## **Trabajo Práctico 4:**

### **Objetivo**

Aprender a trabajar con **Nodemon**, **dotenv**, la transición de **require** a **import/export** en Node.js, y cómo interactuar con el usuario solicitando datos por consola usando el módulo **readline**.

#### **1. ¿Qué es Nodemon?**

Nodemon es una herramienta que reinicia automáticamente el servidor de Node.js cuando se detectan cambios en los archivos del proyecto, lo que mejora la productividad durante el desarrollo.

#### **2. ¿Qué es dotenv?**

dotenv es un módulo que carga las variables de entorno desde un archivo .env y las hace accesibles en process.env. Se utiliza para gestionar información sensible, como credenciales o claves de API.

#### **3. Import/Export vs. Require/Module.exports**

* **require/module.exports** es el sistema de módulos en CommonJS.
* **import/export** es el sistema de módulos estándar de ECMAScript (ES6).
* Para usar import/export, debes configurar "type": "module" en el package.json.

#### **4. Módulo readline**

El módulo **readline** permite interactuar con el usuario en la consola, permitiendo pedir datos como nombre, edad o cualquier otra entrada.

### **Ejercicios**

#### **Ejercicio 1: Configuración de Nodemon y dotenv**

1. **Configuración de un proyecto básico:**
   * Crea un proyecto de Node.js con npm init -y.
   * Instala las dependencias de **dotenv** y **nodemon**.
   * Configura **nodemon** para que el servidor se reinicie automáticamente al realizar cambios en el archivo index.js.
2. **Cargar variables de entorno:**
   * Crea un archivo .env en la raíz del proyecto. Agrega las siguientes variables de entorno:  
     DB\_HOST=localhost

DB\_USER=admin

DB\_PASS=secret

* + En el archivo index.js, usa dotenv para cargar las variables de entorno y muestra un mensaje en consola con las variables DB\_HOST, DB\_USER y DB\_PASS.

#### **Ejercicio 2: Migración de require a import/export**

1. **Uso de require:**
   * Crea un archivo math.js con una función llamada sumar que reciba dos números y los sume.
   * En el archivo index.js, importa la función usando require y muestra el resultado de sumar(5, 3).
2. **Migración a import/export:**
   * Cambia tu proyecto para usar módulos ES6.
   * Reemplaza require por import y module.exports por export en el archivo math.js.
   * En index.js, usa import para traer la función sumar y muestra el resultado de la suma.

#### **Ejercicio 3: Pedir Datos por Consola con readline**

1. **Solicitar datos al usuario:**
   * En el archivo index.js, utiliza el módulo readline para pedir al usuario su nombre.
   * Muestra un mensaje en consola con el nombre que el usuario introdujo.

import readline from 'readline';

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout

});

rl.question('¿Cuál es tu nombre? ', (nombre) => {

console.log(`Hola, ${nombre}!`);

rl.close();

});

1. **Usar los datos solicitados:**
   * Pide al usuario su edad y muestra un mensaje indicando cuántos años tiene. Usa la entrada para hacer una operación, por ejemplo, calcular el año de nacimiento.

const añoNacimiento = new Date().getFullYear() - parseInt(edad);

### 

### **Ejercicio 4: Pedir Datos por Consola, Guardarlos en un Archivo y Leerlos**

1. **Pedir datos al usuario**:  
   * Usa el módulo **readline** para solicitar al usuario su nombre, edad y correo electrónico.
2. **Guardar los datos en un archivo .txt**:  
   * Utiliza el módulo **fs (File System)** para guardar los datos del usuario en un archivo **datos\_usuario.txt**. Los datos deben ser guardados en formato texto, cada uno en una nueva línea.
3. **Leer el archivo y mostrar los datos**:  
   * Después de guardar los datos, usa **fs.readFile** para leer el archivo **datos\_usuario.txt** y mostrar en consola el contenido del archivo.

¿Qué ventajas tiene usar **dotenv** para manejar configuraciones sensibles en vez de tenerlas directamente en el código?

El uso de **dotenv** ayuda a mantener las configuraciones sensibles seguras, facilita la gestión de entornos, mejora la portabilidad del proyecto, y asegura que las configuraciones sean modificables sin comprometer la integridad del código.

Además, la separación de las configuraciones facilita el mantenimiento y la colaboración en equipos.

¿Por qué es importante utilizar **Nodemon** durante el desarrollo en proyectos de Node.js?

**Nodemon** mejora el flujo de trabajo en el desarrollo de aplicaciones Node.js al automatizar el reinicio del servidor cada vez que se realizan cambios en el código.

Esto mejora la productividad y eficiencia y además reduce los errores de codigo, haciendo que el proceso de desarrollo sea más eficiente y sencillo.

¿Qué diferencias encuentras entre el uso de **require** y **import/export**? ¿Por qué es preferible utilizar la sintaxis **ES6** en proyectos modernos?

La diferencia entre **require** y **import/export** es que **require** se usa en programas de Node.js y carga los datos de forma más lenta.

**Import/export** es más moderno, permite que el código sea más claro y rápido, además de que se puede usar tanto en navegadores como en Node.js.