

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:

```
{
  id: 1,
  title: "Titulo 1",
  message: [
    {paragraph: "parrafo1"},
    {paragraph: "parrafo2"},
    {paragraph: "parrafo3"},
    {paragraph: "parrafo4"}
  ],
  date: "01/01/2020",
  autor: "Autor 1"
}
```

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

GET http://localhost:3000/api/sps/helloworld/v1/read Send

200 OK22.8 ms891 B

BodyAuthQueryHeaderDocs



Select a body type from above

PreviewHeader7CookieTimeline

```
1 [
2   {
3     "id": 1,
4     "title": "Titulo 1",
5     "message": [
6       {
7         "paragraph": "parrafo1"
8       },
9       {
10        "paragraph": "parrafo2"
11      },
12      {
13        "paragraph": "parrafo3"
14      },
15      {
16        "paragraph": "parrafo4"
17      }
18    ],
19    "date": "01/01/2020",
20    "autor": "Autor 1"
21  },
22  {
23    "id": 2,
24    "title": "Titulo 2",
25    "message": [
26      {
27        "paragraph": "parrafo1"
28      },
29      {
30        "paragraph": "parrafo2"
31      },
32      {
33        "paragraph": "parrafo3"
34      },
35      {
36        "paragraph": "parrafo4"
37      }
38    ],
39    "date": "02/02/2020",
40    "autor": "Autor 2"
41  },
42  {
43    "id": 3,
44    "title": "Titulo 3",
45    "message": [
46      {
47        "paragraph": "parrafo1"
48      },
```

Petición al endpoint para crear un artículo

POST

http://localhost:3000/api/sps/helloworld/v1/create

Send

201 Created

25.2 ms

1 B

JSON

Auth

Query

Header1

Docs

Preview

Header7

Cookie

1

{

2

"id": 6,

3

"title": "Titulo 6",

4

"message": [

5

{

6

"paragraph": "parrafo1"

7

},

8

{

9

"paragraph": "parrafo2"

10

}

11

],

12

"date": "01/08/2040",

13

"autor": "Autor 6"

14

}

1

6

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

GET

http://localhost:3000/api/sps/helloworld/v1/read

Send

200 OK

13.2 ms

1019 B

Body

Auth

Query

Header


Docs

Preview

Header7

Cookie

Ti



Select a body type from above

71

},

72

{

73

"paragraph": "parrafo3"

74

},

75

{

76

"paragraph": "parrafo4"

77

}

78

],

79

"date": "04/04/2020",

80

"autor": "Autor 4"

81

},

82

{

83

"id": 5,

84

"title": "Titulo 5",

85

"message": [

86

{

87

"paragraph": "parrafo1"

88

},

89

{

90

"paragraph": "parrafo2"

91

},

92

{

93

"paragraph": "parrafo3"

94

},

95

{

96

"paragraph": "parrafo4"

97

}

98

],

99

"date": "05/05/2020",

100

"autor": "Autor 5"

101

},

102

{

103

"id": 6,

104

"title": "Titulo 6",

105

"message": [

106

{

107

"paragraph": "parrafo1"

108

},

109

{

110

"paragraph": "parrafo2"

111

}

112

],

113

"date": "01/08/2040",

114

"autor": "Autor 6"

115

}

116

]

The screenshot shows a VS Code editor with three files open in a dark theme. The first file, `.dockerignore`, contains two lines: `node_modules` and `npm-debug.log`. The second file, `Dockerfile`, contains 11 lines of Docker instructions: `FROM node:12`, `WORKDIR /app`, `COPY package*.json ./`, `RUN npm install`, `COPY . .`, and `CMD ["npm", "start"]`. The third file, `docker-compose.yml`, contains 10 lines: `version: "3"`, `services:`, `web:`, `build:`, `context: .`, `dockerfile: Dockerfile`, `image: sps-restapi`, `container_name: sps-restapi`, `ports:`, and `- "8090:3000"`.

```

.dockerignore X
...
.dockerignore
You, a minute ago | 1 author
1 node_modules
2 npm-debug.log

Dockerfile X
Dockerfile > ...
1 FROM node:12
2
3 WORKDIR /app
4
5 COPY package*.json ./
6
7 RUN npm install
8
9 COPY . .
10
11 CMD ["npm", "start"]

docker-compose.yml X
docker-compose.yml
You, a minute ago | 1 author (You)
1 version: "3"
2 services:
3   web:
4     build:
5       context: .
6       dockerfile: Dockerfile
7     image: sps-restapi
8     container_name: sps-restapi
9     ports:
10      - "8090:3000"

```

7. Prueba el API, ahora corriendo en el contenedor

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

GET

▼

http://localhost:8090/api/sps/helloworld/v1/read

Send

200 OK

21.8 ms

891 B

Body

▼

Auth

▼

Query

Header

Docs

Preview

▼

Header

7

Cookie

1

▼

[

2

▼

{

3 " id": 1,

4 " title": "Titulo 1",

5

▼

 " message": [

6

▼

 {

7 " paragraph": "parrafo1"

8 },

9

▼

 {

10 " paragraph": "parrafo2"

11 },

12

▼

 {

13 " paragraph": "parrafo3"

14 },

15

▼

 {

16 " paragraph": "parrafo4"

17 }

18],

19 " date": "01/01/2020",

20 " autor": "Autor 1"

21 },

22

▼

{

23 " id": 2,

24 " title": "Titulo 2",

25

▼

 " message": [

26

▼

 {

27 " paragraph": "parrafo1"

28 },

29

▼

 {

30 " paragraph": "parrafo2"

31 },

32

▼

 {

33 " paragraph": "parrafo3"

34 },

35

▼

 {

36 " paragraph": "parrafo4"

37 }

38],

39 " date": "02/02/2020",

40 " autor": "Autor 2"

41 },

42

▼

{

43 " id": 3,

44 " title": "Titulo 3",

45

▼

 " message": [

46

▼

 {

47 " paragraph": "parrafo1"

48 },

Select a body type from above

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

POST

http://localhost:8090/api/sps/helloworld/v1/create

Send

201 Created

17.4 ms

1 B

http://localhost:8090/api/sps/helloworld/v1/read

JSON

Auth

Query

Header

Docs

1

{

2

"id": 6,

3

"title": "Titulo 6",

4

"message": [

5

{

6

"paragraph": "parrafo1"

7

},

8

{

9

"paragraph": "parrafo2"

10

}

11

],

12

"date": "01/08/2040",

13

"autor": "Autor 6"

14

}

Preview

Header

Cook

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

GET

http://localhost:8090/api/sps/helloworld/v1/read/api/s

Send

200 OK

13.2 ms

1019 B

Body

Auth

Query

Header

Docs

Preview

Header

Cookie

71

},

72

{

73

"paragraph": "parrafo3"

74

},

75

{

76

"paragraph": "parrafo4"

77

}

78

],

79

"date": "04/04/2020",

80

"autor": "Autor 4"

81

},

82

{

83

"id": 5,

84

"title": "Titulo 5",

85

"message": [

86

{

87

"paragraph": "parrafo1"

88

},

89

{

90

"paragraph": "parrafo2"

91

},

92

{

93

"paragraph": "parrafo3"

94

},

95

{

96

"paragraph": "parrafo4"

97

}

98

],

99

"date": "05/05/2020",

100

"autor": "Autor 5"

101

},

102

{

103

"id": 6,

104

"title": "Titulo 6",

105

"message": [

106

{

107

"paragraph": "parrafo1"

108

},

109

{

110

"paragraph": "parrafo2"

111

}

112

],

113

"date": "01/08/2040",

114

"autor": "Autor 6"

115

}

116

]

Select a body type from above

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

GET

http://localhost:3000/api/sps/helloworld/v1/read

Send

Error

0 ms

0 B

Body

Auth

Query

Header

Docs

Preview

Header

Cookie

Error: Couldn't connect to server

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