NODE LISTA DE TAREAS

Iniciamos npm init para configurar todo el archivo config.json..

Instalamos yargs y color npm install –save yargs colors

Save por que quieremos que sean dependencias obligatorias

Creamos el archivo app.js e importamos raygs

Const argv = require(‘yargs’).argv;

Console.log(argv);

Tenemos que crear funciones de tareas usando comandos, ejemplo;

Node app crear -d “Pasear al perro”

Creamos el archivo de configuración e los comandos

let comando = argv.\_[0];

switch (comando) {

    case 'crear':

        console.log('Crear lista por hacer')

        break;

    case 'listar':

        console.log(' lista las tareas')

        break;

    case 'actualizar':

        console.log('Actialuza lista por hacer')

        break;

    default:

        console.log('Comando erroneo')

}

Lo probamos con node app listar

**Validaciones automáticas con Yargs**

Hacemos la configuración de yargs, creamos carpeta nueva config, con el archivo yargs.js

Y lo importamos en el archivo principal.

//configuracion

const argv = require('yargs')

    .command('crear', 'Crea un elemento por hacer', {

        descripcion: {

            demand: true,

            alias: 'd',

            desc: 'Descripcion de la tarea por hacer'

        }

    })

    .command('actualizar', 'Actualiza el estado completado de una tarea', {

        descripcion: {

            demand: true,

            alias: 'd',

            desc: 'Descripcion de la tarea por hacer'

        },

        completado: {

            alias: 'c',

            default: true,

            desc: 'Marca como completado una tarea'

        }

    })

    .help()

    .argv; //retorna el objeto global

module.exports = {

    argv //exporta toda la función que hicimos

}

**CREA UNA TAREA PERSISTENTE**

Ahora vamos a hacer que al ejecutar ese comando se guarde en un archivo.

Creo una carpeta por- hacer y el archivo js con su mismo nombre, una función que recibe como parámetro la descripción que le pasamos en la consola que regrtese un objeto y lo guarde en un array.

const fs = require('fs');

//todo se guarda en un arreglo

let listadoPorHacer = [];

const crear = (descripcion) => {

    let porHacer = {

        descripcion,

        completado: false

    };

    listadoPorHacer.push(porHacer);

    return porHacer;

}

module.exports = {

    crear

}

Vamos a guardar esa información en la BD.

Creamos una nueva carpeta DB y el archivo en formato json (data.json) Aquí vamos a guardar la info.

Creamos una nueva función en por-hacer.js para que guarde el array como un archivo JSON

//guarda en la BD

const guardarDB = () => {

    let data = JSON.stringify(listadoPorHacer);

    //lo guarda en la capeta DB

    fs.writeFile('db/data.json', data, (err) => {

        if (err) throw new Error('Error inesperado', err);

    })}

Y ejecuto la función después de agregarlo al array dentro del método crear

   listadoPorHacer.push(porHacer);

    guardarDB()

Ya se guardo en el archivo data.json, el problema es que siempre sobreescribe la info de data.json

**LEER INFORMACION DE UN ARCHIVO JSON**

Tenemos que hacer el proceso inverso de guardar la data en el array para cargarlo nuevamente

Creamos una nueva función en por hacer que retorne la base de datos

//Gaurda el archivo en el array

const cargarDB = () => {

    listadoPorHacer = require('../db/data.json');

}

Y la pongo al principio de la función de crear para que carge la base

const crear = (descripcion) => {

    cargarDB();

**SOLUCIONANDO PROBLEMAS**

Si el archivo db.json esta vacio arroja un error. Que la función de cargar DB este en un try-catch si falla que ponga un arreglo vacio por que eso es un JSON Valido

//Gaurda el archivo en el array

const cargarDB = () => {

    try {

        listadoPorHacer = require('../db/data.json');

    } catch (error) {

        listadoPorHacer = []; }}

**COMANDO PARA LISTAR TODAS LAS TAREAS POR HACER**

Hacemos la función

const getListado = () => {

    cargarDB();

    return listadoPorHacer;

}

Exportamos

module.exports = {

    crear,

    getListado

}

Y en el case la trabajamos, y no olvidar exportar el paquete de colors

const colors = require('colors');

 case 'listar':

        let listado = porHcaer.getListado();

        for (let tarea of listado) {

            console.log('\*\*\*\*\*\*\*\*\*\*\*\*'.green);

            console.log(`${tarea.descripcion}`);

            console.log('Estado :', `${tarea.completado}`);

            console.log('\*\*\*\*\*\*\*\*\*\*\*'.green);

        }

        break;

**ACTUALIZAR TAREA POR HACER**

Crear el comando que diga que ya terminamos la tarea como terminada.

Hacemos una nueva función

const actualizar = (descripcion, completado = true) => {

    cargarDB();

    let index = listadoPorHacer.findIndex(tarea => {

        return tarea.descripcion === descripcion;

    })

    //devielve -1 si no encontro

    if (index >= 0) {

        listadoPorHacer[index].completado = completado;

        guardarDB();

        return true;

    } else {

        return false;

    }

}

Nota no olvidar exportarlo

 case 'actualizar':

        let actualizado = porHcaer.actualizar(argv.descripcion,argv.completado);

        console.log(actualizado)

        break;

*BORRAR TAREAS POR HACER.*

*Usamos filter para encontrar un elemento diferente en el arreglo, osea si este elemento es igual al que le metamos en al descripción, excluye los demás del array y lo guarda en la lista nueva, y si en extensión los dos array son diferentes que se guarde la lista nueva en el anterior,Y que guarde los cambios en la BD.*

const borrar = (descripcion) => {

    cargarDB();

    //tarera el elemento que es diferente en el array

    const listado = listadoPorHacer.filter(

        listadoPorHacer => listadoPorHacer.descripcion !== descripcion)

    if (listadoPorHacer.length === listado.length) {

        return false;

    } else {

        listadoPorHacer = listado;

        guardarDB();

        return true;

    }

}

En el archivo app

 case 'borrar':

        let borrado = porHcaer.borrar(argv.descripcion);

        console.log(borrado);

        break;

comando

.command('borrar', 'Borra una tarea de la base', {

        descripcion: {

            alias: 'd',

            desc: 'Borra una tarea de la BD'

        }

    })

**APLICANDO OPTIMIZACIONES**

**Guardamos como objeto los comandos**

const descripcion = {

    demand: true,

    alias: 'd',

    desc: 'Descripcion de la tarea por hacer'

};

const completado = {

    alias: 'c',

    default: true,

    desc: 'Marca como completado una tarea'

};

//configuracion

const argv = require('yargs')

    .command('crear', 'Crea un elemento por hacer', {

        descripcion

    })

    .command('actualizar', 'Actualiza el estado completado de una tarea', {

        descripcion,completado})