Laboratorio 3

API REST usando el framework Express
Framework Express

Objetivos

- Introducir conceptos de desarrollo back-end
- Construir un API REST usando los métodos POST, GET, PUT y DELETE

Importante: Se requiere tener instalado

- NodeJS
- Insomnia
- Typescript: npm install -g typescript

1- Crear un directorio con el nombre de laboratorio3
2- Abrir el directorio “laboratorio3” desde Visual Studio Code
3- Abrir el terminal VSC y ejecutar el siguiente comando:
>> npm init

Dar el nombre del paquete “laboratorio3”, y las demás opciones damos la tecla “enter”. Finalmente damos la opción “yes” y se genera el archivo package.json

Responda las siguientes preguntas
¿Para qué sirve el archivo package.json?
¿Qué significa la abreviación NPM?
¿ Qué significa la palabra global dentro de la instalación typescript?

4- Crear un directorio llamado “dist”
5- Inicializar TypeScript para eso ejecutaremos el comando en la terminal “ tsc --init”. Este debe permitir generar el archivo tsconfig.json.

Responda las siguientes preguntas
¿ Qué significa las siglas TSC?
¿ Qué es EMACScript?
¿ Para qué sirve el archivo tsconfig.json?
¿ Qué significa la abreviación dist?
¿Qué es un API REST, RESTFUL y Webservice? ¿Son lo mismo?, Sí no son lo mismo justique.

6- Editar el archivo tsconfig.json para cambiar la ruta de compilación del JS.

Figura 1. Editar el archivo tsconfig.json

7- Crear un archivo llamado app.ts
8- Instalar el framework express JS. Abrir el terminal y ejecutamos el comando **npm install express**
9- Instalamos las siguientes dependencias:

npm i --save-dev @types/node
npm i --save-dev @types/express
npm i body-parser
npm i cors

Responda las siguientes preguntas

¿Cuál es la funcionalidad del CORS?
¿Cuál es la funcionalidad del body-parser?
¿Cuál es la funcionalidad del @types/express?
¿Qué es express?

10- Editar el archivo app.ts e importar los módulos express.

```
const express=require("express");
const bodyParser = require('body-parser')
const cors=require('cors')
```

Adicionar la siguiente instrucción al código const app=express();

11- Creamos el servidor usando hostname y un número de puerto.

```
//creamos servidor
const configuracion={
    hostname: "127.0.0.1",
    port: 3000,
}

app.listen(configuracion, () => {
    console.log(`Conectando al servidor http://localhost:${configuracion.port}`)
})
```

12- Creamos un archivo JSON llamado data.json con la siguiente estructura:

```
[{
    "rut": "122121-1",
    "nombre": "Juan",
    "edad": 30,
    "esEstudiante": false,
    "dirección": {
      "calle": "Calle Falsa 123",
      "ciudad": "Madrid",
      "códigoPostal": "28080"
    },
    "hobbies": ["fútbol", "pintura", "lectura"]
  }]
```

Adicione más datos al JSON.

13- Adicionamos la dependencia fs para leer un archivo JSON. Por lo tanto, debemos adicionar la siguiente línea en el código:

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

14- Implementar el método get:

```
app.get('/',(req:any, res:any)=>{
    fs.readFile('./data.json', 'utf8', (err:any, data:any) => {
    if (err) {
        res.status(500).send('Error reading file');
        return;
    }
    try {
        const jsonData = JSON.parse(data);
        res.send((JSON.stringify(jsonData)));
    } catch (error) {
        res.status(500).send('Error de JSON');
    }
  });
});
```

15- Vamos a probar que el método está funcionando correctamente. Primero originamos el JS, por lo que debemos ejecutar el comando tsc –watch. Una vez originado el js en el directorio dist, podemos ejecutar el archivo app.js con el siguiente comando: **node dis/app.js**

16- Abrir Insomnia y probar el método get.

17- Ahora se debe crear un método POST para consultar un dato, por lo que, requiere pasar parámetros tipo JSON. En este caso se requiere el POST para consultar el rut.

Este módulo se usa para procesar datos de solicitudes HTTP con JSON o datos de formularios. Por lo que, se debe adicionar la siguiente instrucción:

```
var jsonParser = bodyParser.json()
```

El método POST quedaría de la siguiente manera:

```
app.post('/',json,(req:any, res:any)=>{
    let rut=req.body.rut;
    fs.readFile('./data.json', 'utf8', (err:any, data:any) => {
        if (err) {
            res.status(500).send('Error reading file');
            return;
        }
        try {
            const jsonData = JSON.parse(data);
            const newElemento=jsonData.find((x:any)=> x.rut===rut);
            console.log(newElemento);
            if(newElemento)
                res.status(200).send("dato encontrado")
            else
                res.status(200).send("dato no encontrado")
        } catch (err) {
            res.status(500).send(err);
        }
    });

});
```

18- Para probar en Insomnia se debe crear datos tipo input en formato JSON:

19- Implemente los métodos PUT y DELETE.

Considere que para eliminar un dato en JSON debe considerar el siguiente código:

```
const jsonData = JSON.parse(data);
const index=jsonData.findIndex((x:any)=> x.rut===rut);

if (index != -1) {
    console.log(index);
    jsonData.splice(index, 1);
    fs.writeFile('./data.json',JSON.stringify(jsonData),function(err:any){
        if(err) throw err;
    })

    res.status(201).send("dato eliminado");

}
```

Entregar:

- El código fuente del API REST
- Un archivo en formato pdf con las respuestas.