Evidencias – Prueba Técnica Quality Control Engineer

Hugo Manuel Ruiz Hernández

Double V Partners NYX

Tabla de contenido

Precondiciones generales

Para todos los casos de prueba API se debe contar con:

1. **Environment en Postman** llamado FakeStoreAPI.
    - Variable: baseUrl
    - Valor: https://fakestoreapi.com
2. Colección **YourStore API – Products** creada en Postman.
3. Conexión a internet disponible para consumir los endpoints de la API.

Parte A – Casos Positivos

Test Cases 1: Consulta de productos que pertenezcan a la categoría de “electronics”.

Endpoint: GET {{baseUrl}}/products/category/electronics
Objetivo: Validar que el servicio API responda exitosamente con **Status 200 OK** y retorne solo productos cuya categoría sea *electronics*.

Precondición:

- Cumplir las precondiciones generales definidas al inicio del documento (environment FakeStoreAPI configurado).

Pasos ejecutados:

1. Abrir Postman.
2. Crear un nuevo request en la colección YourStore API - Products:
    a. Método: GET
    b. URL: {{baseUrl}}/products/category/electronics
3. Guardar el request en la colección (botón Save).
4. Hacer Send.
5. Verificar que la respuesta tenga status **200 OK** y que todos los productos devueltos pertenezcan a la categoría *electronics*.

Resultado esperado:

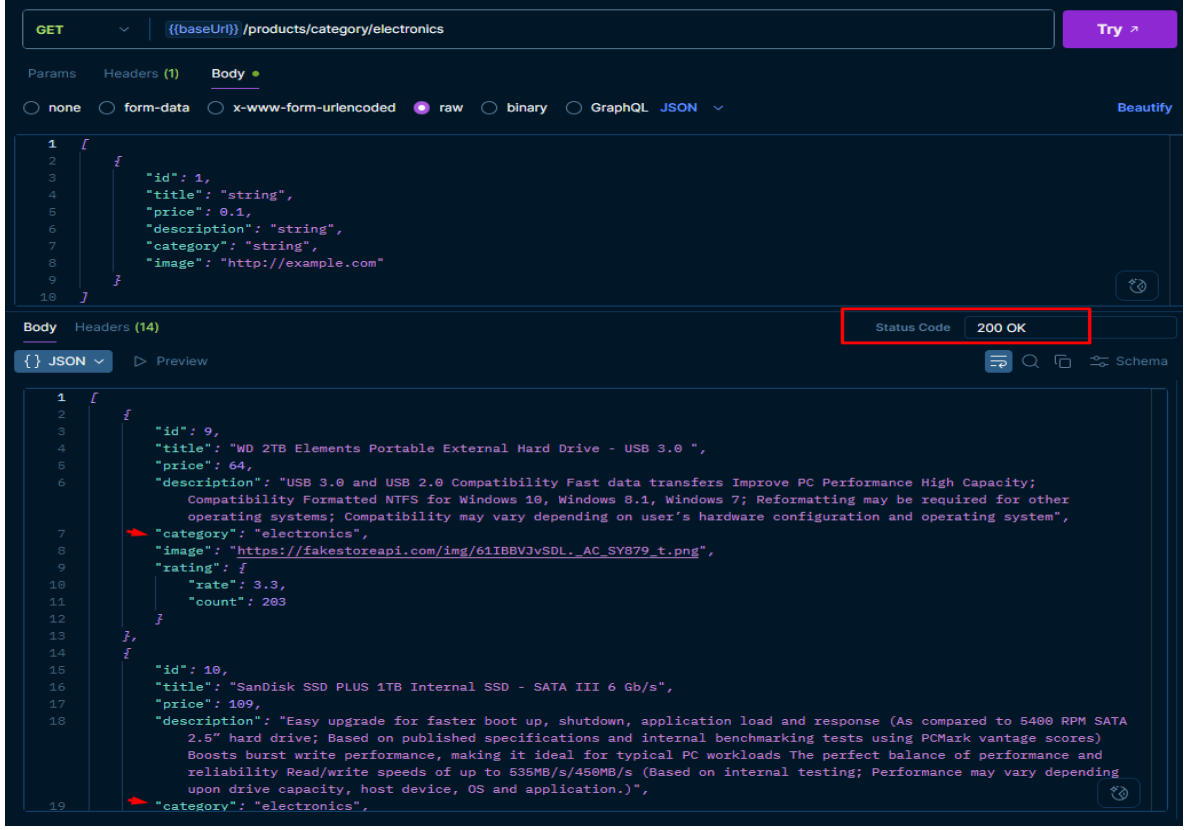
- Status 200 OK.
- Respuesta JSON en formato array de objetos con los campos: id, title, price, description, category, image, rating.
- Todos los productos devueltos con "category": "electronics".

Resultado obtenido:

- El servicio respondió con 200 OK.

- Se evidenció que todos los productos obtenidos pertenecen a la categoría electronics.
- Ejemplo de un producto recibido:


```
{
  "id": 9,
  "title": "WD 2TB Elements Portable External Hard Drive - USB 3.0 ",
  "price": 64,
  "description": "USB 3.0 and USB 2.0 Compatibility Fast data transfers Improve PC Performance High Capacity; Compatibility Formatted NTFS for Windows 10, Windows 8.1, Windows 7; Reformatting may be required for other operating systems; Compatibility may vary depending on user's hardware configuration and operating system",
  "category": "electronics",
  "image": "https://fakestoreapi.com/img/61IBBVJvSDL._AC_SY879_t.png",
  "rating": {
    "rate": 3.3,
    "count": 203
  }
}
```

Test Cases 2: Consultar un producto por ID

Endpoint: GET {{baseUrl}}/products/1

Objetivo: Validar que el servicio API responda exitosamente con Status 200 OK y retorne la información del producto correspondiente al ID solicitado.

Precondición:

- Cumplir las precondiciones generales definidas al inicio del documento (environment FakeStoreAPI configurado).

Pasos ejecutados:

1. Abrir Postman.
2. Crear un nuevo request en la colección YourStore API – Products:
    - Método: GET
    - URL: {{baseUrl}}/products/1
3. Guardar el request en la colección (botón Save).
4. Hacer Send.
5. Verificar que la respuesta tenga status **200 OK** y que el objeto retornado corresponda al producto con ID=1.

Resultado esperado:

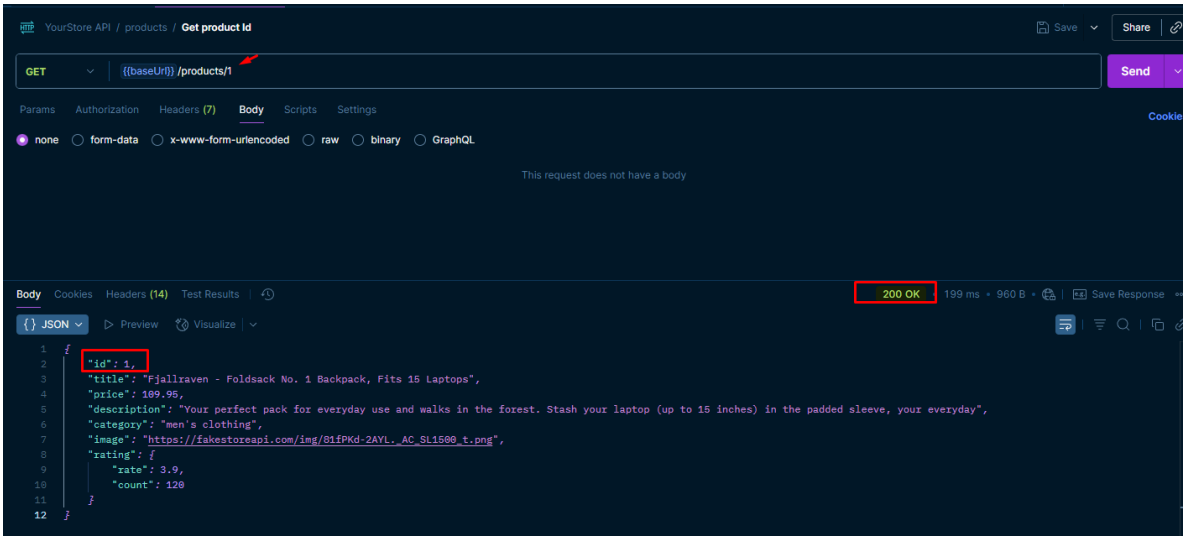
- Status **200 OK**.
- Respuesta JSON con un único objeto que contiene los campos:
  id, title, price, description, category, image, rating.
- El campo "id" debe ser igual a 1.

Resultado obtenido:

- El servicio respondió con **200 OK**.
- Se recibió la información correspondiente al producto con ID=1.

- Ejemplo de la respuesta obtenida:


```
{
  "id": 1,
  "title": "Fjallraven - Foldsack No. 1 Backpack, Fits 15 Laptops",
  "price": 109.95,
  "description": "Your perfect pack for everyday use and walks in the forest. Stash your laptop (up to 15 inches) in the padded sleeve, your everyday",
  "category": "men's clothing",
  "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_t.png",
  "rating": {
    "rate": 3.9,
    "count": 120
  }
}
```

Test Cases 3: Crear un producto

Endpoint: POST {{baseUrl}}/products

Objetivo: Validar que el servicio API permita la creación de un producto y retorne una respuesta exitosa con **Status 201** y los datos del producto creado, incluyendo el nuevo id.

Precondición:

- Cumplir **las** precondiciones generales definidas al inicio del documento (environment FakeStoreAPI configurado).

Pasos ejecutados:

1. Abrir Postman.
2. Seleccionar la colección YourStore API – Products.
3. Crear un nuevo request:
    - Método: POST
    - URL: {{baseUrl}}/products
    - Header:
        - Content-Type: application/json
        - Accept: application/json
    - Body (formato raw, JSON), ejemplo:

```
{
  "title": "QA Test Product",
  "price": 199.99,
  "description": "Producto de prueba técnica",
  "category": "electronics",
  "image": https://i.pravatar.cc
}
```

4. Guardar el request en la colección (botón Save).
5. Hacer Send.

6. Verificar que la respuesta tenga status 201 Created y que se devuelva el objeto creado con un id asignado.

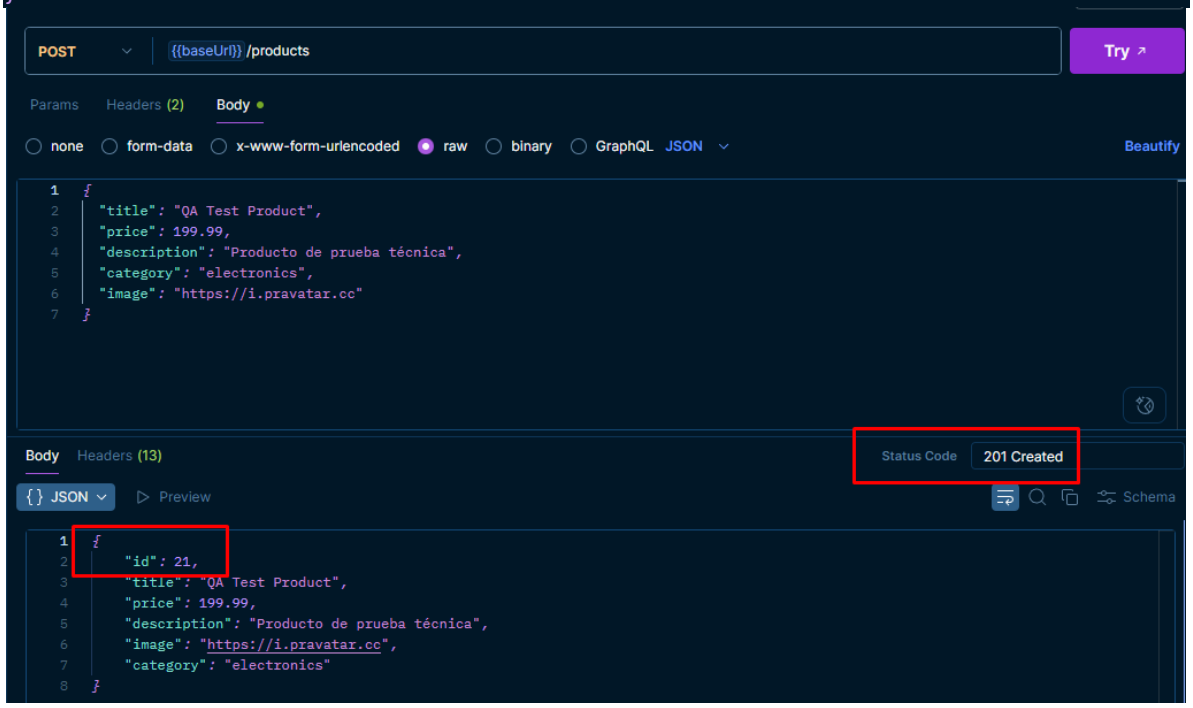
Resultado esperado:

- Status 201 **Created**.
- El objeto creado debe contener los mismos campos enviados en el body (title, price, description, image, category) más un nuevo id generado por la API.

Resultado obtenido:

- El servicio respondió exitosamente con **201 Created**.
- Se generó un nuevo id para el producto creado.
- Ejemplo de la respuesta obtenida:


```
{
  "id": 21,
  "title": "QA Test Product",
  "price": 199.99,
  "description": "Producto de prueba técnica",
  "image": "https://i.pravatar.cc",
  "category": "electronics"
}
```

Test Cases 4: Actualizar la imagen del producto creado

Endpoint: PUT {{baseUrl}}/products/{id}

Objetivo: Validar que el servicio API permita actualizar la información de un producto existente, específicamente la imagen, y que retorne una respuesta exitosa con Status 200 OK.

Precondición:

- Cumplir las precondiciones generales (environment FakeStoreAPI configurado).
- Haber creado previamente un producto (Caso 3) y disponer del id generado (ejemplo: id = 21).

Pasos ejecutados:

1. Abrir Postman.
2. Seleccionar la colección YourStore API – Products.
3. Crear un nuevo request:
    - Método: PUT
    - URL: {{baseUrl}}/products/21 (reemplazar 21 por el ID real creado en el caso anterior).
    - Header:
        - Content-Type: application/json
    - Body (formato raw, JSON), ejemplo:

```
{
 "id": 21,
 "title": "QA Test Product",
 "price": 199.99,
 "description": "Producto de prueba técnica",
 "category": "electronics",
 "image": "https://i.pravatar.cc/150?img=6"
}
```

4. Guardar el request en la colección (botón Save).

5. Hacer Send.
6. Verificar que la respuesta tenga status **200 OK** y que el campo image se actualice con la nueva URL.

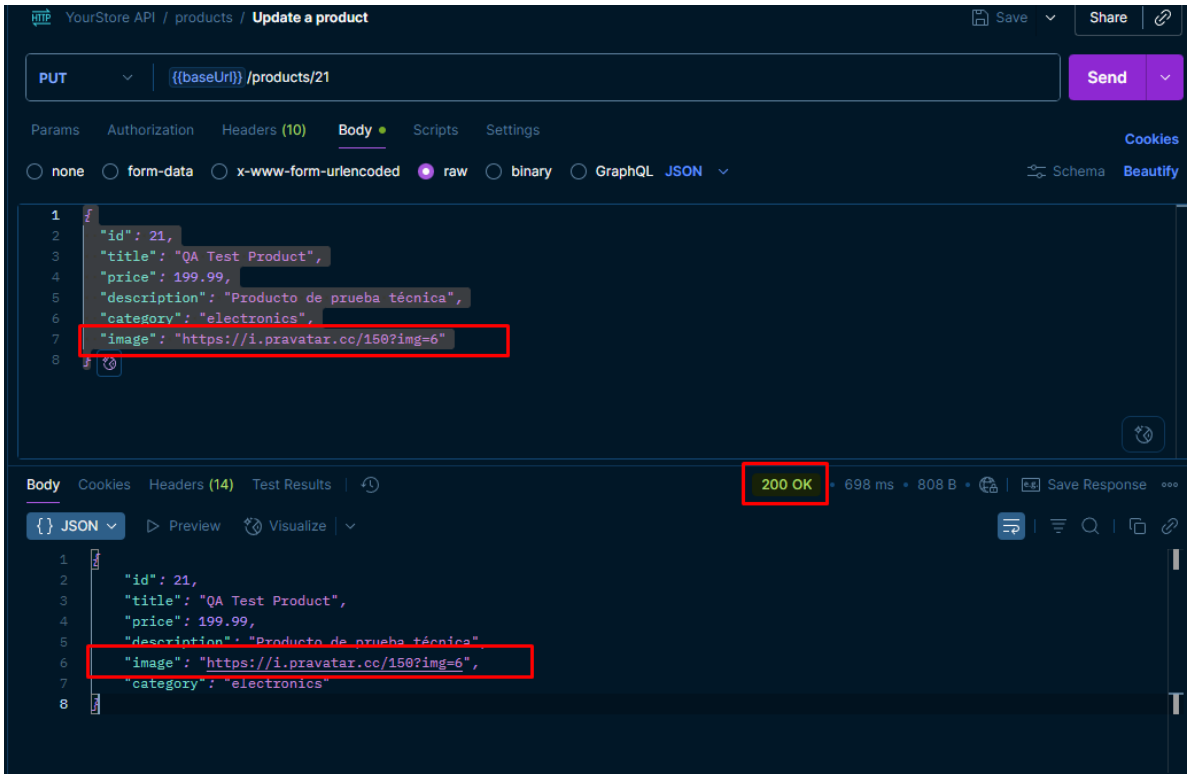
Resultado esperado:

- Status **200 OK**.
- El objeto del producto actualizado debe contener la nueva imagen ("image": "https://i.pravatar.cc/150?img=6").
- Los demás atributos del producto permanecen iguales a los creados en el Caso 3.

Resultado obtenido:

- El servicio respondió exitosamente con **200 OK**.
- La propiedad image fue actualizada al nuevo valor.
- Ejemplo de la respuesta obtenida:


```
{
  "id": 21,
  "title": "QA Test Product",
  "price": 199.99,
  "description": "Producto de prueba técnica",
  "image": "https://i.pravatar.cc/150?img=6",
  "category": "electronics"
}
```



Parte B – Casos Negativos

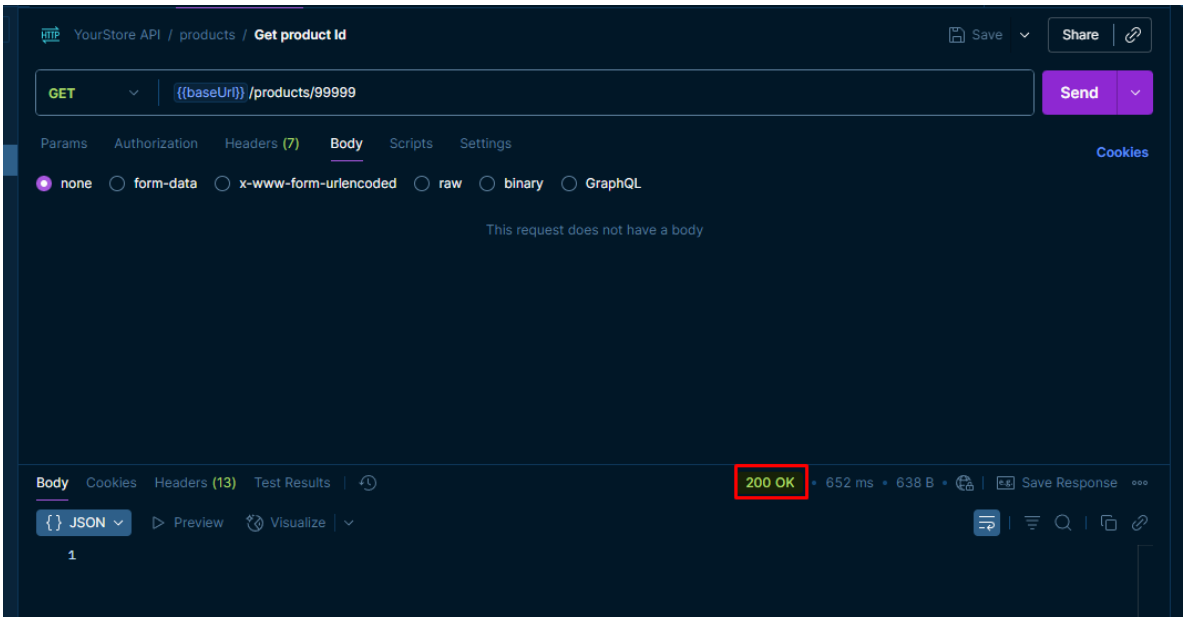
Caso Negativo 1: Consultar producto inexistente

Endpoint: GET {{baseUrl}}/products/99999

Resultado esperado: El servicio debería devolver 404 Not Found o un mensaje de error indicando que el producto no existe.

Resultado obtenido: El servicio respondió con 200 OK y un objeto genérico, lo que no corresponde al comportamiento esperado en un entorno productivo.

Observación: Se documenta la inconsistencia, ya que la API de prueba (FakeStoreAPI) no valida correctamente IDs inexistentes.



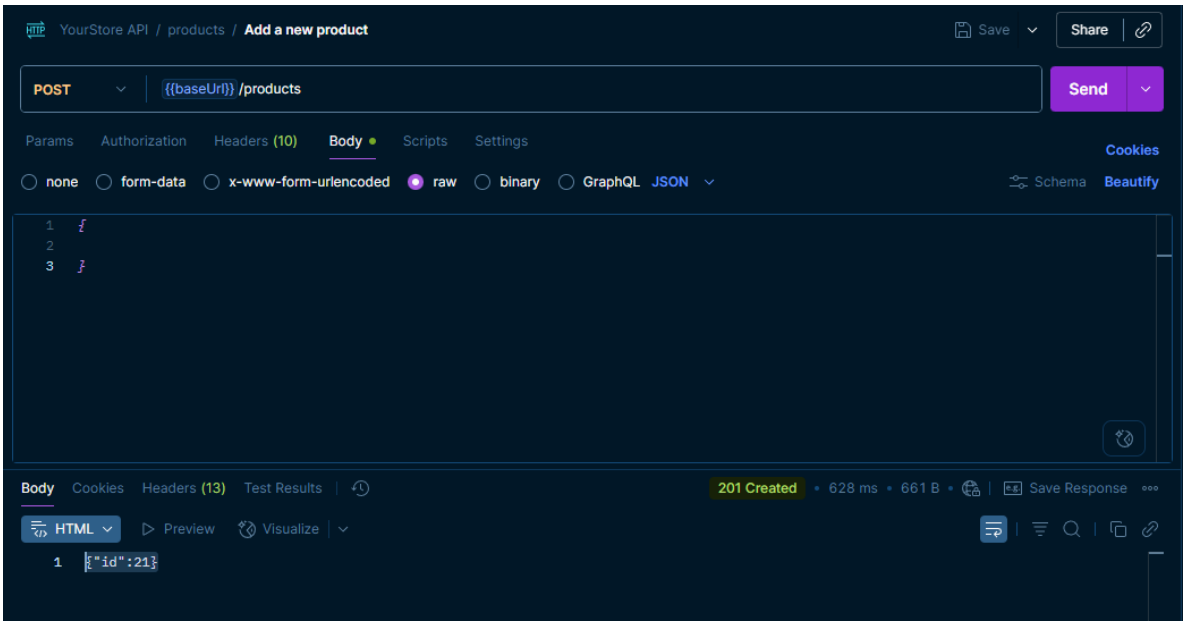
Caso Negativo 2: Crear producto con body vacío

Endpoint: POST {{baseUrl}}/products

Resultado esperado: El servicio debería responder con 400 Bad Request, indicando que faltan campos obligatorios.

Resultado obtenido: El servicio respondió con 201 Created y generó un nuevo id, a pesar de que el body estaba vacío.

Observación: Este comportamiento refleja que la API de prueba no valida datos obligatorios, lo cual en un entorno real sería considerado un defecto crítico.



Caso Negativo 3: Actualizar producto con ID inválido

Endpoint: PUT {{baseUrl}}/products/abc

Objetivo: Validar que el servicio rechace un request de actualización cuando el ID del producto es inválido.

Pasos ejecutados:

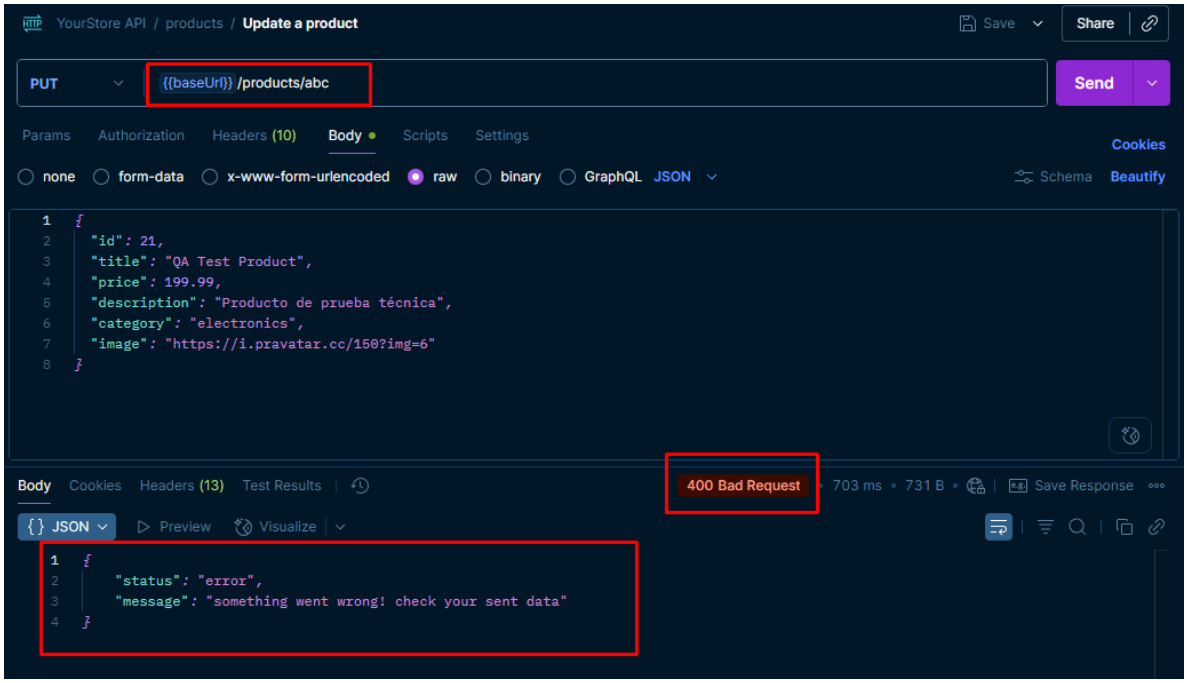
1. Enviar un request PUT a {{baseUrl}}/products/abc con body JSON válido.
2. Verificar la respuesta.

Resultado esperado:

- Status 400 Bad Request o 404 Not Found**.**
- Mensaje de error indicando ID inválido.

Resultado obtenido:

- El servicio devolvió error esperado.



Caso Negativo 4: Crear producto con tipos de datos incorrectos

Endpoint: POST {{baseUrl}}/products

Objetivo: Validar que el servicio rechace la creación de un producto con un tipo de dato inválido en el campo price.

Pasos ejecutados:

1. Enviar un request POST a {{baseUrl}}/products con el siguiente body:

```
{
```

```
  "title": "Producto inválido",
  "price": "texto",
  "description": "Prueba negativa",
  "image": "https://i.pravatar.cc",
  "category": "electronics"
}
```

2. Verificar la respuesta.

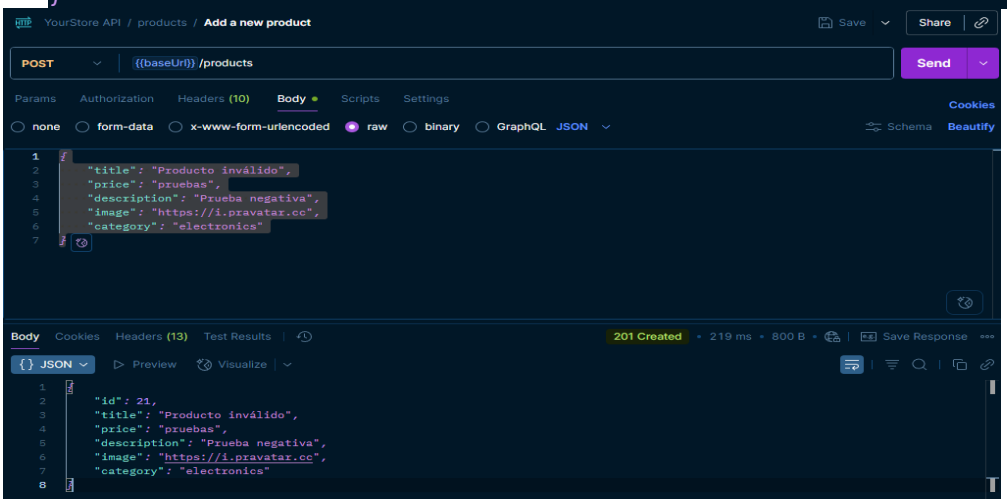
Resultado esperado:

- Status 400 Bad Request.
- Mensaje de error indicando que el campo price debe ser numérico.

Resultado obtenido:

- El servicio respondió con **201 Created**, aceptando un valor de tipo texto en el campo price.
- Ejemplo de respuesta:

```
{
  "id": 21,
  "title": "Producto inválido",
  "price": "pruebas",
  "description": "Prueba negativa",
  "image": "https://i.pravatar.cc",
  "category": "electronics"
}
```



Caso Opcional: Eliminar un producto existente

Endpoint: DELETE {{baseUrl}}/products/{id}

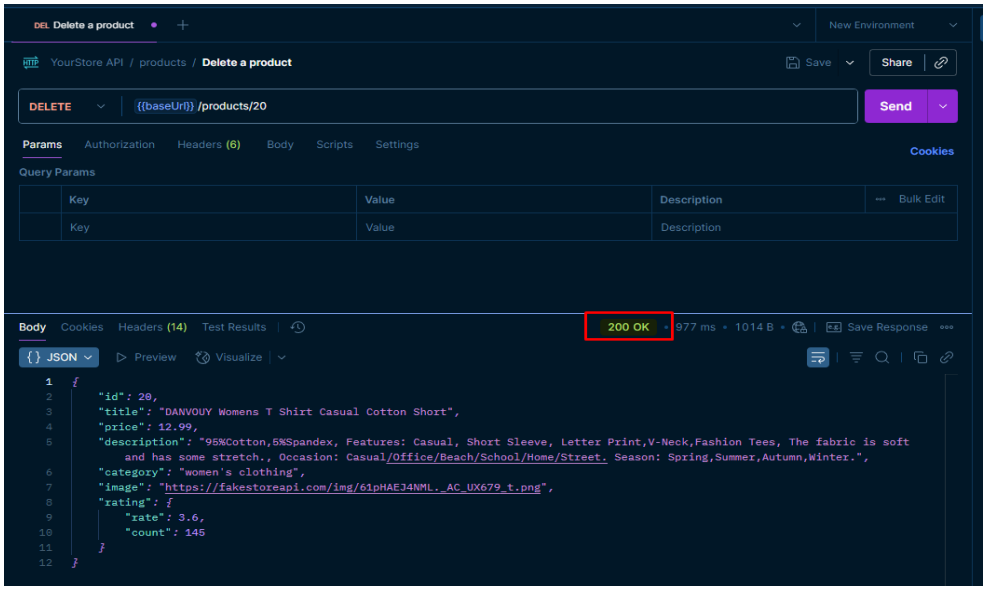
Objetivo: Validar que el servicio API permita eliminar un producto existente y devuelva respuesta exitosa.

Pasos ejecutados:

1. Enviar DELETE {{baseUrl}}/products/{id}.
2. Verificar status **200 OK**.

Resultado esperado: Confirmación de que el producto fue eliminado.

Resultado obtenido: La API responde con el objeto eliminado



Parte C → Pruebas de Performance (JMeter)

Caso de Prueba 5: Prueba de Carga – 150 usuarios concurrentes

- **Herramienta**: Apache JMeter
- **Objetivo**: Validar el desempeño de la API bajo una carga de 150 usuarios concurrentes durante 2 minutos, midiendo tiempo de respuesta, tasa de error y estabilidad.

Precondiciones:

- JMeter instalado y configurado en la máquina de pruebas.
- Archivo de prueba (.jmx) configurado con:
- Thread Group → 150 threads (usuarios)
- Ramp-up time → 30 segundos
- Duración → 120 segundos (2 minutos)
- Endpoints a probar:
  - GET {{baseUrl}}/products

    - POST {{baseUrl}}/products (con body válido para creación de producto)
- Conexión estable a internet.

Pasos ejecutados:

1. Abrir JMeter.
2. Importar el plan de prueba (LoadTest_150Users.jmx).
3. Configurar el Thread Group:
    a. Número de usuarios: 150
    b. Ramp-up: 30 segundos
    c. Duración: 2 minutos (120 s)
4. Agregar dos HTTP Requests:
    a. GET → {{baseUrl}}/products
    b. POST → {{baseUrl}}/products (con body JSON válido).
5. Agregar Listeners:
    a. Summary Report
    b. Aggregate Report
    c. Graph Results
    d. View Results Tree
6. Ejecutar el plan de prueba.

Resultado esperado:

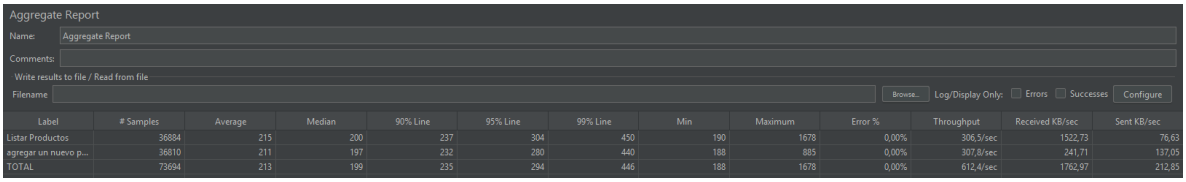
- Status 200 OK en >95% de solicitudes.
- Tiempo de respuesta promedio < 2000 ms.
- Tasa de error ≤ 2%.

Resultado obtenido (a partir del reporte JMeter):

- **Solicitudes totales:** 68,721 (34,325 POST /products, 34,396 GET /products).
- **Tasa de error:** 0% (todas exitosas).
- **Tiempo de respuesta promedio:**
    - General: **229 ms**
    - POST /products: **227 ms**
    - GET /products: **233 ms**
- **Percentiles:**
    - P90 ≈ 202–203 ms
    - P95 ≈ 276–299 ms
    - P99 ≈ 367–383 ms

- **Throughput:**
    - General: **537 req/segundo**
    - POST /products: **540 req/segundo**
    - GET /products: **547 req/segundo**

Conclusión:
La API manejó 150 usuarios concurrentes durante 2 minutos con **cero errores**, **tiempos promedio muy bajos (< 250 ms)** y un **throughput estable de ~540 req/segundo**, cumpliendo con los criterios de éxito definidos.


Aggregate Report

Name: Aggregate Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
Listar Productos	36884	215	200	237	304	450	190	1678	0,00%	306,5/sec	1522,73	76,63
agregar un nuevo p...	36810	211	197	232	280	440	188	885	0,00%	307,8/sec	241,71	137,05
TOTAL	73694	213	199	235	294	446	188	1678	0,00%	612,4/sec	1762,97	212,85

Caso de Prueba 6: Prueba de Estrés – Escalado de usuarios

- **Herramienta:** Apache JMeter
- **Objetivo:** Validar la capacidad de la API bajo un incremento progresivo de carga, escalando de **100 a 1000 usuarios concurrentes** en intervalos de **150 usuarios**, para identificar el punto de degradación del servicio.

Precondiciones:

- JMeter instalado y configurado en la máquina de pruebas.
- Archivo .jmx diseñado para escalado progresivo:
    - Se puede usar **Ultimate Thread Group** o **Stepping Thread Group** (plugin de JMeter Plugins) para controlar el incremento de usuarios.
- Endpoints a probar:
    - GET {{baseUrl}}/products
    - POST {{baseUrl}}/products
- Conexión estable a internet.

Pasos ejecutados:

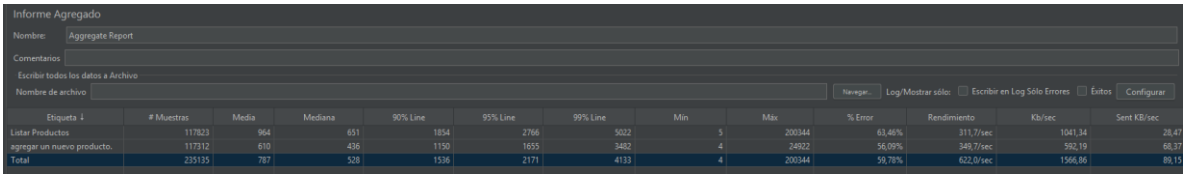
1. Abrir JMeter.
2. Crear un Thread Group avanzado (Ultimate/Stepping):
    - Iniciar en **100 usuarios**.
    - Incrementar en bloques de **150 usuarios** cada X segundos hasta llegar a **1000 usuarios**.
    - Mantener cada nivel de carga durante 1-2 minutos antes de escalar.
3. Agregar dos **HTTP Samplers**:
    - GET /products
    - POST /products
4. Agregar **Listeners**:
    - Summary Report
    - Aggregate Report
    - Graph Results
5. Ejecutar la prueba.

Resultado esperado:

- Status 200 OK en >95% de solicitudes.
- Tiempo de respuesta promedio < 2000 ms.
- Tasa de error ≤ 2%

Conclusión

- La API experimenta una tasa de error elevada (≈48%), lo que está muy por encima del límite esperado del 2%, indicando problemas bajo la carga actual.
- El tiempo promedio se mantiene dentro de los límites (<1,500 ms), pero los percentiles P95 y P99 muestran latencias muy altas, especialmente en GET /products (hasta ~9.5 s).
- El throughput se mantiene relativamente estable, mostrando que la API procesa solicitudes, pero con mucha degradación en la confiabilidad.


Informe Agregado

Nombre: Aggregate Report

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Mostrar sólo: | Escribir en Log Sólo Errores | Éxitos | Configurar

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
Listar Productos	117823	964	651	1854	2766	5022	5	200344	63,46%	311,7/sec	1041,34	28,47
agregar un nuevo producto.	117312	610	436	1150	1655	3482	4	24922	56,09%	349,7/sec	592,19	68,37
Total	235135	787	528	1536	2171	4133	4	200344	59,78%	622,0/sec	1566,86	89,15