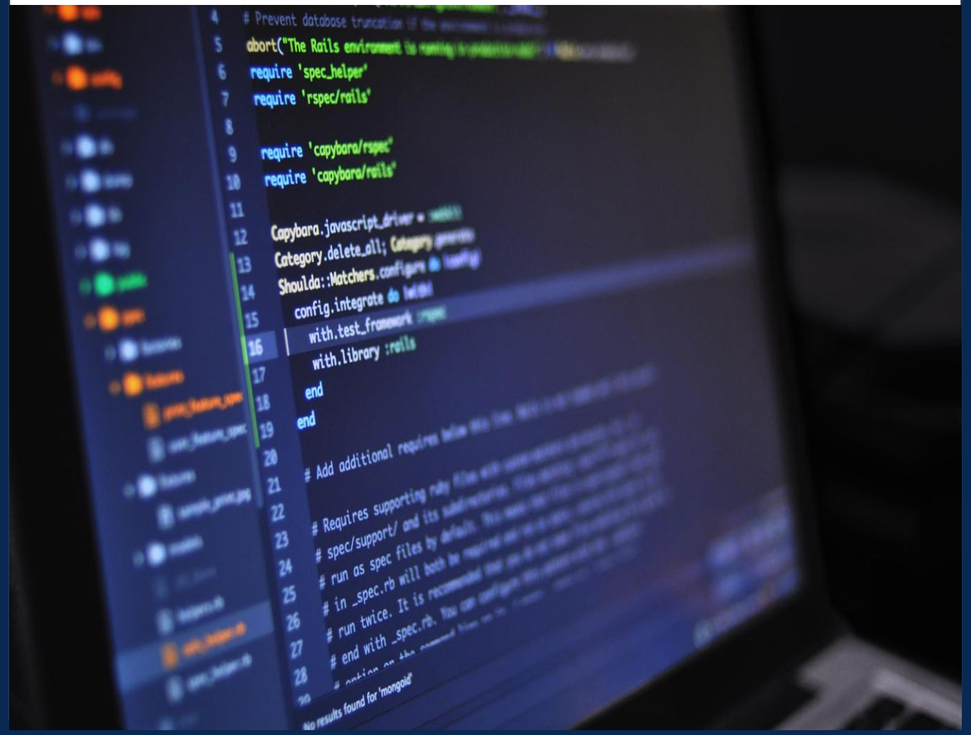
Bloque TypeScript



Tema 8: JSON SERVER

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

¿QUÉ ES JSON Server ?

JSON Server permite crear una API REST falsa (mock API) a partir de un archivo JSON.

Simula un backend real para desarrollo y pruebas, ideal para practicar operaciones CRUD y probar interfaces.

Las ventajas que tiene son:

a) Rapidez de configuración
b) Ideal para pruebas y aprendizaje
c) Desarrollo paralelo frontend–backend
d) Reinicio rápido de datos
e) Compatible con React, Vue, Angular, etc.

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

I N S T A L A C I Ó N

- Requisitos previos:

Para poder utilizar JSON Server, necesitas tener instalado Node.js y npm (Node Package Manager) en tu sistema.

- Node.js permite ejecutar código JavaScript fuera del navegador, lo que facilita crear y gestionar servidores.

- npm se instala automáticamente junto con Node.js y sirve para descargar, instalar y gestionar librerías de JavaScript, como JSON Server.

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

INSTALACIÓN

1.Instalación global

Permite ejecutar el comando json-server desde cualquier carpeta de tu sistema.

```
npm install -g json-server
```

Con esto, podrás iniciar un servidor en cualquier proyecto simplemente ejecutando:

```
json-server db.json
```

```
D:\DWEC\json>json-server db.json
JSON Server started on PORT :3000
Press CTRL-C to stop
Watching db.json...

(^_^)

Index:
http://localhost:3000/

Static files:
Serving ./public directory if it exists

Endpoints:
http://localhost:3000/usuarios
```

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

INSTALACIÓN

1.Instalación local:

Instala JSON Server solo dentro de una carpeta o proyecto, registrándolo como dependencia de desarrollo.

npm install json-server --save-dev

```
D:\DWEC\json2>dir
 El volumen de la unidad D es DATOS
 El número de serie del volumen es: 8062-1FF6

 Directorio de D:\DWEC\json2

08/10/2025  12:54    <DIR>          .
08/10/2025  12:54    <DIR>          ..
08/10/2025  12:54    <DIR>          node_modules
08/10/2025  12:54            20.243 package-lock.json
08/10/2025  12:54                66 package.json
               2 archivos         20.309 bytes
               3 dirs  255.306.465.280 bytes libres
```

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

INSTALACIÓN

Fichero package.json

Contiene toda la información necesaria para que Node y npm (el gestor de paquetes) sepan cómo ejecutar, gestionar y compartir tu aplicación.En otras palabras, es el “corazón” del proyecto, ya que define:

a) Qué dependencias (librerías) usa.
b) Qué scripts se pueden ejecutar.
c) Qué versión tiene el proyecto.
d) Y otros metadatos importantes.

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

I N S T A L A C I Ó N


```
{
  "name": "api-falsa-jsonserver",
  "version": "1.0.0",
  "description": "API simulada con JSON Server para pruebas y desarrollo",
  "main": "index.js",
  "scripts": {
                "start": "json-server db.json --port 3000"
              },
  "author": "Tu Nombre",
  "license": "MIT",
  "dependencies": {
    "json-server": "^0.17.0"
  }
}
```

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

Estructura db.json

base de datos simulada que está compuesta por **colecciones de datos**, representadas como **arrays (listas)** de objetos. Cada colección funciona como una “tabla” o “recurso” dentro de la API.

```
{
  "usuarios": [
    { "id": 1, "nombre": "Ana", "email": "ana@example.com" },
    { "id": 2, "nombre": "Carlos", "email": "carlos@example.com" }
  ],

  "tareas": [
    { "id": 1, "titulo": "Estudiar JSON Server", "completada": false }
  ]
}
```

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

Estructura db.json

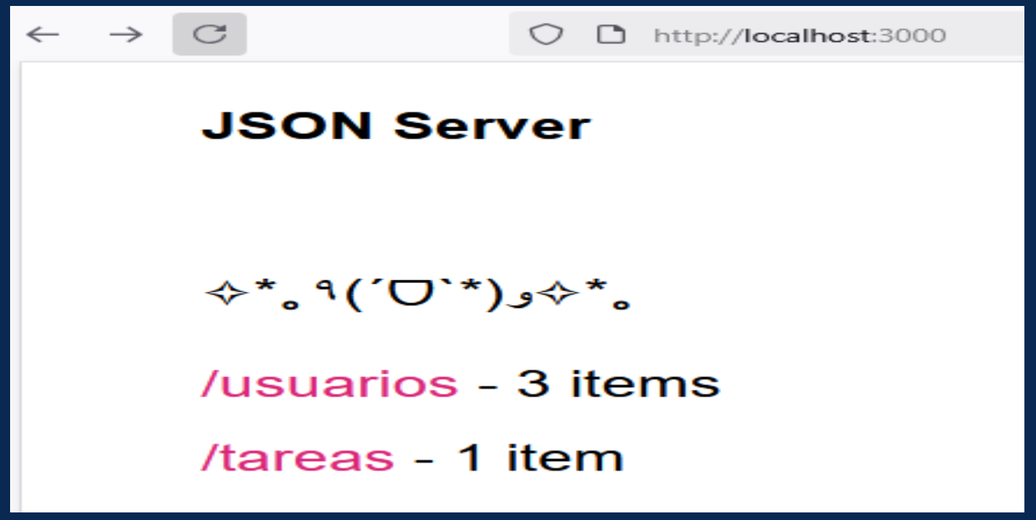
```
D:\DWEC\json>json-server db.json
JSON Server started on PORT :3000
Press CTRL-C to stop
Watching db.json...

ᕕ( ᐛ )

Index:
http://localhost:3000/

Static files:
Serving ./public directory if it exists

Endpoints:
http://localhost:3000/usuarios
http://localhost:3000/tareas
```



/usuarios será un endpoint que devuelve la lista de usuarios.

/tareas será otro endpoint que devuelve las tareas.

endpoint es la "*puerta de entrada*" a un conjunto de datos o funciones dentro de un servidor.

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

Operaciones CRUD

Create (crear) **R**ead (leer) **U**pdate (actualizar) **D**elete (eliminar)

En el caso de **JSON Server**, estas operaciones se implementan mediante los **métodos HTTP**: POST, GET, PUT, PATCH y DELETE.

MÉTODO	ACCIÓN	DESCRIPCIÓN	EJEMPLO
GET	Leer	Obtiene datos del servidor. Puede usarse para listar todos los registros o uno específico.	`/usuarios` `/usuarios/1`
POST	Crear	Envía datos al servidor para crear un nuevo registro.	`/usuarios`
PUT	Actualizar (completo)	Reemplaza totalmente un registro existente por uno nuevo.	`/usuarios/1`
PATCH	Actualizar (parcial)	Modifica solo algunos campos de un registro.	`/usuarios/1`
DELETE	Eliminar	Borra un registro existente.	`/usuarios/1`

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

Operaciones CRUD

Archivo base (db.json)

```
{
  "usuarios": [
    { "id": 1, "nombre": "Carlos López", "email": "carlos@example.com", "activo": true },
    { "id": 2, "nombre": "Marta Díaz", "email": "marta@example.com", "activo": false }
  ]
}
```

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

Operaciones CRUD

Create (crear) => POST

```
// Dar de alta un usuario
fetch('http://localhost:3000/usuarios', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    nombre: 'María López',
    email: 'maria@example.com'
  })
})
  .then(response => response.json())
  .then(data => console.log('Usuario creado:', data))
  .catch(error => console.error('Error al crear usuario:', error));
```

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

Operaciones CRUD

Read (leer) => GET.

```
// Obtener todos los usuarios
fetch('http://localhost:3000/usuarios')
  .then(response => response.json())
  .then(data => {
    console.log('Usuarios:', data);
  })
  .catch(error => console.error('Error al obtener usuarios:', error));

// Obtener un usuario específico por ID
fetch('http://localhost:3000/usuarios/1')
  .then(response => response.json())
  .then(usuario => console.log('Usuario con ID 1:', usuario))
  .catch(error => console.error('Error al obtener usuario:', error));
```

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

Operaciones CRUD

Update (actualizar completo) => PUT

```
// Actualizar un registro completo
fetch('http://localhost:3000/usuarios/2', {
  method: 'PUT',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    id: 2,
    nombre: 'Luis Pérez Actualizado',
    email: 'luis.actualizado@example.com'
  })
})
  .then(response => response.json())
  .then(data => console.log('Usuario actualizado completamente:', data))
  .catch(error => console.error('Error al actualizar usuario:', error));
```

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

Operaciones CRUD

Update (actualizar parcial) => PACTH

```
// Actualizar un registro parcialmente
fetch('http://localhost:3000/usuarios/1', {
  method: 'PATCH',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    email: 'ana.actualizado@example.com'
  })
})
  .then(response => response.json())
  .then(data => console.log('Usuario modificado parcialmente:', data))
  .catch(error => console.error('Error al modificar usuario:', error));
```

un evento es lo que se produce cuando el navegador detecta determinados eventos; por ejemplo, cuando se pulsa en un enlace, se activa un botón o se carga una página.

Operaciones CRUD

Delete (eliminar)

```
// Borrar un registro
fetch('http://localhost:3000/usuarios/1', {
  method: 'DELETE'
})
  .then(() => console.log('Usuario eliminado correctamente'))
  .catch(error => console.error('Error al eliminar usuario:', error));
```