# Desarrollo de Software: Backend

Objetivo: Crear una aplicación backend con una interface WebApiRest, programada en NodeJs/Javascript, usando el framework express y con acceso a una base de datos (sqlite) mediante un ORM (Sequelize)

* Version final del proyecto: <https://labsys.frc.utn.edu.ar/dds-backend-2024/>
* Version final del proyecto con frontend funcionando: <https://labsys.frc.utn.edu.ar/dds-frontend-2024/>
* Requisitos tener instalado:
  + Visual studio Code
  + Node.js
  + Todos los comandos se ejecutan desde una consola de git bash

## 

## 

## Etapa 1

## Crear proyecto básico

* creamos la carpeta del proyecto: dds-backend
* Ubicado en la carpeta, inicializamos el proyecto node, con el comando:  
  npm init

| **parámetro** | **valor** |
| --- | --- |
| name | dds-backend |
| description | backend con express |
| entry point | index.js |
| etc | etc |

* Nota: Podríamos usar la sintaxis de import en lugar de require, agregando en el package.json: "type":"module".
* instalamos la librería express, con el comando:  
  npm i express
* creamos el archivo inicial de la aplicación: index.js
  + codificamos la aplicación web básica:

const express = require("express");

*// crear servidor*

const app = express();

*// controlar ruta*

app.get("/", (req, res) => {

res.send("Backend inicial dds-backend!");

});

*// levantar servidor*

const port = 3000;

app.listen(port, () => {

console.log(`sitio escuchando en el puerto ${port}`);

});

* Ejecutamos el proyecto:  
   node index.js
* Testeamos la aplicación desde el explorador, url: localhost:3000
* Inicializamos repositorio y hacemos el primer commit.  
  git init
* agregamos el archivo .gitignore, con el siguiente contenido:

node\_modules/

* ejecutamos:

git add --all

* y luego hacemos commit

git commit -m "etapa 1 completa"

Para mejorar la experiencia de desarrollo, haremos uso de nodemon, instalamos nodemon, mediante el comando:  
 npm i nodemon -D

Agregamos a package.json en "scripts" el siguiente script:  
 "dev": "nodemon index.js"

deberia quedarnos algo similar a:

| {  "name": "dds-backend",  "version": "1.0.0",  "description": "",  "main": "index.js",  "scripts": {  "test": "echo \"Error: no test specified\" && exit 1",  "dev": "nodemon index.js"  },  "keywords": [],  "author": "",  "license": "ISC",  "dependencies": {  "[express](https://npmjs.com/package/express)": "[^4.18.2](https://npmjs.com/package/express)"  },  "devDependencies": {  "[nodemon](https://npmjs.com/package/nodemon)": "^3.1.0"  }  } |
| --- |

Finalmente para ejecutar el proyecto de aquí en adelante usaremos:  
 npm run dev

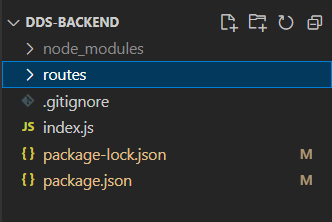
¿que diferencia hay en ejecutar el proyecto con?

* node index.js
* npm run dev

## Etapa 2

## weapi articulosfamiliasmock

A continuación construiremos una api que no accede a base de datos sino que simulando dicho acceso trabaja con un array de datos hardcodeados.

* Agregamos al proyecto la carpeta "routes" en donde pondremos los controladores de las diferentes rutas de los recursos de la webapi  
  
* dentro de la carpeta routes creamos el archivo "articulosfamiliasmock.js” que gestionará el recurso articulosfamiliasmock, con el siguiente código:

const express = require('express');

const router = express.Router();

let arr\_ArticulosFamiliasMock = [

{

"IdArticuloFamilia": 1,

"Nombre": "Accesorios"

},

{

"IdArticuloFamilia": 2,

"Nombre": "Audio"

},

{

"IdArticuloFamilia": 3,

"Nombre": "Celulares"

},

{

"IdArticuloFamilia": 4,

"Nombre": "Cuidado Personal"

},

{

"IdArticuloFamilia": 5,

"Nombre": "Dvd"

},

{

"IdArticuloFamilia": 6,

"Nombre": "Fotografia"

},

{

"IdArticuloFamilia": 7,

"Nombre": "Frio-Calor"

},

{

"IdArticuloFamilia": 8,

"Nombre": "Gps"

},

{

"IdArticuloFamilia": 9,

"Nombre": "Informatica"

},

{

"IdArticuloFamilia": 10,

"Nombre": "Led - Lcd"

}

];

router.get('/api/articulosfamiliasmock', async function (req, res) {

res.json(arr\_ArticulosFamiliasMock);

});

module.exports = router;

**Observe:**

* la clase express.Router para crear controladores de rutas montables y modulares.
* la definición moqueada del array de datos de articulosfamilias.
* el controlador GET de la ruta "/api/articulosfamilasmock" que devolverá serializado como json el array de datos.
* La función se define como asíncrona "async", que aunque no tenga sentido actualmente, la usamos previendo cuando obtengamos datos desde la base de datos donde será necesaria.

Una vez definido el controlador de nuestro recurso debemos vincularlo a nuestra aplicación express, cargando el módulo de ruta en el archivo index.js antes de levantar el servidor (app.listen…)

const articulosfamiliasmockRouter = require("./routes/articulosfamiliasmock");

app.use(articulosfamiliasmockRouter);

Para testear nuestro recurso, iniciemos nuestra aplicación y consultemos desde el explorador la siguiente url:

<http://localhost:3000/api/articulosfamiliasmock>

Agregaremos ahora el método GET que permite obtener un recurso según su id, al archivo articulosfamiliasmock.js le agregamos este código, antes de la instrucción module-exports …

router.get('/api/articulosfamiliasmock/:id', async function (req, res) {

let articuloFamilia = arr\_ArticulosFamiliasMock.find(

(x) => x.IdArticuloFamilia == req.params.id

);

if (articuloFamilia) res.json(articuloFamilia);

else res.status(404).json({ message: 'articulofamilia no encontrado' });

});

**Observe:**

* cómo se recupera el id del segmento de la url, mediante la colección params
* como se busca en el array el dato solicitado
  + si se encuentra se devuelve el mismo en formato de json
  + si no se encuentra se devuelve un error 404 com un mensaje adecuado.

Para testearlo, iniciemos nuestra aplicación y consultemos desde el explorador la siguiente url:

<http://localhost:3000/api/articulosfamiliasmock/1>

\*\* Testemos cambiando el número final de la url que indica el id del recurso a buscar.

Agregamos ahora el método post, que permite a agregar un recurso, usaremos el siguiente código (siempre antes del module.exports):

router.post('/api/articulosfamiliasmock/', (req, res) => {

const { Nombre } = req.body;

let articuloFamilia = {

Nombre,

IdArticuloFamilia: Math.floor(Math.random()\*100000),

};

*// aqui agregar a la coleccion*

arr\_ArticulosFamiliasMock.push(articuloFamilia);

res.status(201).json(articuloFamilia);

});

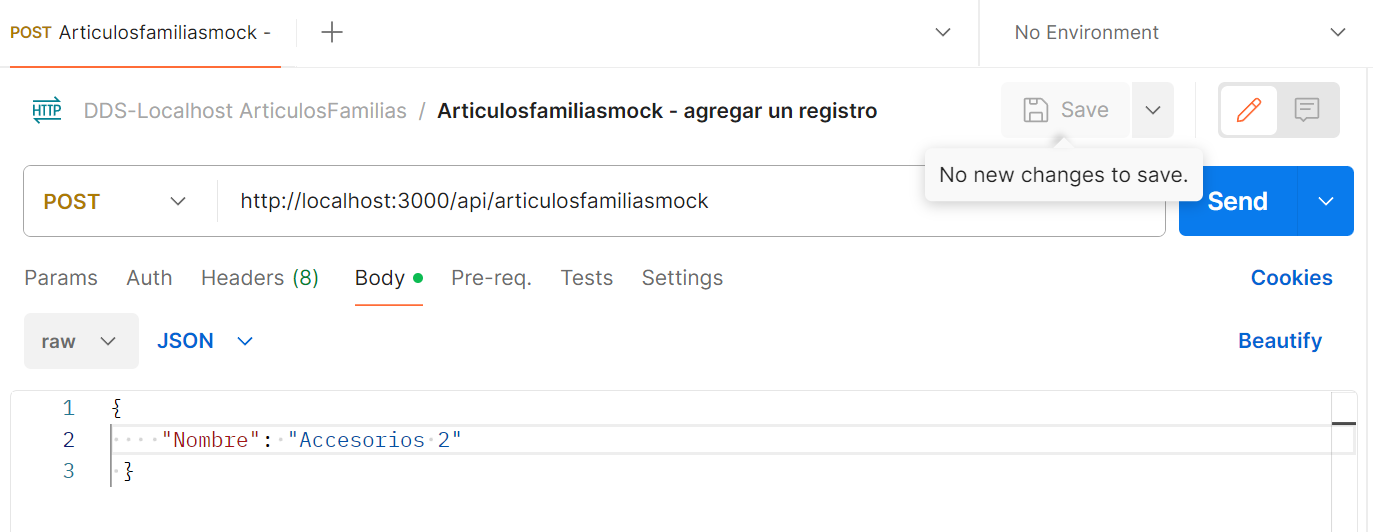
**Observe:**

* cómo se recupera el dato del Nombre desde el objeto "body" del request
* el campo IdArticuloFamilia, en base de datos sería un autonumérico, aquí usamos un solución poco fiable pero sencilla, solo válida para una demostración rápida.
* devolvemos el código de status 201 y el objeto recién creado; tal vez quien consuma esta api buscara allí, entre otros valores, el IdArticuloFamilia recién generado.

Para que este método funcione, express necesita un middleware que le permita interpretar el json que recibe en el body, para lo cual agregamos en el index.js, luego de crear la constante app, el código siguiente:

app.use(express.json()); *// para poder leer json en el body*

Testeamos este método, con la ayuda de la aplicación Postman que nos facilitara invocar la url con el verbo Post y los parámetros necesarios.



Agregamos ahora el método PUT, que permite a modificar un recurso, usaremos el siguiente código:

router.put('/api/articulosfamiliasmock/:id', (req, res) => {

let articuloFamilia = arr\_ArticulosFamiliasMock.find(

(x) => x.IdArticuloFamilia == req.params.id

);

if (articuloFamilia) {

const { Nombre } = req.body;

articuloFamilia.Nombre = Nombre;

res.json({ message: 'articulofamilia actualizado' });

} else {

res.status(404).json({ message: 'articulofamilia no encontrado' })

}

});

**Observe:**

* el uso del método find para buscar el recurso a modificar
* la modificación del objeto encontrado y la devolución de un mensaje de éxito.
* la devolución del código de status 404 si no se encuentra el recurso a modificar

Testeamos este modelo, con la ayuda de la aplicación Postman que nos facilitará invocar la url con el verbo PUT y los parámetros necesarios.



Finalmente agregamos el método DELETE, que permite a eliminar un recurso, usaremos el siguiente código:

router.delete('/api/articulosfamiliasmock/:id', (req, res) => {

let articuloFamilia = arr\_ArticulosFamiliasMock.find(

(x) => x.IdArticuloFamilia == req.params.id

);

if (articuloFamilia) {

arr\_ArticulosFamiliasMock = arr\_ArticulosFamiliasMock.filter(

(x) => x.IdArticuloFamilia != req.params.id

);

res.json({ message: 'articulofamilia eliminado' });

} else {

res.status(404).json({ message: 'articulofamilia no encontrado' })

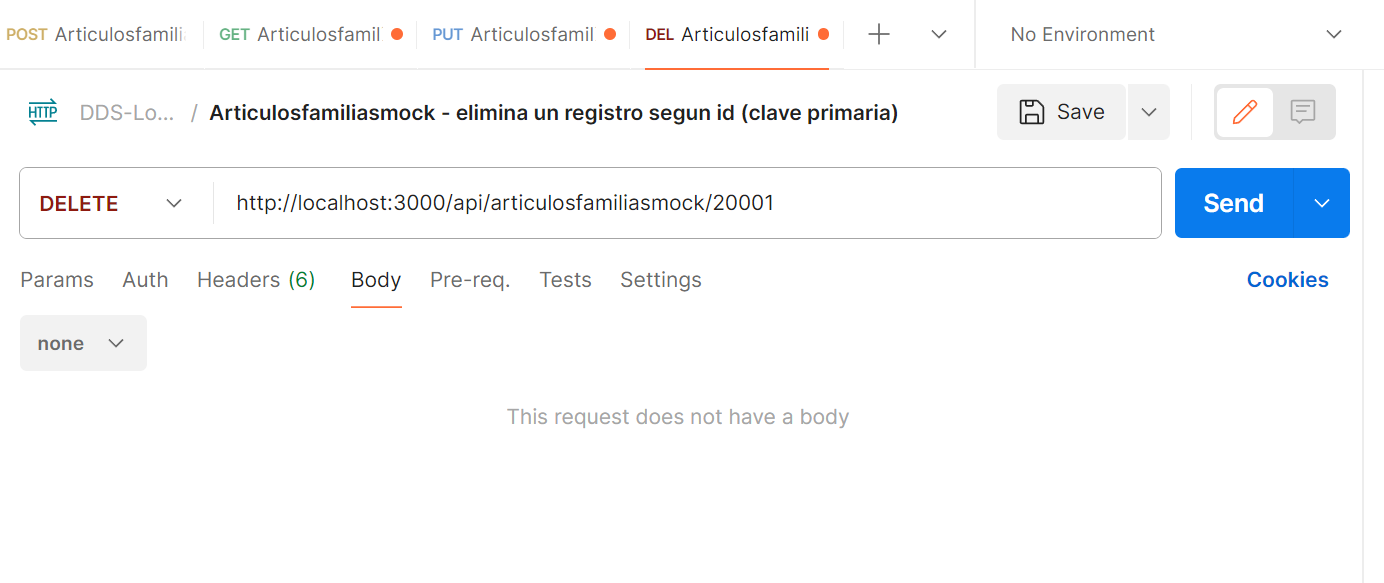
}

});

**Observe:**

* el uso del método filter, como estrategia para eliminar el elemento del array
* la devolución de un mensaje de éxito.
* la devolución del código de status 404 si no se encuentra el recurso a eliminar

Testeamos este método, con la ayuda de la aplicación Postman que nos facilitara invocar la url con el verbo DELETE y los parámetros necesarios.



* Ejercicio: implementar una mejora al método GET que devuelva todos los articulosfamilias. Deberá retornar solo aquellos que coincidan con un parámetro opcional: "Nombre", si no se recibiese dicho parámetro, seguirá funcionando como antes devolviendo todos los registros.

Tips: para leer el parámetro usaremos el objeto "query" del request

## 

## Etapa 3

## webapi ArticulosFamilias

Esta api accedera a la base de datos sqlite: Pymes.db, mediante el orm Sequelize.

* Agregamos al proyecto la carpeta ".data" en donde se alojará el archivo de base de datos de sqlite: "pymes.db", el mismo será creado mediante código. Usamos la nomenclatura “.data” por compatibilidad a la plataforma de desarrollo stackblitz, esto puede traer algunos inconvenientes en la visibilidad de la carpeta en windows.
* Agregamos al proyecto la carpeta "base-orm" en donde pondremos el código relacionado a la base de datos

Inicialmente instalaremos los paquetes necesarios para acceder a la base de datos "sqlite3", una librería para simplificar el acceso asíncrono "aa-sqlite" y el ORM elegido "sequelize". Todo mediante el siguiente comando:

npm i sqlite3 aa-sqlite sequelize

* Agregamos en la carpeta base-orm, el archivo "sqlite-init.js" que contiene el código con la ejecución del script para crear la base de datos:

*// acceder a la base usando aa-sqlite*

const db = require("aa-sqlite");

async function CrearBaseSiNoExiste() {

*// abrir base, si no existe el archivo/base lo crea*

await db.open("./.data/pymes.db");

*//await db.open(process.env.base);*

let existe = false;

let res = null;

res = await db.get(

"SELECT count(\*) as contar FROM sqlite\_schema WHERE type = 'table' and name= 'usuarios'",

[]

);

if (res.contar > 0) existe = true;

if (!existe) {

await db.run(

"CREATE table usuarios( IdUsuario INTEGER PRIMARY KEY AUTOINCREMENT, Nombre text NOT NULL UNIQUE, Clave text NOT NULL, Rol text NOT NULL);"

);

console.log("tabla usuarios creada!");

await db.run(

"insert into usuarios values (1,'admin','123','admin'),(2,'juan','123','member');"

);

}

existe = false;

res = await db.get(

"SELECT count(\*) as contar FROM sqlite\_schema WHERE type = 'table' and name= 'articulosfamilias'",

[]

);

if (res.contar > 0) existe = true;

if (!existe) {

await db.run(

"CREATE table articulosfamilias( IdArticuloFamilia INTEGER PRIMARY KEY AUTOINCREMENT, Nombre text NOT NULL UNIQUE);"

);

console.log("tabla articulosfamilias creada!");

await db.run(

"insert into articulosfamilias values (1,'ACCESORIOS'),(2,'AUDIO'),(3,'CELULARES'),(4,'CUIDADO PERSONAL'),(5,'DVD'),(6,'FOTOGRAFIA'),(7,'FRIO-CALOR'),(8,'GPS'),(9,'INFORMATICA'),(10,'LED - LCD');"

);

}

existe = false;

sql =

"SELECT count(\*) as contar FROM sqlite\_schema WHERE type = 'table' and name= 'articulos'";

res = await db.get(sql, []);

if (res.contar > 0) existe = true;

if (!existe) {

await db.run(

`CREATE table articulos(

IdArticulo INTEGER PRIMARY KEY AUTOINCREMENT

, Nombre text NOT NULL UNIQUE

, Precio real

, CodigoDeBarra

, IdArticuloFamilia integer

, Stock integer

, FechaAlta text

, Activo boolean,

FOREIGN KEY (IdArticuloFamilia) REFERENCES articulosfamilias(IdArticuloFamilia)

);`

);

console.log("tabla articulos creada!");

await db.run(

`insert into articulos values

(1,'KIT DIRECT TV PREPA 0.60MT',299.00, '0779815559001', 10, 329,'2017-01-19', 1 ),

(2,'KIT DIRECT TV PREPA 0.90MT',349.00, '0779815559002', 10, 468,'2017-01-31', 1 ),

(3,'LED 22" LG FHD 22MN42APM',2669.00, '0779808338808', 10, 536,'2017-01-12', 1 ),

(4,'LED 24" ILO HD DIGITAL MOD LDH24ILO02',2999.00, '0779696260024', 10, 169,'2017-01-30', 1 ),

(5,'LED 24" LG HD 24MN42A-PM',3129.00, '0779808338809', 10, 296,'2016-12-28', 1 ),

(7,'LED 32" BGH HD BLE3214D',4830.00, '0779688540133', 10, 998,'2017-01-01', 1 ),

(8,'LED 32" BGH SMART TV BLE3213RT',5405.00, '0779688540117', 10, 650,'2017-01-18', 1 ),

(9,'LED 32" HISENSE IPTV HLE3213RT',5290.00, '0779688540119', 10, 51,'2017-02-03', 1 ),

(10,'LED 32" HITACHI HD CDHLE32FD10',4837.00, '0779694109973', 10, 838,'2016-12-25', 1 ),

(11,'LED 32" ILO HD DIGITAL LDH32ILO02',4199.00, '0779696260132', 10, 501,'2017-01-25', 1 ),

(12,'LED 32" JVC HD IPTV LT32DR930',6699.00, '0779818058057', 10, 906,'2017-01-25', 1 ),

(13,'LED 32" JVC HD LT32DA330',4499.00, '0779696266323', 10, 435,'2017-02-07', 1 ),

(14,'LED 32" LG 3D 32LA613B',6299.00, '0779808338816', 10, 329,'2017-02-06', 1 ),

(15,'LED 32" PHILIPS FHD 32PFL3018D/77',6799.00, '0871258168715', 10, 971,'2016-12-25', 1 ),

(16,'LED 32" PHILIPS FHD IPTV 32PFL4508G/77',7699.00, '0871258167198', 10, 636,'2017-02-07', 1 ),

(17,'LED 32" PHILIPS HD 32PFL3008D/77',5799.00, '0871258167218', 10, 67,'2016-12-27', 1 ),

(18,'LED 32" PHILIPS SMART TV 32PFL3518G/77',7399.00, '0871258167225', 10, 250,'2017-01-08', 1 ),

(19,'LED 32" RCA HD L32S80DIGI',4499.00, '0779694101214', 10, 857,'2017-01-23', 1 ),

(20,'LED 32" SAMSUNG FHD UN32F5000',6094.00, '0880608543154', 10, 636,'2016-12-30', 1 ),

(21,'LED 32" SAMSUNG HD UN32F4000',5519.00, '0880608543153', 10, 37,'2017-01-23', 1 ),

(22,'LED 32" SAMSUNG SMART UN32F5500',6899.00, '0880608548607', 10, 214,'2017-01-24', 1 ),

(23,'LED 32" SONY HD KDL32R425',6199.00, '0490552491740', 10, 642,'2017-01-17', 1 ),

(24,'LED 32" SONY SMART TV KDL32W655',6999.00, '0490552491687', 10, 50,'2017-02-04', 1 ),

(25,'LED 39" ILO DIG FHD LDF39ILO2',5699.00, '0779696260394', 10, 951,'2017-01-19', 1 ),

(26,'LED 39" PHILIPS FHD IPTV 39PFL3508G/77',8799.00, '0871258168717', 10, 889,'2017-02-03', 1 ),

(27,'LED 39" RCA FHD L39S85DIGIFHD',6499.00, '0779694101215', 10, 487,'2016-12-25', 1 ),

(28,'LED 40" BGH FHD BLE4014D',7245.00, '0779688540132', 10, 480,'2016-12-27', 1 ),

(29,'LED 40" SAMSUNG 3D SMART UN40F6800',13224.00, '0880608565606', 10, 734,'2017-01-26', 1 ),

(30,'LED 40" SAMSUNG 3D UN40F6100',9999.00, '0880608544958', 10, 835,'2017-01-19', 1 ),

(31,'LED 40" SAMSUNG FHD UN40F5000',8164.00, '0880608543156', 10, 436,'2017-02-01', 1 ),

(32,'LED 40" SAMSUNG SMART UN40F5500',9774.00, '0880608565438', 10, 639,'2017-01-20', 1 ),

(33,'LED 40" SONY FHD KDL40R485',7499.00, '0490552493532', 10, 862,'2017-01-07', 1 ),

(34,'LED 42" LG 3D 42LA6130',9199.00, '0779808338817', 10, 560,'2017-01-05', 1 ),

(35,'LED 42" LG FHD 42LN5400',8099.00, '0779808338818', 10, 48,'2017-01-28', 1 ),

(36,'LED 42" LG SMART TV 42LN5700',9799.00, '0779808338823', 10, 967,'2017-01-27', 1 ),

(37,'LED 42" PANASONIC 3D SMART TV TCL42ET60',11249.00, '0779805518074', 10, 570,'2017-01-19', 1 ),

(38,'LED 42" PHILIPS 3D SMART TV 42PFL5008G/7',11599.00, '0871258167039', 10, 802,'2017-02-04', 1 ),

(39,'LED 42" PHILIPS FHD 42PFL3008D/77',8499.00, '0871258167221', 10, 193,'2017-02-04', 1 ),

(40,'LED 42" PHILIPS SMART TV 42PFL3508G/77',9499.00, '0871258167227', 10, 693,'2016-12-30', 1 ),

(41,'LED 42" PIONEER 3D SMART PLE42FZP2',12299.00, '0498802821943', 10, 907,'2017-02-01', 1 ),

(42,'LED 42" SONY FHD KDL42R475',7999.00, '0490552491728', 10, 140,'2017-01-13', 1 ),

(43,'LED 46" PHILIPS SMART TV 46PFL4508G/7',13999.00, '0871258168718', 10, 236,'2017-01-31', 1 ),

(44,'LED 46" SAMSUNG 3D SMART TV UN46F7500',23574.00, '0880608565943', 10, 143,'2016-12-25', 1 ),

(45,'LED 46" SAMSUNG SMART UN46F5500',13224.00, '0880608548610', 10, 345,'2017-01-07', 1 ),

(46,'LED 46" SANYO SMART TV LCE46IF12',10599.00, '0779696260612', 10, 557,'2017-02-03', 1 ),

(47,'LED 47" LG SMART TV 47LN5700',13199.00, '0779808338824', 10, 599,'2017-01-20', 1 ),

(48,'LED 47" PIONEER 3D SMART PLE47FZP1',15999.00, '0498802821947', 10, 310,'2017-02-07', 1 ),

(49,'LED 47" SONY 3D SMART TV KDL47W805',17199.00, '0490552494098', 10, 526,'2017-01-31', 1 ),

(50,'LED 55" NOBLEX 3D IPTV 55LD856DI',20799.00, '0779696260000', 10, 362,'2017-01-26', 1 ),

(51,'LED 55" PHILIPS 3D SMART TV 55PFL8008G/77',29999.00, '0871258166949', 10, 841,'2017-01-06', 1 ),

(52,'SOPORTE LCD / LED DE 14" A 42" TANGWOOD',599.00, '0779814176493', 10, 527,'2017-02-07', 1 ),

(53,'SOPORTE LCD / LED DE 17 '' A 40 ''',499.00, '0779814176654', 10, 588,'2016-12-23', 1 ),

(54,'SOPORTE LCD / LED DE 17" A 37" TANGWOOD',225.00, '0779814176489', 10, 687,'2017-01-29', 1 ),

(55,'SOPORTE LCD / LED DE 23 '' A 50 ''',350.00, '0779814176652', 10, 519,'2016-12-25', 1 ),

(56,'SOPORTE LCD / LED DE 26" A 47" TANGWOOD',350.00, '0779814176442', 10, 81,'2017-01-28', 1 ),

(57,'SOPORTE LCD / LED TGW DE 17 '' A 37 ''',199.00, '0779814176648', 10, 164,'2017-01-17', 1 ),

(58,'SOPORTE LCD 10" TAGWOOD',375.00, '0779814176490', 10, 217,'2017-01-31', 1 ),

(59,'SOPORTE LCD 32" NAKAN',199.00, '0779803504550', 10, 873,'2017-01-01', 1 ),

(60,'SOPORTE LCD 32" ONE FOR ALL',259.00, '0871618404213', 10, 585,'2017-01-30', 1 ),

(61,'SOPORTE LCD 40" ONE FOR ALL',519.00, '0871618404215', 10, 809,'2017-01-22', 1 ),

(62,'SOPORTE LCD/LED 23 A 46"',399.00, '0779814176617', 10, 470,'2017-01-21', 1 ),

(68,'SOPORTE GPS',119.00, '0779814176084', 8, 524,'2017-01-14', 1 ),

(69,'SOPORTE GPS NEGRO MOTO 3,5" - 5,5"',259.00, '0779808004535', 8, 800,'2017-02-05', 1 ),

(70,'GPS GARMIN NUVI 2595',2899.00, '0075375999226', 8, 745,'2017-02-07', 1 ),

(71,'GPS GARMIN NUVI 52',2149.00, '0075375999808', 8, 274,'2016-12-22', 1 ),

(72,'GPS X VIEW VENTURA TV 7"',1849.00, '0779804220262', 8, 150,'2016-12-30', 1 ),

(73,'GPS XVIEW VENTURA TV',1509.00, '0779804220220', 8, 183,'2017-01-05', 1 ),

(74,'MOUSE HP 2.4G SILVER WIRELESS OPT CAN/EN',199.00, '0088496276058', 9, 40,'2017-02-03', 1 ),

(75,'PENDRIVE KINGSTONE DT101G2 8GB',129.00, '0074061716983', 9, 537,'2016-12-21', 1 ),

(76,'PENDRIVE SANDISK BLADE 4GB',129.00, '0061965900041', 9, 340,'2017-02-02', 1 ),

(77,'PENDRIVE SANDISK CRUZAR ORBIT 8GB',159.00, '0061965909040', 9, 696,'2017-02-07', 1 ),

(78,'PENDRIVE SANDISK POP BLACK 8GB',159.00, '0061965908448', 9, 431,'2017-01-08', 1 ),

(79,'PENDRIVE SANDISK POP PAIN 8GB',159.00, '0061965908156', 9, 521,'2017-02-01', 1 ),

(80,'CARTUCHO EPSON 732 CYAN',10290.00, '0001034385887', 9, 234,'2017-01-26', 1 ),

(81,'CARTUCHO EPSON T133120-AL MAGENTA',9690.00, '0001034387695', 9, 374,'2016-12-26', 1 ),

(82,'CARTUCHO EPSON T133120-AL NEGRO',8479.00, '0001034387692', 9, 836,'2017-01-25', 1 ),

(83,'CARTUCHO EPSON T133420-AL AMARILLO',9690.00, '0001034387696', 9, 796,'2016-12-28', 1 ),

(84,'CARTUCHO HP 122 NEGRO',149.00, '0088496298354', 9, 373,'2017-02-05', 1 ),

(85,'CARTUCHO HP 22 COLOR',299.00, '0082916090222', 9, 199,'2017-01-01', 1 ),

(86,'CARTUCHO HP 60 COLOR',289.00, '0088358598319', 9, 801,'2017-01-31', 1 ),

(87,'CARTUCHO HP 60 NEGRO',199.00, '0088358598317', 9, 655,'2017-01-08', 1 ),

(88,'PC ALL IN ONE 120-1156LA + TECLADO INAL + MOUSE',5499.00, '0088611278012', 9, 331,'2017-01-19', 1 ),

(90,'IMPRESORA MULTIFUNCION EPSON L355',3999.00, '0001034390469', 9, 293,'2017-01-01', 1 ),

(91,'MULTIFUNCION EPSON L210 + SISTEMA CONTINUO',3399.00, '0001034390433', 9, 689,'2017-01-09', 1 ),

(92,'MULTIFUNCION EPSON XP211',1199.00, '0001034390754', 9, 693,'2017-01-08', 1 ),

(93,'MULTIFUNCION EPSON XP401',1799.00, '0001034390348', 9, 363,'2017-01-17', 1 ),

(94,'NOTEBOOK BGH C-530 3D',4999.00, '0779816664067', 9, 401,'2017-01-30', 1 ),

(95,'NOTEBOOK BGH C-550',5799.00, '0779816664065', 9, 230,'2017-01-04', 1 ),

(96,'NOTEBOOK BGH C-565',6299.00, '0779816664069', 9, 876,'2017-02-06', 1 ),

(97,'NOTEBOOK BGH C-570',7299.00, '0779816664070', 9, 929,'2017-01-17', 1 ),

(98,'NOTEBOOK BGH QL 300 MINI',3699.00, '0779816664101', 9, 176,'2017-01-28', 1 ),

(99,'NOTEBOOK DELL INSPIRON 14 3421 I14I32\_45',6599.00, '0789948950198', 9, 758,'2016-12-31', 1 ),

(100,'NOTEBOOK DELL INSPIRON 14 3421 I14V997\_4',5999.00, '0779801657005', 9, 666,'2016-12-20', 1 ),

(101,'NOTEBOOK LENOVO G485 C-70',4399.00, '0088761972842', 9, 115,'2017-01-21', 1 ),

(102,'NOTEBOOK NOBLEX CEVEN GFAST',4499.00, '0779808041201', 9, 853,'2017-02-07', 1 ),

(103,'NOTEBOOK POSITIVO BGH F-810N NEGRA',4999.00, '0779816664059', 9, 48,'2017-01-21', 1 ),

(104,'NOTEBOOK SAMSUNG NP300E4C',6999.00, '0880608528173', 9, 272,'2017-01-08', 1 ),

(105,'NOTEBOOK SAMSUNG NP300E5A AD4AR',4799.00, '0880608500428', 9, 194,'2017-01-18', 1 ),

(106,'ULTRABOOK ACER S3-391-6867',9793.00, '0471219655495', 9, 974,'2017-01-23', 1 ),

(107,'ADAPTADOR PCI WIFI TL-WN751ND',259.00, '0693536405056', 9, 171,'2017-01-15', 0 ),

(110,'ANTENA TP-LINK TL-ANT2408C',249.00, '0693536405216', 9, 689,'2016-12-26', 1 ),

(111,'MINI ADAPATADOR USB TP LINK WN723N',185.00, '0693536405055', 9, 382,'2016-12-31', 1 ),

(112,'ROUTER MR3420 3G TP-LINK',649.00, '0693536405149', 9, 143,'2016-12-21', 1 ),

(113,'ROUTER PORTATIL TP LINK TL-MR3020',499.00, '0693536405170', 9, 594,'2017-02-01', 1 ),

(114,'ROUTER TL-WR941ND TP LINK',759.00, '0693536405127', 9, 646,'2017-02-06', 1 ),

(115,'ROUTER TP-LINK TL-WR720N',309.00, '0693536405198', 9, 867,'2017-01-01', 1 ),

(116,'ROUTER WR740 TP-LINK',389.00, '0693536405133', 9, 925,'2017-01-28', 1 ),

(117,'ROUTER WR841 TP-LINK',469.00, '0693536405124', 9, 624,'2017-01-29', 1 ),

(118,'TABLET MAGNUM TECH 7"',2599.00, '0779813546539', 9, 344,'2016-12-26', 1 ),

(119,'TABLET 10" MAGNUM TECH 8GB 1GBM',3799.00, '0779813546540', 9, 751,'2017-01-24', 1 ),

(120,'TABLET 10" NOBLEX NB1012',3549.00, '0779696292015', 9, 319,'2017-01-13', 1 ),

(121,'TABLET ALCATEL AB10',1799.00, '0695508989953', 9, 939,'2017-02-01', 1 ),

(122,'TABLET EUROCASE ARS 708',1099.00, '0779813546928', 9, 534,'2017-01-26', 1 ),

(123,'TABLET FUNTAB PRO',1699.00, '0081770701101', 9, 869,'2017-01-23', 1 ),

(124,'TABLET IDEAPAD LENOVO A1000L',2799.00, '0088794260611', 9, 597,'2017-01-05', 1 ),

(125,'TABLET LENOVO IDEAPAD A1000 7"',2299.00, '0088777046041', 9, 510,'2017-02-04', 1 ),

(126,'TABLET MAGNUM MG-701',1499.00, '0779813546946', 9, 645,'2017-02-05', 1 ),

(127,'TABLET NOBLEX-8013 8''',2149.00, '0779696291801', 9, 850,'2017-01-17', 1 ),

(130,'TABLET OLIPAD SMART 7" 3G',1499.00, '0802033432056', 9, 489,'2017-02-07', 1 ),

(131,'TABLET PC 7001 TITAN',999.00, '0076113310158', 9, 850,'2016-12-24', 1 ),

(132,'TABLET PC BOX T700U 7" DUAL CORE',1999.00, '0779815876409', 9, 769,'2017-02-06', 1 ),

(133,'TABLET PC FIRSTAR MID070A 8650',799.00, '0779815467080', 9, 9,'2017-01-23', 1 ),

(134,'TABLET PCBOX MOD T900',2799.00, '0779815876410', 9, 501,'2017-01-25', 1 ),

(135,'TABLET POLAROID MID1000 10',4299.00, '0358417655560', 9, 151,'2016-12-23', 1 ),

(136,'TABLET SYNKOM 7"',2499.00, '0779816920041', 9, 695,'2016-12-23', 1 ),

(137,'TABLET XVIEW ALPHA2 8GB',1899.00, '0779804220264', 9, 565,'2017-02-05', 1 ),

(138,'TABLET XVIEW PROTON',1699.00, '0779804220247', 9, 3,'2016-12-28', 1 ),

(139,'AIRE ACONDICIONADO DAEWOO 3200FC DWT23200FC',5898.00, '0779816944014', 7, 668,'2018-01-04', 1 ),

(140,'AIRE ACONDICIONADO DURABRAND 3500FC DUS35WCL4',5499.00, '0779688543933', 7, 945,'2017-01-20', 1 ),

(141,'AIRE ACONDICIONADO DURABRAND 4500FC DUS53WCL4',7499.00, '0779688543937', 7, 962,'2016-12-29', 1 ),

(142,'AIRE ACONDICIONADO KELVINATOR 2500WFC COD1056',4499.00, '0779694101056', 7, 670,'2017-01-03', 1 ),

(143,'AIRE ACONDICIONADO LG 3000 FC H126TNW0',7499.00, '0779808338858', 7, 441,'2017-01-09', 1 ),

(144,'AIRE ACONDICIONADO LG 4500 FC H1865NW0',10399.00, '0779808338859', 7, 971,'2016-12-23', 1 ),

(145,'AIRE ACONDICIONADO LG 5500 FC H2465NW0',12699.00, '0779808338860', 7, 648,'2017-01-15', 1 ),

(146,'AIRE ACONDICIONADO LG ARTCOOL 2300FC H096EFT0',7999.00, '0779808338853', 7, 659,'2017-01-01', 1 ),

(147,'AIRE ACONDICIONADO LG ARTCOOL 4500FC H1868FT0',12899.00, '0779808338855', 7, 712,'2016-12-25', 1 ),

(148,'AIRE ACONDICIONADO PHILCO 3200W FC PHS32H13X',6199.00, '0779696244974', 7, 588,'2017-01-09', 1 ),

(149,'AIRE ACONDICIONADO PHILCO 5000W FC PHS50H13X',9099.00, '0779696242975', 7, 275,'2016-12-22', 1 ),

(150,'AIRE ACONDICIONADO PORTATIL DURABRAND 2500FS LGACD01',4999.00, '0073621119267', 7, 995,'2017-01-26', 1 ),

(151,'AIRE ACONDICIONADO SAMSUNG 3000FC AR12FQFTAUR',7949.00, '0880608575497', 7, 34,'2017-01-03', 1 ),

(152,'AIRE ACONDICIONADO SANYO 2600W FC KC913HSAN',6099.00, '0779696244956', 7, 372,'2017-01-23', 1 ),

(153,'AIRE ACONDICIONADO SANYO 3200W FC KC1213HSAN',6899.00, '0779696242957', 7, 260,'2017-02-02', 1 ),

(154,'AIRE ACONDICIONADO SURREYPRIA 2250FC 553EPQ0913F',6929.00, '0779708708630', 7, 38,'2016-12-30', 1 ),

(155,'AIRE ACONDICIONADO SURREYPRIA 3000FC 553EPQ1213F',7949.00, '0779708708631', 7, 180,'2017-01-04', 1 ),

(156,'AIRE ACONDICIONADO SURREYPRIA 4500FC 553EPQ1813F',11849.00, '0779708708632', 7, 232,'2017-01-07', 1 ),

(157,'AIRE ACONDICIONADO SURREYPRIA 5500FC 553EPQ2213F',14329.00, '0779708708633', 7, 909,'2017-01-10', 1 ),

(158,'CALEFACTOR SIN SALIDA 4000 KCAL VOLCAN',1159.00, '0779703781219', 7, 598,'2016-12-23', 1 ),

(159,'CALEFACTOR SIN SALIDA ORBIS 4200 KCAL',1469.00, '0779703781123', 7, 504,'2017-01-11', 0 ),

(160,'ESTUFA ORBIS TIRO BALANCEADO 5000 K',2019.00, '0779703781129', 7, 600,'2017-01-17', 1 ),

(161,'ESTUFA VOLCAN TIRO BALANCEADO 2000 KCAL 42312V',1439.00, '0779703781220', 7, 602,'2016-12-28', 1 ),

(162,'ESTUFA VOLCAN TIRO BALANCEADO NEGRO 3800 43712V',1679.00, '0779703781221', 7, 650,'2017-02-04', 1 ),

(163,'TIRO BALANCEADO 3500 KCAL EMEGE',1605.00, '0779135400180', 7, 474,'2017-01-29', 1 ),

(164,'CALEFACTOR ELECTRICO CLEVER VIDRIO H1107',1950.00, '0779815957117', 7, 459,'2016-12-29', 1 ),

(165,'CALEFACTOR ELECTRICO CONVECCION CON-1800',1599.00, '0779814958212', 7, 10,'2017-01-13', 1 ),

(166,'CALEFACTOR ELECTRICO CONVECCION CON-2000N',790.00, '0779815957180', 7, 112,'2017-01-11', 1 ),

(167,'CALEFACTOR ELECTRICO CONVECCION CON-2000R',790.00, '0779815957181', 7, 141,'2017-01-26', 1 ),

(168,'CALEFACTOR LILIANA INFRARROJO CI062',345.00, '0779386200687', 7, 516,'2016-12-27', 1 ),

(169,'CALEFACTOR PANEL 500 WATTS',769.00, '0779813482002', 7, 804,'2017-01-03', 1 ),

(170,'CALOVENTOR 2000 W AXEL AX-CA100',249.00, '0779811896139', 7, 780,'2017-01-10', 1 ),

(171,'CALOVENTOR DE PARED 2000 W KENBROWN',839.00, '0779811320136', 7, 737,'2016-12-28', 1 ),

(172,'CALOVENTOR DE PARED PROTALIA CP200A',799.00, '0779811559131', 7, 833,'2017-01-30', 1 ),

(173,'CALOVENTOR ELECTRICO BLANCO 1500W LE1500B',599.00, '0779815957245', 7, 492,'2017-01-04', 1 ),

(174,'CALOVENTOR ELECTRICO LE1500ROJO',599.00, '0779815957247', 7, 437,'2017-01-29', 1 ),

(175,'CALOVENTOR ELECTRICO NEGRO 1500W LE1500N',599.00, '0779815957246', 7, 875,'2017-01-09', 1 ),

(176,'CALOVENTOR ELECTROLUX SPLIT CONTROL REMOTO',999.00, '0779386200613', 7, 675,'2016-12-20', 1 ),

(177,'CALOVENTOR KEN BROWN 2000 W',319.00, '0779811320075', 7, 76,'2017-01-23', 1 ),

(178,'CALOVENTOR RESISTENCIA CERAMICA',319.00, '0557306319076', 7, 243,'2017-01-08', 1 ),

(179,'CIRCULADOR DE AIRE FRIO CALOR DURABRAND',1049.00, '0073621119287', 7, 121,'2017-01-30', 1 ),

(180,'CONVECTOR AXEL 2000 W AX-COT100',689.00, '0779811896141', 7, 357,'2016-12-24', 1 ),

(181,'CONVECTOR AXEL 2000 W CON TURBO AX-COT',609.00, '0779811896131', 7, 246,'2017-01-16', 1 ),

(182,'CONVECTOR CLEVER CLEVERBLANCO CON2000B',790.00, '0779815957179', 7, 229,'2017-01-09', 1 ),

(183,'CONVECTOR TELEFUNKEN 2000 WATT C1009',479.00, '0779724533114', 7, 642,'2016-12-29', 1 ),

(184,'ESTUFA ELECTROLUX HALOGENAS HAL18G',549.00, '0779386200254', 7, 295,'2017-01-15', 1 ),

(185,'ESTUFA ELECTRICA KEN BROWN 2 VELAS 800 KB 22',245.00, '0779811320288', 7, 598,'2016-12-24', 1 ),

(186,'ESTUFA HALOGENA 3 VELAS KEN BROWN',409.00, '0779811320134', 7, 580,'2016-12-24', 1 ),

(187,'ESTUFA HALOGENA 4 VELAS KEN BROWN',449.00, '0779811320135', 7, 741,'2017-01-28', 1 ),

(188,'ESTUFA HALOGENA ELECTROLUX 1600W SIN OSCILACION HAL18A',499.00, '0779386200253', 7, 632,'2016-12-23', 1 ),

(189,'ESTUFA HALOGENA MAGIC 1200 W C1007',189.00, '0779724533112', 7, 518,'2016-12-26', 1 ),

(190,'PANEL 1000W ATMA',99999.00, '0779696280631', 7, 951,'2017-01-17', 1 ),

(191,'PANEL 2000 W NEGRO ENERGY SAVE',1499.00, '0779814951036', 7, 647,'2016-12-20', 1 ),

(192,'PANEL 500 W ECOSOL',1119.00, '0779813482029', 7, 805,'2017-01-18', 1 ),

(193,'PANEL 900W ECOSOL 1-502',1869.00, '0779813482031', 7, 726,'2017-02-01', 1 ),

(194,'PANEL MICA ELECTROLUX RMIC15',999.00, '0779386200256', 7, 331,'2016-12-26', 1 ),

(195,'PANEL PIETRA 500 W PEISA',699.00, '0779808116284', 7, 171,'2017-01-27', 1 ),

(196,'RADIADOR DE MICA ELECTROLUX 1000W RALU01',699.00, '0779817317015', 7, 987,'2017-01-24', 1 ),

(197,'TURBO CALENTADOR 2000W TCAL2000',590.00, '0779815957248', 7, 539,'2017-01-03', 1 ),

(198,'VENTILADOR DE PIE DURABRAND 18" VP21',122.00, '0779797170650', 7, 318,'2017-01-31', 1 ),

(199,'CAMARA DIGITAL C1433 SLVER GE',899.00, '0084695100018', 6, 528,'2017-02-02', 1 ),

(200,'LIMPIADOR CD SV 8336 ONE FOR ALL',55.00, '0871618404342', 1, 508,'2016-12-27', 1 ),

(201,'LIMPIADOR LCD SV 8410 ONE FOR ALL',102.00, '0871618404333', 1, 186,'2017-02-02', 1 )

;`

);

}

*// cerrar la base*

db.close();

}

CrearBaseSiNoExiste();

module.exports = CrearBaseSiNoExiste;

**Observe:**

* inicialmente se crea el archivo correspondiente a la base: pymes.db
* Se verifica consultado el esquema de sqlite si existe la tabla usuarios y si corresponde, se la crea. si se creo la tabla, se insertan un conjunto de registros
* Se verifica consultado el esquema de sqlite si existe la tabla articulosfamilias y si corresponde, se la crea. si se creo la tabla, se insertan un conjunto de registros
* se verifica consultando el esquema de sqlite si existe la tabla articulos y si corresponde, se la crea. Si se creó la tabla artículos, se insertan un conjunto de registros.

Ahora ejecutaremos únicamente el código recién creado, para testear su correcto funcionamiento, verificando si efectivamente crea la base de datos:

* + comando: node base-orm/sqlite-init
  + verificamos en el carpeta data, buscando el archivo pymes.db el que podemos abrir con alguna aplicación/extensión adecuada para ver su contenido.

por ej DBeaver



Una vez probado nuestro código de creación de base de datos, lo invocamos en la aplicación en el index.js justo después de crear el objeto "app", mediante la siguiente línea:

require("./base-orm/sqlite-init"); *// crear base si no existe*

* Agregamos el archivo "sequelize-init.js" (dentro de la carpeta /base-orm) que contiene la definición del modelo de datos del ORM sequelize:

*// configurar ORM sequelize*

const { Sequelize, DataTypes } = require("sequelize");

*//const sequelize = new Sequelize("sqlite:" + process.env.base );*

const sequelize = new Sequelize("sqlite:" + "./.data/pymes.db");

*// definicion del modelo de datos*

const articulosfamilias = sequelize.define(

"articulosfamilias",

{

IdArticuloFamilia: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true,

},

Nombre: {

*// todo evitar que string autocomplete con espacios en blanco, debería ser varchar sin espacios*

type: DataTypes.STRING(30),

allowNull: false,

validate: {

notEmpty: {

args: true,

msg: "Nombre es requerido",

},

len: {

args: [5, 30],

msg: "Nombre debe ser tipo caracteres, entre 5 y 30 de longitud",

},

},

},

},

{

*// pasar a mayusculas*

hooks: {

beforeValidate: function (articulofamilia, options) {

if (typeof articulofamilia.Nombre === "string") {

articulofamilia.Nombre = articulofamilia.Nombre.toUpperCase().trim();

}

},

},

timestamps: false,

}

);

const articulos = sequelize.define(

"articulos",

{

IdArticulo: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true,

},

Nombre: {

type: DataTypes.STRING(60),

allowNull: false,

validate: {

notEmpty: {

args: true,

msg: "Nombre es requerido",

},

len: {

args: [5, 60],

msg: "Nombre debe ser tipo caracteres, entre 5 y 60 de longitud",

},

},

unique: {

args: true,

msg: "este Nombre ya existe en la tabla!",

},

},

Precio: {

type: DataTypes.DECIMAL(10, 2),

allowNull: false,

validate: {

notNull: {

args: true,

msg: "Precio es requerido",

}

}

},

CodigoDeBarra: {

type: DataTypes.STRING(13),

allowNull: false,

validate: {

notNull: {

args: true,

msg: "Codigo De Barra es requerido",

},

is: {

args: ["^[0-9]{13}$", "i"],

msg: "Codigo de Barra debe ser numérico de 13 digitos",

},

},

},

IdArticuloFamilia: {

type: DataTypes.INTEGER,

allowNull: false,

validate: {

notNull: {

args: true,

msg: "IdArticuloFamilia es requerido",

}

}

},

Stock: {

type: DataTypes.INTEGER,

allowNull: false,

validate: {

notNull: {

args: true,

msg: "Stock es requerido",

}

}

},

FechaAlta: {

type: DataTypes.STRING,

allowNull: false,

validate: {

notNull: {

args: true,

msg: "Fecha Alta es requerido",

}

}

},

Activo: {

type: DataTypes.BOOLEAN,

allowNull: false,

validate: {

notNull: {

args: true,

msg: "Activo es requerido",

}

}

},

},

{

*// pasar a mayusculas*

hooks: {

beforeValidate: function (articulo, options) {

if (typeof articulo.Nombre === "string") {

articulo.Nombre = articulo.Nombre.toUpperCase().trim();

}

},

},

timestamps: false,

}

);

module.exports = {

sequelize,

articulosfamilias,

articulos,

};

**Observe:**

1. la definición del modelo de articulosfamilias
2. la definición del modelo de articulos
3. las validaciones del modelo con sus mensajes de error
4. los hooks para pasar a mayúsculas los datos y evitan que se ingresen datos con espacios en blanco al inicio o al final, antes de validarlos; y que junto a estilos en el frontend, dan coherencia a los datos ingresados por el usuario.

En la carpeta routes creamos el archivo "articulosfamilias.js" que gestionará el recurso articulosfamilias con los datos provenientes de la base de datos a través del ORM; con el siguiente código:

const express = require("express");

const router = express.Router();

const db = require("../base-orm/sequelize-init");

router.get("/api/articulosfamilias", async function (req, res, next) {

let data = await db.articulosfamilias.findAll({

attributes: ["IdArticuloFamilia", "Nombre"],

});

res.json(data);

});

module.exports = router;

**Observe:**

* el acceso al ORM mediante el módulo: sequelize-init
* el controlador GET de la ruta "/api/articulosfamilas" que devolverá serializado como json el array de datos, obtenido desde la base

Una vez definido el controlador de nuestro recurso debemos vincularlo a nuestra aplicación express, cargando el módulo de ruta en el archivo index.js antes de levantar el servidor

const articulosfamiliasRouter = require("./routes/articulosfamilias");

app.use(articulosfamiliasRouter);

Para testear nuestro recurso, iniciemos nuestra aplicación y consultemos desde el explorador la siguiente url: <http://localhost:3000/api/articulosfamilias>

***Ejercicio:*** implementar el método GET que devuelve un articulosfamilias según su id. Deberá retornar el registro específico solicitado, si no existiese devolver un error adecuado.

* + tips: usar la misma firma/estructura del articulosfamiliasmock

## 

## Etapa 4

## webapi Articulos

Ahora implementaremos la webapi artículos, que contendrá toda la funcionalidad para la gestión del recurso artículos (CRUD Create,Read,Update,Delete = ABMC Alta,Baja,Modificacion,Consulta)

En la carpeta routes creamos el archivo "articulos.js", con el siguiente código:

const express = require("express");

const router = express.Router();

const db = require("../base-orm/sequelize-init");

const { Op, ValidationError } = require("sequelize");

router.get("/api/articulos", async function (req, res, next) {

*// #swagger.tags = ['Articulos']*

*// #swagger.summary = 'obtiene todos los Articulos'*

*// consulta de artículos con filtros y paginacion*

let where = {};

if (req.query.Nombre != undefined && req.query.Nombre !== "") {

where.Nombre = {

[Op.like]: "%" + req.query.Nombre + "%",

};

}

if (req.query.Activo != undefined && req.query.Activo !== "") {

*// true o false en el modelo, en base de datos es 1 o 0*

*// convertir el string a booleano*

where.Activo = req.query.Activo === "true";

}

const Pagina = req.query.Pagina ?? 1;

const TamañoPagina = 10;

const { count, rows } = await db.articulos.findAndCountAll({

attributes: [

"IdArticulo",

"Nombre",

"Precio",

"Stock",

"FechaAlta",

"Activo",

],

order: [["Nombre", "ASC"]],

where,

offset: (Pagina - 1) \* TamañoPagina,

limit: TamañoPagina,

});

return res.json({ Items: rows, RegistrosTotal: count });

});

router.get("/api/articulos/:id", async function (req, res, next) {

*// #swagger.tags = ['Articulos']*

*// #swagger.summary = 'obtiene un Articulo'*

*// #swagger.parameters['id'] = { description: 'identificador del Articulo...' }*

let items = await db.articulos.findOne({

attributes: [

"IdArticulo",

"Nombre",

"Precio",

"CodigoDeBarra",

"IdArticuloFamilia",

"Stock",

"FechaAlta",

"Activo",

],

where: { IdArticulo: req.params.id },

});

res.json(items);

});

router.post("/api/articulos/", async (req, res) => {

*// #swagger.tags = ['Articulos']*

*// #swagger.summary = 'agrega un Articulo'*

*/\* #swagger.parameters['item'] = {*

*in: 'body',*

*description: 'nuevo Artículo',*

*schema: { $ref: '#/definitions/Articulos' }*

*} \*/*

try {

let data = await db.articulos.create({

Nombre: req.body.Nombre,

Precio: req.body.Precio,

CodigoDeBarra: req.body.CodigoDeBarra,

IdArticuloFamilia: req.body.IdArticuloFamilia,

Stock: req.body.Stock,

FechaAlta: req.body.FechaAlta,

Activo: req.body.Activo,

});

res.status(200).json(data.dataValues); *// devolvemos el registro agregado!*

} catch (err) {

if (err instanceof ValidationError) {

*// si son errores de validación, los devolvemos*

let messages = '';

err.errors.forEach((x) => messages += (x.path ?? 'campo') + ": " + x.message + '\n');

res.status(400).json({message : messages});

} else {

*// si son errores desconocidos, los dejamos que los controle el middleware de errores*

throw err;

}

}

});

router.put("/api/articulos/:id", async (req, res) => {

*// #swagger.tags = ['Articulos']*

*// #swagger.summary = 'actualiza un Artículo'*

*// #swagger.parameters['id'] = { description: 'identificador del Artículo...' }*

*/\* #swagger.parameters['Articulo'] = {*

*in: 'body',*

*description: 'Articulo a actualizar',*

*schema: { $ref: '#/definitions/Articulos' }*

*} \*/*

try {

let item = await db.articulos.findOne({

attributes: [

"IdArticulo",

"Nombre",

"Precio",

"CodigoDeBarra",

"IdArticuloFamilia",

"Stock",

"FechaAlta",

"Activo",

],

where: { IdArticulo: req.params.id },

});

if (!item) {

res.status(404).json({ message: "Artículo no encontrado" });

return;

}

item.Nombre = req.body.Nombre;

item.Precio = req.body.Precio;

item.CodigoDeBarra = req.body.CodigoDeBarra;

item.IdArticuloFamilia = req.body.IdArticuloFamilia;

item.Stock = req.body.Stock;

item.FechaAlta = req.body.FechaAlta;

item.Activo = req.body.Activo;

await item.save();

*// otra forma de hacerlo*

*// let data = await db.articulos.update(*

*// {*

*// Nombre: req.body.Nombre,*

*// Precio: req.body.Precio,*

*// CodigoDeBarra: req.body.CodigoDeBarra,*

*// IdArticuloFamilia: req.body.IdArticuloFamilia,*

*// Stock: req.body.Stock,*

*// FechaAlta: req.body.FechaAlta,*

*// Activo: req.body.Activo,*

*// },*

*// { where: { IdArticulo: req.params.id } }*

*// );*

res.sendStatus(204);

} catch (err) {

if (err instanceof ValidationError) {

*// si son errores de validación, los devolvemos*

let messages = '';

err.errors.forEach((x) => messages += x.path + ": " + x.message + '\n');

res.status(400).json({message : messages});

} else {

*// si son errores desconocidos, los dejamos que los controle el middleware de errores*

throw err;

}

}

});

router.delete("/api/articulos/:id", async (req, res) => {

*// #swagger.tags = ['Articulos']*

*// #swagger.summary = 'elimina un Articulo'*

*// #swagger.parameters['id'] = { description: 'identificador del Articulo..' }*

let bajaFisica = false;

if (bajaFisica) {

*// baja fisica*

let filasBorradas = await db.articulos.destroy({

where: { IdArticulo: req.params.id },

});

if (filasBorradas == 1) res.sendStatus(200);

else res.sendStatus(404);

} else {

*// baja lógica*

try {

let data = await db.sequelize.query(

"UPDATE articulos SET Activo = case when Activo = 1 then 0 else 1 end WHERE IdArticulo = :IdArticulo",

{

replacements: { IdArticulo: +req.params.id },

}

);

res.sendStatus(200);

} catch (err) {

if (err instanceof ValidationError) {

*// si son errores de validación, los devolvemos*

const messages = err.errors.map((x) => x.message);

res.status(400).json(messages);

} else {

*// si son errores desconocidos, los dejamos que los controle el middleware de errores*

throw err;

}

}

}

});

module.exports = router;

**Observe:**

* el uso de la librería swagger-jsdoc para documentar la api
* el uso de la librería sequelize para acceder a la base de datos

Una vez definido el controlador de nuestro recurso debemos vincularlo a nuestra aplicación express, cargando el módulo de ruta en el archivo index.js antes de levantar el servidor

const articulosRouter = require("./routes/articulos");

app.use(articulosRouter);

Haremos testing de esta webapi desde postman:

En postman importar colección de pruebas desde el archivo: dds-backend.postman\_collection.json

* verifique los errores devueltos por el modelo de ORM
  + Nombre: entre 4 y 50 caracteres
* verifique los errores devueltos por las restricciones de la base de datos:
  + clave única sobre el campo Nombre

Nota: recuerde que puede reinicializar los datos simplemente eliminando el archivo de base de datos /.data/pymes.db (se volverá a recrear)

***Ejercicio 1:*** modifique el método GET que devuelve todos los registros, para que acepte opcionalmente los parámetros Nombre y Activo, y si vienen estos aplicarlos como filtro para traer los registros. También considerar un parámetro Pagina que permitirá paginar el resultado (pagina de a 10 registros), devolviendo únicamente la página solicitada. Finalmente el método deberá devolver los datos con la siguiente estructura de salida: {Items, RegistrosTotal} En donde items serán los registros filtrados y paginados, mientras que RegistrosTotal indicará la cantidad total de registros en la base que cumplen con el filtro solicitado antes de paginar.

***Ejercicio 2:*** modifique el método DELETE para que el mismo no haga una baja física, sino una baja lógica; mediante el cambio del valor del campo Activo (1 o 0)

Configurar el servidor para nuestras webapi, puedan ser consumidos desde otra url que no sea la propia, seguimos estos 2 pasos:

instalamos la libreria cors:

npm i cors

y en el archivo index.js antes de levantar el servidor configuramos, en este caso, solo a modo de ejemplo, que cualquier url pueda consumir nuestra aplicación:

## // configurar servidor

## const cors = require("cors");

## app.use(

## cors({

## origin: "\*", // origin: 'https://dds-frontend.azurewebsites.net'

## })

## );

## 

## Etapa 5

En esta etapa vamos a agregar seguridad a la webapi, para ello vamos a utilizar JWT (JSON Web Token) para la autenticación y autorización.

* instalamos las dependencias necesarias

npm install jsonwebtoken

* crearemos el middleware de seguridad, que será el encargado de validar el token de acceso y autorizar el acceso a las rutas seguras, para lo cual crearemos el archivo: seguridad/auth.js, con el siguiente contenido:

const jwt = require("jsonwebtoken");

const accessTokenSecret = "youraccesstokensecret";

const refreshTokenSecret = "yourrefreshtokensecrethere";

const authenticateJWT = (req, res, next) => {

const authHeader = req.headers.authorization;

if (authHeader) {

const token = authHeader.split(" ")[1];

jwt.verify(token, accessTokenSecret, (err, user) => {

if (err) {

*//return res.sendStatus(400);*

return res.status(403).json({ message: "token no es valido" });

}

res.locals.user = user;

next();

});

} else {

*//res.sendStatus(401);*

res.status(401).json({ message: "Acceso denegado" });

}

};

module.exports = { authenticateJWT, accessTokenSecret, refreshTokenSecret };

**Observe:**

* el uso de la librería jsonwebtoken para validar el token que se recibe en el header de la petición
* accessTokenSecret: es la clave secreta para firmar el token de acceso
* refreshTokenSecret: es la clave secreta para firmar el token de refresco
* si el token es válido, se guarda el usuario en el objeto res.locals.user, para que pueda ser utilizado para luego autorizar las rutas seguras
* si el token no es válido, se devuelve un error 401 (acceso denegado) o 403 (token no válido)

Seguidamente en la carpeta routes, crearemos el archivo: seguridad.js, con el siguiente contenido:

const express = require("express");

const router = express.Router();

const jwt = require("jsonwebtoken");

const auth = require("../seguridad/auth");

const users = [

{

usuario: "admin",

clave: "123",

rol: "admin",

},

{

usuario: "juan",

clave: "123",

rol: "member",

},

];

let refreshTokens = [];

router.post("/api/login", (req, res) => {

*// #swagger.tags = ['Seguridad']*

*// #swagger.summary = 'Login de usuarios: admin:123(rol administrador), juan:123(rol miembro)'*

const { usuario, clave } = req.body;

*// Filter user from the users array by usuario and clave*

const user = users.find((u) => {

return u.usuario === usuario && u.clave === clave;

});

if (user) {

*// Generate an access token*

const accessToken = jwt.sign(

{ usuario: user.usuario, rol: user.rol },

auth.accessTokenSecret,

{ expiresIn: "20m" }

);

*// Avanzado!*

const refreshToken = jwt.sign(

{ usuario: user.usuario, rol: user.rol },

auth.refreshTokenSecret

);

refreshTokens.push(refreshToken);

res.json({

accessToken,

refreshToken,

message: "Bienvenido " + user.usuario + "!",

});

} else {

res.json({ message: "usuario or clave incorrecto" });

}

});

router.post("/api/logout", (req, res) => {

*// #swagger.tags = ['Seguridad']*

*// #swagger.summary = 'Logout: invalida el refresh token (no invalida el token actual!!!)'*

*// recordar que el token sigue válido hasta que expire, aquí evitamos que pueda renovarse cuando expire!*

let message = "Logout inválido!";

const { token } = req.body;

if (refreshTokens.includes(token)) {

message = "Usuario deslogueado correctamente!";

}

refreshTokens = refreshTokens.filter((t) => t !== token);

res.json({ message });

});

router.post("/api/token", (req, res) => {

*// #swagger.tags = ['Seguridad']*

*// #swagger.summary = 'refresh token'*

const { refreshToken } = req.body;

if (!refreshToken) {

return res.sendStatus(401);

}

if (!refreshTokens.includes(refreshToken)) {

return res.sendStatus(403);

}

jwt.verify(refreshToken, auth.refreshTokenSecret, (err, user) => {

if (err) {

return res.sendStatus(403);

}

const accessToken = jwt.sign(

{ usuario: user.usuario, rol: user.rol },

auth.accessTokenSecret,

{ expiresIn: "20m" }

);

res.json({

accessToken,

});

});

});

module.exports = router;

**Observe:**

* la definición de usuarios (clave y roles) en forma hardcodeada para simplificar el ejemplo.
* router.post("/api/login"): es el método que se encarga de autenticar al usuario, para lo cual se debe enviar el usuario y clave
* router.post("/api/logout"): es el método que se encarga de invalidar el token de refresco, para lo cual se debe enviar el token de refresco
* router.post("/api/token"): es el método que se encarga de refrescar el token de acceso, para lo cual se debe enviar el token de refresco

**Autenticación:** finalmente haremos uso del middleware de autenticación en la webapi, para lo cual modificaremos el archivo: routes/articulos.js, agregando una ruta segura para obtener todos los articulos, para lo cual agregaremos el siguiente código, antes de la instrucción module.exports

*//------------------------------------*

*//-- SEGURIDAD ---------------------------*

*//------------------------------------*

router.get(

"/api/articulosJWT",

auth.authenticateJWT,

async function (req, res, next) {

*/\* #swagger.security = [{*

*"bearerAuth1": []*

*}] \*/*

*// #swagger.tags = ['Articulos']*

*// #swagger.summary = 'obtiene todos los Artículos, con seguridad JWT, solo para rol: admin (usuario:admin, clave:123)'*

const { rol } = res.locals.user;

if (rol !== "admin") {

return res.status(403).json({ message: "usuario no autorizado!" });

}

let items = await db.articulos.findAll({

attributes: [

"IdArticulo",

"Nombre",

"Precio",

"CodigoDeBarra",

"IdArticuloFamilia",

"Stock",

"FechaAlta",

"Activo",

],

order: [["Nombre", "ASC"]],

});

res.json(items);

}

);

**Observe:**

* router.get("/api/articulosJWT", ...): es la ruta segur, que devuelve todos los articulos y que solo puede ser accedida por usuarios con rol: admin
* auth.authenticateJWT: es el middleware que se encarga de validar el token de acceso y autorizar el acceso a las rutas seguras
* En un escenario real la ruta de acceso sin seguridad “api/articulos” debería eliminarse ya que solo quedaría la ruta "/api/articulosJWT" que es la que tiene la validación de seguridad recién implementada.

Recuerde que para usar un middleware debe importarlo, esto lo haremos en las primeras lineas de codigo de nuestro archivo articulos.js con siguiente instrucción:

const auth = require("../seguridad/auth");

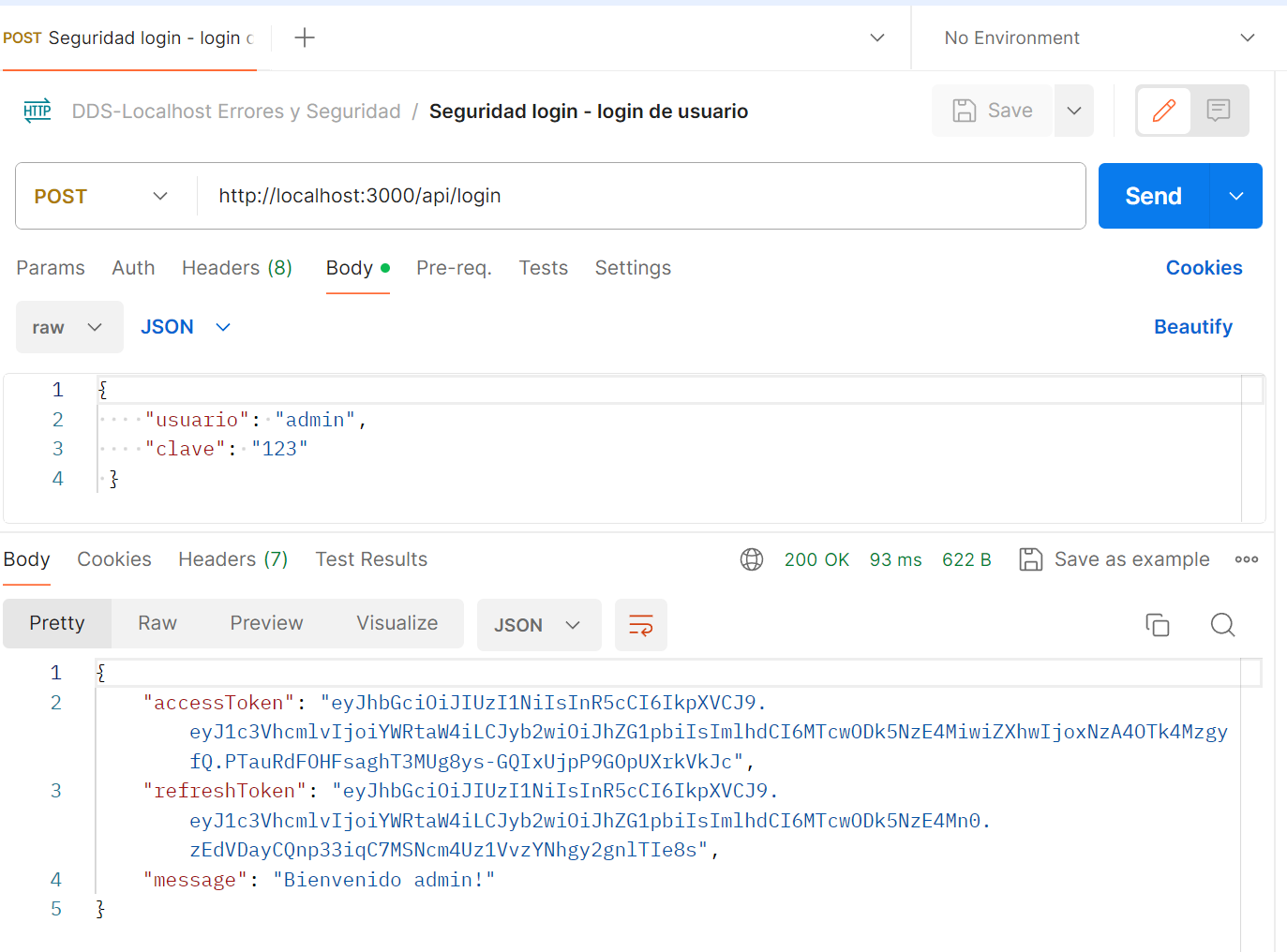
Una vez definido las rutas de seguridad de nuestro sistema, como todo controlador de ruta debemos vincularlo a nuestra aplicación express, cargando el módulo de ruta en el archivo index.js antes de levantar el servidor:

const seguridadRouter = require("./routes/seguridad");

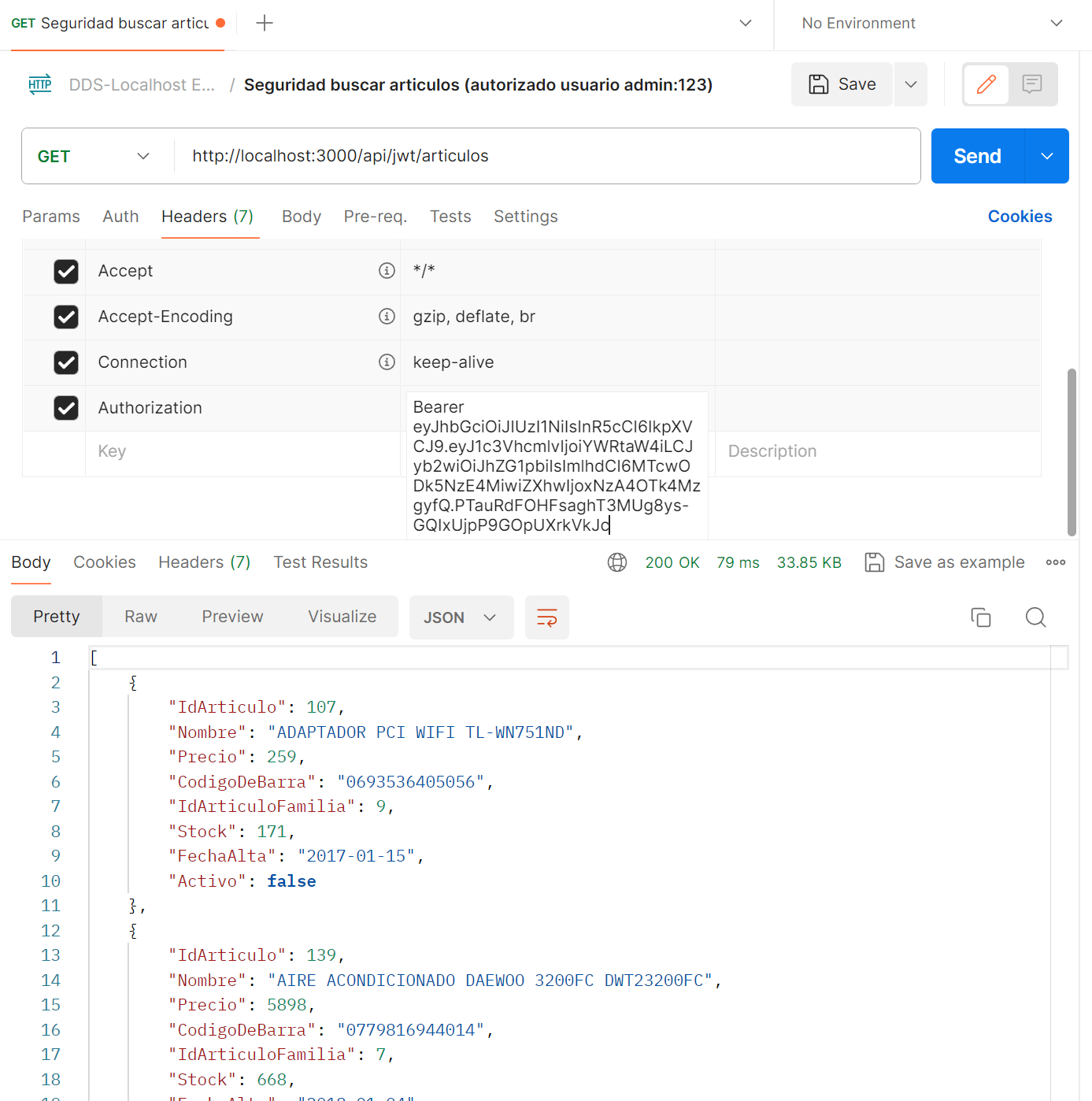
app.use(seguridadRouter);

Seguidamente haremos testing de esta webapi desde postman:

* Primero debemos invocar la api “/api/login”, pasando por parámetro usuario y clave con la cual obtendremos entre otros datos, el accessToken de nuestra aplicacion.



* Segundo invocamos alguna ruta segura de nuestra aplicación, las cuales exigen que les pasemos un accessToken, el cual sera validado para permitir o denegar el acceso al recurso.



***Ejercicio 1:*** modifique la ruta /api/articulosJWT, para que permita acceder a la misma exclusivamente a los usuarios con el rol “member”

## 

## 

## Etapa 6

En esta etapa implementaremos test unitarios para validar las webapis desarrolladas, para lo cual utilizaremos las librerías: jest y supertest. Para iniciar nos aseguramos tener instalada a nivel global la librería jest, con la cual se ejecutan las pruebas unitarias:

npm install -g jest

luego a nivel de nuestro proyecto instalaremos como dependencia de desarrollo la librería supertest:

npm install --save-dev supertest

Ya instaladas las librerías necesarias escribiremos nuestro primer archivo de test, para lo cual crearemos el archivo: test/pruebainicial.test.js, con el siguiente contenido:

const request = require("supertest");

const app = require("../index");

describe("Ejemplo simple, test que no falla", () => {

it("Simplemente compruebo si true === true", () => {

expect(true).toBe(true);

});

});

describe("GET Hola mundo!", () => {

it("Debería devolver Hola mundo!", async () => {

const res = await request(app).get("/");

expect(res.statusCode).toEqual(200);

expect(res.text).toEqual('Hola mundo!');

});

});

describe("GET \_isalive", () => {

it("Deberia devolver ejecutándose desde ...", async () => {

const res = await request(app).get("/\_isalive");

expect(res.statusCode).toEqual(200);

expect(res.text).toContain('Ejecutandose desde:');

});

});

describe("GET 404", () => {

it("Debería devolver error 404 y su texto apropiado", async () => {

const res = await request(app).get("/urlinexistente");

expect(res.statusCode).toEqual(404);

expect(res.text).toEqual("No encontrada!");

});

});

Antes de ejecutarlo necesitamos hacer un cambio a nuestra aplicación, para que la misma no inicie el servidor web al momento de ejecutar los test, para lo cual modificaremos el archivo: index.js, condicionando el inicio del servidor web, para que solo se ejecute cuando no se esté ejecutando los test y también exporte la aplicación express, para lo cual haremos el siguiente cambio:

Reemplazar:

const port = process.env.PORT || 3000; *// en producción se usa el puerto de la variable de entorno PORT*

app.locals.fechaInicio = new Date();

app.listen(port, () => {

console.log(`sitio escuchando en el puerto ${port}`);

});

por:

if (!module.parent) { *// si no es llamado por otro módulo, es decir, si es el módulo principal -> levantamos el servidor*

const port = process.env.PORT || 3000; *// en producción se usa el puerto de la variable de entorno PORT*

app.locals.fechaInicio = new Date();

app.listen(port, () => {

console.log(`sitio escuchando en el puerto ${port}`);

});

}

module.exports = app; *// para testing*

**Observe:**

* module.parent: es una variable que se define cuando se ejecuta un módulo desde otro módulo, en este caso cuando se ejecuta el test desde el archivo: test/pruebainicial.test.js, esta variable se define, por lo tanto el servidor web no se inicia, pero si se ejecuta el test desde el navegador, esta variable no se define, por lo tanto el servidor web se inicia

Ahora ejecutaremos el test, para lo cual ejecutaremos el siguiente comando:

jest test/pruebainicial.test.js

**Observe:**

* si ejecuta el comando: jest, sin especificar el archivo de test, se ejecutarán todos los test que se encuentren en la carpeta test
* si alguna prueba falla, indica que dicha prueba no pasó, y muestra el error que se produjo.

***Ejercicios:***

* En el caso de la funcionalidad Hola mundo!, si la misma no está implementada en la aplicación, le proponemos implementarla y volver a verificarla.
* En el caso de la funcionalidad \_isalive, si la misma no está implementada en la aplicación, le proponemos implementar y volver a verificarla.

Seguidamente crearemos un test para validar la webapi de articulosfamilias, para lo cual crearemos el archivo: test/articulosfamilias.test.js, con el siguiente contenido:

const request = require("supertest");

const app = require("../index");

describe("GET /api/articulosfamilias", function () {

it("Devolveria todos los artciulosfamilias", async function () {

const res = await request(app)

.get("/api/articulosfamilias")

.set("content-type", "application/json");

expect(res.headers["content-type"]).toEqual(

"application/json; charset=utf-8"

);

expect(res.statusCode).toEqual(200);

expect(res.body).toEqual(

expect.arrayContaining([

expect.objectContaining({

IdArticuloFamilia: expect.any(Number),

Nombre: expect.any(String),

}),

])

);

});

});

describe("GET /api/articulosfamilias/:id", function () {

it("respond with json containing a single artciulosfamilias", async function () {

const res = await request(app)

.get("/api/articulosfamilias/1");

expect(res.statusCode).toEqual(200);

expect(res.body).toEqual(

expect.objectContaining({

IdArticuloFamilia: 1,

Nombre: expect.any(String),

})

);

});

});

**Observe:**

* solo se testean los métodos GET; el primero testea la webapi de articulosfamilias y verifica que la respuesta sea un array con objetos que contengan los atributos IdArticuloFamilia y Nombre. El segundo testea la webapi de articulosfamilias/:id y verifica que la respuesta sea un objeto que contenga los atributos IdArticuloFamilia = 1 y Nombre sea un texto.

Ahora ejecutaremos el test, para lo cual ejecutaremos el siguiente comando:

jest test/articulosfamilias.test.js

***Ejercicio:*** Implemente los test para métodos faltantes de la webapi de articulosfamilias.

Ahora continuaremos con la implementación de test para la webapi de articulos, para lo cual crearemos el archivo: test/articulos.test.js, con el siguiente contenido:

const request = require("supertest");

const app = require("../index");

const articuloAlta = {

Nombre: "Articulo " + (() => (Math.random() + 1).toString(36).substring(2))(), *// Genera un nombre aleatorio*

Precio: 10.5,

CodigoDeBarra: "1234567890123",

IdArticuloFamilia: 1,

Stock: 11,

FechaAlta: new Date().toISOString(),

Activo: true,

};

const articuloModificacion = {

IdArticulo: 1,

Nombre: "Articulo " + (() => (Math.random() + 1).toString(36).substring(2))(), *// Genera un nombre aleatorio*

Precio: 10.5,

CodigoDeBarra: "1234567890123",

IdArticuloFamilia: 1,

Stock: 11,

FechaAlta: new Date().toISOString(),

Activo: true,

};

*// test route/articulos GET*

describe("GET /api/articulos", () => {

it("Deberia devolver todos los artículos paginados", async () => {

const res = await request(app).get("/api/articulos?Pagina=1");

expect(res.statusCode).toEqual(200);

expect(res.body).toEqual(

expect.objectContaining({

Items: expect.arrayContaining([

expect.objectContaining({

IdArticulo: expect.any(Number),

Nombre: expect.any(String),

Precio: expect.any(Number),

Stock: expect.any(Number),

FechaAlta: expect.any(String),

Activo: expect.any(Boolean)

}),

]),

RegistrosTotal: expect.any(Number),

})

);

});

});

*// test route/articulos GET*

describe("GET /api/articulos con filtros", () => {

it("Deberia devolver los articulos según filtro ", async () => {

const res = await request(app).get("/api/articulos?Nombre=AIRE&Activo=true&Pagina=1");

expect(res.statusCode).toEqual(200);

expect(verificarPropiedades(res.body.Items) ).toEqual(true );

function verificarPropiedades(array) {

for (let i = 0; i < array.length; i++) {

if ( !array[i].Nombre.includes("AIRE") || !array[i].Activo ) {

return false;

}

}

return true;

}

});

});

*// test route/articulos/:id GET*

describe("GET /api/articulos/:id", () => {

it("Deberia devolver el artículo con el id 1", async () => {

const res = await request(app).get("/api/articulos/1");

expect(res.statusCode).toEqual(200);

expect(res.body).toEqual(

expect.objectContaining({

IdArticulo: expect.any(Number),

Nombre: expect.any(String),

Precio: expect.any(Number),

CodigoDeBarra: expect.any(String),

IdArticuloFamilia: expect.any(Number),

Stock: expect.any(Number),

FechaAlta: expect.any(String),

Activo: expect.any(Boolean),

})

);

});

});

*// test route/articulos POST*

describe("POST /api/articulos", () => {

it("Deberia devolver el articulo que acabo de crear", async () => {

const res = await request(app).post("/api/articulos").send(articuloAlta);

expect(res.statusCode).toEqual(200);

expect(res.body).toEqual(

expect.objectContaining({

IdArticulo: expect.any(Number),

Nombre: expect.any(String),

Precio: expect.any(Number),

CodigoDeBarra: expect.any(String),

IdArticuloFamilia: expect.any(Number),

Stock: expect.any(Number),

FechaAlta: expect.any(String),

Activo: expect.any(Boolean),

})

);

});

});

*// test route/articulos/:id PUT*

describe("PUT /api/articulos/:id", () => {

it("Deberia devolver el articulo con el id 1 modificado", async () => {

const res = await request(app)

.put("/api/articulos/1")

.send(articuloModificacion);

expect(res.statusCode).toEqual(204);

});

});

*// test route/articulos/:id DELETE*

describe("DELETE /api/articulos/:id", () => {

it("Debería devolver el artículo con el id 1 borrado", async () => {

const res = await request(app).delete("/api/articulos/1");

expect(res.statusCode).toEqual(200);

*// baja lógica, no se borra realmente*

*// expect(res.body).toEqual(*

*// expect.objectContaining({*

*// IdArticulo: expect.any(Number),*

*// Nombre: expect.any(String),*

*// Precio: expect.any(Number),*

*// })*

*// );*

});

});

**Observe:**

* se testean los métodos GET, POST, PUT y DELETE de la webapi de artículos
* se prepara un objeto articuloAlta para testear el método POST
* se prepara un objeto articuloModificacion para testear el método PUT
* Tanto para el alta como para la modificación se genera un nombre aleatorio para el artículo, así no se repiten los nombres de los artículos en la base de datos, lo que es exigido en base de datos.

Ahora ejecutaremos el test, para lo cual ejecutaremos el siguiente comando:

jest test/articulos.test.js

Continuaremos con un test para probar las webapi de seguridad de nuestra aplicación, para lo cual crearemos el archivo: test/seguridad.test.js, con el siguiente contenido:

const request = require("supertest");

const app = require("../index");

const usuarioAdmin = { usuario: "admin", clave: "123" };

const usuarioMiembro = { usuario: "juan", clave: "123" };

describe("POST /api/login admin", function () {

it("Devolveria error de autenticacion, porque tiene clave errónea", async function () {

const res = await request(app)

.post("/api/login")

*//.set("Content-type", "application/json")*

.send({ usuario: "admin", clave: "errónea" });

expect(res.statusCode).toEqual(200);

expect(res.body.message).toEqual("usuario or clave incorrecto");

});

it("Devolvería el token para usuario admin", async function () {

const res = await request(app).post("/api/login").send(usuarioAdmin);

expect(res.statusCode).toEqual(200);

expect(res.body.accessToken).toEqual(expect.any(String));

});

});

describe("GET /api/articulosJWT", () => {

it("Devolveria error, porque falta token de autorización", async function () {

const res = await request(app).get("/api/articulosJWT");

expect(res.statusCode).toEqual(401);

expect(res.body.message).toEqual("Acceso denegado");

});

it("Devolveria error, porque el token no es válido", async function () {

const res = await request(app).get("/api/articulosJWT")

.set("Authorization", 'Bearer invalido');

expect(res.statusCode).toEqual(403);

expect(res.body.message).toEqual("token no es valido");

});

it("Devolvería todos los articulos, solo autorizado para administradores", async function () {

const res1 = await request(app)

.post("/api/login")

.set("Content-type", "application/json")

.send(usuarioAdmin);

expect(res1.statusCode).toEqual(200);

let token = res1.body.accessToken;

const res = await request(app)

.get("/api/articulosJWT")

.set("Authorization", `Bearer ${token}`);

expect(res.statusCode).toEqual(200);

expect(res.body).toEqual(

expect.arrayContaining([

expect.objectContaining({

IdArticulo: expect.any(Number),

Nombre: expect.any(String),

Precio: expect.any(Number),

CodigoDeBarra: expect.any(String),

IdArticuloFamilia: expect.any(Number),

Stock: expect.any(Number),

FechaAlta: expect.any(String),

Activo: expect.any(Boolean),

}),

])

);

});

it("Devolvería error de autorizacion, porque solo están autorizados los administradores", async function () {

const res1 = await request(app)

.post("/api/login")

.set("Content-type", "application/json")

.send(usuarioMiembro);

expect(res1.statusCode).toEqual(200);

let token = res1.body.accessToken;

const res = await request(app)

.get("/api/articulosJWT")

.set("Authorization", `Bearer ${token}`);

expect(res.statusCode).toEqual(403);

expect(res.body.message).toEqual('usuario no autorizado!');

});

});

**Observe:**

* como se testean los métodos con resultados exitosos
* como se testean los métodos con resultados erróneos

Ahora ejecutaremos el test, para lo cual ejecutaremos el siguiente comando:

jest test/seguridad.test.js

Finalmente, para poder ejecutar todos los tests, como un scripts dentro del archivo package.json, reemplazaremos la propiedad del objeto script:

"test": "echo \"Error: no test specified\" && exit 1",

por:

"test": "jest --testTimeout=10000",

con lo cual podremos ejecutar los test con el comando:

npm run test