

# Challenge

## Instalación:

- Clonar el repositorio
- Instalar dependencias a través de consola con el comando -> npm i

## Instrucciones de uso:

- Para iniciar el servidor, comando -> npm start
- Una vez iniciado el servidor accedemos a una petición mediante una url similar a esta -> `http://localhost:3000/api/roma`

Es decir `http://localhost:3000/api/` + [ciudad a buscar]

Nos entregará la temperatura actual consultando a una api externa, podemos chequerarlo en el texto recibido: "Source: API weather". Si Consultamos nuevamente la misma ciudad, esta será traída de nuestra base de datos mongoDB, por lo que veremos "Source: DB"

En caso que la petición a la API externa falle, el código hará dos intentos más y en caso de una respuesta negativa, devolverá un error explicando el motivo.

The screenshot shows a REST client interface with a dark theme. The top bar displays the method 'GET', the URL 'http://localhost:3000/api/roma', and a 'Send' button. Below the top bar, there are tabs for 'Query', 'Headers', 'Auth', 'Body', and 'Tests'. The 'Query' tab is selected, showing 'Query Parameters' with a table with two columns: 'parameter' and 'value'. The right panel shows the 'Response' tab selected, displaying the status '200 OK', size '63 Bytes', and time '1.22 s'. The response body is '1 The temperature consulted is 11.75 degrees. Source: API weather'. Two red arrows point to the words 'API' and 'weather' in the response body.

parameter	value
-----------	-------

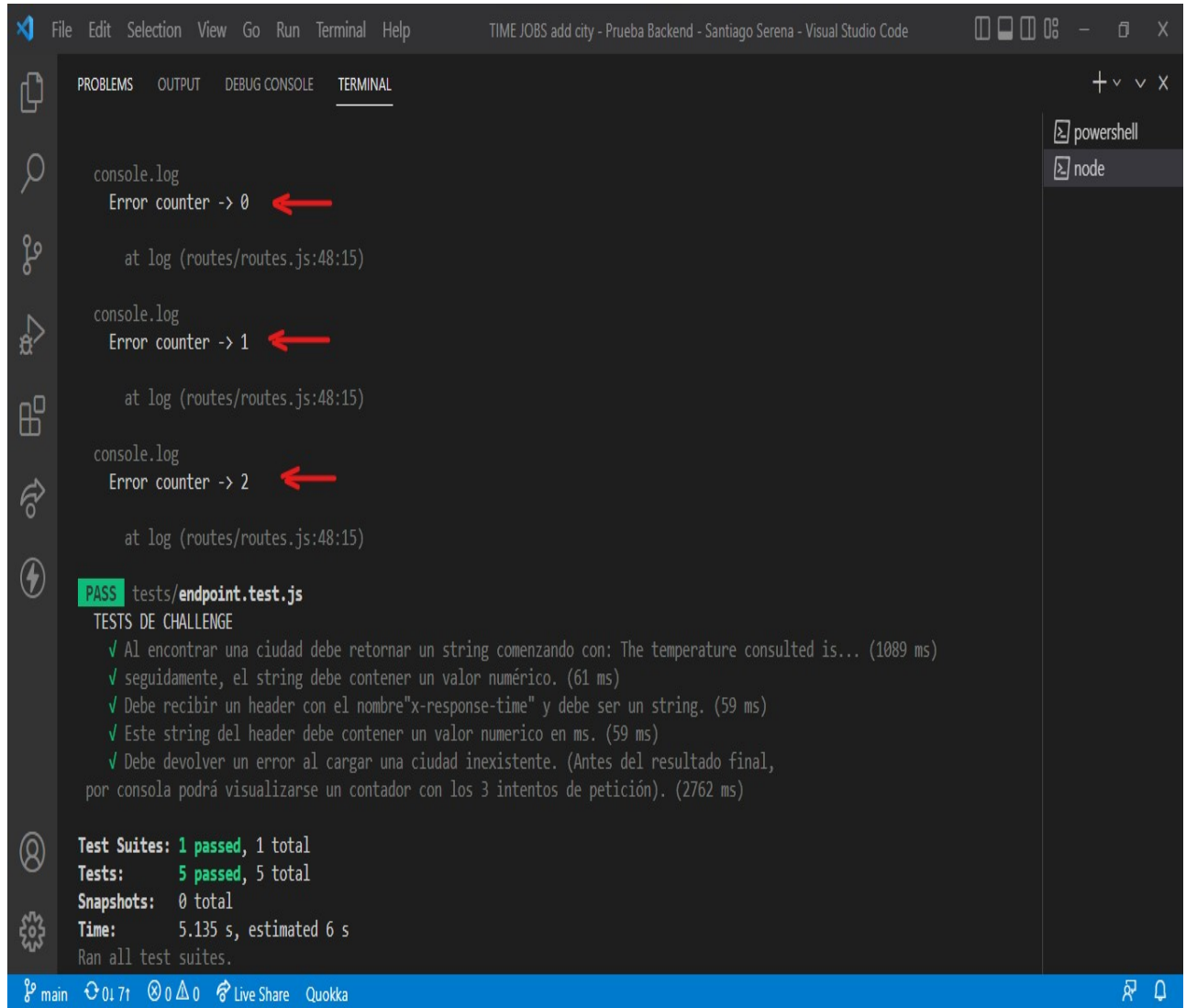
Status: 200 OK Size: 63 Bytes Time: 1.22 s

Response Headers Cookies Results Docs {} ≡

1 The temperature consulted is 11.75 degrees. Source: API weather

## Tests:

- Sin el servidor activo, podemos correr los tests con el comando -> npm test
- tendremos un contador de errores para corroborar los 3 intentos, en caso de introducir una ciudad inexistente o un problema en la API externa (observar imagen).
- Los test a diferencia del resto del código fueron escritos en español para una mejor comprensión del usuario.



The screenshot shows the Visual Studio Code interface with the terminal open. The terminal output displays the results of running tests for 'tests/endpoint.test.js'. The output is as follows:

```
console.log
Error counter -> 0
    at log (routes/routes.js:48:15)

console.log
Error counter -> 1
    at log (routes/routes.js:48:15)

console.log
Error counter -> 2

PASS tests/endpoint.test.js
TESTS DE CHALLENGE
  ✓ Al encontrar una ciudad debe retornar un string comenzando con: The temperature consulted is... (1089 ms)
  ✓ seguidamente, el string debe contener un valor numérico. (61 ms)
  ✓ Debe recibir un header con el nombre "x-response-time" y debe ser un string. (59 ms)
  ✓ Este string del header debe contener un valor numerico en ms. (59 ms)
  ✓ Debe devolver un error al cargar una ciudad inexistente. (Antes del resultado final, por consola podrá visualizarse un contador con los 3 intentos de petición). (2762 ms)

Test Suites: 1 passed, 1 total
Tests: 5 passed, 5 total
Snapshots: 0 total
Time: 5.135 s, estimated 6 s
Ran all test suites.
```

## Header de tiempo de respuesta:

- Contamos con un header de respuesta que mide el tiempo en ms, denominado -> x-response-time (observar imagen).

GET

▼

http://localhost:3000/api/roma

Send

Query

Headers<sup>2</sup>

Auth

Body

Tests

Query Parameters

☐

parameter

value

Status: 200 OK   Size: 63 Bytes   Time: 1.22 s

Response

Headers<sup>7</sup>

Cookies

Results

Docs

{}

≡

Response Headers

Header	Value
x-powered-by	Express
content-type	text/html; charset=utf-8
content-length	63
etag	W/"3f-hhjOURz6HrXDsb+ff9XroePr8A"
x-response-time	1219.249ms
date	Fri, 07 Oct 2022 20:57:44 GMT
connection	close