

Introducción a Node.js y Express.js

Ficha de trabajo – Reto 2 “El Puesto de Mando”

1. ¿Qué es Node.js?

Node.js es un entorno que permite ejecutar **JavaScript fuera del navegador**, es decir, en el **lado del servidor**. Con él podemos crear programas, servicios, y sobre todo **servidores web** que reciben peticiones y envían respuestas.

■ **En resumen:** Node.js convierte a JavaScript en un lenguaje capaz de manejar tanto la parte del cliente (**frontend**) como del servidor (**backend**).

¿Para qué se usa?

- Crear **servidores web** (como Apache o IIS, pero hechos en JS)
- Desarrollar **APIs REST** que envían y reciben datos
- Conectarnos a **bases de datos** (MongoDB, MySQL, PostgreSQL...)
- Ejecutar tareas automáticas o procesos en segundo plano

¿Qué es npm?

npm (Node Package Manager) es el sistema de gestión de paquetes de Node.js. Permite instalar librerías y herramientas listas para usar.

Ejemplo: `npm install express`

2. ¿Qué es Express.js?

Express.js (o simplemente **Express**) es una **librería de Node.js** que facilita la creación de servidores web y APIs REST. Es un **framework** que simplifica el uso de Node.js para manejar peticiones HTTP.

Ejemplo con Node.js puro:

```
import http from "http";

const server = http.createServer((req, res) => {
  res.writeHead(200, { "Content-Type": "text/plain" });
  res.end("¡Servidor funcionando!");
});

server.listen(3000);
```

Ejemplo con Express:

```
import express from "express";
const app = express();

app.get("/", (req, res) => {
  res.send("¡Servidor funcionando con Express!");
});
```

```
app.listen(3000);
```

➡■ **Conclusión:** Express es una forma más cómoda, limpia y moderna de crear servidores web con Node.js.

3. ¿Por qué vamos a usar Node.js y Express en el Reto 2?

En el **Reto 2 – “El Puesto de Mando”**, vamos a desarrollar una **aplicación de tres capas**:

1. Capa de presentación: App móvil (el usuario)
2. Capa intermedia: Servidor con **Node.js + Express** (lógica de negocio)
3. Capa de datos: Base de datos (MongoDB u Odoó)

Nuestro servidor Express será el **cerebro** que recibe las peticiones, las procesa y devuelve datos en formato JSON.

4. Conceptos clave que deben recordar

Concepto	Significado
Node.js	Entorno para ejecutar JavaScript en el servidor.
npm	Gestor de librerías y módulos para Node.js.
Express.js	Framework que simplifica la creación de servidores y APIs REST.
API REST	Interfaz que permite enviar y recibir datos usando HTTP (GET, POST, PUT, DELETE).
Puerto	‘Puerta’ de entrada/salida del servidor (por ejemplo, 3000).
localhost	Nombre que se usa para referirse al propio equipo.

5. Mini práctica introductoria

1. Instalar Node.js desde <https://nodejs.org>

2. Verificar la instalación:

node -v

npm -v

3. Crear un proyecto nuevo y ejecutar:

npm init -y

npm install express

4. Crear el archivo **index.js** con el siguiente código:

```
import express from 'express';
const app = express();

app.get('/', (req, res) => {
  res.send('¡Servidor funcionando!');
});

app.listen(3000, () => {
  console.log('Servidor en http://localhost:3000');
});
```

5. Ejecutar el servidor: `node index.js`
6. Abrir el navegador y visitar <http://localhost:3000>

■ Resultado esperado:

- En la terminal: Servidor en <http://localhost:3000>
- En el navegador: **¡Servidor funcionando!**

⇒ ■ Reflexiona y responde:

1. ¿Qué diferencia hay entre Node.js y Express?

.....

2. ¿Qué papel cumple el archivo package.json?

.....

3. ¿Por qué crees que Express se usa tanto en proyectos modernos?

.....

4. ¿Qué sucede si cambias el número de puerto 3000 por otro?

.....