Técnicas de Programación

Carrera Programador full-stack

Persistencia

Agenda

- Concepto de persistencia.
- Librerías *Node* para persistir datos en archivos de texto.
- Lectura y escritura de archivos de texto
- Ejercicios

¿Qué es persistencia?

En programación, la **persistencia** es la acción de preservar la información de forma permanente (guardado), pero a su vez también se refiere a poder recuperar la información del mismo (leerlo) para que pueda ser nuevamente utilizado.

Persistencia

En una App funcionando, utilizariamos una base de datos.

En esta instancia, al no tener ese conocimiento, utilizamos persistencia para “simular” una base de datos y poder trabajar con ellos sin tener que cargar datos cada vez que corremos nuestros proyectos o que los datos que ingresemos sean solo momentaneos.

Para esto, vamos a utilizar un modulo de Node js que nos va a permitir guardar la informacion en un archivo .txt y luego poder leer esa informacion.

File System

Node.js cuenta con un módulo centralizado, llamado File System (fs), diseñado específicamente para la manipulación de archivos. Este módulo brinda una interfaz que asegura la compatibilidad independientemente del sistema operativo (SO) en el que se ejecute Node.js, proporcionando un enfoque estándar para operaciones relacionadas con archivos.

writeFileSync()

Este modulo (fs), tiene métodos para escribir, leer o borrar un archivo.

En este caso, nos vamos a centrar en el metodo para escribir un archivo y asi lograr que la información persista en nuestro programa.

Para acceder a él vamos a importar fs a nuestro archivo:

```
import fs from 'node:fs';
```

Sintaxis

```
fs.writeFileSync('./test.txt', texto);
```

Módulo File System importado desde Node JS

Método para guardar información en archivo .txt

URL del archivo .txt. (Si no existe se creará automáticamente)

Texto a guardar (Puede ser un string o una variable que lo contenga)

readFileSync()

Este método también pertenece al modulo “fs” y lo utilizamos para leer archivos .txt.

De la misma manera que el método anterior le pasamos como parámetro la URL del archivo a leer y como segundo parámetro el formato que queremos, en nuestro caso “utf8”:

```
const datos = fs.readFileSync('./test.txt', "utf8");
```

LO GUARDAMOS EN UNA VARIABLE!!

readFileSync()

Luego mostramos esos datos por consola, y vamos a ver que los recibimos como un string.

¿Como hacemos para que vuelva a ser un array de datos?

Vamos a utilizar el metodo trim() para que nos saque los espacios por detras y delante del texto y “split” para separar strings y crear un nuevo array:

Ejemplo

```
  const sinEspacio = datos.trim()
  const dataRecuperada= sinEspacio.split(" ");
```

Salida:

```
Karen@DESKTOP-VODAUQN MINGW64 ~/Desktop/CEP
$ ts-node persistencia.ts
Finalizado con exito
Karen Juan Pedro
[ 'Karen', 'Juan', 'Pedro' ]
```

Técnicas de Programación

Carrera Programador full-stack

Persistencia

Ejercicio obligatorio entrega 14/06

Guarda la información de los siguientes arrays:

```
const precios: number[] = [525, 3500, 400, 1999];
const productos: string[] = ["Leche", "Galletitas", "Harina",
"Queso"];
```

en un archivo “precios.txt” y “productos.txt” respectivamente.
Luego recupera la información en forma de array nuevamente y mostrala por consola.

Técnicas de Programación

Carrera Programador full-stack

Persistencia