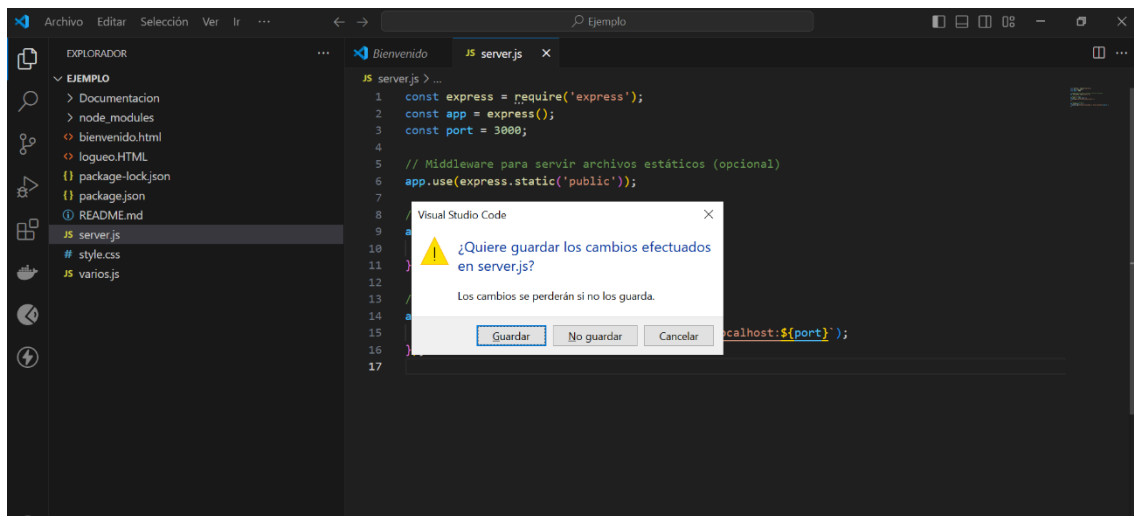
**app.listen():** Este método hace que la aplicación **empiece a escuchar** en el puerto que hemos especificado (en este caso, el puerto 3000).

- El primer argumento de `app.listen()` es el puerto en el que el servidor escuchará las solicitudes.

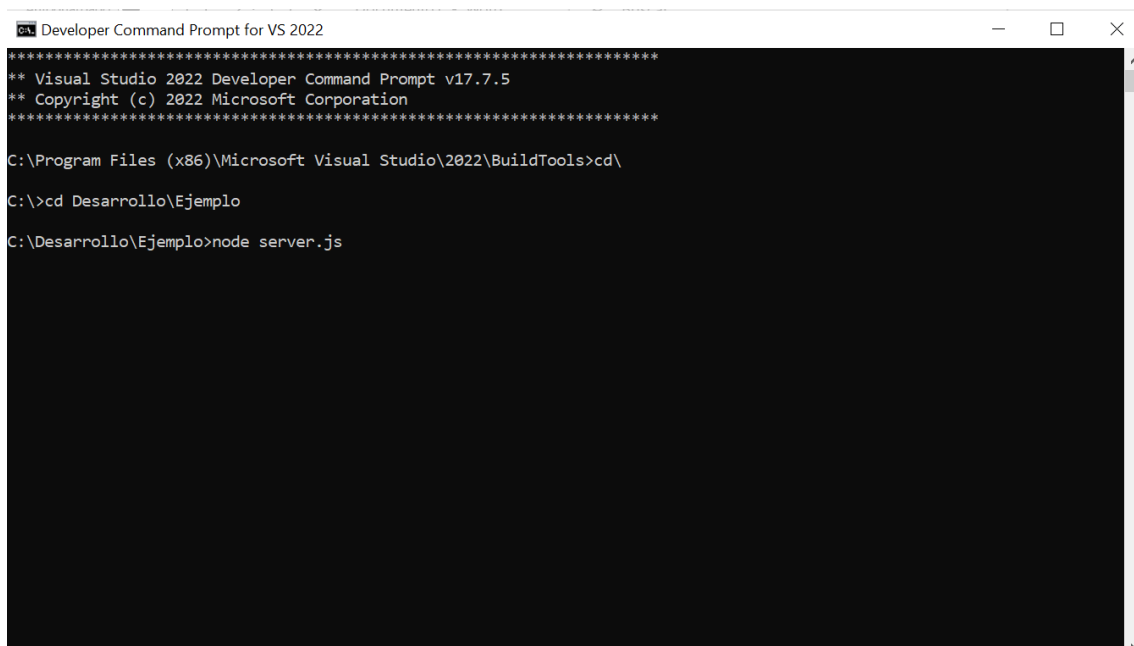
- El segundo argumento es una **función de callback** que se ejecuta cuando el servidor está listo para recibir solicitudes. Dentro de esta función, simplemente imprimimos un mensaje en la consola diciendo que el servidor está corriendo y en qué dirección se puede acceder a él.

Este ejemplo genérico solamente da una respuesta básica al puerto, es una breve porción de código para poder identificar que fue debidamente creado. Recuerden que siempre está en nuestra aplicación y sus usos, por lo que este archivo genérico vamos a modificarlo a nuestras necesidades. Se encuentra subido al repositorio de Ejemplo.

Una vez instanciado debemos guardar los cambios del server.js:

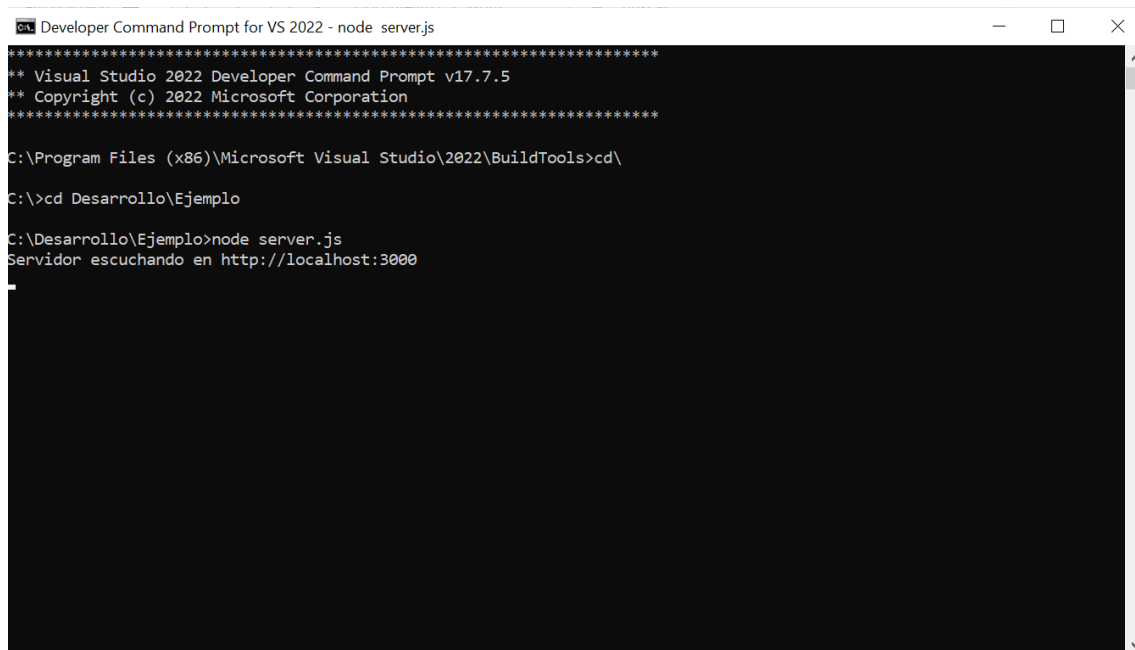


Por último, para inicializar el servidor. Debemos pararnos sobre el proyecto y ejecutar *node server.js*





Damos enter para ejecutar, esto hará que el puerto configurado (3000) esté escuchando la petición y atento a requerimientos. Debería aparecer la siguiente pantalla:



```
Developer Command Prompt for VS 2022 - node server.js
*****
** Visual Studio 2022 Developer Command Prompt v17.7.5
** Copyright (c) 2022 Microsoft Corporation
*****

C:\Program Files (x86)\Microsoft Visual Studio\2022\BuildTools>cd\

C:\>cd Desarrollo\Ejemplo

C:\Desarrollo\Ejemplo>node server.js
Servidor escuchando en http://localhost:3000
```

Para tener la última comprobación, podemos:

- abrir navegador web e ir a <http://localhost:3000>.
- ver el mensaje: "¡Hola Mundo desde Express!".