#### Practica 1

# 1. Elige el lenguaje de programación que más te guste

**Javascript**, ya que es el lenguaje que al día de hoy mejor manejo y cuento con los conocimientos necesarios para desarrollar una API bajo el tiempo estipulado.

2. Define mensajes de entrada y salida para un API de tipo REST, en formato XML o JSON. Decide cuál es el formato y la estructura de los mensajes. Tu define la lógica interna del servicio.

La API recibirá un objeto JSON con la siguiente estructura:

Si se manda una petición GET al endpoint principal, regresará todos los artículos.

Y si se manda una petición POST al endpoint, de create, permitirá registrar un nuevo artículo.

3. Crea un endpoint en la ruta: /api/sps/helloworld/v1 que reciba y devuelva los mensajes del punto anterior

```
PS F:\repositorios\sps> npm init
This utility will walk you through creating a package.json file.

It only covers the most common items, and tries to guess sensible defaults.

See 'npm help json' for definitive documentation on these fields and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (sps)
version: (1.0.0)
description: Practica #1 para entrar a como becario a la consultora SPS entry point: (index.js)
test command:
git repository:
keywords: API REST javascript
author: Eduardo Rodriguez Ricardez
license: (ISC)
About to write to F:\repositorios\sps\package.json:

{
    "name": "sps",
    "version": "1.0.0",
    "description": "Practica #1 para entrar a como becario a la consultora SPS",
    "main": "index.js",
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1"
        },
        "keywords": [
            "API",
            "REST",
            "javascript"
        ],
        "author": "Eduardo Rodriguez Ricardez",
        "license": "ISC"
}
```

## 4. Despliega el API REST en tu máquina.

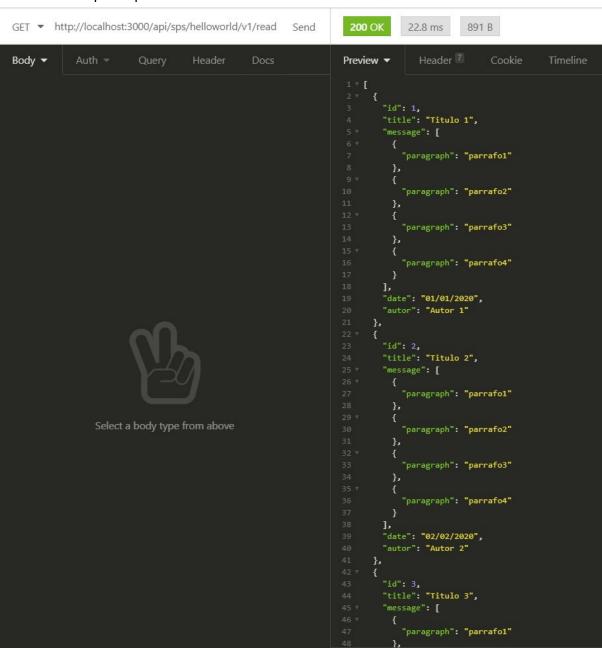
```
PS F:\repositorios\sps\practica1> npm run dev

> sps-practica1@1.0.0 dev F:\repositorios\sps\practica1
> nodemon index

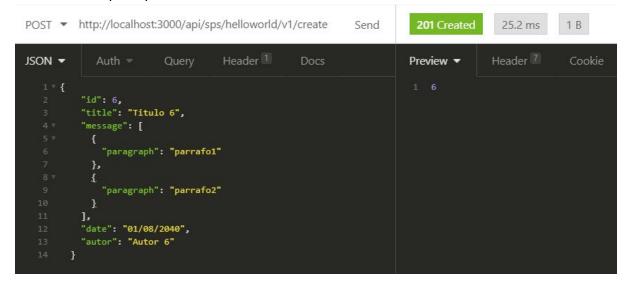
[nodemon] 2.0.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
Escuchando en 3000
```

# 5. Prueba el api con el cliente REST que más te guste.

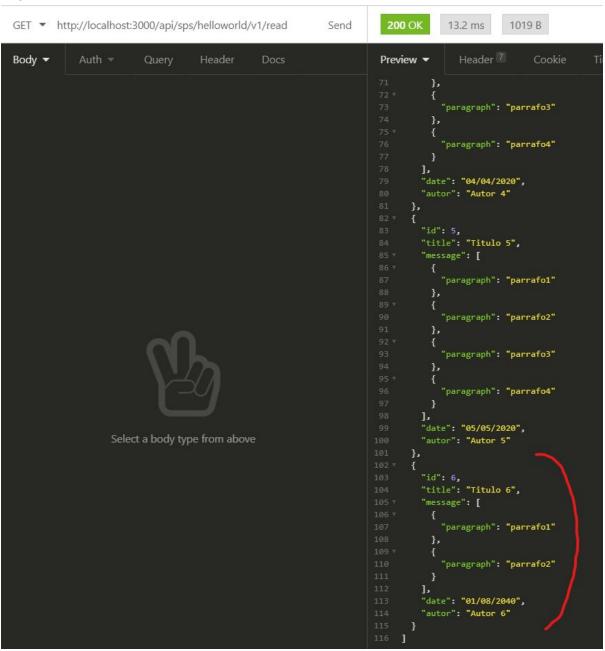
Petición al endpoint que contiene todos los artículos



#### Petición al endpoint para crear un artículo

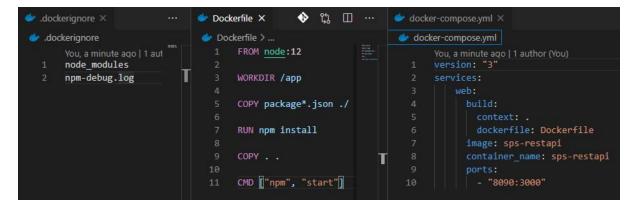


Segunda petición al endpoint que trae todos los artículos, después de crear uno.



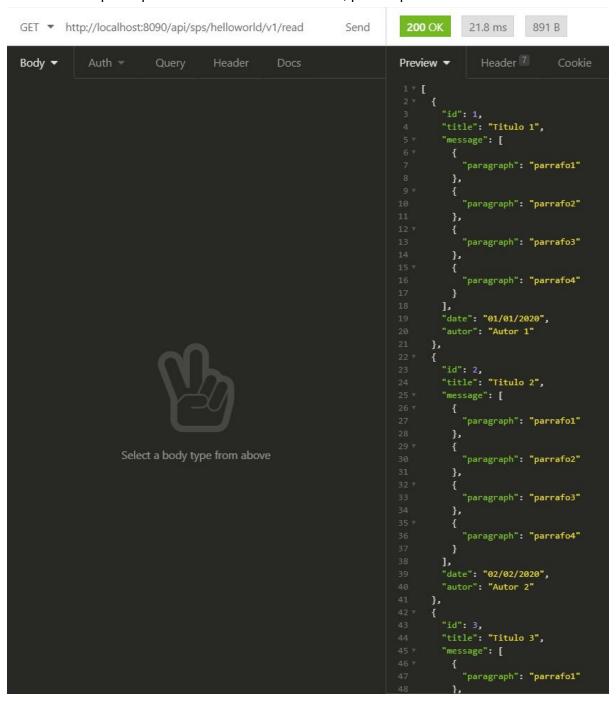
# 6. Conteneriza la aplicación y levanta el contenedor. Expón tu servicio por el puerto 8090

Con estos 3 archivos configuramos nuestro contenedor. Claro, previamente se debió instalar Docker en el equipo correspondiente.



### 7. Prueba el API, ahora corriendo en el contenedor

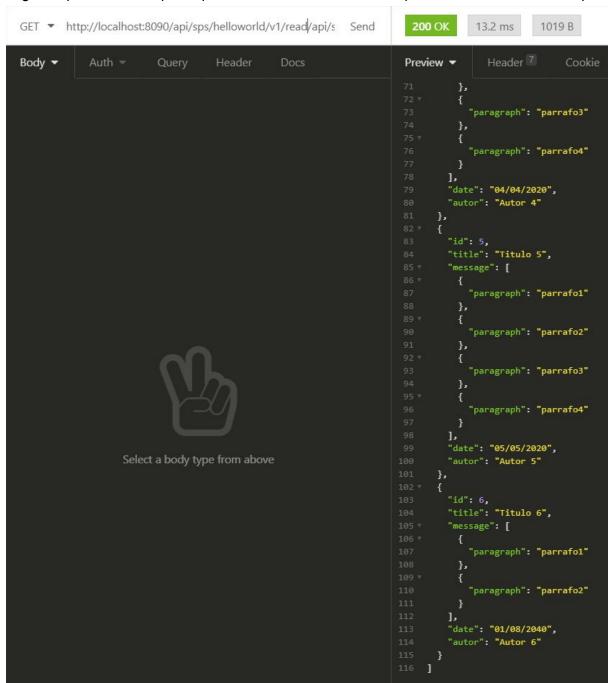
Petición al endpoint que contiene todos los artículos, pero al puerto 8090



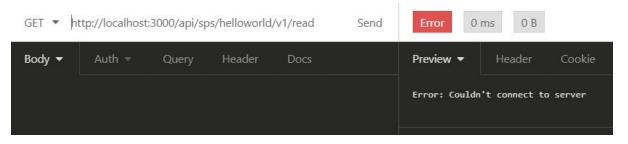
Petición al endpoint para crear un artículo, pero al puerto 8090



Segunda petición al endpoint que trae todos los artículos, después de crear uno. Pero al puerto 8090



Si intento alguna petición al puerto anterior 3000, me da un error. Al no encontrar el servidor



8. Sube todo el código a un repositorio público basado en Git, el que más te guste. Sube tu documentación también.

Después de clonar mi repositorio, solo es necesario correr estos comandos para hacer uso de mi API docker-compose up -d // Levantar servicio docker-compose ps // Visualizar los servicios compose que se están ejecutando docker-compose down // Detener el servicio