

Microsoft Azure Devops

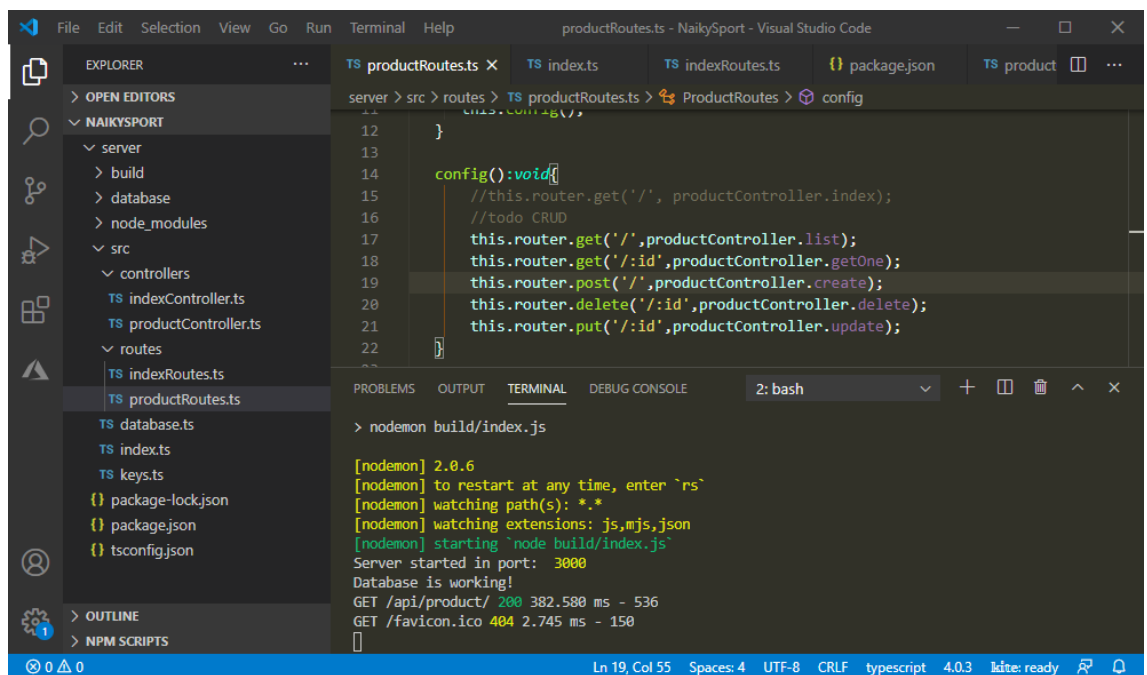
Lab: API Testing con Postman

Objetivo:

- Mostrar al participante el proceso de testing usando la herramienta de test Postman.

Procedimiento:

- Iniciar la API product desarrollado en node y typescript.



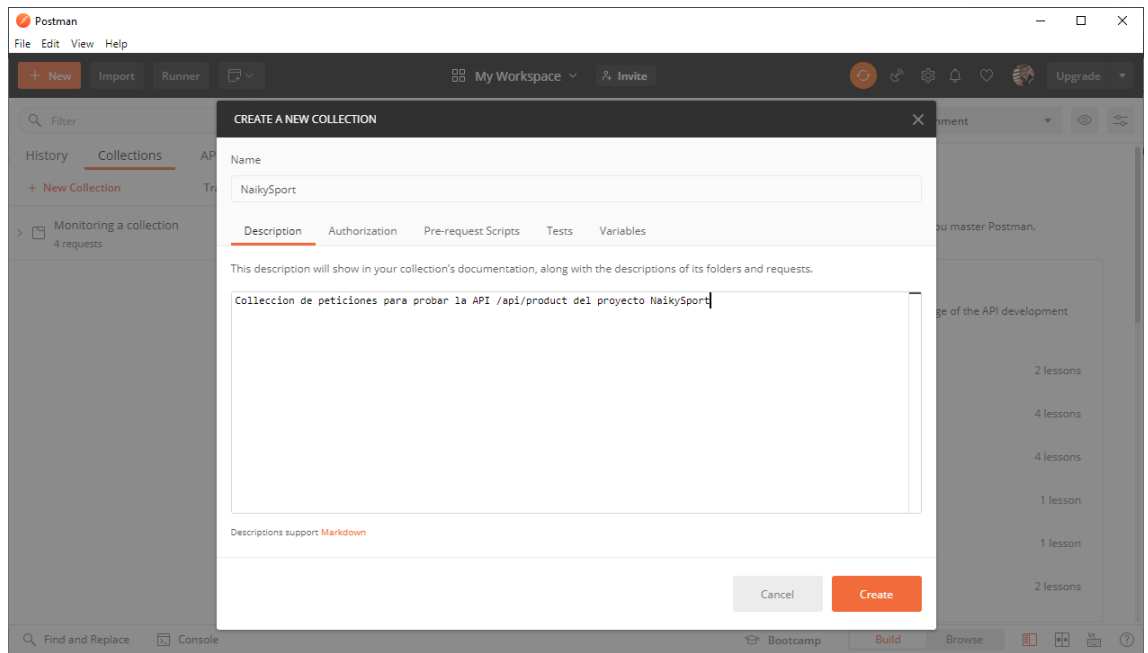
```
server > src > routes > TS productRoutes.ts > ProductRoutes > config
12  }
13
14  config():void{
15      //this.router.get('/', productController.index);
16      //todo CRUD
17      this.router.get('/',productController.list);
18      this.router.get('/:id',productController.getOne);
19      this.router.post('/',productController.create);
20      this.router.delete('/:id',productController.delete);
21      this.router.put('/:id',productController.update);
22  }
```

```
> nodemon build/index.js
[nodemon] 2.0.6
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node build/index.js`
Server started in port: 3000
Database is working!
GET /api/product/ 200 382.580 ms - 536
GET /favicon.ico 404 2.745 ms - 150
```

La API producto desarrollado en node y typescript, tiene las siguientes rutas.

- [GET] /api/product -> La lista de todos los productos de la tienda.
- [GET] /api/product/{id} -> Encuentra un producto por si identificador.
- [POST] /api/product -> Crear un nuevo producto en la tienda.
- [PUT] /api/product/{id} -> Actualizar la información de un producto por si identificador
- [DELETE] /api/product/{id} -> Eliminar un producto de la tienda

- Crea una colección en Postman llamada NaikySport.

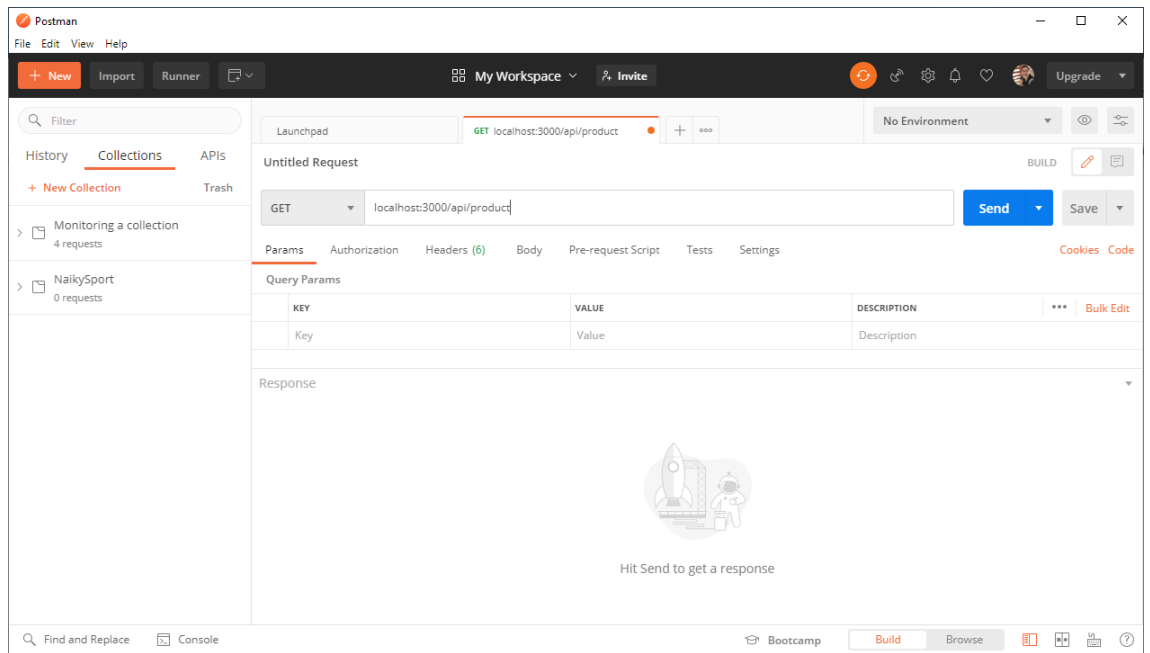


Nota. -

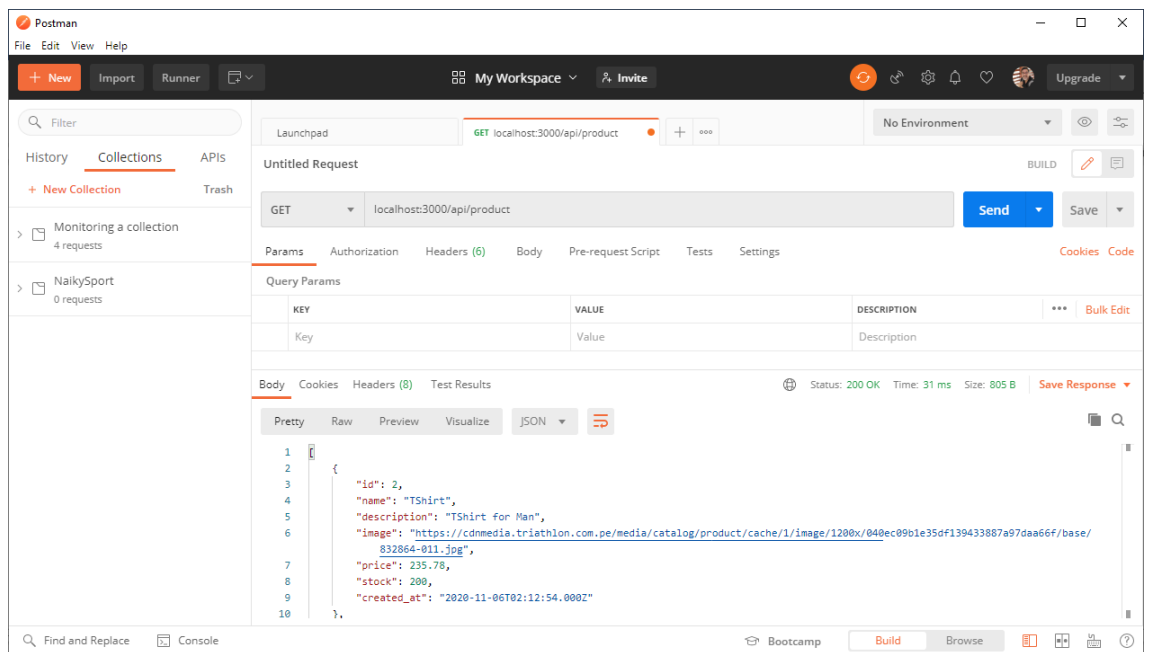
- Si nos fijamos en esta misma ventana hay dos aspectos interesantes relacionados con el testing. El primero, la pestaña de "Tests". Estos tests se ejecutarán cada vez que cualquier petición de esta colección se haya servido. En la pestaña "Variables" podremos crear variables que podremos reutilizar en todas nuestras peticiones, pero estas son variables que se definen para la colección actual, con lo que si queremos que las variables que utilicemos para nuestras peticiones cambien de valor dependiendo del entorno es mejor que configuremos entornos.

3. Añadir una petición a la colección.

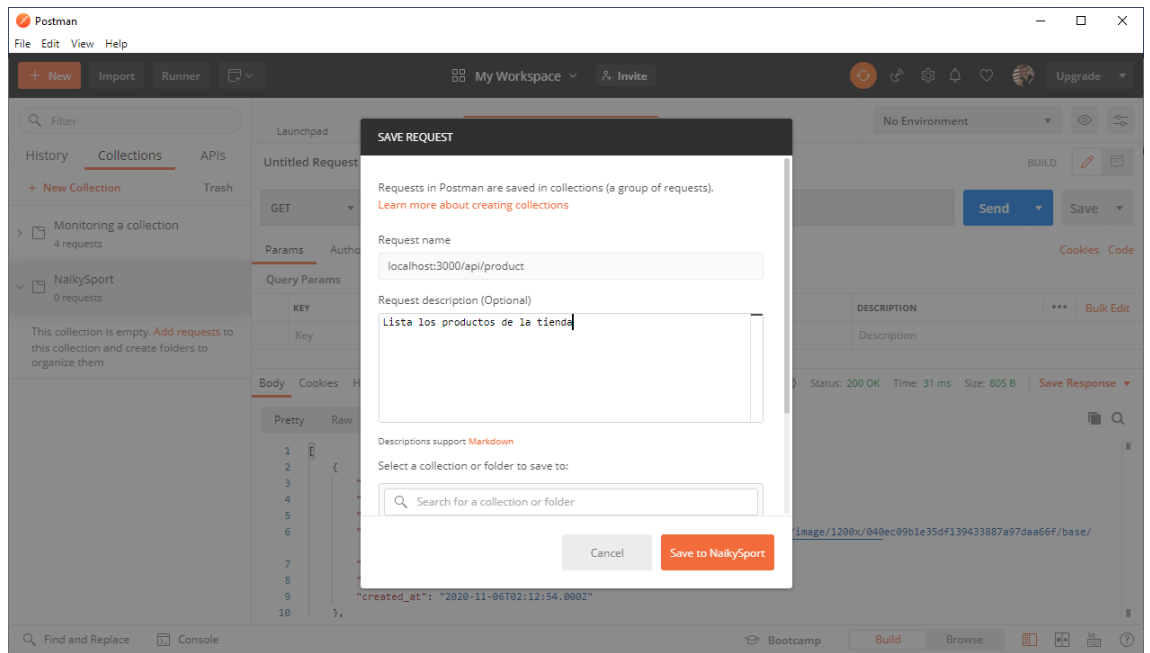
Una vez ya tenemos la colección creada, ya podemos añadir los endpoints que queremos implementar y probar. Nos vamos a centrar en una sola petición, GET **/api/product**, que nos permite obtener la lista de productos de nuestra tienda.



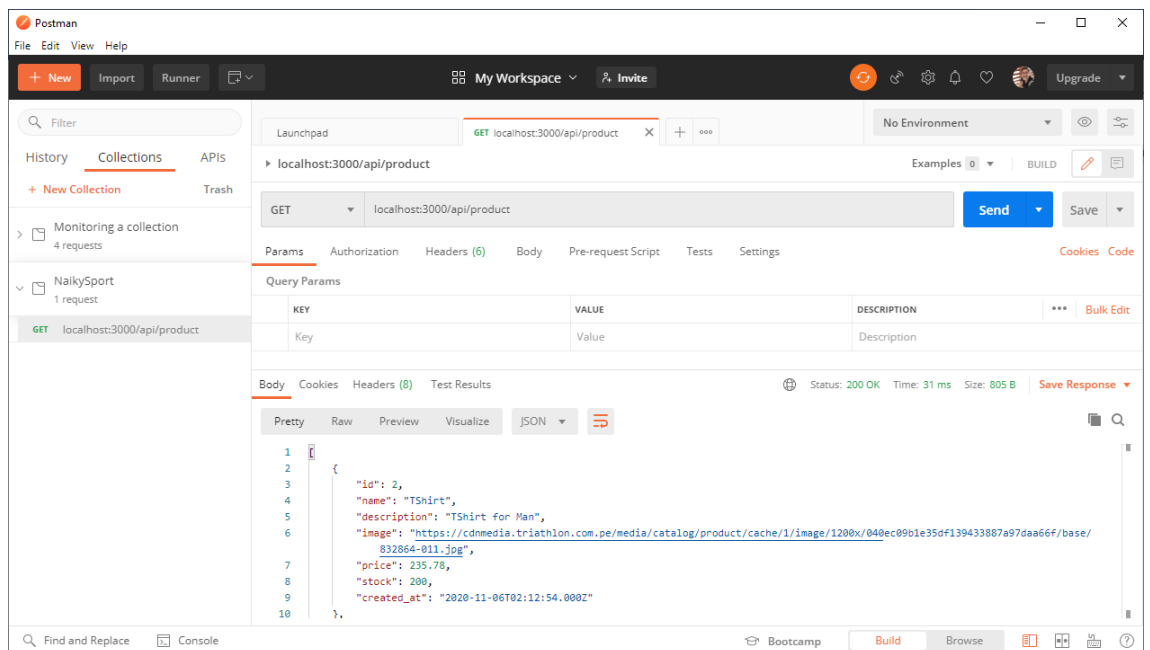
Clic en Send



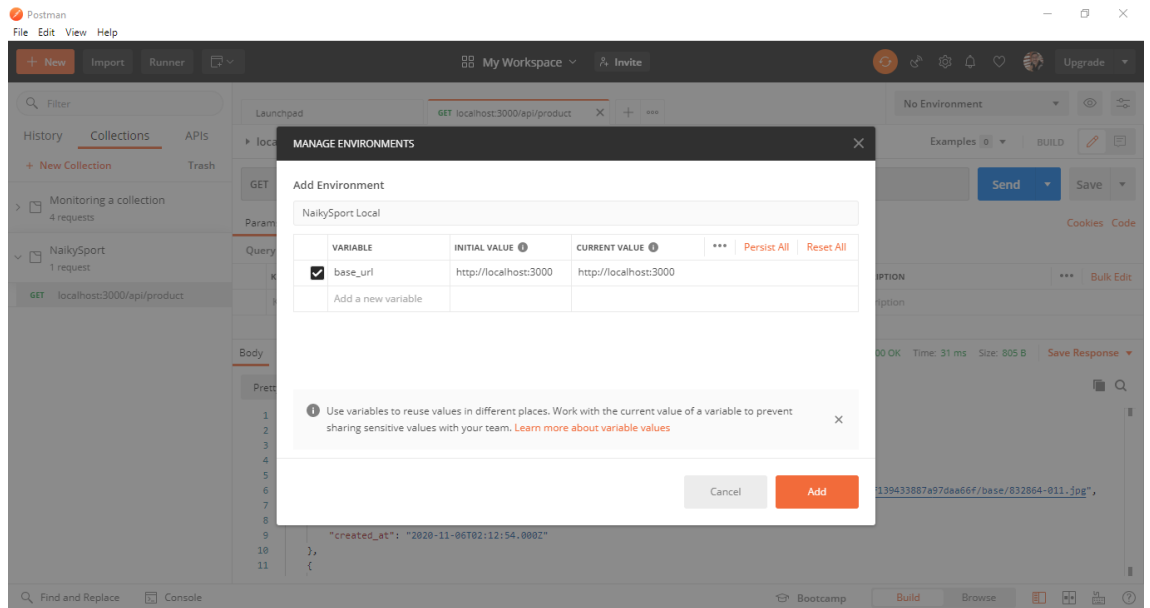
4. Guardar la petición en la colección NaikySport.



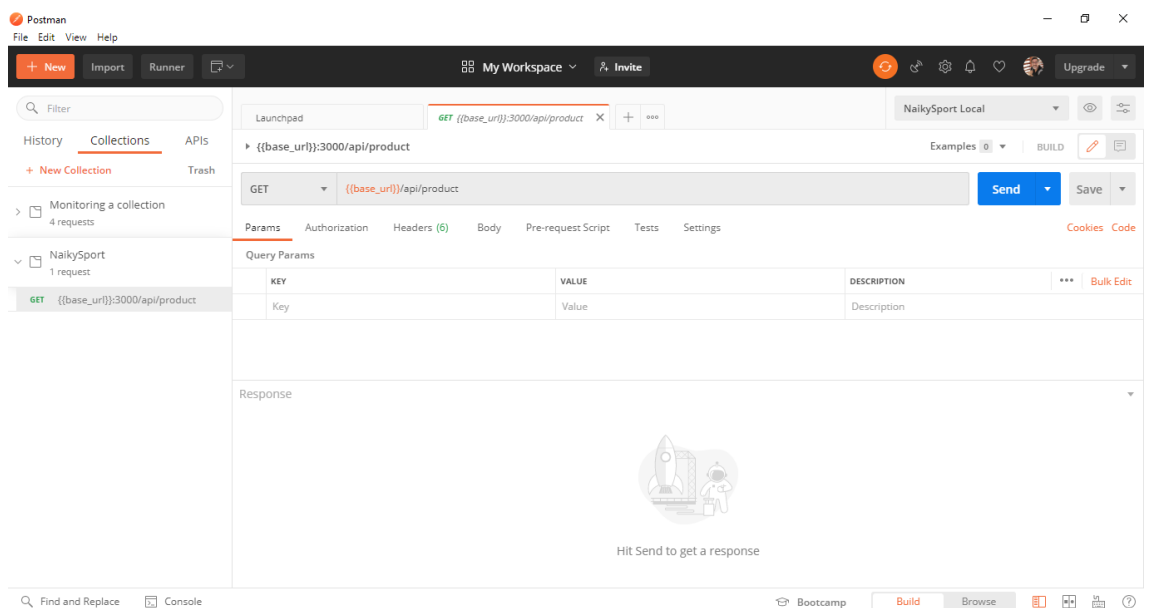
Haz clic en “Save to NaikySport”



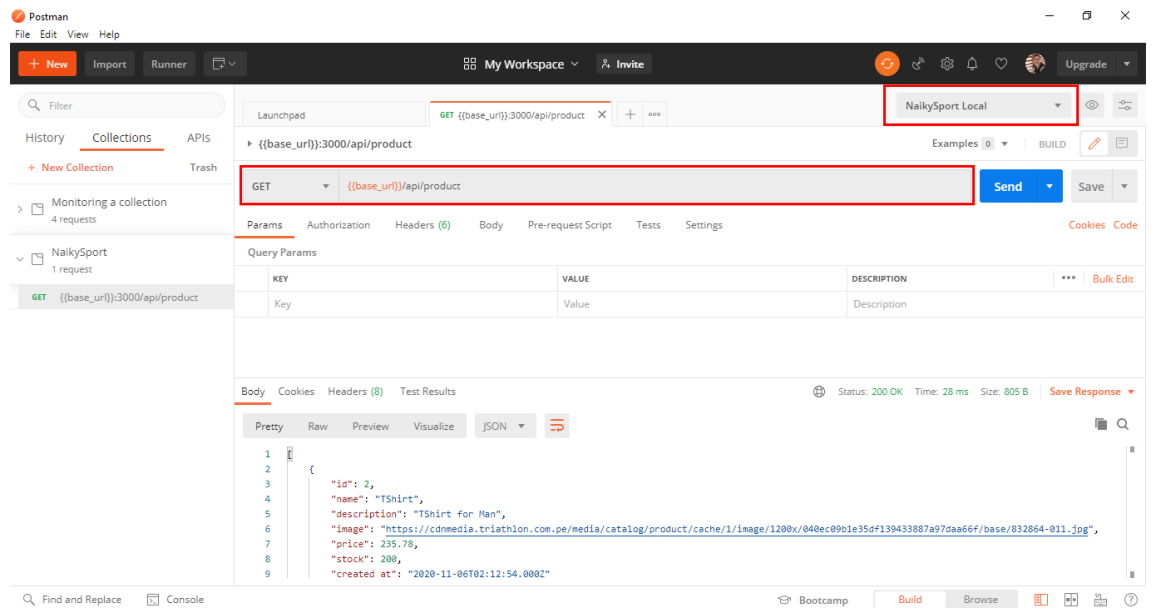
5. Creación de entornos (**environment**) para las pruebas. Con los “**Environments**” podemos crear configuraciones de entornos que usen las mismas variables pero que tengan valores diferentes dependiendo del entorno. Crearemos el entorno NaikySport Local con una variable **base_url**. Para crear los entornos, presionamos el icono de la rueda en la parte superior derecha y en la ventana de “**Manage environments**” presionamos “**Add**”.



Luego el Request GET en la colección NaikySport .



Haz clic en Send.



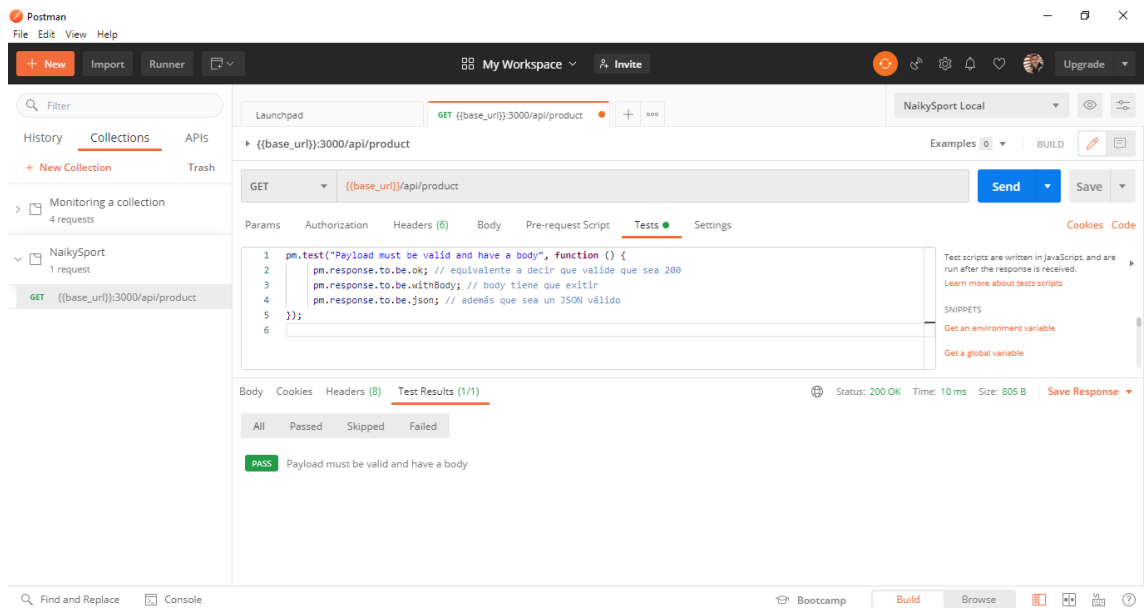
6. Asociar un test en js a la petición. Escribir un test tendrá básicamente el siguiente formato:

```
pm.test('nombre_del_test', function() {  
    // closure function que contendrá todos los asserts a ejecutar  
});
```

Vemos un ejemplo:

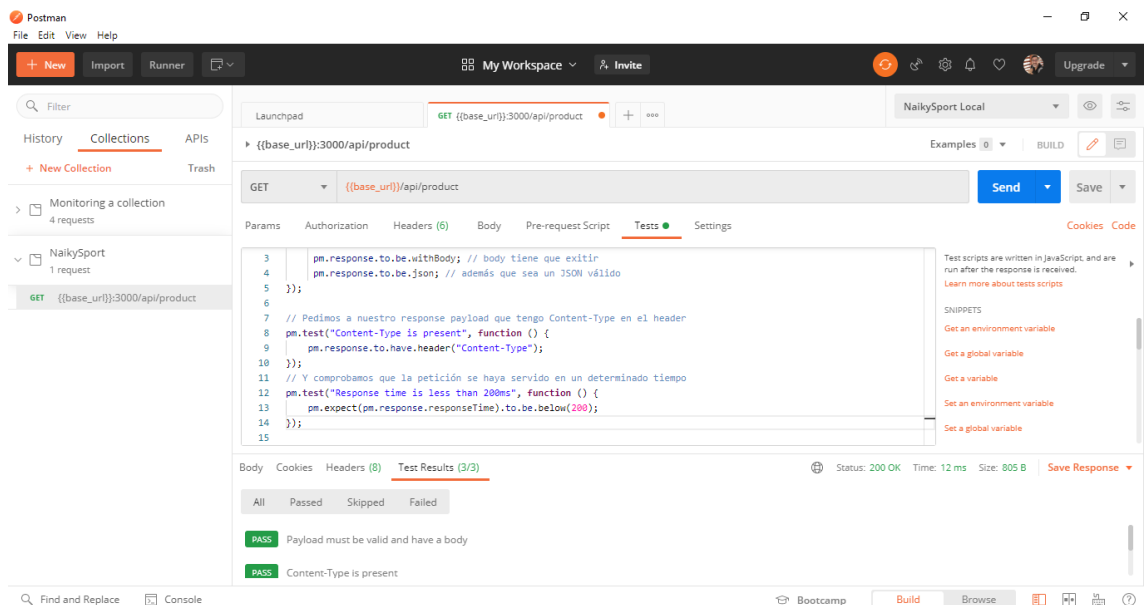
```
pm.test("Payload must be valid and have a body", function () {  
    pm.response.to.be.ok; // equivalente a decir que valide que sea 200  
    pm.response.to.be.withBody; // body tiene que existir  
    pm.response.to.be.json; // además que sea un JSON válido  
});
```

Los test se agregan en la pestaña “Tests” de la petición y el resultado se observa después de enviar la petición en “Tests Results”.



Añadimos más tests.

```
// Pedimos a nuestro response payload que tengo Content-Type en el header
pm.test("Content-Type is present", function () {
  pm.response.to.have.header("Content-Type");
});
// Y comprobamos que la petición se haya servido en un determinado tiempo
pm.test("Response time is less than 200ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(200);
});
```



- Validación de los datos retornados mediante un Schema. Postman integra la librería Tiny Validator que usa las especificaciones de JSON schema v4 para validar objetos JSON.

```

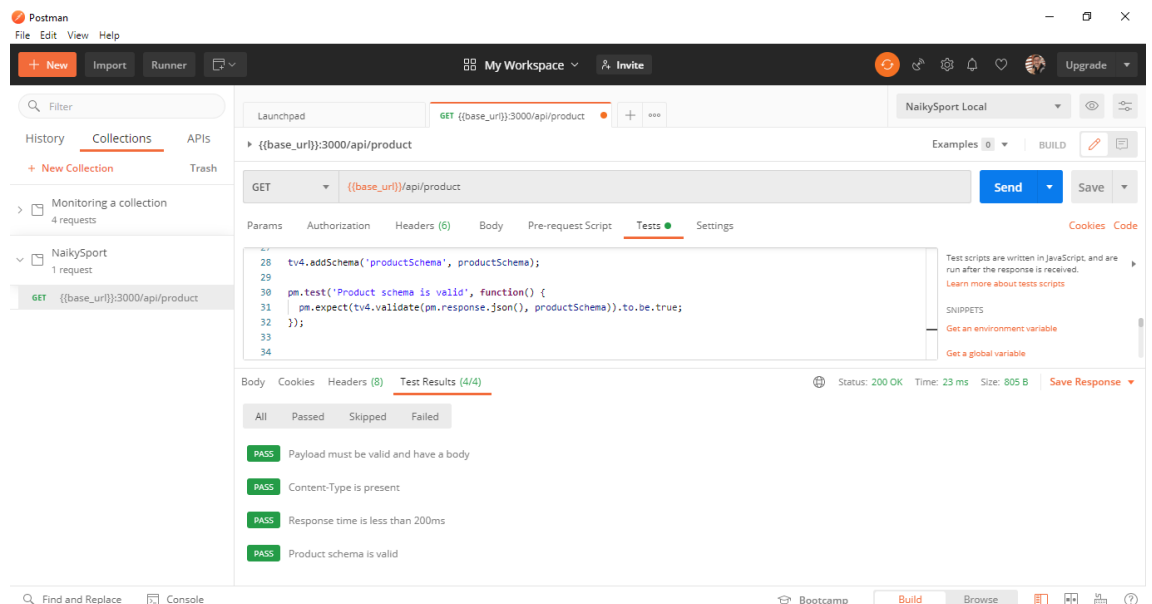
var productSchema = {
  "properties": {
    "id": { "type": "integer" },
    "name": { "type": "string" },
    "image": { "type": "string" },
    "price": { "type": "decimal" },
    "stock": { "type": "integer" },
    "created_at": { "type": "string" }
  }
};

tv4.addSchema('productSchema', productSchema);

pm.test('Product schema is valid', function() {
  pm.expect(tv4.validate(pm.response.json(), productSchema)).to.be.true;
});

```

Envía la petición y verifica la ejecución de los test.



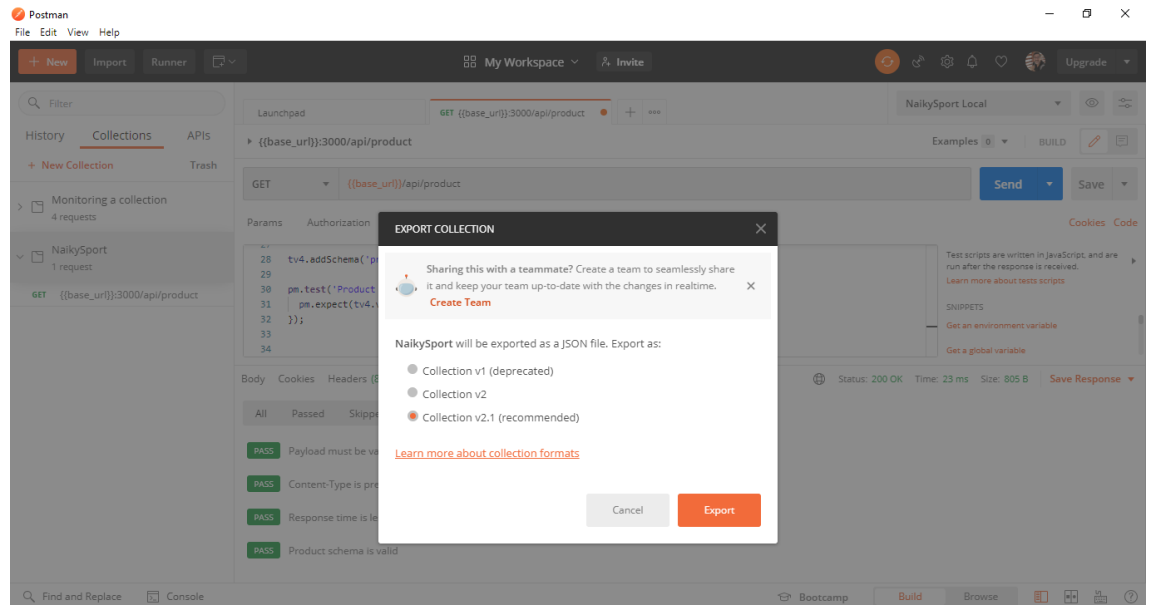
8. Ejecución de las pruebas por línea de comandos. En el entorno de Visual Studio Code instala Newman.

```

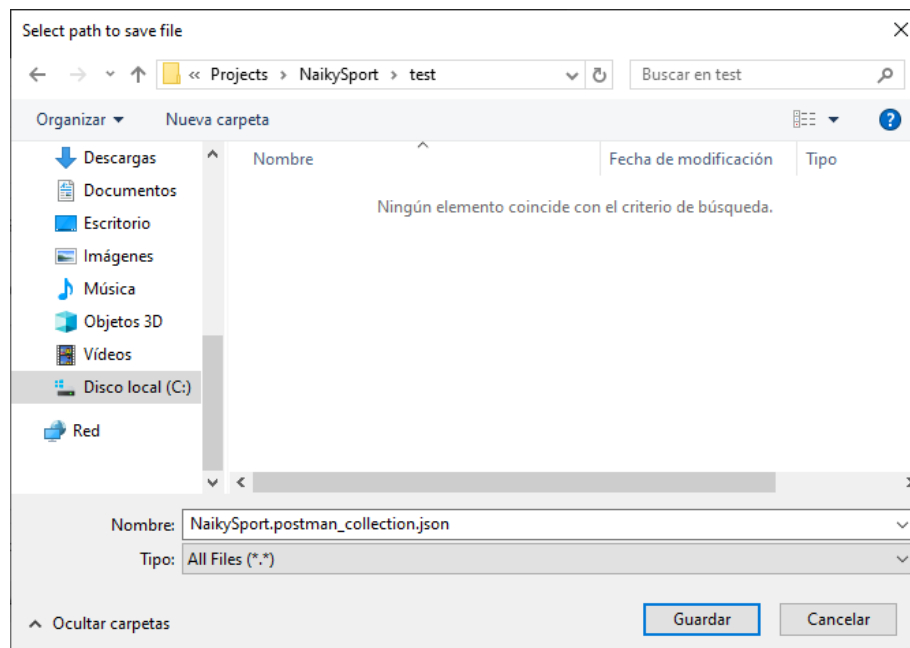
$ npm install -g newman
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
C:\Users\Daddy\AppData\Roaming\npm\newman ->
C:\Users\Daddy\AppData\Roaming\npm\node_modules\newman\bin\newman.js
+ newman@5.2.1
added 157 packages from 199 contributors in 118.868s

```

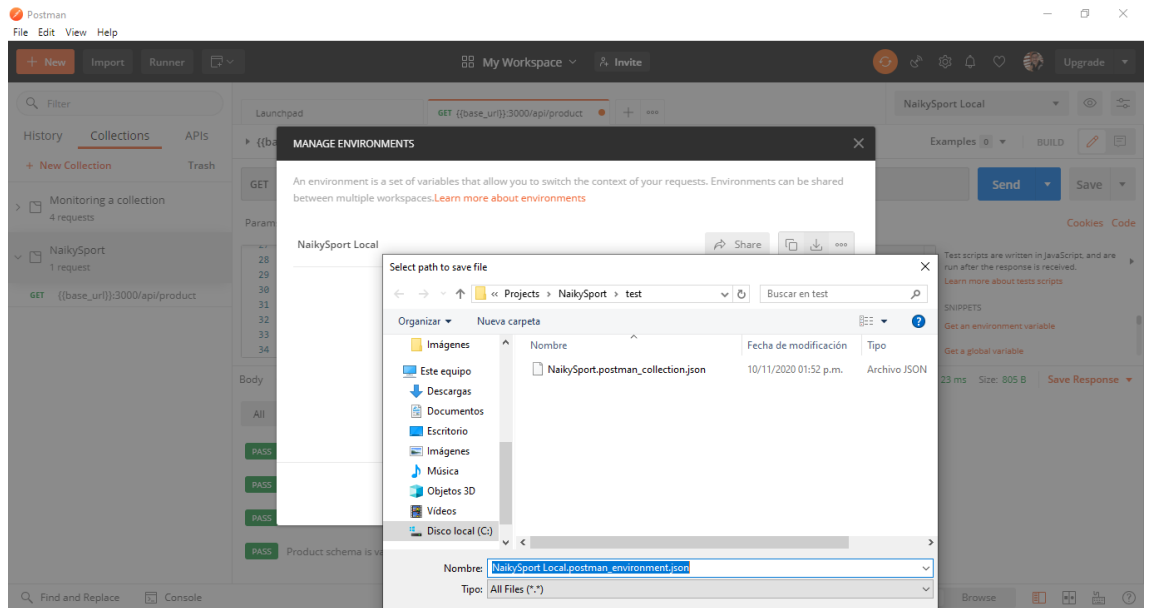

9. Exporta desde Postman la colección de pruebas que desear probar.



Exporta el archivo “NaikySport.postman_collection.json” en el directorio C:/Projects/NaikySport/test, si no existe créalo.



10. Descarga la configuración del entorno NaikySport Local, en el directorio C:/Projects/NaikySport/test



11. Ejecuta el comando desde la consola para ejecutar los test usando Newman.

```
Daddy@Daddy-PC MINGW64 /c/Projects/NaikySport/server
$ newman run -e "C://Projects//NaikySport//test//NaikySport
Local.postman_environment.json"
C://Projects//NaikySport//test//NaikySport.postman_collection.json
```

