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 backend: https://labsys.frc.utn.edu.ar/dds-backend-2025/
- Version final del proyecto con frontend: https://labsys.frc.utn.edu.ar/dds-frontend-2025/

- Requisitos tener instalado:
  - Visual studio Code
  - Node.js
  - Todos los comandos se ejecutan desde una consola de `git bash (debian laboratorio) o consola cmd (windows) o consola powershell (windows con permiso)`

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.locals.fechaInicio = new Date();  // fecha y hora inicio de aplicacion
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 (si este proyecto ya es parte de un mono-repositorio, por ej su portafolio, no ejecute la inicialización del git y adecue los msj de los 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, para que la aplicación se reinicie cuando hagamos cambios en el código, 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"

debería quedarnos algo similar a:

```
{
  "name": "dds-backend",
  "version": "1.0.0",
  "description": "backend con express",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "dev": "nodemon index.js"
  },
```

```
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.21.2"
  },
  "devDependencies": {
    "nodemon": "^3.1.9"
  }
}
```

Finalmente para ejecutar el proyecto de aquí en adelante usaremos:

```
npm run dev
```

¿que diferencia hay en ejecutar el proyecto con?

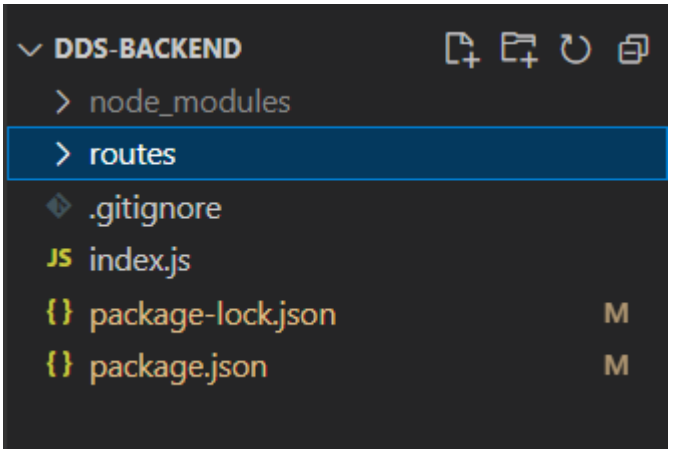
- `node index.js`
- `npm run dev`

Etapa 2

webapi categoriasmock

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 "categoriasmock.js" que gestionará el recurso categoriasmock, con el siguiente código:

```
const express = require('express');
const router = express.Router();

let arr_CategoriasMock = [
  {
    "IdCategoria": 1,
    "Nombre": "ACCESORIOS"
  },
  {
    "IdCategoria": 2,
    "Nombre": "AUDIO"
  },
  {
    "IdCategoria": 3,
```

```
    "Nombre": "CELULARES"
  },
  {
    "IdCategoria": 4,
    "Nombre": "CUIDADO PERSONAL"
  },
  {
    "IdCategoria": 5,
    "Nombre": "DVD"
  },
  {
    "IdCategoria": 6,
    "Nombre": "FOTOGRAFIA"
  },
  {
    "IdCategoria": 7,
    "Nombre": "FRIO-CALOR"
  },
  {
    "IdCategoria": 8,
    "Nombre": "GPS"
  },
  {
    "IdCategoria": 9,
    "Nombre": "INFORMATICA"
  },
  {
    "IdCategoria": 10,
    "Nombre": "LED-LCD"
  }
];

router.get('/api/categoriasmock', async function (req, res) {
  res.json(arr_CategoriasMock);
});
module.exports = router;
```

Observe:

- la clase express.Router para crear controladores de rutas montables y modulares.
- la definición moqueada del array de datos de categorias.
- el controlador GET de la ruta "/api/categoriasmock" 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 categoriasmockRouter = require("./routes/categoriasmock");
app.use(categoriasmockRouter);
```

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

http://localhost:3000/api/categoriasmock

Seguimos trabajando en el archivo categoriasmock.js, agregaremos ahora el método GET que permite obtener un recurso según su id, al archivo categoriasmock.js le agregamos este código, antes de la instrucción module.exports …

```
router.get('/api/categoriasmock/:id', async function (req, res) {
  let categoria = arr_CategoriasMock.find(
    (x) => x.IdCategoria == req.params.id
  );
  if (categoria) res.json(categoria);
  else res.status(404).json({ message: 'categoria 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 con un mensaje adecuado.

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

http://localhost:3000/api/categoriasmock/1

** Probemos 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/categoriasmock/', (req, res) => {
  const { Nombre } = req.body;
  let categoria = {
    Nombre,
    IdCategoria: Math.floor(Math.random()*100000),
  };

  // aqui agregar a la coleccion
  arr_CategoriasMock.push(categoria);

  res.status(201).json(categoria);
});
```

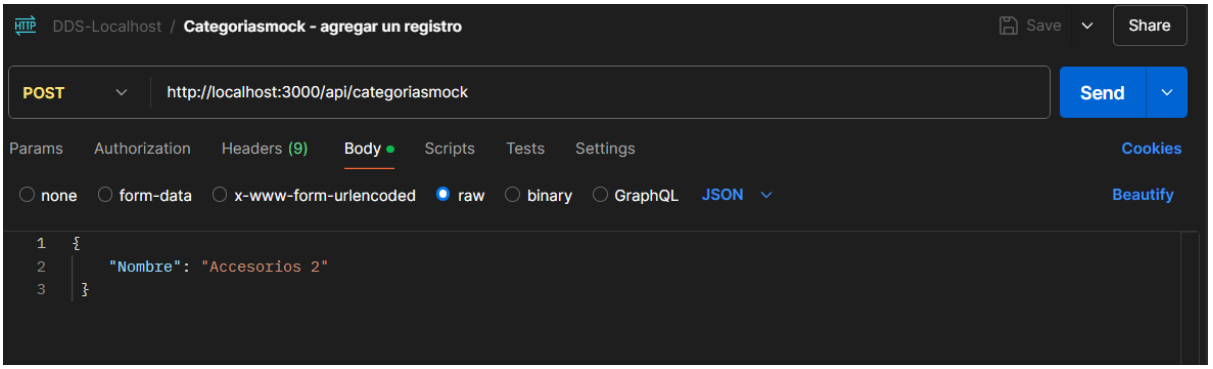
Observe:

- cómo se recupera el dato del Nombre desde el objeto "body" del request
- el campo IdCategoria, 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 IdCategoria 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 (siempre antes del module.exports):

```
router.put('/api/categoriasmock/:id', (req, res) => {
  let categoria = arr_CategoriasMock.find(
    (x) => x.IdCategoria == req.params.id
  );

  if (categoria) {
    const { Nombre } = req.body;
    categoria.Nombre = Nombre;
    res.json({ message: 'categoria actualizado' });
  } else {
    res.status(404).json({ message: 'categoria 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:

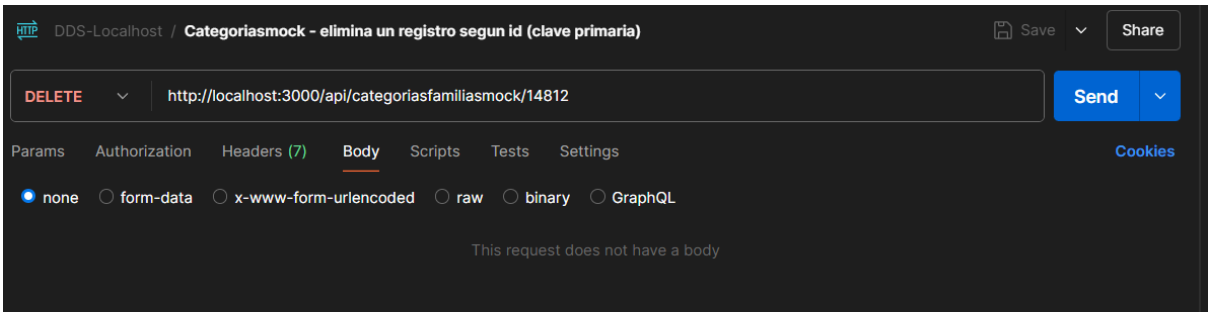
```
router.delete('/api/categoriasmock/:id', (req, res) => {
  let categoria= arr_CategoriasMock.find(
    (x) => x.IdCategoria == req.params.id
  );

  if (categoria) {
    arr_CategoriasMock = arr_CategoriasMock.filter(
      (x) => x.IdCategoria != req.params.id
    );
    res.json({ message: 'categoria eliminado' });
  } else {
    res.status(404).json({ message: 'categoria 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 devuelve todos los categorias. 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 Categorias

Antes de desarrollar nuestra primera webapi con acceso a datos, necesitamos configurar nuestra aplicacion, los mismo se almacenarán en base de datos sqlite que llamaremos Pymes.db y se acceden mediante un ORM, en nuestro caso Sequelize.

Vamos a aprovechar una forma de crear nuestra base de datos, mediante una funcionalidad que nos ofrecen muchos ORM, incluyendo Sequelize.

Los pasos a seguir serán:

- Crear los modelos de datos
- Definir el contenido de los mismos
- Sincronizar modelos y contenido con nuestra base de datos.

Crear los modelos de datos:

Inicialmente crearemos una carpeta, llamada “models” en donde estarán las definiciones de los modelos. Todos los modelos tienen algunas características en común, que configuraremos en forma global, para lo cual crearemos el archivo “/models/configurarSequelize.js” con el siguiente contenido:

```
const { Sequelize } = require('sequelize');

// Configuración de la base de datos SQLite
const sequelize = new Sequelize({
  dialect: 'sqlite',
  storage: './.data/pymes.db', // Nombre del archivo de la base de datos  (ruta desde la raíz del proyecto)
  define: {
    freezeTableName: true,  // no pluraliza los nombres de las tablas, modelo = tabla
    timestamps: false,  // no crea campos de fecha de creación y modificación

  },
});


module.exports = sequelize;
```

Agregamos al proyecto la carpeta ".data" en donde se alojará el archivo de base de datos de

sqlite: "pymes.db" (este último será creado mediante código) Para nombrar la carpeta, utilizamos “.data” por compatibilidad con la plataforma de desarrollo stackblitz. Tenga en cuenta que las carpetas o archivos que inician con un punto por defecto no están visibles en el explorador de archivos (windows/linux.), pero seria bueno cambiar está configuración para poder verlos y estar conscientes de su existencia

Observe:

Nuestro primer modelo de sequelize corresponde a la definición de Categorias, para lo cual crearemos el archivo categoriasModel.js con el siguiente contenido:

```
const { DataTypes } = require('sequelize');
const sequelize = require('./configurarSequelize');

const categorias = sequelize.define('categorias', {
  IdCategoria: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  Nombre: {
    type: DataTypes.STRING,
    allowNull: false
  }
});

module.exports = categorias;
```

Continuamos con el modelo de Articulos, para lo cual creamos el archivo articulosModel.js con el siguiente contenido:

```
const { DataTypes } = require('sequelize');
const sequelize = require('./configurarSequelize');


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",
      },
    },
  },
  IdCategoria: {
    type: DataTypes.INTEGER,
    allowNull: false,
    validate: {
      notNull: {
        args: true,
        msg: "IdCategoria 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();
     }
    },
   },
  }
);

module.exports = articulos;
```

Observe:

1. las validaciones del modelo con sus mensajes de error
2. 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.

y finalmente continuamos con el modelo de Usuarios, para lo cual creamos el archivo usuariosModel.js con el siguiente contenido:

```
const { DataTypes } = require('sequelize');
const sequelize = require('./configurarSequelize');

const usuarios = sequelize.define('usuarios', {
  IdUsuario: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true
  },
  Nombre: {
    type: DataTypes.STRING,
    allowNull: false
  },
  Clave: {
    type: DataTypes.STRING,
    allowNull: false
  },
  Rol: {
    type: DataTypes.STRING,
    allowNull: false
  }

});
console.log('usuariosModel.js');

module.exports = usuarios;
```

Una vez que tenemos la configuración general de sequelize, y la definición de los modelos que usaremos, vamos a escribir el código necesario para crear las tablas según la definición de los modelos y también les daremos datos iniciales para poder hacer nuestras pruebas. Todo el código necesario lo podremos en el archivo /models/inicializarBase.js que tiene el siguiente contenido:

```
const sequelize = require('./configurarSequelize');
const categorias = require('./categoriasModel');
const articulos = require('./articulosModel');
const usuarios = require('./usuariosModel');

if (require.main === module) {
  inicializarBase();    // si se ejecuta este archivo directamente, inicializa la base de datos, si no
```

```
se lo ejecutará antes de levantar el servidor
}


async function inicializarBase() {
  try {

   // verfica si el archivo de base de datos existe
    const fs = require('fs');
    const path = require('path');
    const archivo = path.join(__dirname, '.././data/pymes.db');
    if (fs.existsSync(archivo)) {
      return;   // si existe lo deja tal cual
    }
    // si no existe, crea la base de datos

    // Sincroniza los modelos con la base de datos
    await sequelize.sync({ force: true }); // `force: true` elimina las tablas existentes y las vuelve a
crear (¡cuidado en producción!)

    // Crea datos de prueba
    await DatosCategorias();
    await DatosArticulos();
    await DatosUsuarios();

    console.log('Base de datos inicializada y datos de prueba creados.');

  } catch (error) {
    console.error('Error al inicializar la base de datos:', error);
  }
}


async function DatosArticulos() {
  await articulos.bulkCreate([
    { ArticuloId: 1, Nombre: 'KIT DIRECT TV PREPA 0.60MT', Precio: 299.00, CodigoDeBarra:
'0779815559001', IdCategoria: 10, Stock: 329, FechaAlta: '2017-01-19', Activo: true },
    { ArticuloId: 2, Nombre: 'KIT DIRECT TV PREPA 0.90MT', Precio: 349.00, CodigoDeBarra:
'0779815559002', IdCategoria: 10, Stock: 468, FechaAlta: '2017-01-31', Activo: true },
    { ArticuloId: 3, Nombre: 'LED 22" LG FHD 22MN42APM', Precio: 2669.00, CodigoDeBarra:
'0779808338808', IdCategoria: 10, Stock: 536, FechaAlta: '2017-01-12', Activo: true },
    { ArticuloId: 4, Nombre: 'LED 24" ILO HD DIGITAL MOD LDH24ILO02', Precio: 2999.00,
CodigoDeBarra: '0779696260024', IdCategoria: 10, Stock: 169, FechaAlta: '2017-01-30', Activo:
true },
    { ArticuloId: 5, Nombre: 'LED 24" LG HD 24MN42A-PM', Precio: 3129.00, CodigoDeBarra:
'0779808338809', IdCategoria: 10, Stock: 296, FechaAlta: '2016-12-28', Activo: true },
    { ArticuloId: 7, Nombre: 'LED 32" BGH HD BLE3214D', Precio: 4830.00, CodigoDeBarra:
'0779688540133', IdCategoria: 10, Stock: 998, FechaAlta: '2017-01-01', Activo: true },
    { ArticuloId: 8, Nombre: 'LED 32" BGH SMART TV BLE3213RT', Precio: 5405.00,
CodigoDeBarra: '0779688540117', IdCategoria: 10, Stock: 650, FechaAlta: '2017-01-18', Activo:
true },
    { ArticuloId: 9, Nombre: 'LED 32" HISENSE IPTV HLE3213RT', Precio: 5290.00,
CodigoDeBarra: '0779688540119', IdCategoria: 10, Stock: 51, FechaAlta: '2017-02-03', Activo:
```

true },
{ ArticuloId: 10, Nombre: 'LED 32" HITACHI HD CDHLE32FD10', Precio: 4837.00, CodigoDeBarra: '0779694109973', IdCategoria: 10, Stock: 838, FechaAlta: '2016-12-25', Activo: true },
{ ArticuloId: 11, Nombre: 'LED 32" ILO HD DIGITAL LDH32ILO02', Precio: 4199.00, CodigoDeBarra: '0779696260132', IdCategoria: 10, Stock: 501, FechaAlta: '2017-01-25', Activo: true },
{ ArticuloId: 12, Nombre: 'LED 32" JVC HD IPTV LT32DR930', Precio: 6699.00, CodigoDeBarra: '0779818058057', IdCategoria: 10, Stock: 906, FechaAlta: '2017-01-25', Activo: true },
{ ArticuloId: 13, Nombre: 'LED 32" JVC HD LT32DA330', Precio: 4499.00, CodigoDeBarra: '0779696266323', IdCategoria: 10, Stock: 435, FechaAlta: '2017-02-07', Activo: true },
{ ArticuloId: 14, Nombre: 'LED 32" LG 3D 32LA613B', Precio: 6299.00, CodigoDeBarra: '0779808338816', IdCategoria: 10, Stock: 329, FechaAlta: '2017-02-06', Activo: true },
{ ArticuloId: 15, Nombre: 'LED 32" PHILIPS FHD 32PFL3018D/77', Precio: 6799.00, CodigoDeBarra: '0871258168715', IdCategoria: 10, Stock: 971, FechaAlta: '2016-12-25', Activo: true },
{ ArticuloId: 16, Nombre: 'LED 32" PHILIPS FHD IPTV 32PFL4508G/77', Precio: 7699.00, CodigoDeBarra: '0871258167198', IdCategoria: 10, Stock: 636, FechaAlta: '2017-02-07', Activo: true },
{ ArticuloId: 17, Nombre: 'LED 32" PHILIPS HD 32PFL3008D/77', Precio: 5799.00, CodigoDeBarra: '0871258167218', IdCategoria: 10, Stock: 67, FechaAlta: '2016-12-27', Activo: true },
{ ArticuloId: 18, Nombre: 'LED 32" PHILIPS SMART TV 32PFL3518G/77', Precio: 7399.00, CodigoDeBarra: '0871258167225', IdCategoria: 10, Stock: 250, FechaAlta: '2017-01-08', Activo: true },
{ ArticuloId: 19, Nombre: 'LED 32" RCA HD L32S80DIGI', Precio: 4499.00, CodigoDeBarra: '0779694101214', IdCategoria: 10, Stock: 857, FechaAlta: '2017-01-23', Activo: true },
{ ArticuloId: 20, Nombre: 'LED 32" SAMSUNG FHD UN32F5000', Precio: 6094.00, CodigoDeBarra: '0880608543154', IdCategoria: 10, Stock: 636, FechaAlta: '2016-12-30', Activo: true },
{ ArticuloId: 21, Nombre: 'LED 32" SAMSUNG HD UN32F4000', Precio: 5519.00, CodigoDeBarra: '0880608543153', IdCategoria: 10, Stock: 37, FechaAlta: '2017-01-23', Activo: true },
{ ArticuloId: 22, Nombre: 'LED 32" SAMSUNG SMART UN32F5500', Precio: 6899.00, CodigoDeBarra: '0880608548607', IdCategoria: 10, Stock: 214, FechaAlta: '2017-01-24', Activo: true },
{ ArticuloId: 23, Nombre: 'LED 32" SONY HD KDL32R425', Precio: 6199.00, CodigoDeBarra: '0490552491740', IdCategoria: 10, Stock: 642, FechaAlta: '2017-01-17', Activo: true },
{ ArticuloId: 24, Nombre: 'LED 32" SONY SMART TV KDL32W655', Precio: 6999.00, CodigoDeBarra: '0490552491687', IdCategoria: 10, Stock: 50, FechaAlta: '2017-02-04', Activo: true },
{ ArticuloId: 25, Nombre: 'LED 39" ILO DIG FHD LDF39ILO2', Precio: 5699.00, CodigoDeBarra: '0779696260394', IdCategoria: 10, Stock: 951, FechaAlta: '2017-01-19', Activo: true },
{ ArticuloId: 26, Nombre: 'LED 39" PHILIPS FHD IPTV 39PFL3508G/77', Precio: 8799.00, CodigoDeBarra: '0871258168717', IdCategoria: 10, Stock: 889, FechaAlta: '2017-02-03', Activo: true },
{ ArticuloId: 27, Nombre: 'LED 39" RCA FHD L39S85DIGIFHD', Precio: 6499.00, CodigoDeBarra: '0779694101215', IdCategoria: 10, Stock: 487, FechaAlta: '2016-12-25', Activo: true },
{ ArticuloId: 28, Nombre: 'LED 40" BGH FHD BLE4014D', Precio: 7245.00, CodigoDeBarra: '0779688540132', IdCategoria: 10, Stock: 480, FechaAlta: '2016-12-27', Activo: true },

{ ArticuloId: 29, Nombre: 'LED 40" SAMSUNG 3D SMART UN40F6800', Precio: 13224.00, CodigoDeBarra: '0880608565606', IdCategoria: 10, Stock: 734, FechaAlta: '2017-01-26', Activo: true },
{ ArticuloId: 30, Nombre: 'LED 40" SAMSUNG 3D UN40F6100', Precio: 9999.00, CodigoDeBarra: '0880608544958', IdCategoria: 10, Stock: 835, FechaAlta: '2017-01-19', Activo: true },
{ ArticuloId: 31, Nombre: 'LED 40" SAMSUNG FHD UN40F5000', Precio: 8164.00, CodigoDeBarra: '0880608543156', IdCategoria: 10, Stock: 436, FechaAlta: '2017-02-01', Activo: true },
{ ArticuloId: 32, Nombre: 'LED 40" SAMSUNG SMART UN40F5500', Precio: 9774.00, CodigoDeBarra: '0880608565438', IdCategoria: 10, Stock: 639, FechaAlta: '2017-01-20', Activo: true },
{ ArticuloId: 33, Nombre: 'LED 40" SONY FHD KDL40R485', Precio: 7499.00, CodigoDeBarra: '0490552493532', IdCategoria: 10, Stock: 862, FechaAlta: '2017-01-07', Activo: true },
{ ArticuloId: 34, Nombre: 'LED 42" LG 3D 42LA6130', Precio: 9199.00, CodigoDeBarra: '0779808338817', IdCategoria: 10, Stock: 560, FechaAlta: '2017-01-05', Activo: true },
{ ArticuloId: 35, Nombre: 'LED 42" LG FHD 42LN5400', Precio: 8099.00, CodigoDeBarra: '0779808338818', IdCategoria: 10, Stock: 48, FechaAlta: '2017-01-28', Activo: true },
{ ArticuloId: 36, Nombre: 'LED 42" LG SMART TV 42LN5700', Precio: 9799.00, CodigoDeBarra: '0779808338823', IdCategoria: 10, Stock: 967, FechaAlta: '2017-01-27', Activo: true },
{ ArticuloId: 37, Nombre: 'LED 42" PANASONIC 3D SMART TV TCL42ET60', Precio: 11249.00, CodigoDeBarra: '0779805518074', IdCategoria: 10, Stock: 570, FechaAlta: '2017-01-19', Activo: true },
{ ArticuloId: 38, Nombre: 'LED 42" PHILIPS 3D SMART TV 42PFL5008G/7', Precio: 11599.00, CodigoDeBarra: '0871258167039', IdCategoria: 10, Stock: 802, FechaAlta: '2017-02-04', Activo: true },
{ ArticuloId: 39, Nombre: 'LED 42" PHILIPS FHD 42PFL3008D/77', Precio: 8499.00, CodigoDeBarra: '0871258167221', IdCategoria: 10, Stock: 193, FechaAlta: '2017-02-04', Activo: true },
{ ArticuloId: 40, Nombre: 'LED 42" PHILIPS SMART TV 42PFL3508G/77', Precio: 9499.00, CodigoDeBarra: '0871258167227', IdCategoria: 10, Stock: 693, FechaAlta: '2016-12-30', Activo: true },
{ ArticuloId: 41, Nombre: 'LED 42" PIONEER 3D SMART PLE42FZP2', Precio: 12299.00, CodigoDeBarra: '0498802821943', IdCategoria: 10, Stock: 907, FechaAlta: '2017-02-01', Activo: true },
{ ArticuloId: 42, Nombre: 'LED 42" SONY FHD KDL42R475', Precio: 7999.00, CodigoDeBarra: '0490552491728', IdCategoria: 10, Stock: 140, FechaAlta: '2017-01-13', Activo: true },
{ ArticuloId: 43, Nombre: 'LED 46" PHILIPS SMART TV 46PFL4508G/7', Precio: 13999.00, CodigoDeBarra: '0871258168718', IdCategoria: 10, Stock: 236, FechaAlta: '2017-01-31', Activo: true },
{ ArticuloId: 44, Nombre: 'LED 46" SAMSUNG 3D SMART TV UN46F7500', Precio: 23574.00, CodigoDeBarra: '0880608565943', IdCategoria: 10, Stock: 143, FechaAlta: '2016-12-25', Activo: true },
{ ArticuloId: 45, Nombre: 'LED 46" SAMSUNG SMART UN46F5500', Precio: 13224.00, CodigoDeBarra: '0880608548610', IdCategoria: 10, Stock: 345, FechaAlta: '2017-01-07', Activo: true },
{ ArticuloId: 46, Nombre: 'LED 46" SANYO SMART TV LCE46IF12', Precio: 10599.00, CodigoDeBarra: '0779696260612', IdCategoria: 10, Stock: 557, FechaAlta: '2017-02-03', Activo: true },
{ ArticuloId: 47, Nombre: 'LED 47" LG SMART TV 47LN5700', Precio: 13199.00,

CodigoDeBarra: '0779808338824', IdCategoria: 10, Stock: 599, FechaAlta: '2017-01-20', Activo: true },
{ ArticuloId: 48, Nombre: 'LED 47" PIONEER 3D SMART PLE47FZP1', Precio: 15999.00, CodigoDeBarra: '0498802821947', IdCategoria: 10, Stock: 310, FechaAlta: '2017-02-07', Activo: true },
{ ArticuloId: 49, Nombre: 'LED 47" SONY 3D SMART TV KDL47W805', Precio: 17199.00, CodigoDeBarra: '0490552494098', IdCategoria: 10, Stock: 526, FechaAlta: '2017-01-31', Activo: true },
{ ArticuloId: 50, Nombre: 'LED 55" NOBLEX 3D IPTV 55LD856DI', Precio: 20799.00, CodigoDeBarra: '0779696260000', IdCategoria: 10, Stock: 362, FechaAlta: '2017-01-26', Activo: true },
{ ArticuloId: 51, Nombre: 'LED 55" PHILIPS 3D SMART TV 55PFL8008G/77', Precio: 29999.00, CodigoDeBarra: '0871258166949', IdCategoria: 10, Stock: 841, FechaAlta: '2017-01-06', Activo: true },
{ ArticuloId: 52, Nombre: 'SOPORTE LCD / LED DE 14" A 42" TANGWOOD', Precio: 599.00, CodigoDeBarra: '0779814176493', IdCategoria: 10, Stock: 527, FechaAlta: '2017-02-07', Activo: true },
{ ArticuloId: 53, Nombre: 'SOPORTE LCD / LED DE 17 \'\' A 40 \'\'', Precio: 499.00, CodigoDeBarra: '0779814176654', IdCategoria: 10, Stock: 588, FechaAlta: '2016-12-23', Activo: true },
{ ArticuloId: 54, Nombre: 'SOPORTE LCD / LED DE 17" A 37" TANGWOOD', Precio: 225.00, CodigoDeBarra: '0779814176489', IdCategoria: 10, Stock: 687, FechaAlta: '2017-01-29', Activo: true },
{ ArticuloId: 55, Nombre: 'SOPORTE LCD / LED DE 23 \'\' A 50 \'\'', Precio: 350.00, CodigoDeBarra: '0779814176652', IdCategoria: 10, Stock: 519, FechaAlta: '2016-12-25', Activo: true },
{ ArticuloId: 56, Nombre: 'SOPORTE LCD / LED DE 26" A 47" TANGWOOD', Precio: 350.00, CodigoDeBarra: '0779814176442', IdCategoria: 10, Stock: 81, FechaAlta: '2017-01-28', Activo: true },
{ ArticuloId: 57, Nombre: 'SOPORTE LCD / LED TGW DE 17 \'\' A 37 \'\'', Precio: 199.00, CodigoDeBarra: '0779814176648', IdCategoria: 10, Stock: 164, FechaAlta: '2017-01-17', Activo: true },
{ ArticuloId: 58, Nombre: 'SOPORTE LCD 10" TAGWOOD', Precio: 375.00, CodigoDeBarra: '0779814176490', IdCategoria: 10, Stock: 217, FechaAlta: '2017-01-31', Activo: true },
{ ArticuloId: 59, Nombre: 'SOPORTE LCD 32" NAKAN', Precio: 199.00, CodigoDeBarra: '0779803504550', IdCategoria: 10, Stock: 873, FechaAlta: '2017-01-01', Activo: true },
{ ArticuloId: 60, Nombre: 'SOPORTE LCD 32" ONE FOR ALL', Precio: 259.00, CodigoDeBarra: '0871618404213', IdCategoria: 10, Stock: 585, FechaAlta: '2017-01-30', Activo: true },
{ ArticuloId: 61, Nombre: 'SOPORTE LCD 40" ONE FOR ALL', Precio: 519.00, CodigoDeBarra: '0871618404215', IdCategoria: 10, Stock: 809, FechaAlta: '2017-01-22', Activo: true },
{ ArticuloId: 62, Nombre: 'SOPORTE LCD/LED 23 A 46"', Precio: 399.00, CodigoDeBarra: '0779814176617', IdCategoria: 10, Stock: 470, FechaAlta: '2017-01-21', Activo: true },
{ ArticuloId: 68, Nombre: 'SOPORTE GPS', Precio: 119.00, CodigoDeBarra: '0779814176084', IdCategoria: 8, Stock: 524, FechaAlta: '2017-01-14', Activo: true },
{ ArticuloId: 69, Nombre: 'SOPORTE GPS NEGRO MOTO 3,5" - 5,5"', Precio: 259.00, CodigoDeBarra: '0779808004535', IdCategoria: 8, Stock: 800, FechaAlta: '2017-02-05', Activo: true },
{ ArticuloId: 70, Nombre: 'GPS GARMIN NUVI 2595', Precio: 2899.00, CodigoDeBarra: '0075375999226', IdCategoria: 8, Stock: 745, FechaAlta: '2017-02-07', Activo: true },
{ ArticuloId: 71, Nombre: 'GPS GARMIN NUVI 52', Precio: 2149.00, CodigoDeBarra: '0075375999808', IdCategoria: 8, Stock: 274, FechaAlta: '2016-12-22', Activo: true },

{ ArticuloId: 72, Nombre: 'GPS X VIEW VENTURA TV 7"', Precio: 1849.00, CodigoDeBarra: '0779804220262', IdCategoria: 8, Stock: 150, FechaAlta: '2016-12-30', Activo: true },
{ ArticuloId: 73, Nombre: 'GPS XVIEW VENTURA TV', Precio: 1509.00, CodigoDeBarra: '0779804220220', IdCategoria: 8, Stock: 183, FechaAlta: '2017-01-05', Activo: true },
{ ArticuloId: 74, Nombre: 'MOUSE HP 2.4G SILVER WIRELESS OPT CAN/EN', Precio: 199.00, CodigoDeBarra: '0088496276058', IdCategoria: 9, Stock: 40, FechaAlta: '2017-02-03', Activo: true },
{ ArticuloId: 75, Nombre: 'PENDRIVE KINGSTONE DT101G2 8GB', Precio: 129.00, CodigoDeBarra: '0074061716983', IdCategoria: 9, Stock: 537, FechaAlta: '2016-12-21', Activo: true },
{ ArticuloId: 76, Nombre: 'PENDRIVE SANDISK BLADE 4GB', Precio: 129.00, CodigoDeBarra: '0061965900041', IdCategoria: 9, Stock: 340, FechaAlta: '2017-02-02', Activo: true },
{ ArticuloId: 77, Nombre: 'PENDRIVE SANDISK CRUZAR ORBIT 8GB', Precio: 159.00, CodigoDeBarra: '0061965909040', IdCategoria: 9, Stock: 696, FechaAlta: '2017-02-07', Activo: true },
{ ArticuloId: 78, Nombre: 'PENDRIVE SANDISK POP BLACK 8GB', Precio: 159.00, CodigoDeBarra: '0061965908448', IdCategoria: 9, Stock: 431, FechaAlta: '2017-01-08', Activo: true },
{ ArticuloId: 79, Nombre: 'PENDRIVE SANDISK POP PAIN 8GB', Precio: 159.00, CodigoDeBarra: '0061965908156', IdCategoria: 9, Stock: 521, FechaAlta: '2017-02-01', Activo: true },
{ ArticuloId: 80, Nombre: 'CARTUCHO EPSON 732 CYAN', Precio: 10290.00, CodigoDeBarra: '0001034385887', IdCategoria: 9, Stock: 234, FechaAlta: '2017-01-26', Activo: true },
{ ArticuloId: 81, Nombre: 'CARTUCHO EPSON T133120-AL MAGENTA', Precio: 9690.00, CodigoDeBarra: '0001034387695', IdCategoria: 9, Stock: 374, FechaAlta: '2016-12-26', Activo: true },
{ ArticuloId: 82, Nombre: 'CARTUCHO EPSON T133120-AL NEGRO', Precio: 8479.00, CodigoDeBarra: '0001034387692', IdCategoria: 9, Stock: 836, FechaAlta: '2017-01-25', Activo: true },
{ ArticuloId: 83, Nombre: 'CARTUCHO EPSON T133420-AL AMARILLO', Precio: 9690.00, CodigoDeBarra: '0001034387696', IdCategoria: 9, Stock: 796, FechaAlta: '2016-12-28', Activo: true },
{ ArticuloId: 84, Nombre: 'CARTUCHO HP 122 NEGRO', Precio: 149.00, CodigoDeBarra: '0088496298354', IdCategoria: 9, Stock: 373, FechaAlta: '2017-02-05', Activo: true },
{ ArticuloId: 85, Nombre: 'CARTUCHO HP 22 COLOR', Precio: 299.00, CodigoDeBarra: '0082916090222', IdCategoria: 9, Stock: 199, FechaAlta: '2017-01-01', Activo: true },
{ ArticuloId: 86, Nombre: 'CARTUCHO HP 60 COLOR', Precio: 289.00, CodigoDeBarra: '0088358598319', IdCategoria: 9, Stock: 801, FechaAlta: '2017-01-31', Activo: true },
{ ArticuloId: 87, Nombre: 'CARTUCHO HP 60 NEGRO', Precio: 199.00, CodigoDeBarra: '0088358598317', IdCategoria: 9, Stock: 655, FechaAlta: '2017-01-08', Activo: true },
{ ArticuloId: 88, Nombre: 'PC ALL IN ONE 120-1156LA + TECLADO INAL + MOUSE', Precio: 5499.00, CodigoDeBarra: '0088611278012', IdCategoria: 9, Stock: 331, FechaAlta: '2017-01-19', Activo: true },
{ ArticuloId: 90, Nombre: 'IMPRESORA MULTIFUNCION EPSON L355', Precio: 3999.00, CodigoDeBarra: '0001034390469', IdCategoria: 9, Stock: 293, FechaAlta: '2017-01-01', Activo: true },
{ ArticuloId: 91, Nombre: 'MULTIFUNCION EPSON L210 + SISTEMA CONTINUO', Precio: 3399.00, CodigoDeBarra: '0001034390433', IdCategoria: 9, Stock: 689, FechaAlta: '2017-01-09', Activo: true },
{ ArticuloId: 92, Nombre: 'MULTIFUNCION EPSON XP211', Precio: 1199.00, CodigoDeBarra: '0001034390754', IdCategoria: 9, Stock: 693, FechaAlta: '2017-01-08', Activo: true },

{ ArticuloId: 93, Nombre: 'MULTIFUNCION EPSON XP401', Precio: 1799.00, CodigoDeBarra: '0001034390348', IdCategoria: 9, Stock: 363, FechaAlta: '2017-01-17', Activo: true },
{ ArticuloId: 94, Nombre: 'NOTEBOOK BGH C-530 3D', Precio: 4999.00, CodigoDeBarra: '0779816664067', IdCategoria: 9, Stock: 401, FechaAlta: '2017-01-30', Activo: true },
{ ArticuloId: 95, Nombre: 'NOTEBOOK BGH C-550', Precio: 5799.00, CodigoDeBarra: '0779816664065', IdCategoria: 9, Stock: 230, FechaAlta: '2017-01-04', Activo: true },
{ ArticuloId: 96, Nombre: 'NOTEBOOK BGH C-565', Precio: 6299.00, CodigoDeBarra: '0779816664069', IdCategoria: 9, Stock: 876, FechaAlta: '2017-02-06', Activo: true },
{ ArticuloId: 97, Nombre: 'NOTEBOOK BGH C-570', Precio: 7299.00, CodigoDeBarra: '0779816664070', IdCategoria: 9, Stock: 929, FechaAlta: '2017-01-17', Activo: true },
{ ArticuloId: 98, Nombre: 'NOTEBOOK BGH QL 300 MINI', Precio: 3699.00, CodigoDeBarra: '0779816664101', IdCategoria: 9, Stock: 176, FechaAlta: '2017-01-28', Activo: true },
{ ArticuloId: 99, Nombre: 'NOTEBOOK DELL INSPIRON 14 3421 I14I32_45', Precio: 6599.00, CodigoDeBarra: '0789948950198', IdCategoria: 9, Stock: 758, FechaAlta: '2016-12-31', Activo: true },
{ ArticuloId: 100, Nombre: 'NOTEBOOK DELL INSPIRON 14 3421 I14V997_4', Precio: 5999.00, CodigoDeBarra: '0779801657005', IdCategoria: 9, Stock: 666, FechaAlta: '2016-12-20', Activo: true },
{ ArticuloId: 101, Nombre: 'NOTEBOOK LENOVO G485 C-70', Precio: 4399.00, CodigoDeBarra: '0088761972842', IdCategoria: 9, Stock: 115, FechaAlta: '2017-01-21', Activo: true },
{ ArticuloId: 102, Nombre: 'NOTEBOOK NOBLEX CEVEN GFAST', Precio: 4499.00, CodigoDeBarra: '0779808041201', IdCategoria: 9, Stock: 853, FechaAlta: '2017-02-07', Activo: true },
{ ArticuloId: 103, Nombre: 'NOTEBOOK POSITIVO BGH F-810N NEGRA', Precio: 4999.00, CodigoDeBarra: '0779816664059', IdCategoria: 9, Stock: 48, FechaAlta: '2017-01-21', Activo: true },
{ ArticuloId: 104, Nombre: 'NOTEBOOK SAMSUNG NP300E4C', Precio: 6999.00, CodigoDeBarra: '0880608528173', IdCategoria: 9, Stock: 272, FechaAlta: '2017-01-08', Activo: true },
{ ArticuloId: 105, Nombre: 'NOTEBOOK SAMSUNG NP300E5A AD4AR', Precio: 4799.00, CodigoDeBarra: '0880608500428', IdCategoria: 9, Stock: 194, FechaAlta: '2017-01-18', Activo: true },
{ ArticuloId: 106, Nombre: 'ULTRABOOK ACER S3-391-6867', Precio: 9793.00, CodigoDeBarra: '0471219655495', IdCategoria: 9, Stock: 974, FechaAlta: '2017-01-23', Activo: true },
{ ArticuloId: 107, Nombre: 'ADAPTADOR PCI WIFI TL-WN751ND', Precio: 259.00, CodigoDeBarra: '0693536405056', IdCategoria: 9, Stock: 171, FechaAlta: '2017-01-15', Activo: false },
{ ArticuloId: 110, Nombre: 'ANTENA TP-LINK TL-ANT2408C', Precio: 249.00, CodigoDeBarra: '0693536405216', IdCategoria: 9, Stock: 689, FechaAlta: '2016-12-26', Activo: true },
{ ArticuloId: 111, Nombre: 'MINI ADAPATADOR USB TP LINK WN723N', Precio: 185.00, CodigoDeBarra: '0693536405055', IdCategoria: 9, Stock: 382, FechaAlta: '2016-12-31', Activo: true },
{ ArticuloId: 112, Nombre: 'ROUTER MR3420 3G TP-LINK', Precio: 649.00, CodigoDeBarra: '0693536405149', IdCategoria: 9, Stock: 143, FechaAlta: '2016-12-21', Activo: true },
{ ArticuloId: 113, Nombre: 'ROUTER PORTATIL TP LINK TL-MR3020', Precio: 499.00, CodigoDeBarra: '0693536405170', IdCategoria: 9, Stock: 594, FechaAlta: '2017-02-01', Activo: true },
{ ArticuloId: 114, Nombre: 'ROUTER TL-WR941ND TP LINK', Precio: 759.00, CodigoDeBarra: '0693536405127', IdCategoria: 9, Stock: 646, FechaAlta: '2017-02-06', Activo: true },

{ ArticuloId: 115, Nombre: 'ROUTER TP-LINK TL-WR720N', Precio: 309.00, CodigoDeBarra: '0693536405198', IdCategoria: 9, Stock: 867, FechaAlta: '2017-01-01', Activo: true },
{ ArticuloId: 116, Nombre: 'ROUTER WR740 TP-LINK', Precio: 389.00, CodigoDeBarra: '0693536405133', IdCategoria: 9, Stock: 925, FechaAlta: '2017-01-28', Activo: true },
{ ArticuloId: 117, Nombre: 'ROUTER WR841 TP-LINK', Precio: 469.00, CodigoDeBarra: '0693536405124', IdCategoria: 9, Stock: 624, FechaAlta: '2017-01-29', Activo: true },
{ ArticuloId: 118, Nombre: 'TABLET MAGNUM TECH 7"', Precio: 2599.00, CodigoDeBarra: '0779813546539', IdCategoria: 9, Stock: 344, FechaAlta: '2016-12-26', Activo: true },
{ ArticuloId: 119, Nombre: 'TABLET 10" MAGNUM TECH 8GB 1GBM', Precio: 3799.00, CodigoDeBarra: '0779813546540', IdCategoria: 9, Stock: 751, FechaAlta: '2017-01-24', Activo: true },
{ ArticuloId: 120, Nombre: 'TABLET 10" NOBLEX NB1012', Precio: 3549.00, CodigoDeBarra: '0779696292015', IdCategoria: 9, Stock: 319, FechaAlta: '2017-01-13', Activo: true },
{ ArticuloId: 121, Nombre: 'TABLET ALCATEL AB10', Precio: 1799.00, CodigoDeBarra: '0695508989953', IdCategoria: 9, Stock: 939, FechaAlta: '2017-02-01', Activo: true },
{ ArticuloId: 122, Nombre: 'TABLET EUROCASE ARS 708', Precio: 1099.00, CodigoDeBarra: '0779813546928', IdCategoria: 9, Stock: 534, FechaAlta: '2017-01-26', Activo: true },
{ ArticuloId: 123, Nombre: 'TABLET FUNTAB PRO', Precio: 1699.00, CodigoDeBarra: '0081770701101', IdCategoria: 9, Stock: 869, FechaAlta: '2017-01-23', Activo: true },
{ ArticuloId: 124, Nombre: 'TABLET IDEAPAD LENOVO A1000L', Precio: 2799.00, CodigoDeBarra: '0088794260611', IdCategoria: 9, Stock: 597, FechaAlta: '2017-01-05', Activo: true },
{ ArticuloId: 125, Nombre: 'TABLET LENOVO IDEAPAD A1000 7"', Precio: 2299.00, CodigoDeBarra: '0088777046041', IdCategoria: 9, Stock: 510, FechaAlta: '2017-02-04', Activo: true },
{ ArticuloId: 126, Nombre: 'TABLET MAGNUM MG-701', Precio: 1499.00, CodigoDeBarra: '0779813546946', IdCategoria: 9, Stock: 645, FechaAlta: '2017-02-05', Activo: true },
{ ArticuloId: 127, Nombre: 'TABLET NOBLEX-8013 8\", Precio: 2149.00, CodigoDeBarra: '0779696291801', IdCategoria: 9, Stock: 850, FechaAlta: '2017-01-17', Activo: true },
{ ArticuloId: 130, Nombre: 'TABLET OLIPAD SMART 7" 3G', Precio: 1499.00, CodigoDeBarra: '0802033432056', IdCategoria: 9, Stock: 489, FechaAlta: '2017-02-07', Activo: true },
{ ArticuloId: 131, Nombre: 'TABLET PC 7001 TITAN', Precio: 999.00, CodigoDeBarra: '0076113310158', IdCategoria: 9, Stock: 850, FechaAlta: '2016-12-24', Activo: true },
{ ArticuloId: 132, Nombre: 'TABLET PC BOX T700U 7" DUAL CORE', Precio: 1999.00, CodigoDeBarra: '0779815876409', IdCategoria: 9, Stock: 769, FechaAlta: '2017-02-06', Activo: true },
{ ArticuloId: 133, Nombre: 'TABLET PC FIRSTAR MID070A 8650', Precio: 799.00, CodigoDeBarra: '0779815467080', IdCategoria: 9, Stock: 9, FechaAlta: '2017-01-23', Activo: true },
{ ArticuloId: 134, Nombre: 'TABLET PCBOX MOD T900', Precio: 2799.00, CodigoDeBarra: '0779815876410', IdCategoria: 9, Stock: 501, FechaAlta: '2017-01-25', Activo: true },
{ ArticuloId: 135, Nombre: 'TABLET POLAROID MID1000 10', Precio: 4299.00, CodigoDeBarra: '0358417655560', IdCategoria: 9, Stock: 151, FechaAlta: '2016-12-23', Activo: true },
{ ArticuloId: 136, Nombre: 'TABLET SYNKOM 7"', Precio: 2499.00, CodigoDeBarra: '0779816920041', IdCategoria: 9, Stock: 695, FechaAlta: '2016-12-23', Activo: true },
{ ArticuloId: 137, Nombre: 'TABLET XVIEW ALPHA2 8GB', Precio: 1899.00, CodigoDeBarra: '0779804220264', IdCategoria: 9, Stock: 565, FechaAlta: '2017-02-05', Activo: true },
{ ArticuloId: 138, Nombre: 'TABLET XVIEW PROTON', Precio: 1699.00, CodigoDeBarra: '0779804220247', IdCategoria: 9, Stock: 3, FechaAlta: '2016-12-28', Activo: true },
{ ArticuloId: 139, Nombre: 'AIRE ACONDICIONADO DAEWOO 3200FC DWT23200FC', Precio: 5898.00, CodigoDeBarra: '0779816944014', IdCategoria: 7, Stock: 668, FechaAlta: '2018-01-04', Activo: true },

{ ArticuloId: 140, Nombre: 'AIRE ACONDICIONADO DURABRAND 3500FC DUS35WCL4', Precio: 5499.00, CodigoDeBarra: '0779688543933', IdCategoria: 7, Stock: 945, FechaAlta: '2017-01-20', Activo: true },
{ ArticuloId: 141, Nombre: 'AIRE ACONDICIONADO DURABRAND 4500FC DUS53WCL4', Precio: 7499.00, CodigoDeBarra: '0779688543937', IdCategoria: 7, Stock: 962, FechaAlta: '2016-12-29', Activo: true },
{ ArticuloId: 142, Nombre: 'AIRE ACONDICIONADO KELVINATOR 2500WFC COD1056', Precio: 4499.00, CodigoDeBarra: '0779694101056', IdCategoria: 7, Stock: 670, FechaAlta: '2017-01-03', Activo: true },
{ ArticuloId: 143, Nombre: 'AIRE ACONDICIONADO LG 3000 FC H126TNW0', Precio: 7499.00, CodigoDeBarra: '0779808338858', IdCategoria: 7, Stock: 441, FechaAlta: '2017-01-09', Activo: true },
{ ArticuloId: 144, Nombre: 'AIRE ACONDICIONADO LG 4500 FC H1865NW0', Precio: 10399.00, CodigoDeBarra: '0779808338859', IdCategoria: 7, Stock: 971, FechaAlta: '2016-12-23', Activo: true },
{ ArticuloId: 145, Nombre: 'AIRE ACONDICIONADO LG 5500 FC H2465NW0', Precio: 12699.00, CodigoDeBarra: '0779808338860', IdCategoria: 7, Stock: 648, FechaAlta: '2017-01-15', Activo: true },
{ ArticuloId: 146, Nombre: 'AIRE ACONDICIONADO LG ARTCOOL 2300FC H096EFT0', Precio: 7999.00, CodigoDeBarra: '0779808338853', IdCategoria: 7, Stock: 659, FechaAlta: '2017-01-01', Activo: true },
{ ArticuloId: 147, Nombre: 'AIRE ACONDICIONADO LG ARTCOOL 4500FC H1868FT0', Precio: 12899.00, CodigoDeBarra: '0779808338855', IdCategoria: 7, Stock: 712, FechaAlta: '2016-12-25', Activo: true },
{ ArticuloId: 148, Nombre: 'AIRE ACONDICIONADO PHILCO 3200W FC PHS32H13X', Precio: 6199.00, CodigoDeBarra: '0779696244974', IdCategoria: 7, Stock: 588, FechaAlta: '2017-01-09', Activo: true },
{ ArticuloId: 149, Nombre: 'AIRE ACONDICIONADO PHILCO 5000W FC PHS50H13X', Precio: 9099.00, CodigoDeBarra: '0779696242975', IdCategoria: 7, Stock: 275, FechaAlta: '2016-12-22', Activo: true },
{ ArticuloId: 150, Nombre: 'AIRE ACONDICIONADO PORTATIL DURABRAND 2500FS LGACD01', Precio: 4999.00, CodigoDeBarra: '0073621119267', IdCategoria: 7, Stock: 995, FechaAlta: '2017-01-26', Activo: true },
{ ArticuloId: 151, Nombre: 'AIRE ACONDICIONADO SAMSUNG 3000FC AR12FQFTAUR', Precio: 7949.00, CodigoDeBarra: '0880608575497', IdCategoria: 7, Stock: 34, FechaAlta: '2017-01-03', Activo: true },
{ ArticuloId: 152, Nombre: 'AIRE ACONDICIONADO SANYO 2600W FC KC913HSAN', Precio: 6099.00, CodigoDeBarra: '0779696244956', IdCategoria: 7, Stock: 372, FechaAlta: '2017-01-23', Activo: true },
{ ArticuloId: 153, Nombre: 'AIRE ACONDICIONADO SANYO 3200W FC KC1213HSAN', Precio: 6899.00, CodigoDeBarra: '0779696242957', IdCategoria: 7, Stock: 260, FechaAlta: '2017-02-02', Activo: true },
{ ArticuloId: 154, Nombre: 'AIRE ACONDICIONADO SURREYPRIA 2250FC 553EPQ0913F', Precio: 6929.00, CodigoDeBarra: '0779708708630', IdCategoria: 7, Stock: 38, FechaAlta: '2016-12-30', Activo: true },
{ ArticuloId: 155, Nombre: 'AIRE ACONDICIONADO SURREYPRIA 3000FC 553EPQ1213F', Precio: 7949.00, CodigoDeBarra: '0779708708631', IdCategoria: 7, Stock: 180, FechaAlta: '2017-01-04', Activo: true },
{ ArticuloId: 156, Nombre: 'AIRE ACONDICIONADO SURREYPRIA 4500FC 553EPQ1813F', Precio: 11849.00, CodigoDeBarra: '0779708708632', IdCategoria: 7, Stock: 232, FechaAlta: '2017-01-07', Activo: true },
{ ArticuloId: 157, Nombre: 'AIRE ACONDICIONADO SURREYPRIA 5500FC 553EPQ2213F', Precio: 14329.00, CodigoDeBarra: '0779708708633', IdCategoria: 7, Stock: 909, FechaAlta:

'2017-01-10', Activo: true },
{ ArticuloId: 158, Nombre: 'CALEFACTOR SIN SALIDA 4000 KCAL VOLCAN', Precio: 1159.00, CodigoDeBarra: '0779703781219', IdCategoria: 7, Stock: 598, FechaAlta: '2016-12-23', Activo: true },
{ ArticuloId: 159, Nombre: 'CALEFACTOR SIN SALIDA ORBIS 4200 KCAL', Precio: 1469.00, CodigoDeBarra: '0779703781123', IdCategoria: 7, Stock: 504, FechaAlta: '2017-01-11', Activo: false },
{ ArticuloId: 160, Nombre: 'ESTUFA ORBIS TIRO BALANCEADO 5000 K', Precio: 2019.00, CodigoDeBarra: '0779703781129', IdCategoria: 7, Stock: 600, FechaAlta: '2017-01-17', Activo: true },
{ ArticuloId: 161, Nombre: 'ESTUFA VOLCAN TIRO BALANCEADO 2000 KCAL 42312V', Precio: 1439.00, CodigoDeBarra: '0779703781220', IdCategoria: 7, Stock: 602, FechaAlta: '2016-12-28', Activo: true },
{ ArticuloId: 162, Nombre: 'ESTUFA VOLCAN TIRO BALANCEADO NEGRO 3800 43712V', Precio: 1679.00, CodigoDeBarra: '0779703781221', IdCategoria: 7, Stock: 650, FechaAlta: '2017-02-04', Activo: true },
{ ArticuloId: 163, Nombre: 'TIRO BALANCEADO 3500 KCAL EMEGE', Precio: 1605.00, CodigoDeBarra: '0779135400180', IdCategoria: 7, Stock: 474, FechaAlta: '2017-01-29', Activo: true },
{ ArticuloId: 164, Nombre: 'CALEFACTOR ELECTRICO CLEVER VIDRIO H1107', Precio: 1950.00, CodigoDeBarra: '0779815957117', IdCategoria: 7, Stock: 459, FechaAlta: '2016-12-29', Activo: true },
{ ArticuloId: 165, Nombre: 'CALEFACTOR ELECTRICO CONVECCION CON-1800', Precio: 1599.00, CodigoDeBarra: '0779814958212', IdCategoria: 7, Stock: 10, FechaAlta: '2017-01-13', Activo: true },
{ ArticuloId: 166, Nombre: 'CALEFACTOR ELECTRICO CONVECCION CON-2000N', Precio: 790.00, CodigoDeBarra: '0779815957180', IdCategoria: 7, Stock: 112, FechaAlta: '2017-01-11', Activo: true },
{ ArticuloId: 167, Nombre: 'CALEFACTOR ELECTRICO CONVECCION CON-2000R', Precio: 790.00, CodigoDeBarra: '0779815957181', IdCategoria: 7, Stock: 141, FechaAlta: '2017-01-26', Activo: true },
{ ArticuloId: 168, Nombre: 'CALEFACTOR LILIANA INFRARROJO CI062', Precio: 345.00, CodigoDeBarra: '0779386200687', IdCategoria: 7, Stock: 516, FechaAlta: '2016-12-27', Activo: true },
{ ArticuloId: 169, Nombre: 'CALEFACTOR PANEL 500 WATTS', Precio: 769.00, CodigoDeBarra: '0779813482002', IdCategoria: 7, Stock: 804, FechaAlta: '2017-01-03', Activo: true },
{ ArticuloId: 170, Nombre: 'CALOVENTOR 2000 W AXEL AX-CA100', Precio: 249.00, CodigoDeBarra: '0779811896139', IdCategoria: 7, Stock: 780, FechaAlta: '2017-01-10', Activo: true },
{ ArticuloId: 171, Nombre: 'CALOVENTOR DE PARED 2000 W KENBROWN', Precio: 839.00, CodigoDeBarra: '0779811320136', IdCategoria: 7, Stock: 737, FechaAlta: '2016-12-28', Activo: true },
{ ArticuloId: 172, Nombre: 'CALOVENTOR DE PARED PROTALIA CP200A', Precio: 799.00, CodigoDeBarra: '0779811559131', IdCategoria: 7, Stock: 833, FechaAlta: '2017-01-30', Activo: true },
{ ArticuloId: 173, Nombre: 'CALOVENTOR ELECTRICO BLANCO 1500W LE1500B', Precio: 599.00, CodigoDeBarra: '0779815957245', IdCategoria: 7, Stock: 492, FechaAlta: '2017-01-04', Activo: true },
{ ArticuloId: 174, Nombre: 'CALOVENTOR ELECTRICO LE1500ROJO', Precio: 599.00, CodigoDeBarra: '0779815957247', IdCategoria: 7, Stock: 437, FechaAlta: '2017-01-29', Activo: true },
{ ArticuloId: 175, Nombre: 'CALOVENTOR ELECTRICO NEGRO 1500W LE1500N', Precio:

599.00, CodigoDeBarra: '0779815957246', IdCategoria: 7, Stock: 875, FechaAlta: '2017-01-09', Activo: true },
{ ArticuloId: 176, Nombre: 'CALOVENTOR ELECTROLUX SPLIT CONTROL REMOTO', Precio: 999.00, CodigoDeBarra: '0779386200613', IdCategoria: 7, Stock: 675, FechaAlta: '2016-12-20', Activo: true },
{ ArticuloId: 177, Nombre: 'CALOVENTOR KEN BROWN 2000 W', Precio: 319.00, CodigoDeBarra: '0779811320075', IdCategoria: 7, Stock: 76, FechaAlta: '2017-01-23', Activo: true },
{ ArticuloId: 178, Nombre: 'CALOVENTOR RESISTENCIA CERAMICA', Precio: 319.00, CodigoDeBarra: '0557306319076', IdCategoria: 7, Stock: 243, FechaAlta: '2017-01-08', Activo: true },
{ ArticuloId: 179, Nombre: 'CIRCULADOR DE AIRE FRIO CALOR DURABRAND', Precio: 1049.00, CodigoDeBarra: '0073621119287', IdCategoria: 7, Stock: 121, FechaAlta: '2017-01-30', Activo: true },
{ ArticuloId: 180, Nombre: 'CONVECTOR AXEL 2000 W AX-COT100', Precio: 689.00, CodigoDeBarra: '0779811896141', IdCategoria: 7, Stock: 357, FechaAlta: '2016-12-24', Activo: true },
{ ArticuloId: 181, Nombre: 'CONVECTOR AXEL 2000 W CON TURBO AX-COT', Precio: 609.00, CodigoDeBarra: '0779811896131', IdCategoria: 7, Stock: 246, FechaAlta: '2017-01-16', Activo: true },
{ ArticuloId: 182, Nombre: 'CONVECTOR CLEVER CLEVERBLANCO CON2000B', Precio: 790.00, CodigoDeBarra: '0779815957179', IdCategoria: 7, Stock: 229, FechaAlta: '2017-01-09', Activo: true },
{ ArticuloId: 183, Nombre: 'CONVECTOR TELEFUNKEN 2000 WATT C1009', Precio: 479.00, CodigoDeBarra: '0779724533114', IdCategoria: 7, Stock: 642, FechaAlta: '2016-12-29', Activo: true },
{ ArticuloId: 184, Nombre: 'ESTUFA ELECTROLUX HALOGENAS HAL18G', Precio: 549.00, CodigoDeBarra: '0779386200254', IdCategoria: 7, Stock: 295, FechaAlta: '2017-01-15', Activo: true },
{ ArticuloId: 185, Nombre: 'ESTUFA ELECTRICA KEN BROWN 2 VELAS 800 KB 22', Precio: 245.00, CodigoDeBarra: '0779811320288', IdCategoria: 7, Stock: 598, FechaAlta: '2016-12-24', Activo: true },
{ ArticuloId: 186, Nombre: 'ESTUFA HALOGENA 3 VELAS KEN BROWN', Precio: 409.00, CodigoDeBarra: '0779811320134', IdCategoria: 7, Stock: 580, FechaAlta: '2016-12-24', Activo: true },
{ ArticuloId: 187, Nombre: 'ESTUFA HALOGENA 4 VELAS KEN BROWN', Precio: 449.00, CodigoDeBarra: '0779811320135', IdCategoria: 7, Stock: 741, FechaAlta: '2017-01-28', Activo: true },
{ ArticuloId: 188, Nombre: 'ESTUFA HALOGENA ELECTROLUX 1600W SIN OSCILACION HAL18A', Precio: 499.00, CodigoDeBarra: '0779386200253', IdCategoria: 7, Stock: 632, FechaAlta: '2016-12-23', Activo: true },
{ ArticuloId: 189, Nombre: 'ESTUFA HALOGENA MAGIC 1200 W C1007', Precio: 189.00, CodigoDeBarra: '0779724533112', IdCategoria: 7, Stock: 518, FechaAlta: '2016-12-26', Activo: true },
{ ArticuloId: 190, Nombre: 'PANEL 1000W ATMA', Precio: 99999.00, CodigoDeBarra: '0779696280631', IdCategoria: 7, Stock: 951, FechaAlta: '2017-01-17', Activo: true },
{ ArticuloId: 191, Nombre: 'PANEL 2000 W NEGRO ENERGY SAVE', Precio: 1499.00, CodigoDeBarra: '0779814951036', IdCategoria: 7, Stock: 647, FechaAlta: '2016-12-20', Activo: true },
{ ArticuloId: 192, Nombre: 'PANEL 500 W ECOSOL', Precio: 1119.00, CodigoDeBarra: '0779813482029', IdCategoria: 7, Stock: 805, FechaAlta: '2017-01-18', Activo: true },
{ ArticuloId: 193, Nombre: 'PANEL 900W ECOSOL 1-502', Precio: 1869.00, CodigoDeBarra: '0779813482031', IdCategoria: 7, Stock: 726, FechaAlta: '2017-02-01', Activo: true },

```
   { ArticuloId: 194, Nombre: 'PANEL MICA ELECTROLUX RMIC15', Precio: 999.00,
CodigoDeBarra: '0779386200256', IdCategoria: 7, Stock: 331, FechaAlta: '2016-12-26', Activo:
true },
   { ArticuloId: 195, Nombre: 'PANEL PIETRA 500 W PEISA', Precio: 699.00, CodigoDeBarra:
'0779808116284', IdCategoria: 7, Stock: 171, FechaAlta: '2017-01-27', Activo: true },
   { ArticuloId: 196, Nombre: 'RADIADOR DE MICA ELECTROLUX 1000W RALU01', Precio:
699.00, CodigoDeBarra: '0779817317015', IdCategoria: 7, Stock: 987, FechaAlta: '2017-01-24',
Activo: true },
   { ArticuloId: 197, Nombre: 'TURBO CALENTADOR 2000W TCAL2000', Precio: 590.00,
CodigoDeBarra: '0779815957248', IdCategoria: 7, Stock: 539, FechaAlta: '2017-01-03', Activo:
true },
   { ArticuloId: 198, Nombre: 'VENTILADOR DE PIE DURABRAND 18" VP21', Precio: 122.00,
CodigoDeBarra: '0779797170650', IdCategoria: 7, Stock: 318, FechaAlta: '2017-01-31', Activo:
true },
   { ArticuloId: 199, Nombre: 'CAMARA DIGITAL C1433 SLVER GE', Precio: 899.00,
CodigoDeBarra: '0084695100018', IdCategoria: 6, Stock: 528, FechaAlta: '2017-02-02', Activo:
true },
   { ArticuloId: 200, Nombre: 'LIMPIADOR CD SV 8336 ONE FOR ALL', Precio: 55.00,
CodigoDeBarra: '0871618404342', IdCategoria: 1, Stock: 508, FechaAlta: '2016-12-27', Activo:
true },
   { ArticuloId: 201, Nombre: 'LIMPIADOR LCD SV 8410 ONE FOR ALL', Precio: 102.00,
CodigoDeBarra: '0871618404333', IdCategoria: 1, Stock: 186, FechaAlta: '2017-02-02', Activo:
true },
  ]);
}

async function DatosCategorias() {
  await categorias.bulkCreate([
    { Stock: 1, Nombre: 'ACCESORIOS' },
    { Stock: 2, Nombre: 'AUDIO' },
    { Stock: 3, Nombre: 'CELULARES' },
    { Stock: 4, Nombre: 'CUIDADO PERSONAL' },
    { Stock: 5, Nombre: 'DVD' },
    { Stock: 6, Nombre: 'FOTOGRAFIA' },
    { Stock: 7, Nombre: 'FRIO-CALOR' },
    { Stock: 8, Nombre: 'GPS' },
    { Stock: 9, Nombre: 'INFORMATICA' },
    { Stock: 10, Nombre: 'LED - LCD' },
  ]);
}

async function DatosUsuarios() {

  await usuarios.bulkCreate([
    { UsuarioId: 1, Nombre: 'admin', Clave: '123', Rol: 'jefe' },
    { UsuarioId: 2, Nombre: 'juan', Clave: '123', Rol: 'empleado' },
    { UsuarioId: 3, Nombre: 'ana', Clave: 'ana123', Rol: 'empleado' },

  ]);
}

module.exports = inicializarBase;
```

Observe:

- que este código usa todas las definición anteriores importándolas
- que tiene una ejecución condicional si el archivo pymes.db existe, es decir que si lo borramos, podemos asegurarnos que serán nuevamente creados con los datos inicializados.
- que cada modelo utiliza la función bulkCreate para que se inserten en la tabla los datos que están en los arrays correspondientes.

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

- comando: node models/inicializarBase
- verificamos en el carpeta .data, buscando el archivo pymes.db el que podemos abrir con alguna aplicación o extensión de visual studio adecuada para ver su contenido.

  por ej la aplicación 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:

```
const inicializarBase = require("./models/inicializarBase");  // inicializar
base de datos
```

y al final del mismo archivo index.js, cambiamos la forma de iniciar el servidor para que lo haga luego de la ejecución de inicializarbase:

cambiar:

```
app.listen(port, () => {
    console.log(`sitio escuchando en el puerto ${port}`);
});
```

por:

```
inicializarBase().then(() => {
  app.listen(port, () => {
    console.log(`sitio escuchando en el puerto ${port}`);
  });
});
```

Ya teniendo configurado el acceso a datos y los datos de prueba de nuestra aplicación vamos a desarrollar los endpoints de nuestra web api.

En la carpeta routes creamos el archivo "categorias.js" que gestionará el recurso categorias 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 categorias = require('../models/categoriasModel');


// Obtener todas las categorías
router.get('/api/categorias', async (req, res) => {
  try {
    const categoriass = await categorias.findAll();
    res.json(categoriass);
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Error al obtener las categorías' });
  }
});


module.exports = router;
```

Observe:

- el acceso al ORM mediante la definición de su modelo.
- el controlador GET de la ruta "/api/categorias" 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, lo hacemos con las siguiente líneas de código:

```
const categoriasRouter = require("./routes/categorias");
app.use(categoriasRouter);
```

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

---

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

- tips: usar la misma firma/estructura del categoriasmock

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 articulos = require('../models/articulosModel');
const { Op, ValidationError } = require("sequelize");


router.get("/api/articulos", async function (req, res) {
  // #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 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) {
  // #swagger.tags = ['Articulos']
  // #swagger.summary = 'obtiene un Articulo'
  // #swagger.parameters['id'] = { description: 'identificador del
Articulo...' }
   let items = await articulos.findOne({
    attributes: [
      "IdArticulo",
      "Nombre",
      "Precio",
      "CodigoDeBarra",
      "IdCategoria",
      "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 item = await articulos.create({
      Nombre: req.body.Nombre,
      Precio: req.body.Precio,
      CodigoDeBarra: req.body.CodigoDeBarra,
      IdCategoria: req.body.IdCategoria,
      Stock: req.body.Stock,
      FechaAlta: req.body.FechaAlta,
      Activo: req.body.Activo,
    });
    res.status(200).json(item.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 articulos.findOne({
      attributes: [
        "IdArticulo",
        "Nombre",
        "Precio",
        "CodigoDeBarra",
        "IdCategoria",
        "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.IdCategoria = req.body.IdCategoria;
    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 articulos.update(
    //   {
    //     Nombre: req.body.Nombre,
    //     Precio: req.body.Precio,
    //     CodigoDeBarra: req.body.CodigoDeBarra,
```

```
    //     IdCategoria: req.body.IdCategoria,
    //     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 articulos.destroy({
      where: { IdArticulo: req.params.id },
    });
    if (filasBorradas == 1) res.sendStatus(200);
    else res.sendStatus(404);
  } else {
    // baja lógica, si esta activo lo desactiva y viceversa
    try {
      let data = await articulos.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 (app.listen…)

```
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” (está se volverá a recrear automaticamente)

Si el código del frontend no está en mismo dominio que el backend, tenemos que configurar el backend para permitir este acceso, para lo cual seguimos estos 2 pasos

- instalamos la librería 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: 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.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("../auth");

const users = [
  {
    usuario: "admin",
    clave: "123",
    rol: "jefe",
  },
  {
    usuario: "juan",
    clave: "123",
    rol: "empleado",
  },
];
let refreshTokens = [];

router.post("/api/login", (req, res) => {
  // #swagger.tags = ['Seguridad']
  // #swagger.summary = 'Login de usuarios: admin:123(rol jefe), juan:123(rol
empleado)'

  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 + " (rol: " + user.rol + ")",
    });
  } 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 = null;
  const authHeader = req.headers.authorization;
  let token = null;
  if (authHeader) {
     token = authHeader.split(" ")[1];
  }

  if (refreshTokens.includes(token)) {
    message = "Usuario deslogueado correctamente!";
  }
  else {
    message = "Logout inválido!";
  }


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

```
  res.json({ message });
});

router.post("/api/refreshtoken", (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/refreshtoken"): 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 crearemos una ruta nueva: routes/usuarios.js, la cual solo implemente un método get que devolverá el listado de usuarios solo para los usuarios autenticados y autorizados.

El código es el siguiente:

```
const express = require('express');
const router = express.Router();

const usuarios = require('../models/usuariosModel');
const auth = require('../auth');


// Obtener todas los usuarios, con seguridad JWT
router.get('/api/usuarios',
  auth.authenticateJWT,
  async function (req, res, next) {
    try {
      // si llego hasta acá, es porque el token es válido y esta autenticado

      // ahora controlamos autorización, segun el rol
      const user = res.locals.user;
      if (user.rol !== "jefe") {
        return res.status(403).json({ message: "usuario no autorizado!" });
      }

      const items = await usuarios.findAll();
      res.json(items);
    } catch (error) {
      console.error(error);
      res.status(500).json({ error: 'Error al obtener los usuarios' });
    }
  });


module.exports = router;
```

Observe:

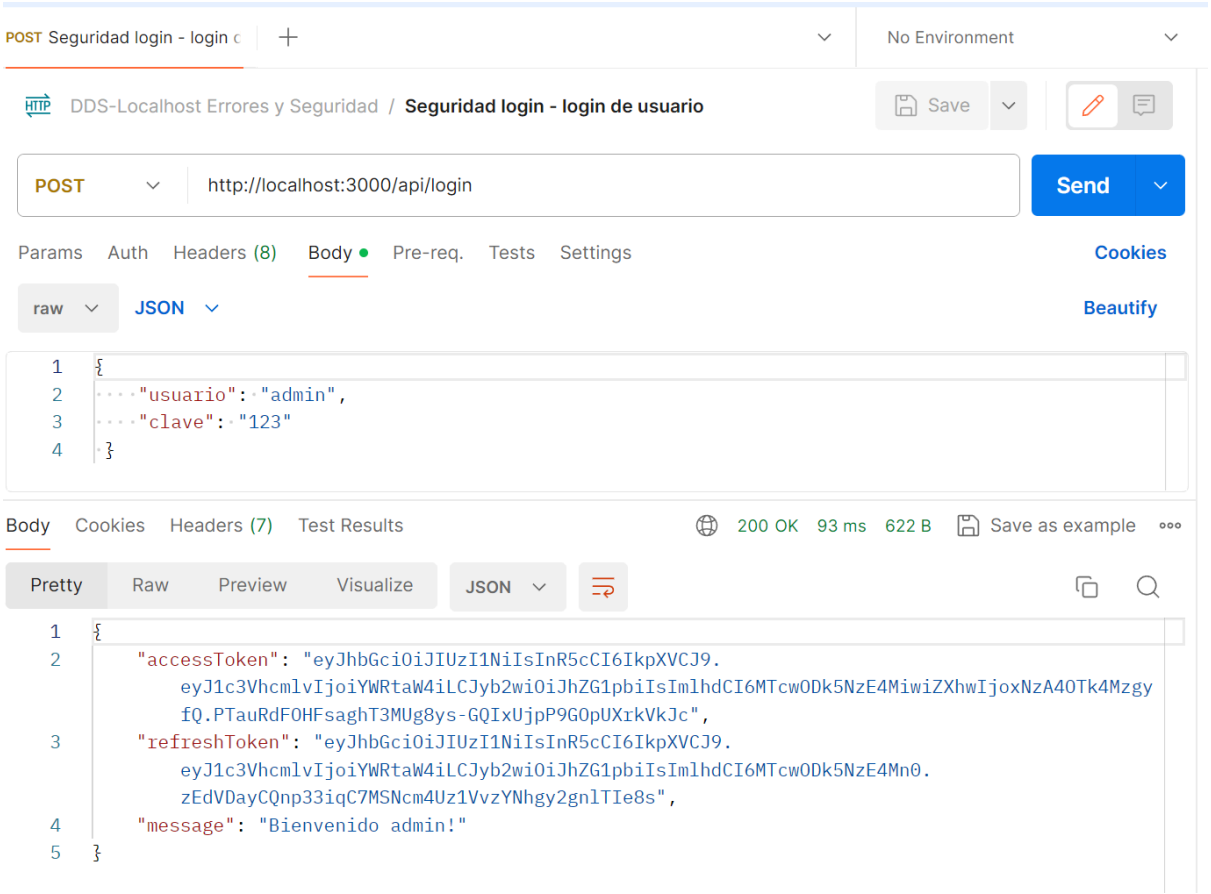
- router.get("/api/usuarios", ...): es la ruta segura, que devuelve todos los usuarios y que solo puede ser accedida por usuarios con rol: jefe
- auth.authenticateJWT: es el middleware que se encarga de validar el token de acceso y autorizar el acceso a las rutas seguras

Una vez definido las rutas de seguridad (login, logout, etc) y el del nuevo recurso (usuarios) 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:

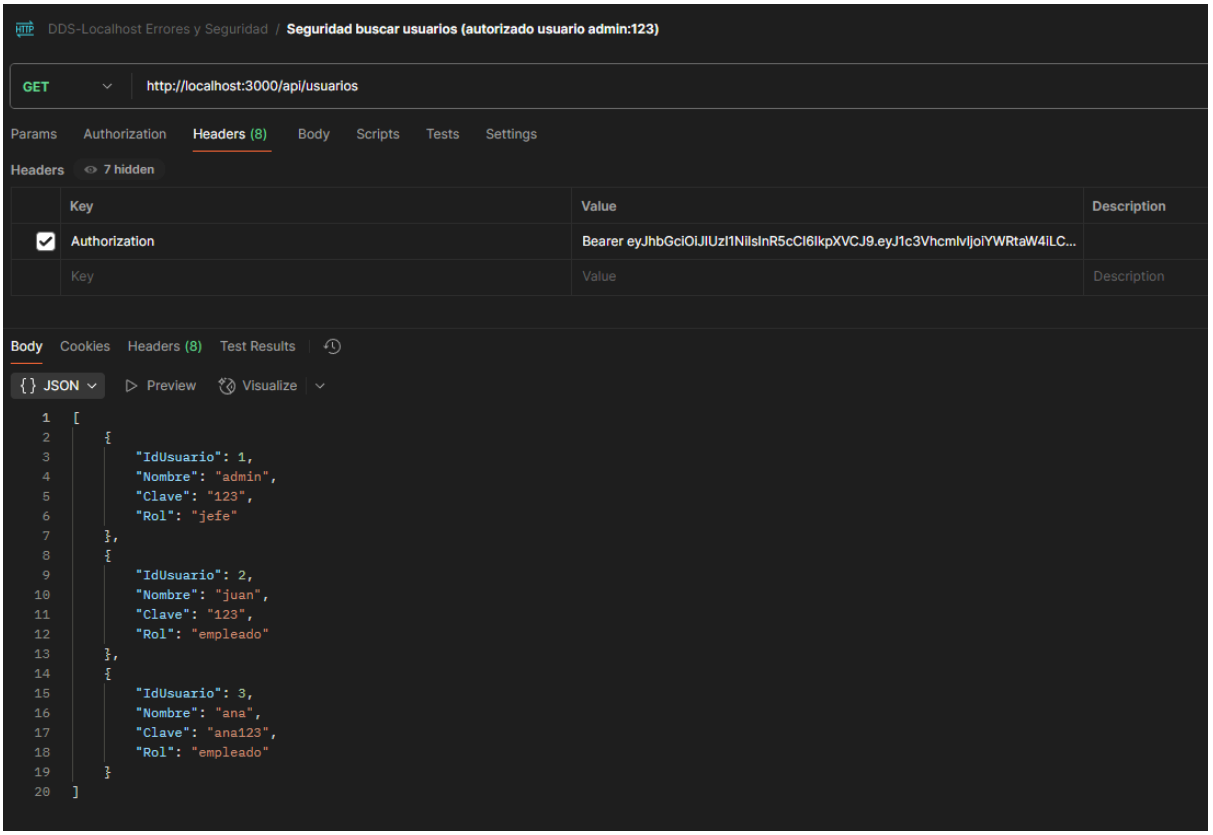
```
const seguridadRouter = require("./routes/seguridad");
app.use(seguridadRouter);
const usuariosRouter = require("./routes/usuarios");
app.use(usuariosRouter);
```

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 aplicación.



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



Observe:

- Seguridad del `accessTokenSecret y refreshTokenSecret`: Son CRÍTICOS. Si se filtran, cualquiera puede generar tokens válidos.
- Contraseñas: NUNCA en texto plano en producción. Usa bcrypt
- Payload del JWT: No incluyas información sensible que no necesites exponer.
- Expiración de Tokens: Define tiempos de expiración razonables.
- Stateless: Los JWT permiten que tu API sea "stateless" (sin estado), ya que el servidor no necesita recordar quién está logueado; la información viene en cada petición.

Ejercicio opcional: Implemente un middleware `authorizedRoles()` que tome como parámetro un array de roles. Este middleware debe verificar si el usuario posee alguno de los roles proporcionados. En caso afirmativo, permitirá el acceso a la ruta. De lo contrario, finalizará la petición con un error apropiado. Implementarlo en la ruta '/api/usuarios' recién vista.

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 Backend inicial dds-backend!", () => {
  it("Debería devolver Backend inicial dds-backend!", async () => {
    const res = await request(app).get("/");
    expect(res.statusCode).toEqual(200);
    expect(res.text).toEqual('Backend inicial dds-backend!');
  });
});

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:

```
inicializarBase().then(() => {
  app.listen(port, () => {
    console.log(`sitio escuchando en el puerto ${port}`);
  });
});
```

por:

```
if (require.main === module) {   // si no es llamado por otro módulo, es decir, si es el módulo principal -> levantamos el servidor
  inicializarBase().then(() => {
   app.listen(port, () => {
    console.log(`sitio escuchando en el puerto ${port}`);
   });
  });
}
```

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

Observe:

- module.parent: es una variable que hace referencia al módulo que invoco al módulo actual, en este caso cuando se ejecuta el test desde el archivo: test/pruebainicial.test.js, esta variable referencia al test, por lo tanto el servidor web no se inicia, pero si se ejecuta el test desde el navegador, esta variable queda indefinida, 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.
- si por alguna razon no puede accederse a la instalación global de jest, intenten ejecutando: npx jest test/pruebainicial.test.js

Ejercicios:

- En el caso de los test "GET 404" y "`_isalive`", que no está implementada en la aplicación, le proponemos implementar y volver a verificarla.

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

```
const request = require("supertest");
const app = require("../index");
```

```
describe("GET /api/categorias", function () {
  it("Devolveria todos los categorias", async function () {
    const res = await request(app)
      .get("/api/categorias")
      .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({
          IdCategoria: expect.any(Number),
          Nombre: expect.any(String),
        }),
      ])
    );
  });
});


describe("GET /api/categorias/:id", function () {
  it("respond with json containing a single categorias", async function () {
    const res = await request(app)
      .get("/api/categorias/1");
    expect(res.statusCode).toEqual(200);
    expect(res.body).toEqual(
      expect.objectContaining({
        IdCategoria: 1,
        Nombre: expect.any(String),
      })
    );
  });
});
```

Observe:

- solo se testean los métodos GET;
- el primero testea la webapi de categorías y verifica que la respuesta sea un array con objetos que contengan los atributos IdCategoria y Nombre.
- El segundo testea la webapi de categorias/:id y verifica que la respuesta sea un objeto que contenga los atributos IdCategoria = 1 y Nombre sea un texto. Este endpoint estaba propuesto en un ejercicio opcional, sino lo desarrolló le dará error.

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

```
jest test/categorias.test.js
```

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

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",
  IdCategoria: 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",
  IdCategoria: 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),
        IdCategoria: 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(201);
    expect(res.body).toEqual(
      expect.objectContaining({
        IdArticulo: expect.any(Number),
        Nombre: expect.any(String),
        Precio: expect.any(Number),
        CodigoDeBarra: expect.any(String),
        IdCategoria: 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(200);
  });
});

// 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 usuarioJefe = { usuario: "admin", clave: "123" };
const usuarioEmpleado = { 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(401);
    expect(res.body.message).toEqual("Usuario o clave incorrectos");
  });

  it("Devolvería el token para usuario admin", async function () {
    const res = await request(app).post("/api/login").send(usuarioJefe);

    expect(res.statusCode).toEqual(200);
    expect(res.body.accessToken).toEqual(expect.any(String));
  });
});

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

  it("Devolveria error, porque falta token de autorización", async function
() {
    const res = await request(app).get("/api/usuarios");
    expect(res.statusCode).toEqual(401);
    expect(res.body.message).toEqual("Acceso denegado");
  });
```

```
  it("Devolvería todos los usuarios, solo autorizado para administradores",
async function () {
    const res1 = await request(app)
    .post("/api/login")
    .set("Content-type", "application/json")
    .send(usuarioJefe);
    expect(res1.statusCode).toEqual(200);
    let token = res1.body.accessToken;

    const res = await request(app)
      .get("/api/usuarios")
      .set("Authorization", `Bearer ${token}`);

    expect(res.statusCode).toEqual(200);
    expect(res.body).toEqual(
      expect.arrayContaining([
        expect.objectContaining({
          IdUsuario: expect.any(Number),
          Usuario: expect.any(String),
          Clave: expect.any(String),
          Rol: expect.any(String),
        }),
      ])
    );
  });

  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(usuarioEmpleado);
    expect(res1.statusCode).toEqual(200);
    let token = res1.body.accessToken;

    const res = await request(app)
      .get("/api/usuarios")
      .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 todos los test del proyecto con el comando:

```
npm run test
```