Se incorporo la clase Products que contiene dos métodos (read & write).
El método de read se encarga de leer un archivo (especificado como parámetro) y devolver su estructura JSON.
El método write se encarga de leer un archivo (especificado como parámetro) obtener su estructura en formato JSON, realizar un push a dicha estructura de un objeto que también se pasa como parámetro, escribir un archivo y finalmente volviendo a leerlo para obtener el resultado y retornar su estructura JSON.

```
import fs from 'fs'
export default function Products(fileName){
    this.fileName=fileName
    this.read=function(){
        try{
            return (JSON.parse(fs.readFileSync(this.fileName,'utf-8')))
        }catch (error){
            console.log(error)
        }
    }
    this.write=function(objeto){
        try{
            const productos=JSON.parse(fs.readFileSync(this.fileName,'utf-8'))
            objeto.id=productos.length
            productos.push(objeto)
            try{
                fs.writeFileSync(this.fileName,JSON.stringify(productos))
                try{
                    return (JSON.parse(fs.readFileSync(this.fileName,'utf-8')))
                }catch (error){
                    console.log(error)
                }
            }catch (error){
                console.log(error)
            }
        }catch (error){
            console.log(error)
        }
    }
}
```

```
import express from 'express';
import fs from 'fs';
import bodyParser from "body-parser";
import Archivo from './modules/Archivo.js';
const app = express();
const archivo = new Archivo('productos.txt');
app.use(bodyParser.json());
const puerto = 8080;
```

Se importa Express, fs, bodyParse y la clase mencionada anteriormente, se relaciona el nombre del archivo de donde se van a leer y a escribir los productos.

Esta ruta se encarga de invocar el método read de la clase Products que se encarga de leer los productos almacenados y responder su resultado en el json del response

```
app.get('/api/productos/listar',(request,response)=>{
    try{
        const productos=archivo.read()
        response.json({'items':productos})
    }catch (error){
        response.json({"error": "Intente de nuevo más tarde"})
    }
})
```

```
app.get('/api/productos/listar/:id',(request,response)=>{
    try{
        const productos=archivo.read()
        const filtro=productos.filter(item => item.id==request.params.id)
        response.json((filtro.length>0)?filtro: {"error": "Producto no encontrado"})
    }catch (error){
        response.json({"error": "Intente de nuevo más tarde"})
    }
})
```

Esta ruta se encarga de invocar el método read de la clase Products y con la información retornada por este método, realizar un filter utilizando el id que llega en los request.params y retornar dicho arreglo.

```
app.post('/api/productos/guardar',(request,response)=>{
    try{
    let producto=request.body
    return response.json({"inserción":{"producto": producto}, "resultado": {"productos": archivo.write(producto)}})
}catch (error){
    response.json({"error": "Intente de nuevo más tarde"})
}

})
```

Esta ruta se encarga de obtener el objeto presente en el request.body, y pasarlo como parámetro al método write de la clase Products. Finalmente se retorna el objeto inicial y el arreglo de productos resultantes.