

SESIÓN 10 – SERVIDORES DE APLICACIONES: TOMCAT, NODE.JS Y EXPRESS

1. ¿Qué es Node.js?

Node.js es un entorno de ejecución de JavaScript que permite ejecutar JavaScript fuera del navegador, es decir, directamente en tu ordenador o servidor.

- Antes: JavaScript solo funcionaba en el navegador (Chrome, Firefox, etc.).
- Ahora, con Node: puedes usar JavaScript para crear servidores, APIs, aplicaciones de escritorio, scripts, etc.

¿Por qué se usa Node.js?

- Para crear servidores web.
- Para hacer APIs REST (como la que tú acabas de hacer).
- Porque es rápido, gracias a su motor V8 (Google Chrome).
- Porque usa un modelo asíncrono y no bloqueante, ideal para apps con miles de conexiones.
- Porque permite usar JavaScript en backend (antes solo era lenguaje de frontend).

Node.js = JavaScript en el servidor.

2. ¿Qué es Express?

Express es un framework (librería) para Node.js que facilita crear servidores y APIs web.

Node por sí solo es muy básico, Express añade:

- Rutas (GET, POST, PUT, DELETE...)

- Manejo de JSON
- Middleware
- Facilidad para crear APIs REST

Sin Express, crear una API sería mucho más largo y complejo.

Con Express, solo escribes unas pocas líneas y ya tienes un servidor funcionando.

3. ¿Para qué se usan Node.js + Express?

Se usan para crear:

APIs REST: es una forma de crear un servicio web que permite que diferentes aplicaciones se comuniquen entre sí usando peticiones HTTP y datos normalmente en JSON.

GET /empleados

Devuelve JSON: ideal para aplicaciones web, móviles, etc.

Páginas web dinámicas

Con motores como EJS, Pug o Handlebars.

Backends de aplicaciones modernas

Ejemplos: React, Angular, Vue → suelen hablar con un backend hecho en Express.

Microservicios

Servicios pequeños, rápidos y escalables.

Aplicaciones en tiempo real

Chats, juegos, notificaciones... gracias a WebSockets.

4. ¿Cómo encaja todo esto?

Imagina:

Node = Motor

- JavaScript funcionando en el servidor.

Express = Coche

- El vehículo que te permite mover datos y crear rutas fácilmente.

API REST = Carretera

- Puntos de acceso donde los clientes piden información.

En el siguiente ejercicio:

- Node ejecuta tu programa (`node app.js`).
- Express crea una API rápida.
- `/empleados` es un endpoint (ruta).
- El navegador recibe JSON.

Node.js permite ejecutar JavaScript en el servidor y Express te permite crear APIs y servidores web de forma sencilla.

Enunciado del ejercicio

Crea una **API REST con Node.js y Express** que:

- Escuche en el **puerto 3000**.
- Tenga un endpoint GET /empleados.
- Devuelva un array de empleados en formato **JSON**:

[

```
{ "id": 1, "nombre": "Luis", "salario": 2000 },  
{ "id": 2, "nombre": "Marta", "salario": 2400 },  
{ "id": 3, "nombre": "Diego", "salario": 1800 }
```

]

Pasos a seguir en Windows

Instalar Node.js

Si ya tienes Node, puedes saltar este paso.

1. Entra en la web de **Node.js** (busca “Node.js download”).
2. Descarga la versión **LTS** para Windows.
3. Ejecuta el .msi → siguiente, siguiente... (deja todo por defecto).
4. Cierra y abre de nuevo **CMD** y comprueba:

```
node -v
```

```
npm -v
```

Si salen versiones, todo correcto.

Crear la carpeta del proyecto

Vamos a crear el proyecto en el Escritorio, por ejemplo.

En **CMD**:

```
cd C:\Users\Usuario\Desktop
```

```
mkdir api-empleados
```

```
cd api-empleados
```

Cambia Usuario por tu nombre de usuario de Windows si es distinto.

Inicializar el proyecto e instalar Express

Dentro de api-empleados:

```
npm init -y
```

```
npm install express
```

- npm init -y crea el package.json.
- npm install express descarga Express en node_modules.

Crear el archivo app.js

Ahora creamos el archivo principal de la API.

1. Abre el **Explorador de archivos** y ve a:
2. C:\Users\Usuario\Desktop\api-empleados
3. Clic derecho → **Nuevo** → **Documento de texto**.
4. Nómbralo **app.js** (si te pregunta por cambiar la extensión, di que **sí**).
5. Ábrelo con **Bloc de notas** (o, mejor, Visual Studio Code si lo usas).

PROMETO

Pega dentro este código:

```
// Importamos el módulo express
const express = require('express');

// Creamos la aplicación express
const app = express();

// Datos de ejemplo: listado de empleados
const empleados = [
    { id: 1, nombre: 'Luis', salario: 2000 },
    { id: 2, nombre: 'Marta', salario: 2400 },
    { id: 3, nombre: 'Diego', salario: 1800 }
];

// Endpoint GET /empleados que devuelve el array en formato JSON
app.get('/empleados', (req, res) => {
    res.json(empleados);
});

// Ponemos la API a escuchar en el puerto 3000
app.listen(3000, () => {
    console.log("API funcionando en puerto 3000");
});
```

Guarda el archivo.

Ejecutar la API

Vuelve a la consola (**CMD**) en la carpeta del proyecto:

```
cd C:\Users\Usuario\Desktop\api-empleados  
node app.js
```

Deberías ver:

API funcionando en puerto 3000

Déjala así, sin cerrar la ventana.

Probar la API en el navegador

Abre un navegador y escribe:

<http://localhost:3000/empleados>

Te debe salir algo así:

```
[  
  { "id": 1, "nombre": "Luis", "salario": 2000 },  
  { "id": 2, "nombre": "Marta", "salario": 2400 },  
  { "id": 3, "nombre": "Diego", "salario": 1800 }  
]
```

Pasos a seguir macOS

Instalar Node.js en Mac

Si ya tienes Node instalado, puedes saltar este paso.

Instalar desde la web oficial

1. Busca en Google: **Node.js download**
2. Entra en la página oficial.
3. Descarga la versión **LTS** para macOS.
4. Ejecuta el archivo .pkg (instalador).
5. Siguiente, siguiente... acepta todo por defecto.
6. Cuando termine → **cierra y vuelve a abrir Terminal**.

Comprobar instalación

En **Terminal**:

```
node -v
```

```
npm -v
```

Si ves versiones (por ejemplo v22.x.x), todo está correcto.

Crear la carpeta del proyecto

Vamos a crear el proyecto en el **Escritorio del Mac**.

En Terminal:

```
cd ~/Desktop
```

```
mkdir api-empleados
```

```
cd api-empleados
```

~ significa tu carpeta de usuario en macOS.

~/Desktop es el Escritorio.

Inicializar el proyecto e instalar Express

En la Terminal, dentro de api-empleados:

```
npm init -y
```

```
npm install express
```

- npm init -y crea el archivo **package.json**
- npm install express descarga Express en node_modules

Crear el archivo app.js

Por ejemplo, en VS Code.

1. Copia y pega este código:

```
// Importamos el módulo express
```

```
const express = require('express');
```

```
// Creamos la aplicación express
```

```
const app = express();
```

```
// Datos de ejemplo: listado de empleados
```

```
const empleados = [
```

PROMETO

```
{ id: 1, nombre: 'Luis', salario: 2000 },  
 { id: 2, nombre: 'Marta', salario: 2400 },  
 { id: 3, nombre: 'Diego', salario: 1800 }  
];  
  
// Endpoint GET /empleados que devuelve el array en formato JSON  
app.get('/empleados', (req, res) => {  
  res.json(empleados);  
});
```

```
// Ponemos la API a escuchar en el puerto 3000  
app.listen(3000, () => {  
  console.log("API funcionando en puerto 3000");  
});
```

4. Guarda el archivo:

- o Nombre: **app.js**
- o Cuidado con la extensión, debe ser si o si .js, si no, dará error.

Asegúrate de que NO lo guardas como app.js.txt.

Para comprobarlo, en Terminal ejecuta:

ls

Debería verse:

app.js

node_modules

package.json

Ejecutar la API

En Terminal:

```
cd ~/Desktop/api-empleados
```

```
node app.js
```

Si todo está correcto verás:

API funcionando en puerto 3000

Déjalo abierto; es tu servidor.

Probar la API en el navegador

Abre Safari, Chrome o Edge y escribe:

```
http://localhost:3000/empleados
```

Resultado esperado:

```
[  
  { "id": 1, "nombre": "Luis", "salario": 2000 },  
  { "id": 2, "nombre": "Marta", "salario": 2400 },  
  { "id": 3, "nombre": "Diego", "salario": 1800 }]
```