Creación de pruebas Unitarias utilizando Jest y Supertest para el backend de una API.

Primero debemos de instalar las bibliotecas que vamos a necesitar para el test.
npm install --save-dev jest supertest

Configurar el jest en el package.json

```
"scripts": {
  "test": "jest"
},
```

Crear un archivo para el marco de pruebas con el nombre **jest.setup.js**

- Donde configuramos el entorno de pruebas que es node.
- Configuración de archivo de prueba donde solicitamos que se ejecute el jest.setup.js que realiza una conexión a una base de datos de prueba.

```
module.exports = {
    testEnvironment: 'node',
    setupFilesAfterEnv: ['./jest.setup.js']
  };
```

Configuración de archivo de conexión a base de datos **jest.setup.js**

```
const mongoose = require('mongoose');

beforeAll(async () => {
    const url = 'mongodb://localhost:27017/testUnit';
    mongoose.connect(url)
    .then(() => console.log('MongoDB pruebas conectado'))
    .catch(err => console.log(err));
});

afterAll(async () => {
    if (mongoose.connection.readyState === 1) {
        await mongoose.connection.close();
        console.log('Conexión a MongoDB cerrada');
        await mongoose.connection.db.dropDatabase();

    }
});
```

Luego creamos nuestro archivo de pruebas, **productos.test.js**
En este archivo notara el uso de it en ves de usar test, enrrealidad es lo mismo simplemente que al usar it damos a entender que tenemos en ejecución múltiples pruebas unitarias anidadas.

```
const request = require('supertest');
const express = require('express');
const mongoose = require('mongoose');
const rutas = require('./rutas/productos');
const Producto = require('./modelos/productos');

const app = express();
app.use(express.json());
app.use('/productos', rutas);

describe('API Productos', () => {

  beforeEach(async () => {
    await Producto.deleteMany();
  });

  afterAll(async () => {
    await mongoose.connection.close();
  });

  it('Agregar producto', async () => {
    const nuevoProducto = {
      producto: 'Producto de prueba',
      categoria: 'Categoría de prueba',
      existencia: 10,
      precio: 99.99
    };

    const response = await request(app)
      .post('/productos')
      .send(nuevoProducto);

    expect(response.status).toBe(201);
    expect(response.body.producto).toBe(nuevoProducto.producto);
  });
```

```
  it('Obtener productos', async () => {
    const producto1 = new Producto({ producto: 'Producto 1', categoria:
'Categoría 1', existencia: 10, precio: 50 });
    const producto2 = new Producto({ producto: 'Producto 2', categoria:
'Categoría 2', existencia: 5, precio: 25 });
    await producto1.save();
    await producto2.save();

    const response = await request(app).get('/productos');

    expect(response.status).toBe(200);
    expect(response.body.length).toBe(2);
  });

  it('Obtener productos por id', async () => {
    const producto = new Producto({ producto: 'Producto 1', categoria:
'Categoría 1', existencia: 10, precio: 50 });
    await producto.save();

    const response = await request(app).get(`/productos/${producto._id}`);

    expect(response.status).toBe(200);
    expect(response.body.producto).toBe(producto.producto);
  });

  it('Actualizar productos', async () => {
    const producto = new Producto({ producto: 'Producto 1', categoria:
'Categoría 1', existencia: 10, precio: 50 });
    await producto.save();

    const updates = { precio: 75 };
    const response = await request(app)
      .put(`/productos/${producto._id}`)
      .send(updates);

    expect(response.status).toBe(200);
    expect(response.body.precio).toBe(75);
  });
```

```
  it('Eliminar productos', async () => {
    const producto = new Producto({ producto: 'Producto 1', categoria:
'Categoría 1', existencia: 10, precio: 50 });
    await producto.save();

    const response = await
request(app).delete(`/productos/${producto._id}`);

    expect(response.status).toBe(200);
    expect(response.body.message).toBe('Producto eliminado');
  });
});
```

Ejecute el archivo de pruebas **npm test.**