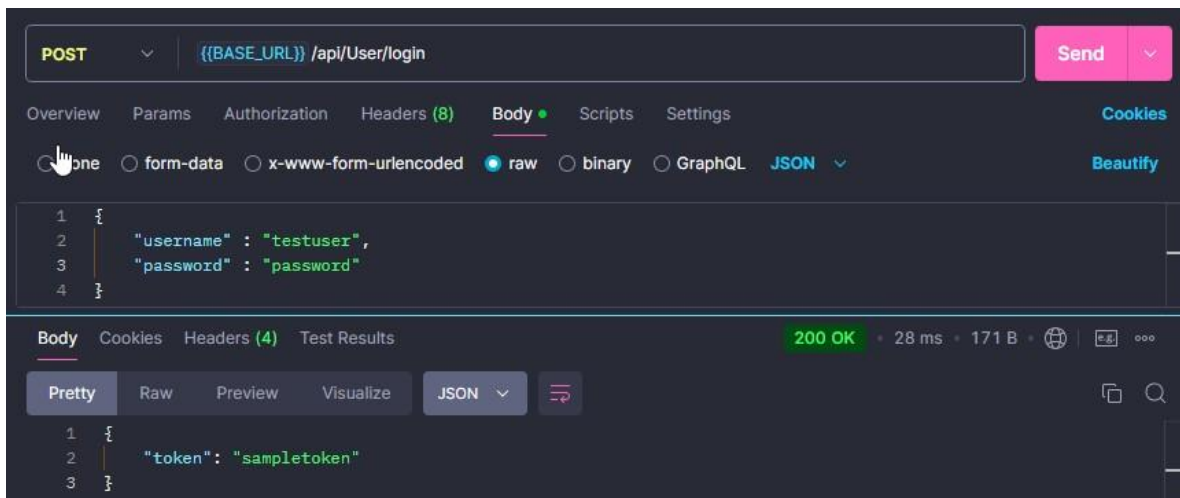


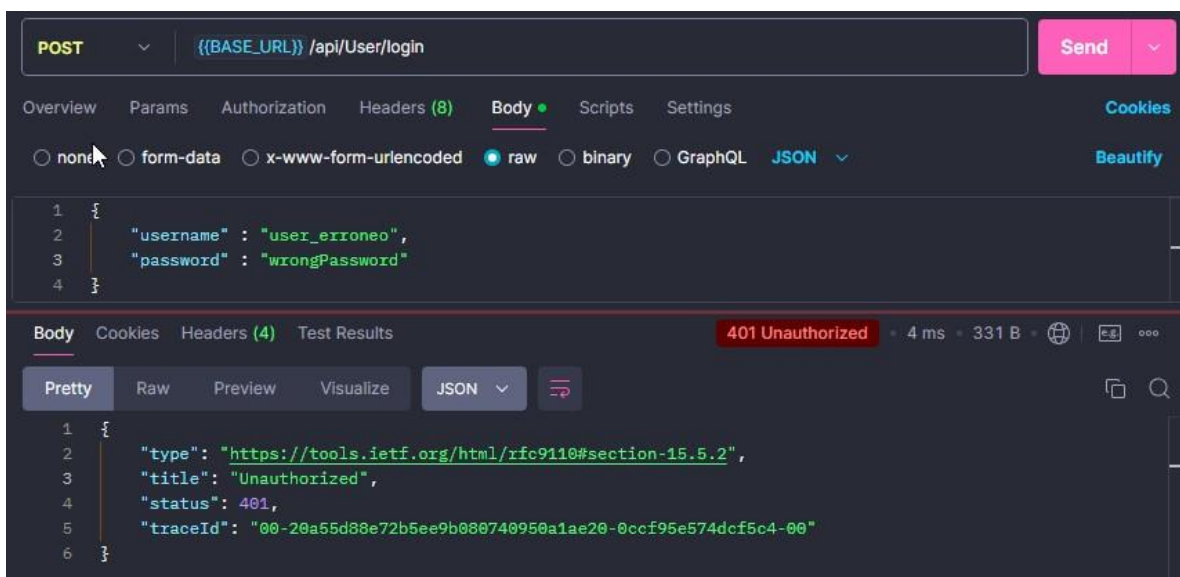
PART 3: TEST EXECUTION AND REPORTING

User Authentication (AU)-BACKEND (C# APIs):

- AU01- Verify login with valid credentials



- AU02 - Login attempt with wrong credentials



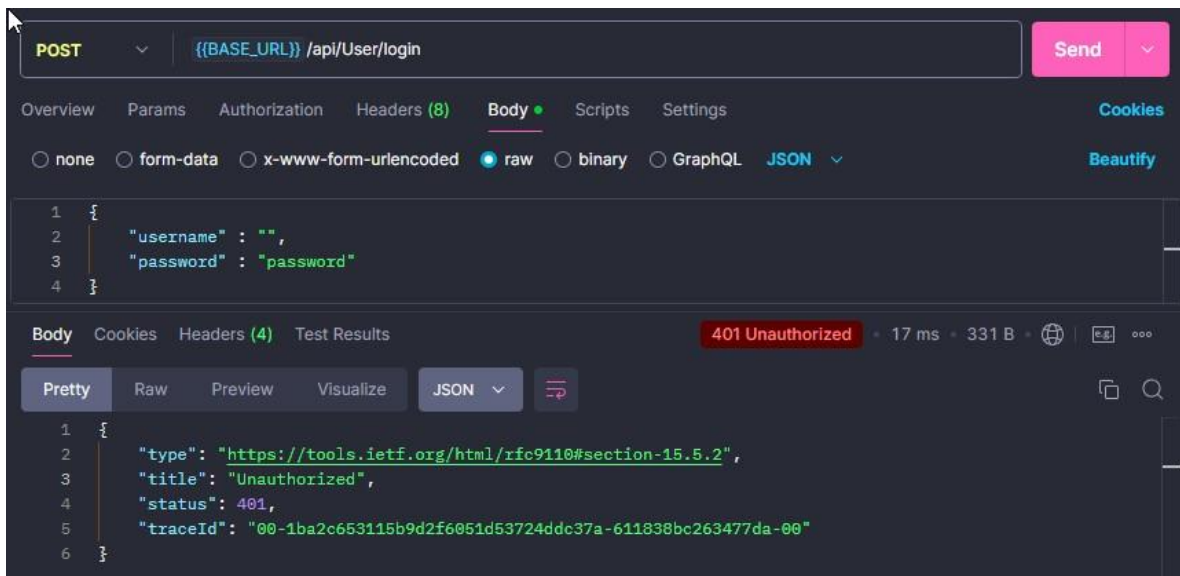
- **AU03 - Login with empty fields**

The screenshot shows a REST client interface with a POST request to `{{BASE_URL}} /api/User/login`. The request body is raw JSON: `{ "username": "", "password": "" }`. The response is a 401 Unauthorized status with a response time of 16 ms and a body size of 331 B. The response body is displayed in JSON format: `{ "type": "https://tools.ietf.org/html/rfc9110#section-15.5.2", "title": "Unauthorized", "status": 401, "traceId": "00-929d1d146316f5f8696e33475165fefe-3667c4ddcd58e6da-00" }`.

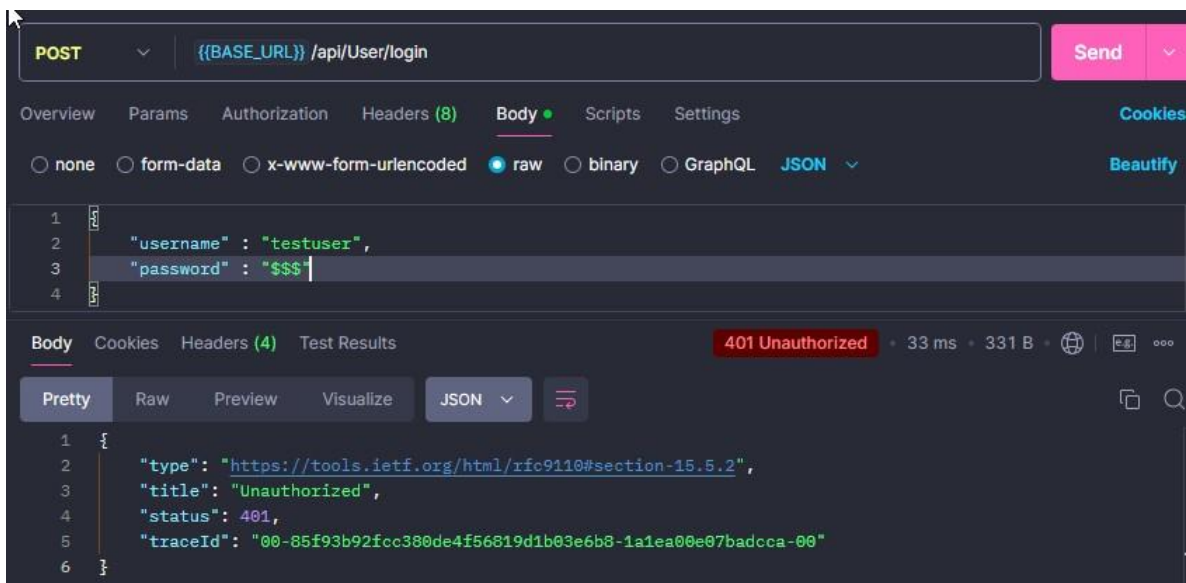
- **AU04- Login with correct username and empty password field.**

The screenshot shows a REST client interface with a POST request to `{{BASE_URL}} /api/User/login`. The request body is raw JSON: `{ "username": "testuser", "password": "" }`. The response is a 401 Unauthorized status with a response time of 39 ms and a body size of 331 B. The response body is displayed in JSON format: `{ "type": "https://tools.ietf.org/html/rfc9110#section-15.5.2", "title": "Unauthorized", "status": 401, "traceId": "00-96f253738c98c94461fca0c3a58462a3-a2e189d18ae702b9-00" }`.

- AU05 - Login with correct password and empty username field

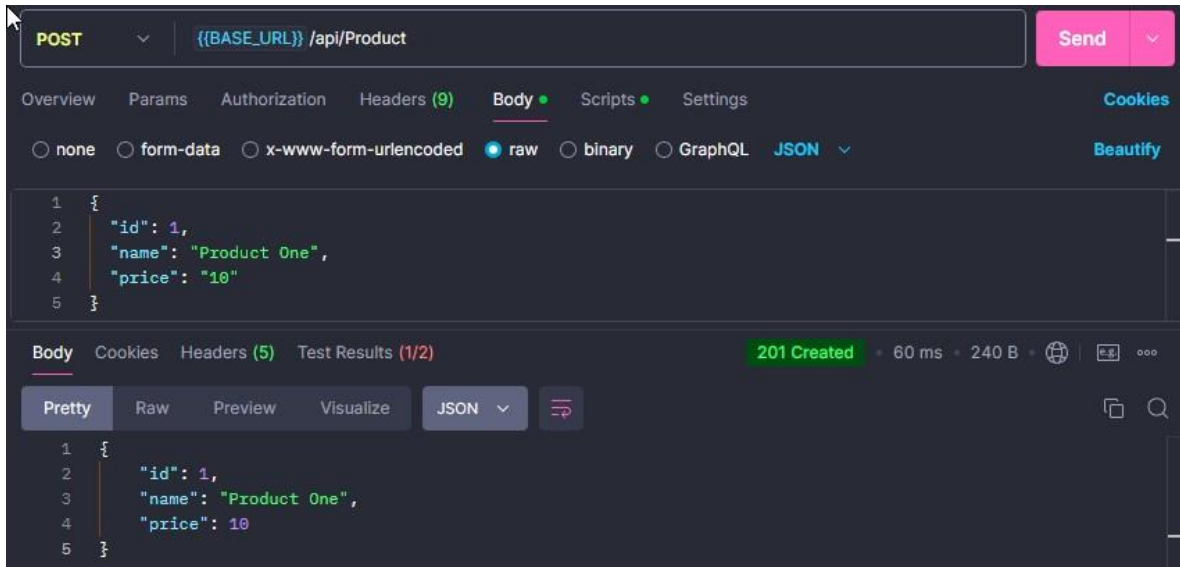


- AU06 - Login attempt with invalid JSON formatting

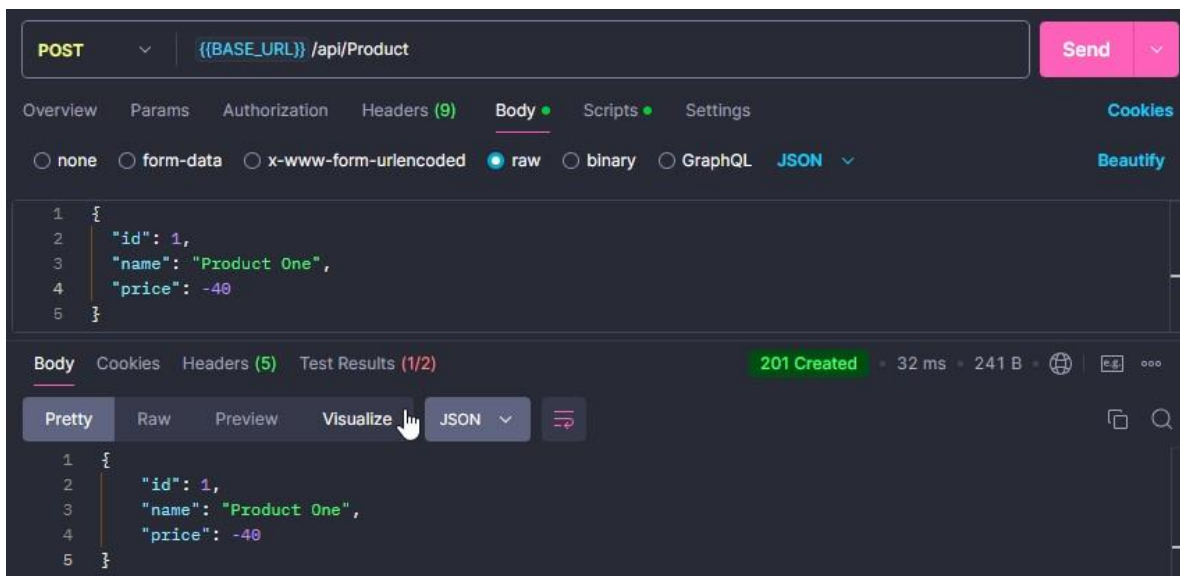


BACKEND (C# APIs): Product Management: (Create Product - POST)

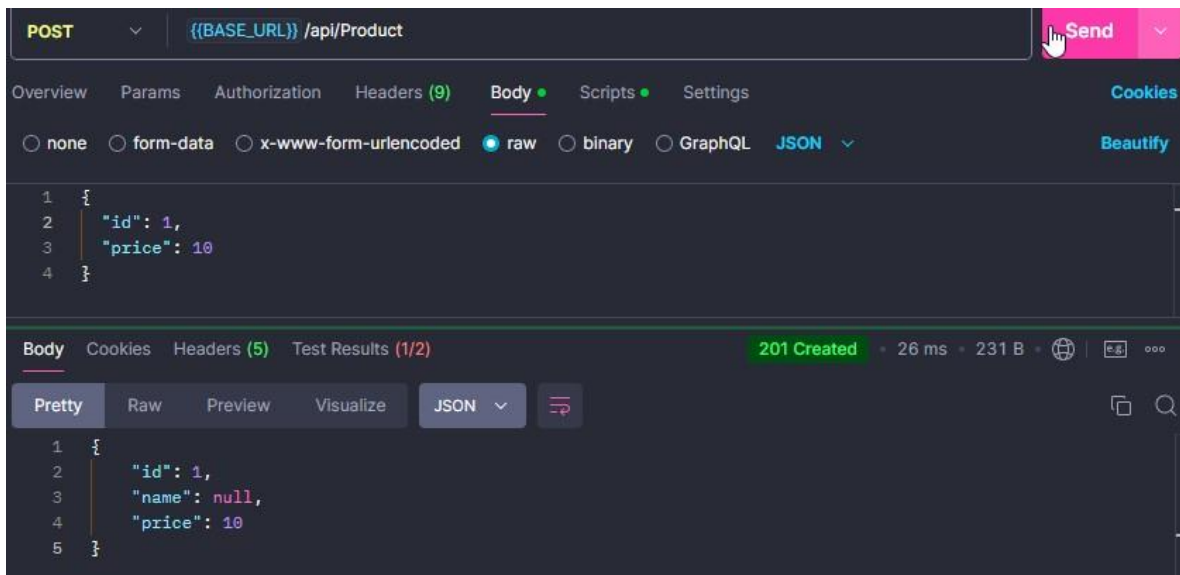
- PC01-Create product with valid credentials



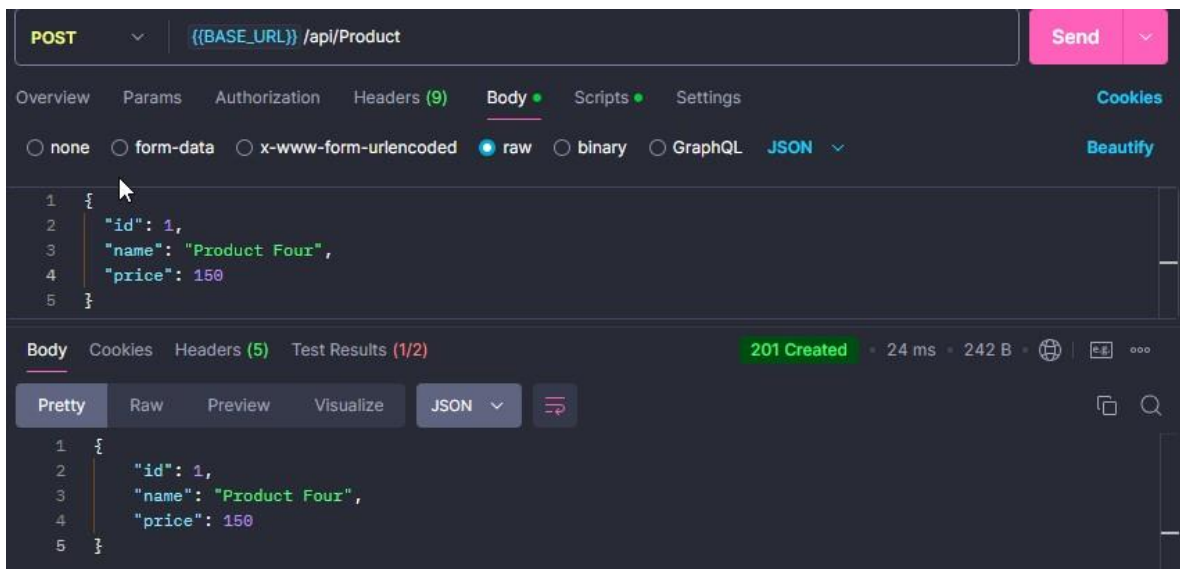
- PC02- Create product with negative price



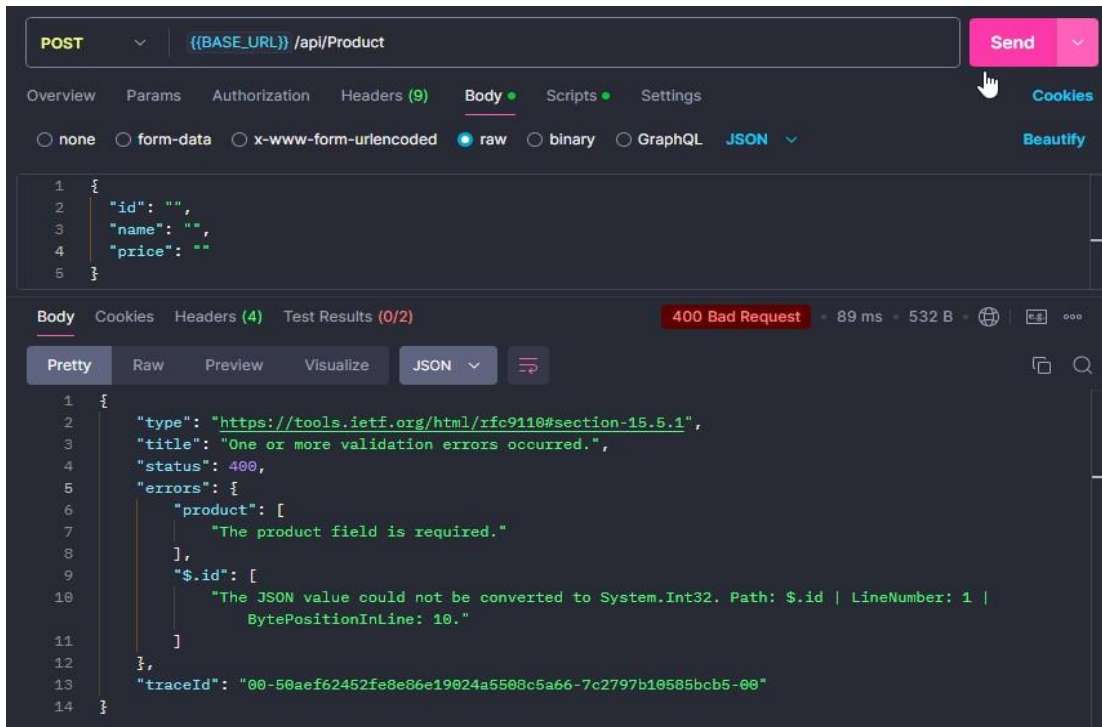
- PC03-Create a product without the “name” field



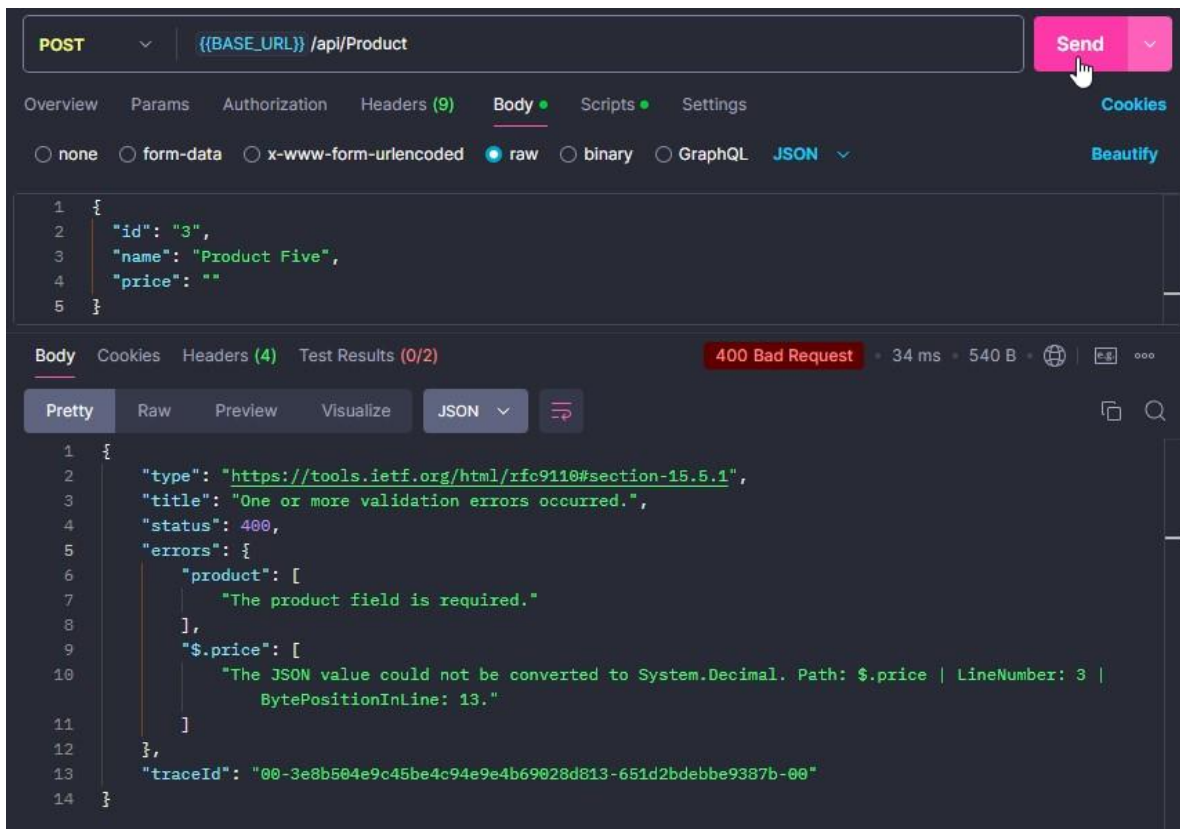
- PC04-Attempting to create a product with a duplicate ID



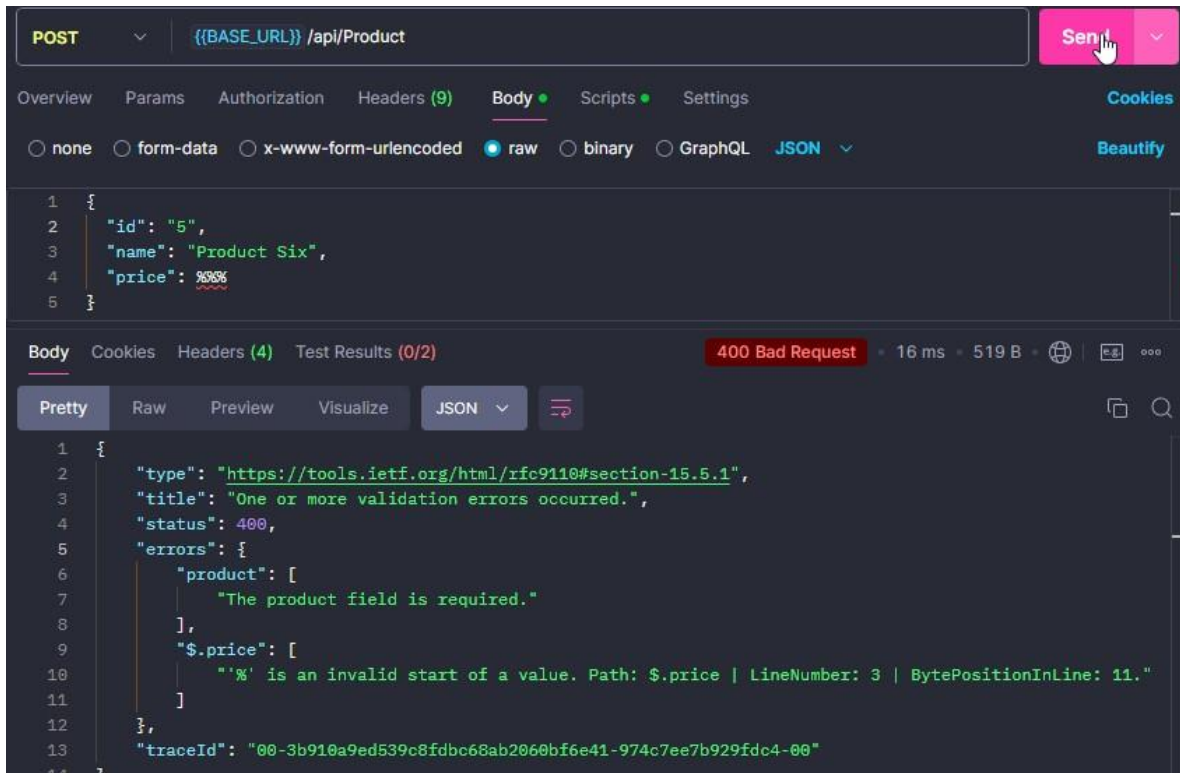
- PC05- Create a product with empty fields



- PC06- Create a product without the “price” field

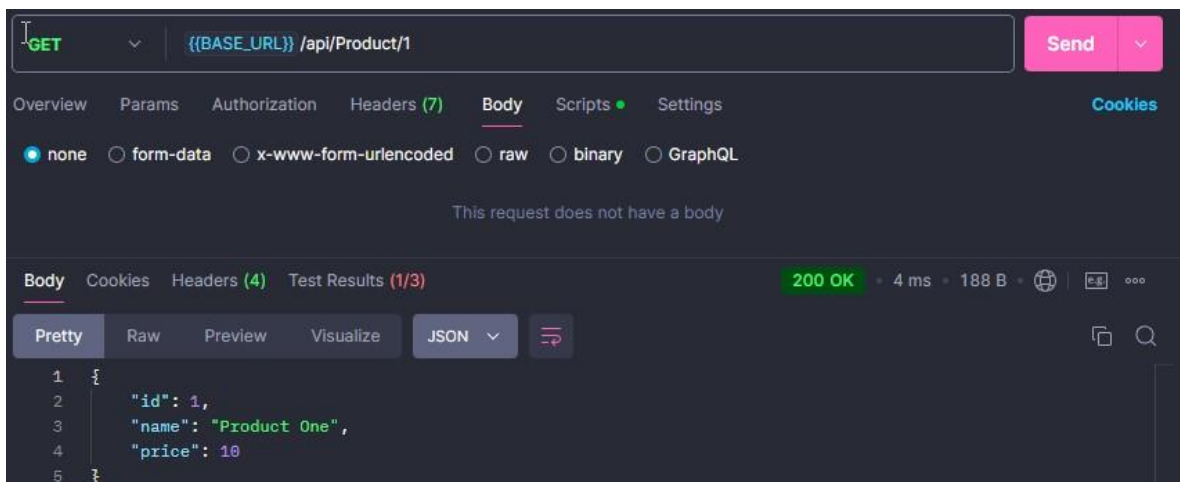


- PC07- Create product with incorrectly formatted price

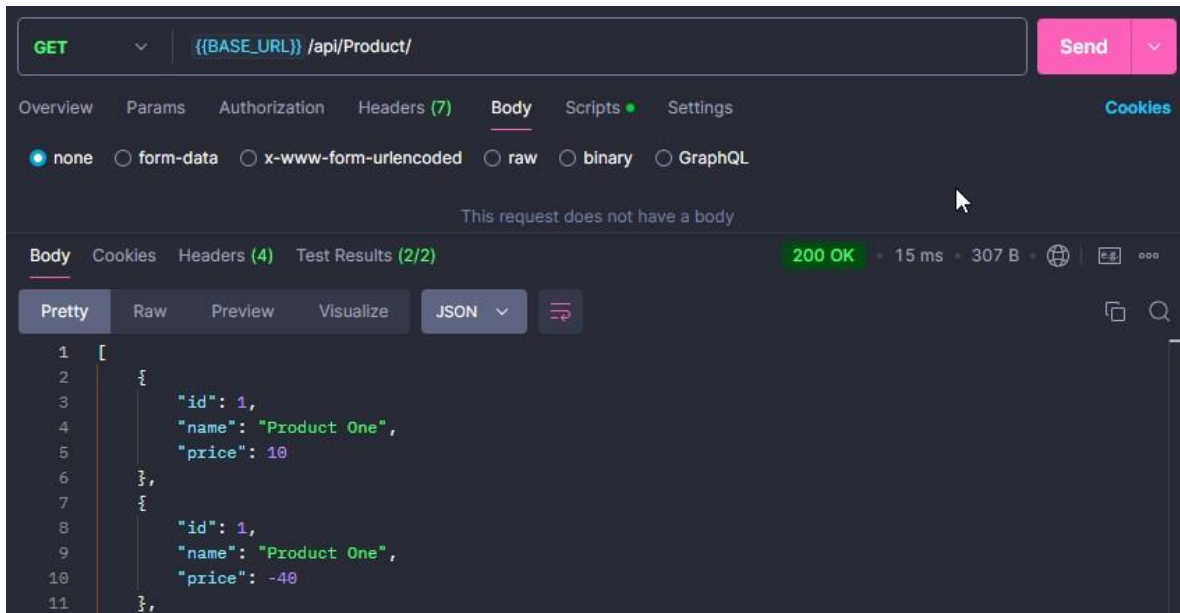


BACKEND (C# APIs): Product Management: (Reding Products - GET)

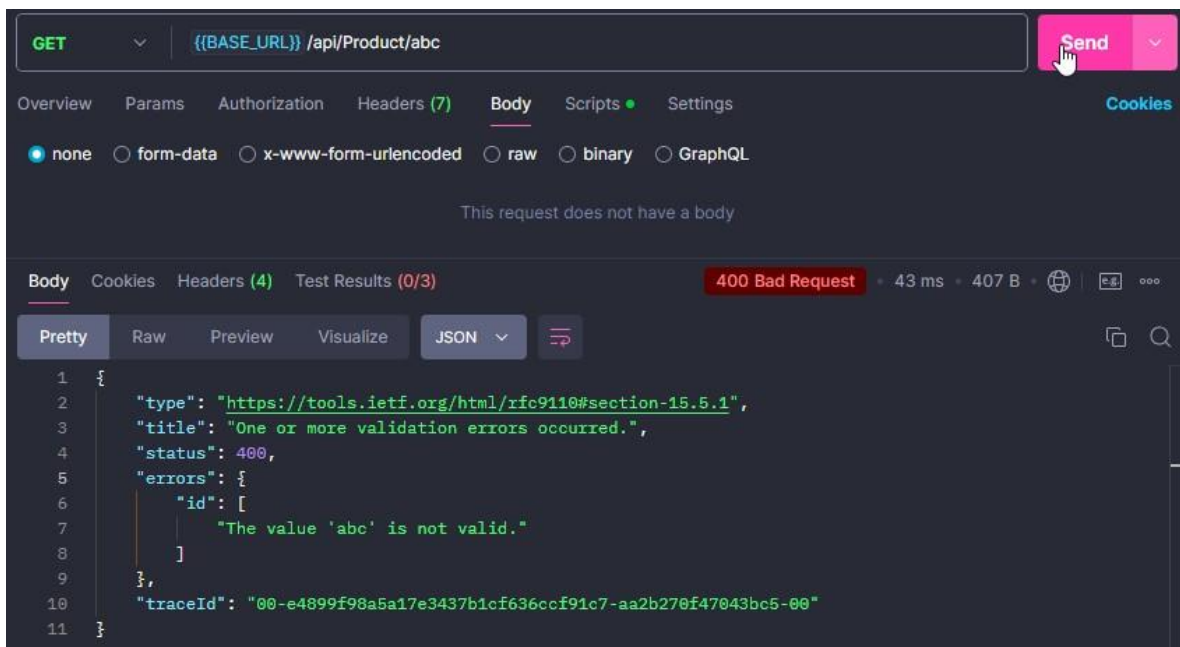
- PC08- Get an existing product by ID



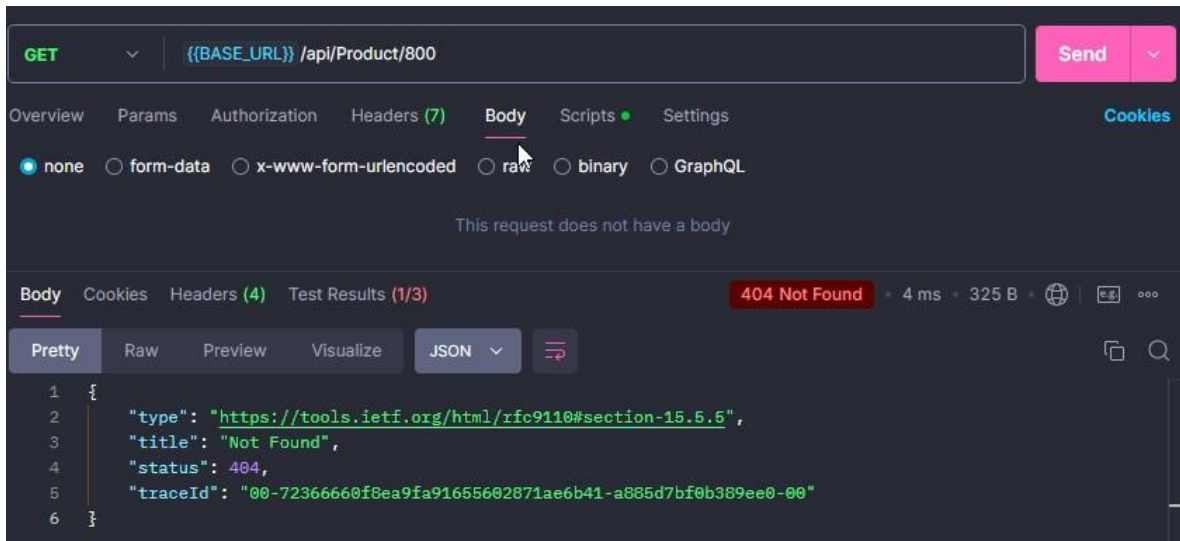
- PC09- Get product list



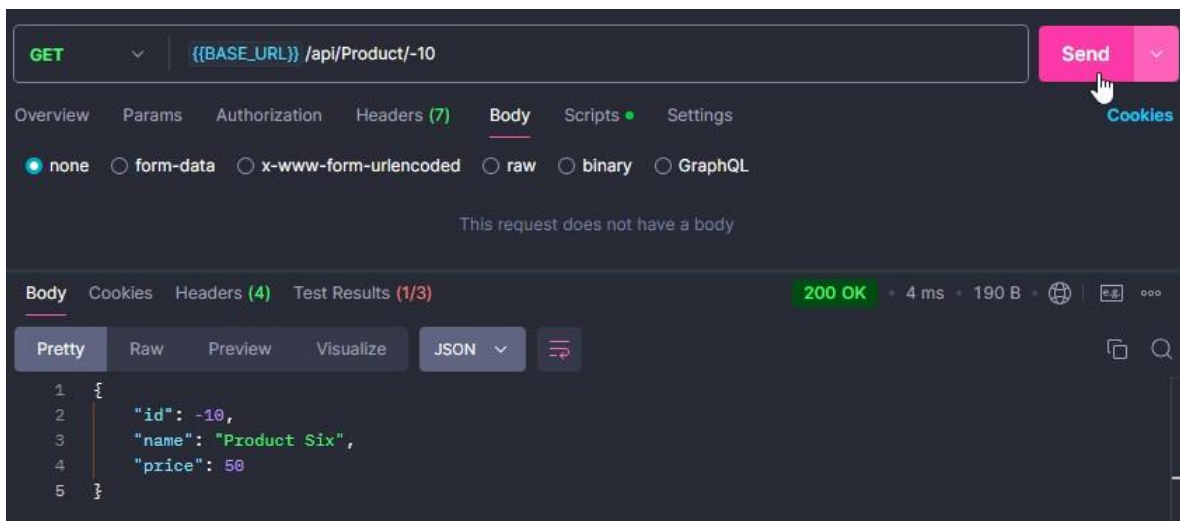
- PC10- Get a product with invalid ID



- **PC11- Get non-existing product**

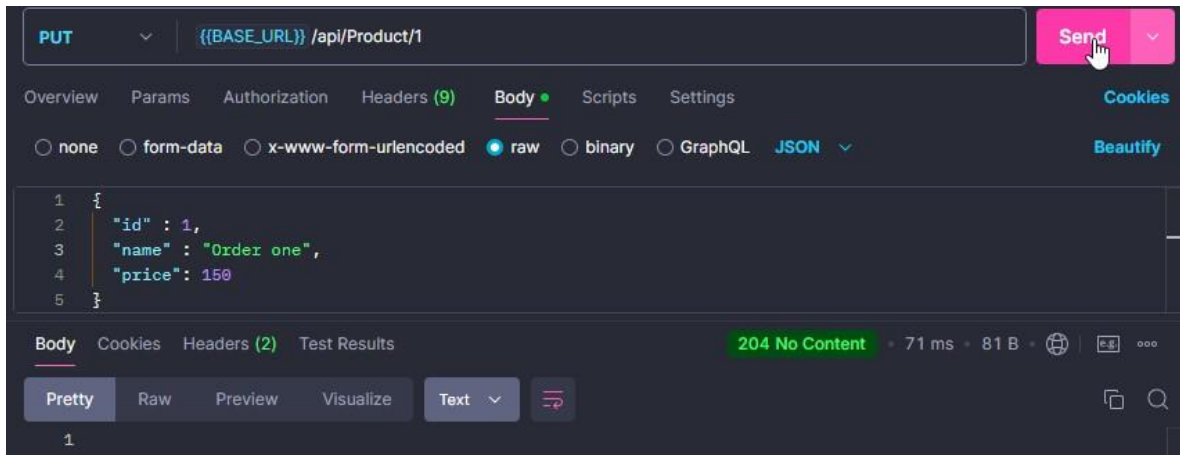


- **PC12- Get a product with negative id**

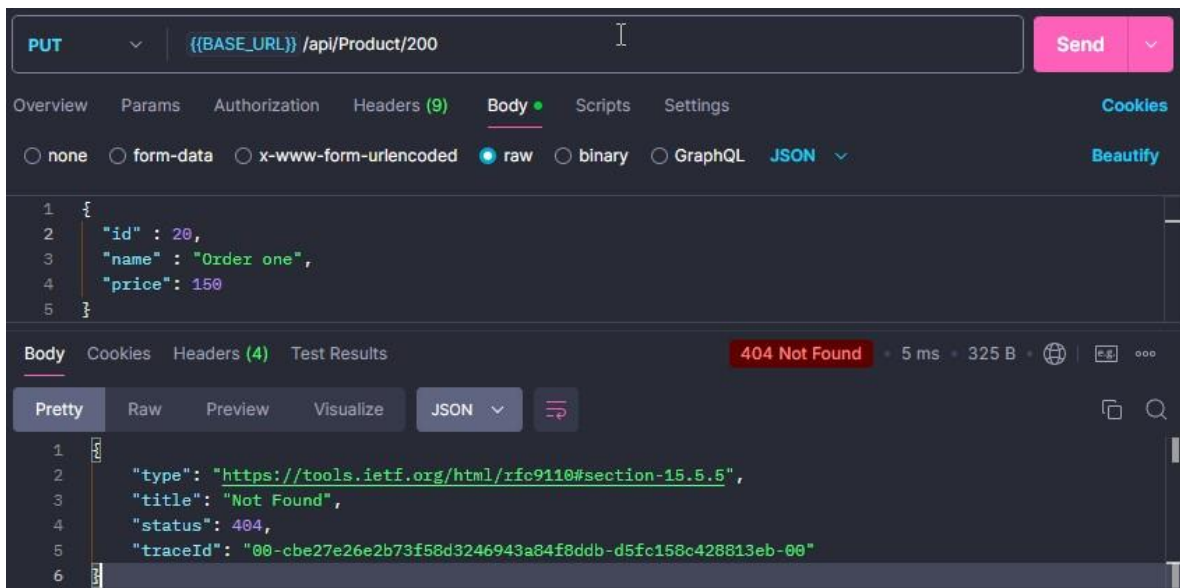


BACKEND (C# APIs): Product Management: (Update Products - PUT)

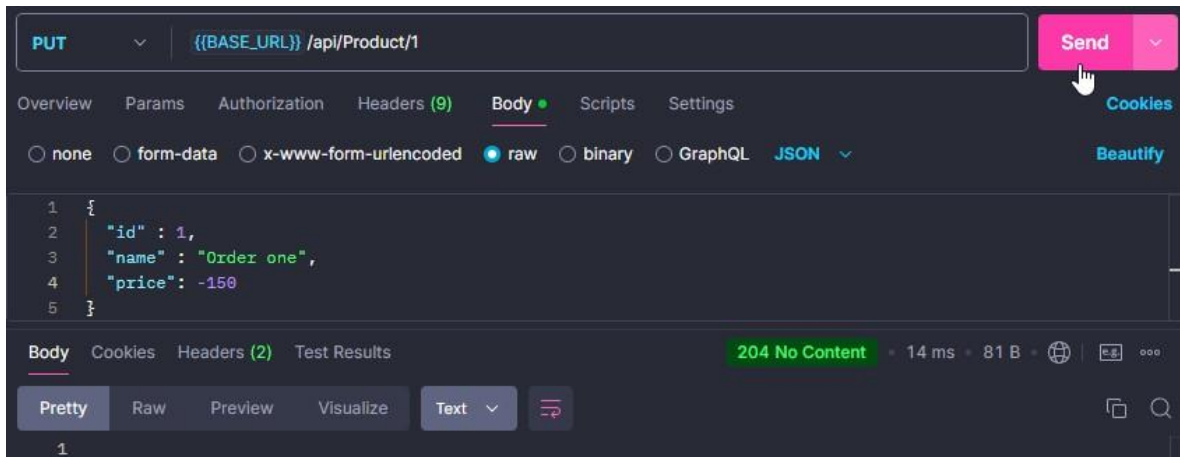
- **PC13- Update an existing product**



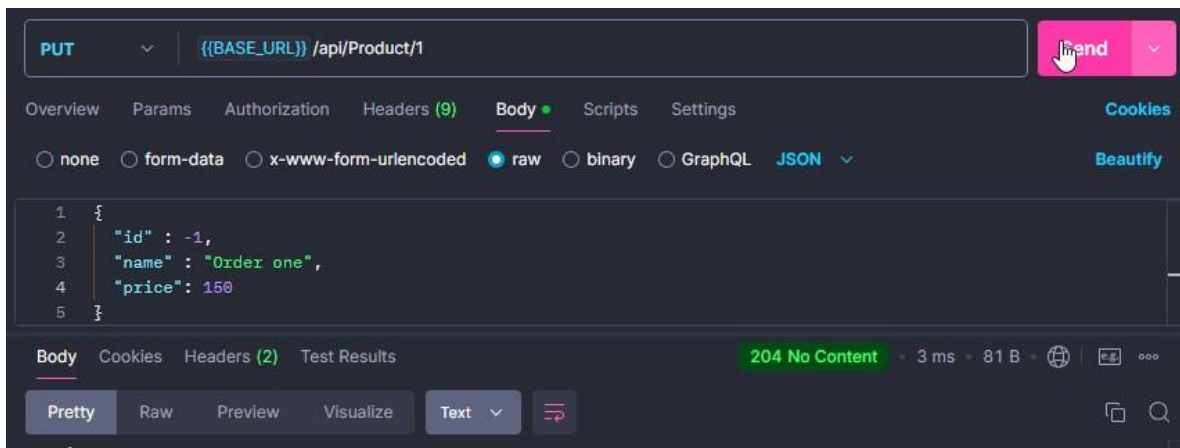
- **PC14- Update non-existent product**



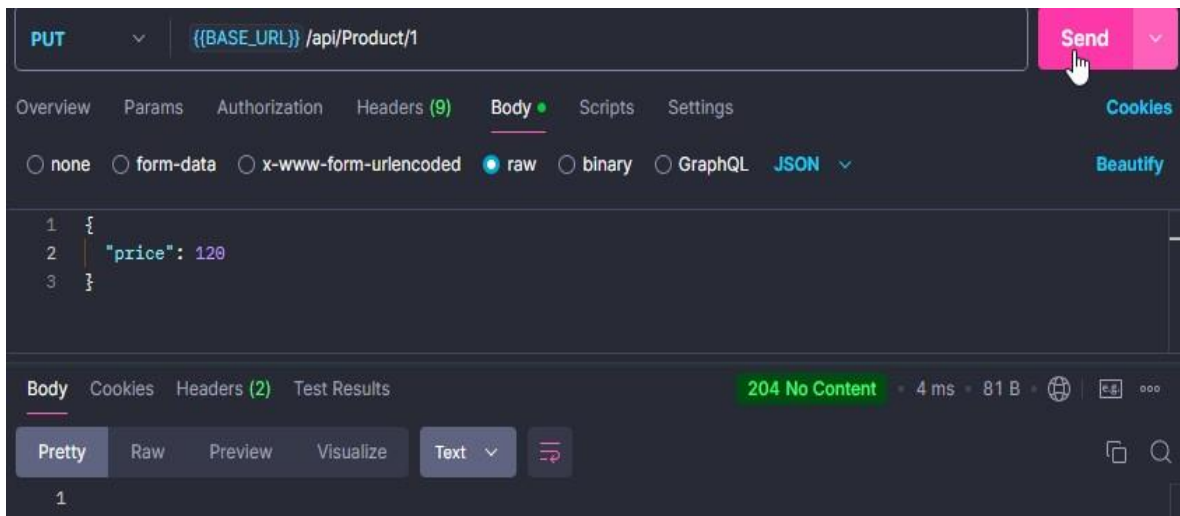
- **PC15- Update product with negative price**



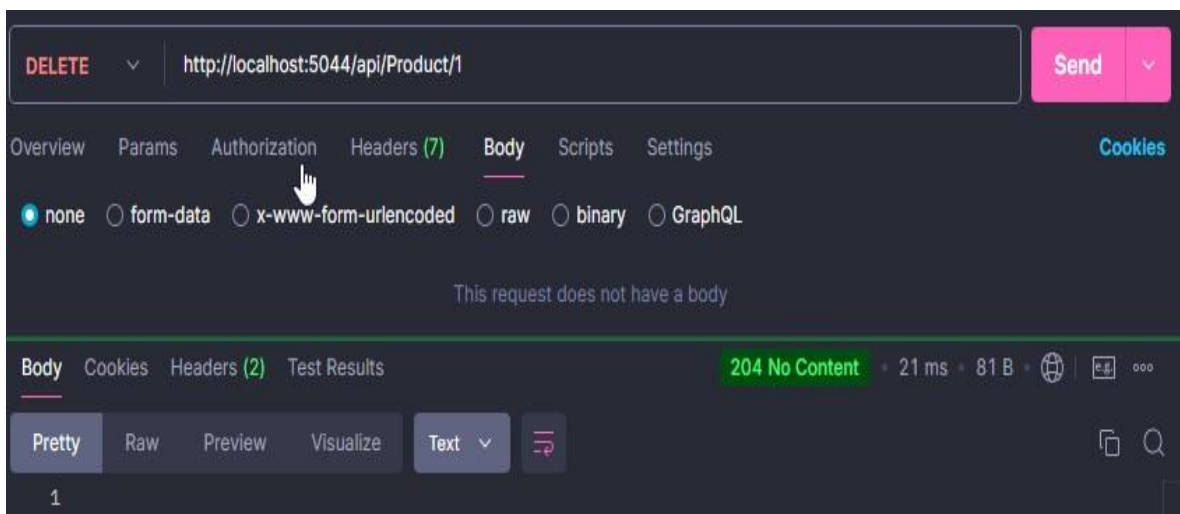
- **PC16- Update a product with negative ID**



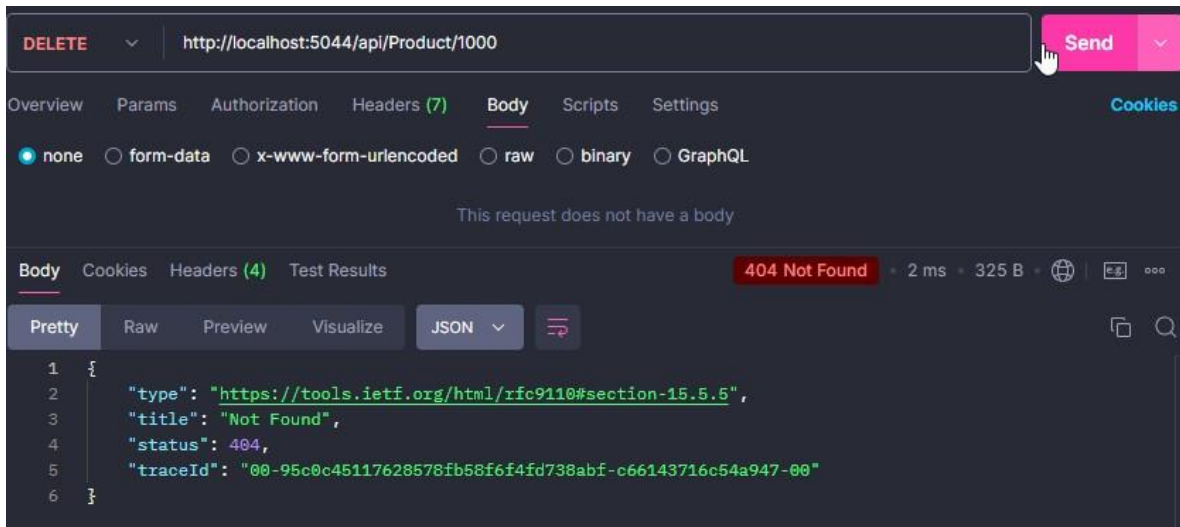
- **PC17- Update a product with missing fields**



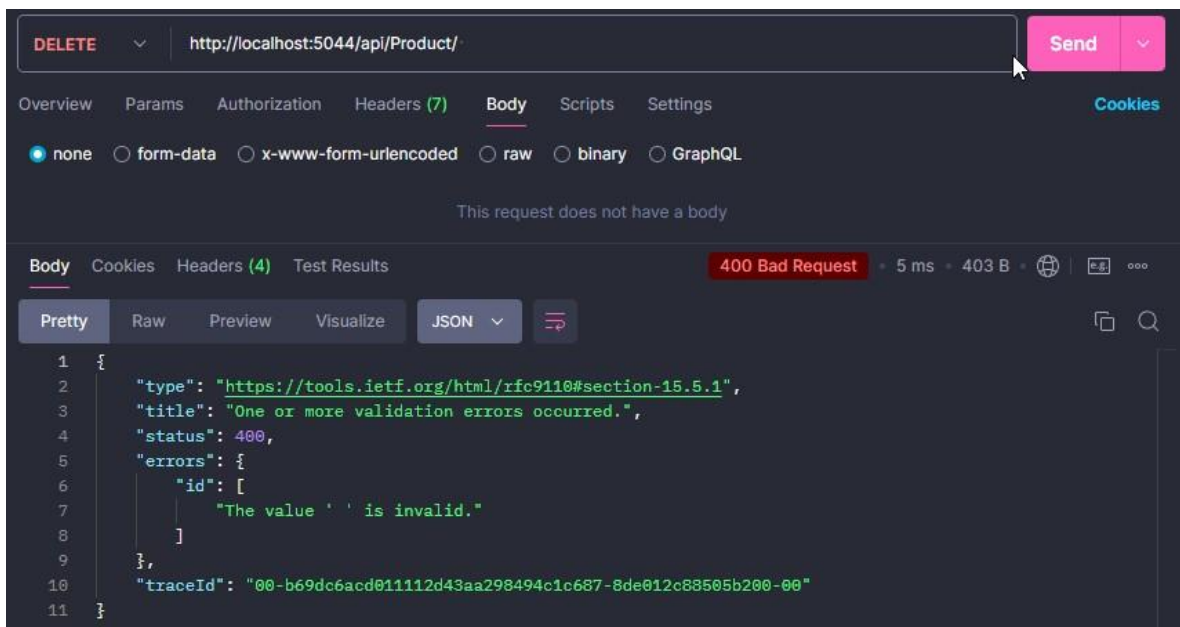
- **PC18- Delete an existing product**



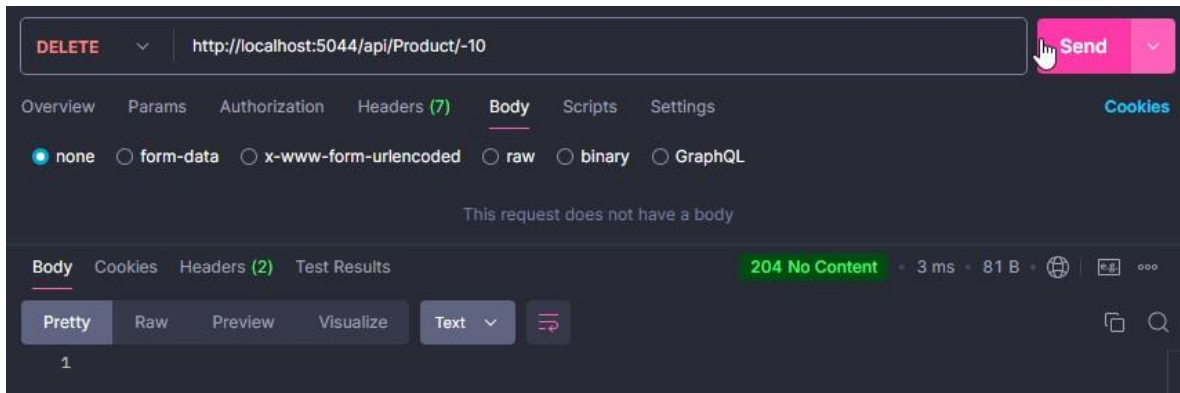
- **PC19- Delete a non-existing product**



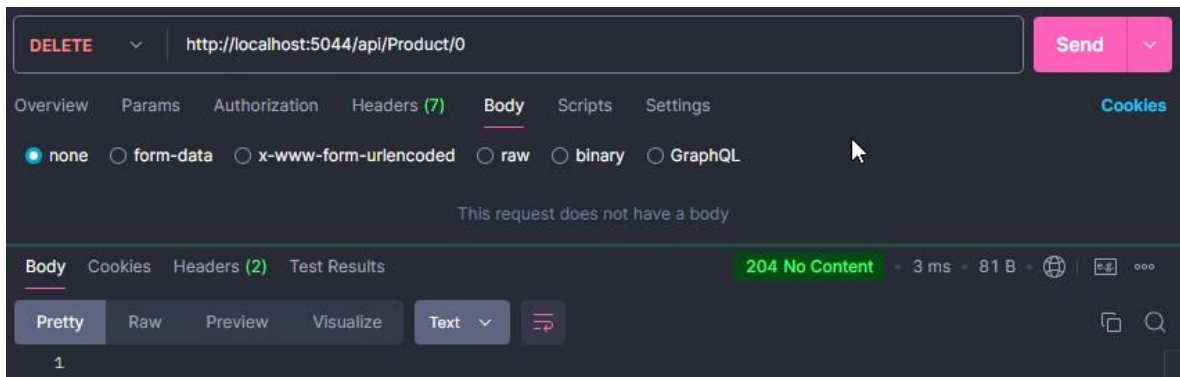
- **PC20- Delete a product without specifying an ID**



- **PC21- Delete a product with invalid negative ID**

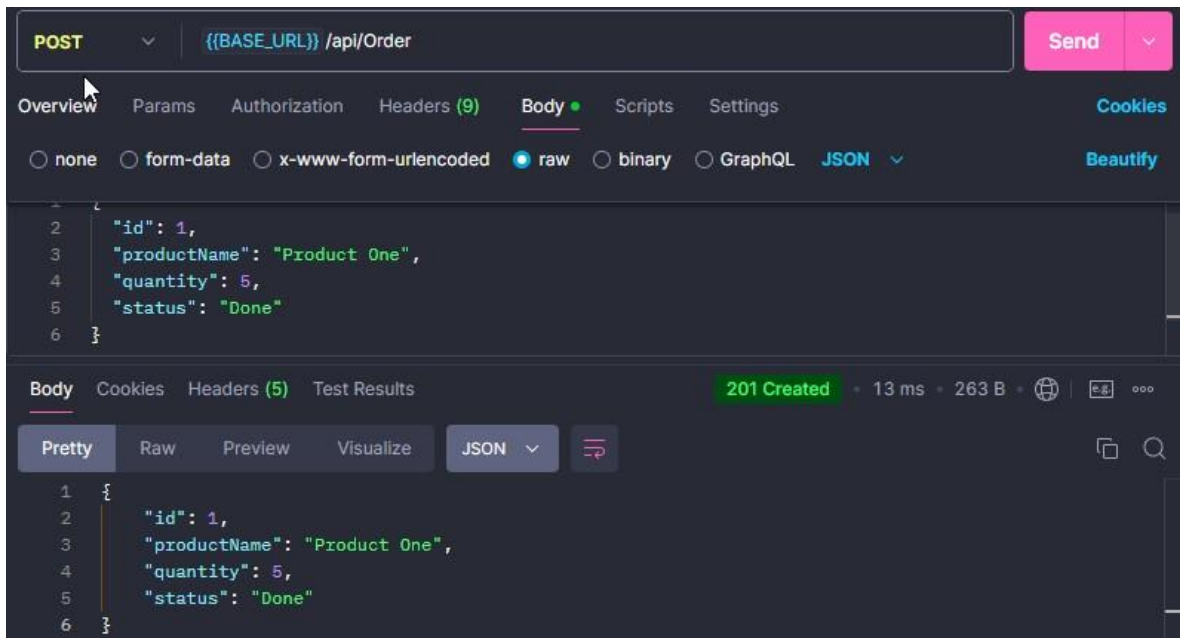


- **PC22- Delete a product with ID set to zero**

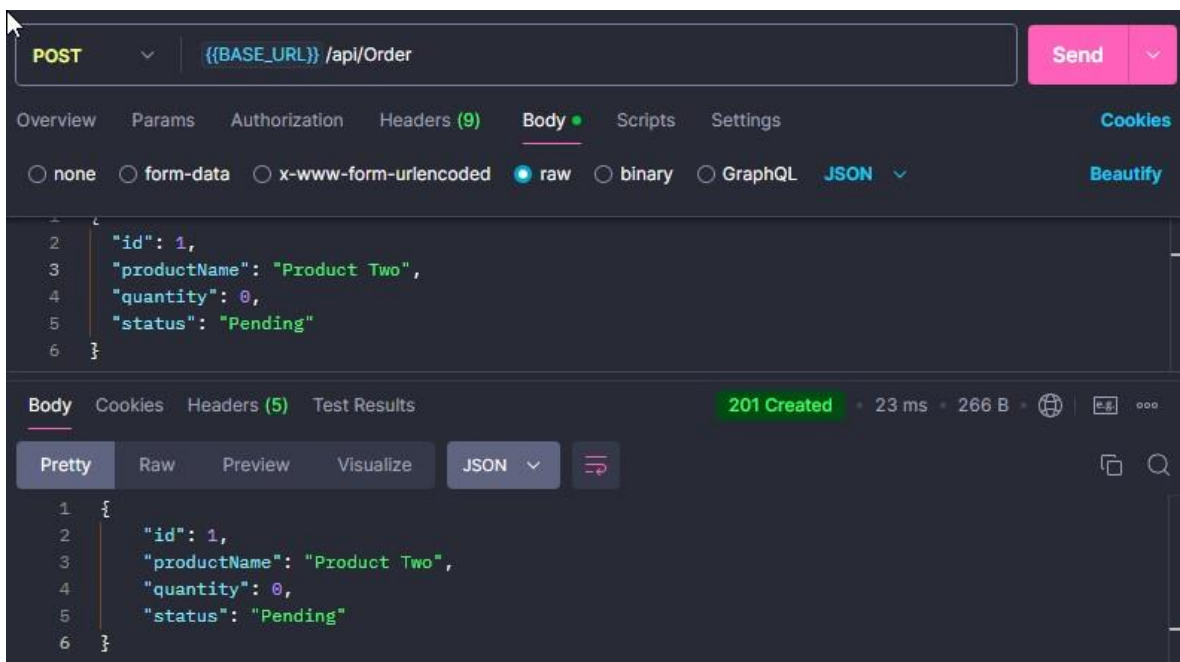


BACKEND (C# APIs): Order Processing: (Create Order- POST)

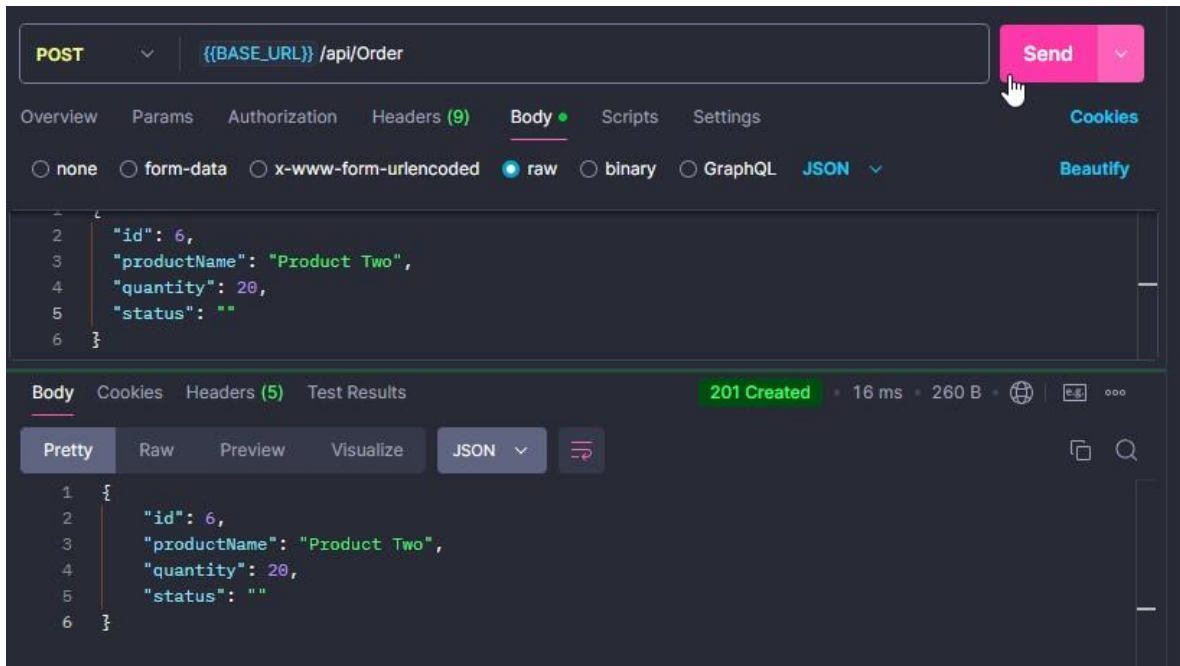
- OC01- Create order with valid data



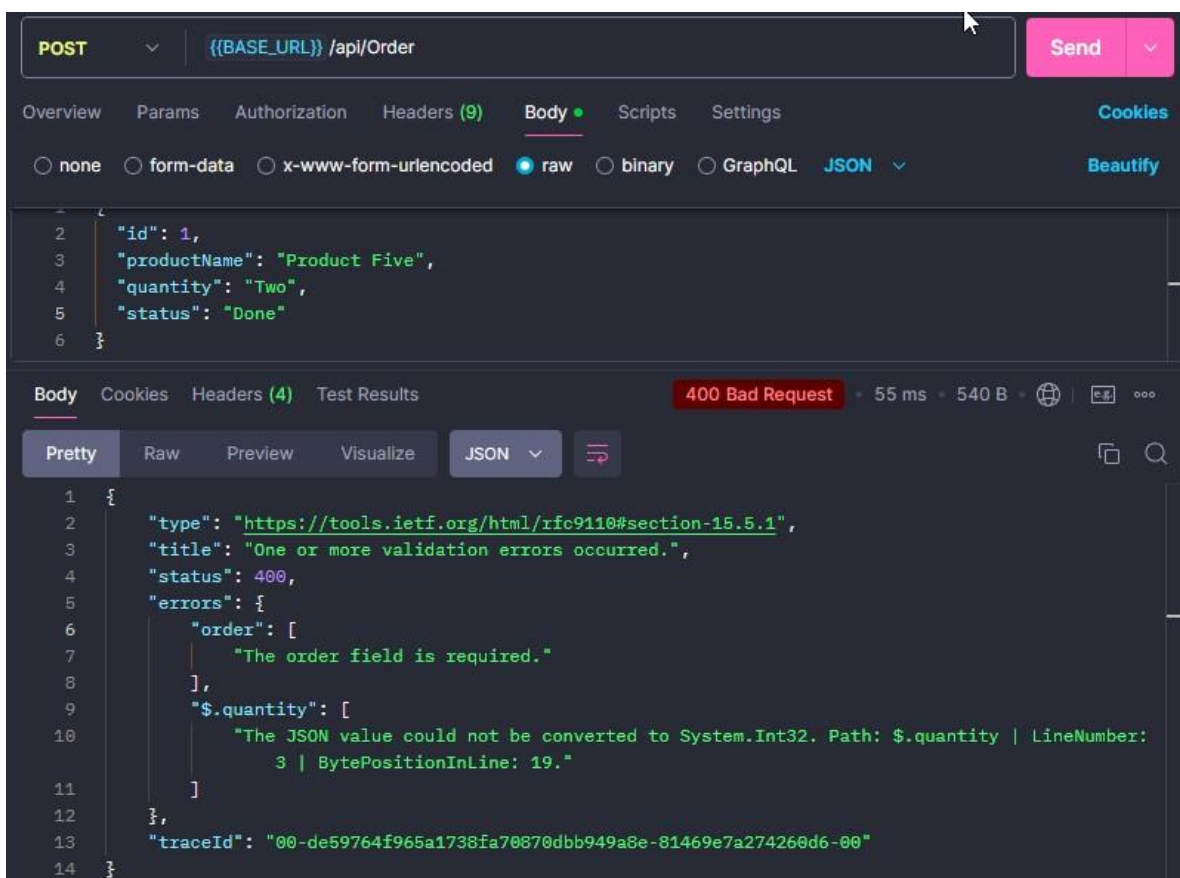
- OC02- Create an order with zero quantity



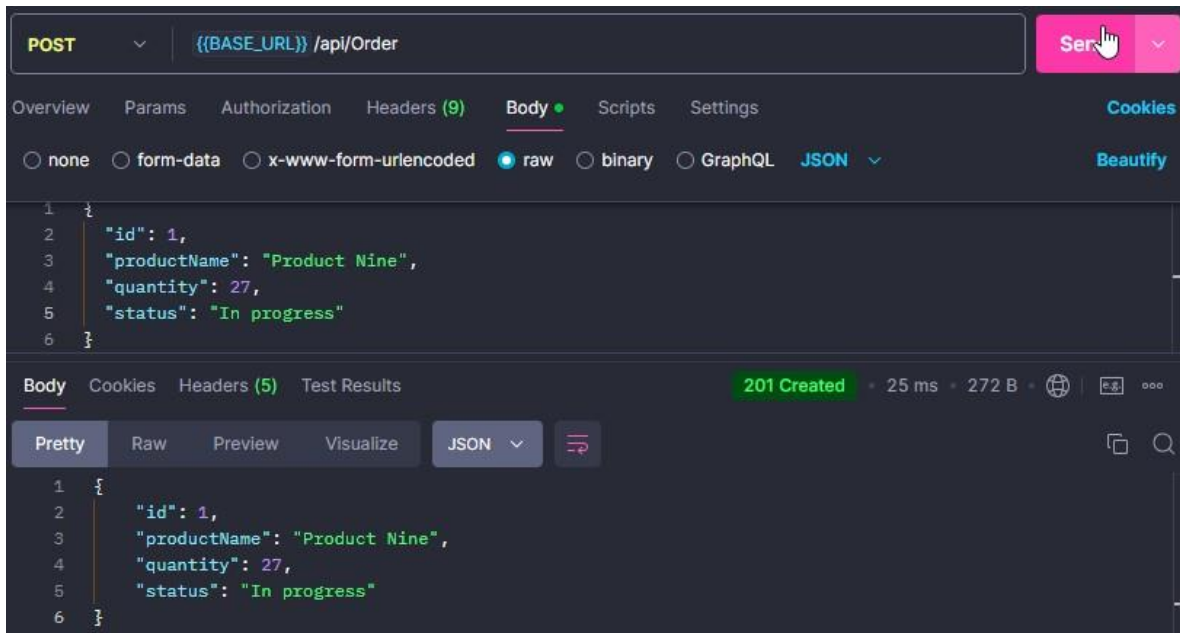
- OC03- Create order with missing status



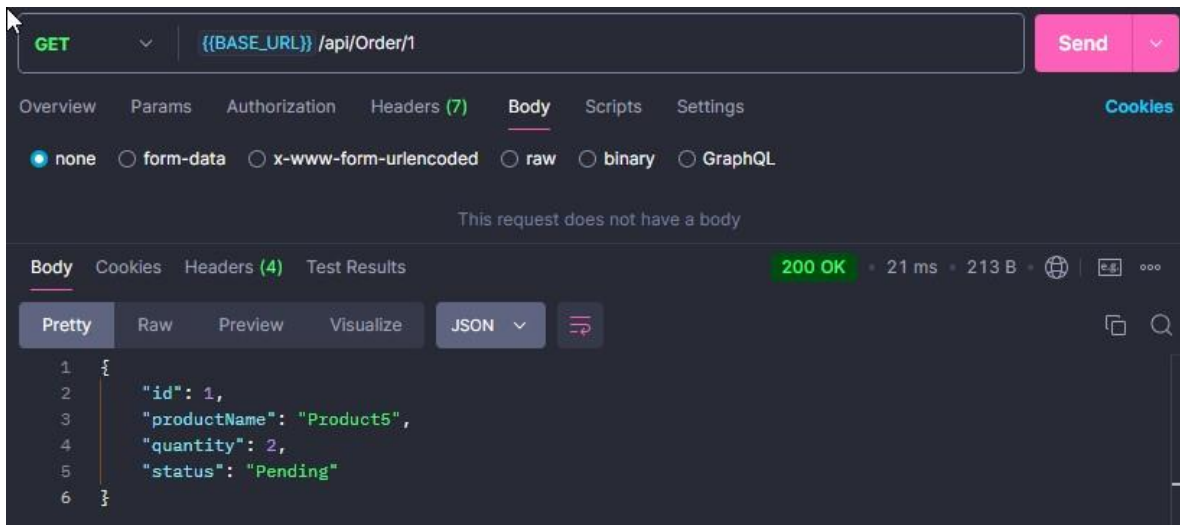
- OC04-Create order with Invalid data types in the fields



- OC05- Create order with an existing order ID



- OC06- Get an existing Order by ID



- OC07- Get Order list

The screenshot shows a REST client interface with a GET request to `{{BASE_URL}} /api/Order/`. The response is a 200 OK status with a response time of 31 ms and a body size of 670 B. The response body is displayed in JSON format, showing a list of two orders:

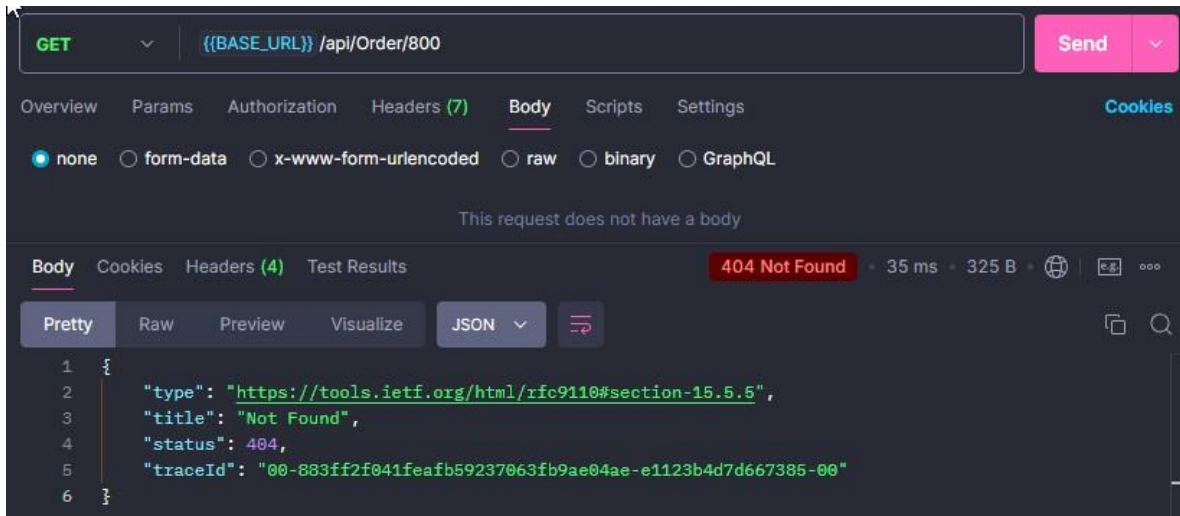
```
1 [
2   {
3     "id": 2,
4     "productName": "Product",
5     "quantity": 0,
6     "status": "Pending"
7   },
8   {
9     "id": 3,
10    "productName": "Product3",
11    "quantity": 5,
12    "status": ""
13  },
14  ]
```

- OC08- Get an order with invalid ID

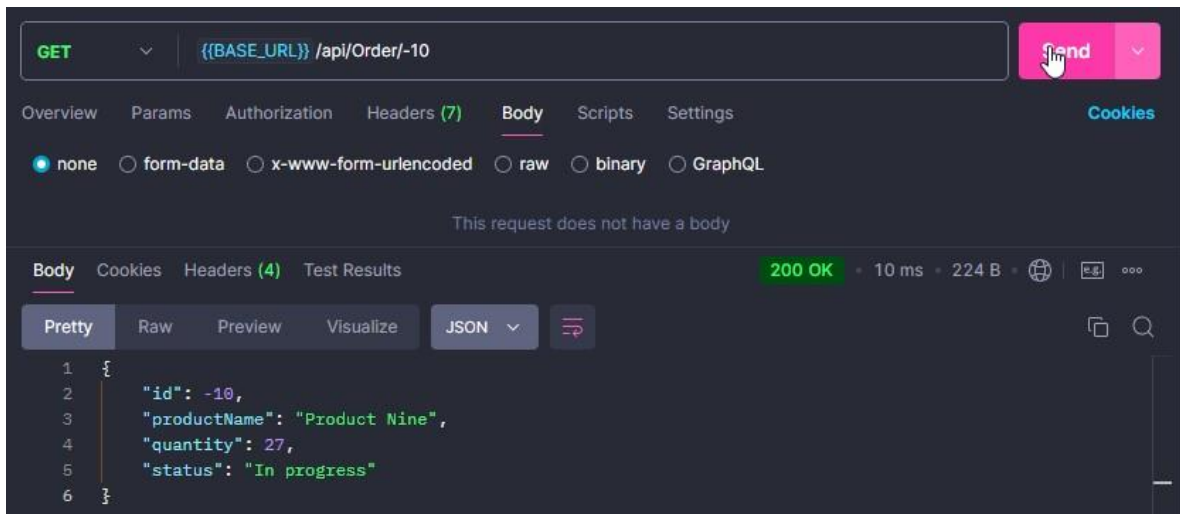
The screenshot shows a REST client interface with a GET request to `{{BASE_URL}} /api/Order/abc`. The response is a 400 Bad Request status with a response time of 88 ms and a body size of 407 B. The response body is displayed in JSON format, indicating a validation error:

```
1 {
2   "type": "https://tools.ietf.org/html/rfc9110#section-15.5.1",
3   "title": "One or more validation errors occurred.",
4   "status": 400,
5   "errors": {
6     "id": [
7       "The value 'abc' is not valid."
8     ]
9   },
10  "traceId": "00-537aa79a938e85091db03658b6980aca-81a8414f3e5cf0be-00"
11 }
```

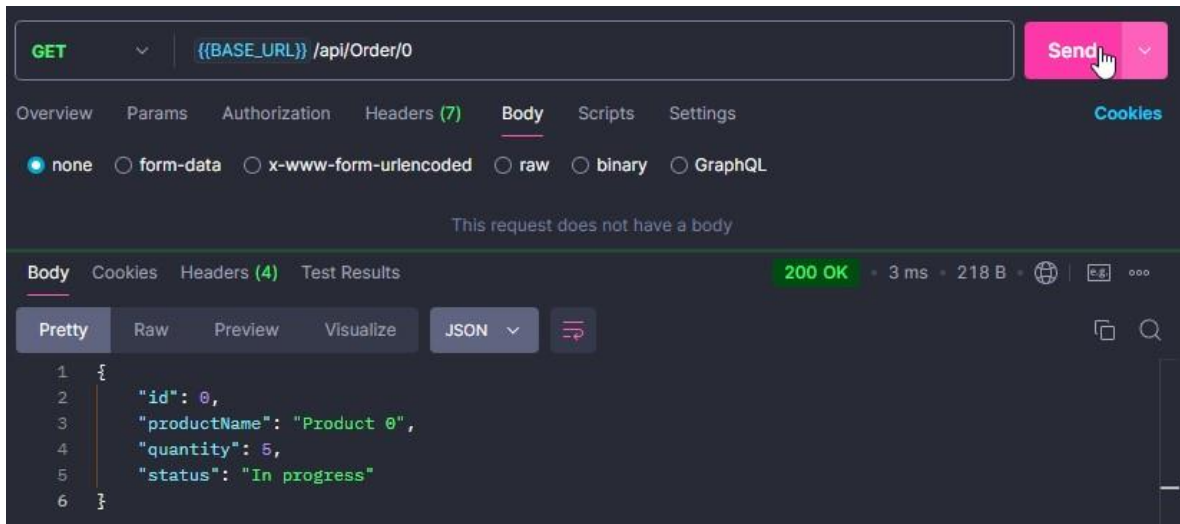
- OC09- Get non-existing order



- OC10 - Get an order with negative id

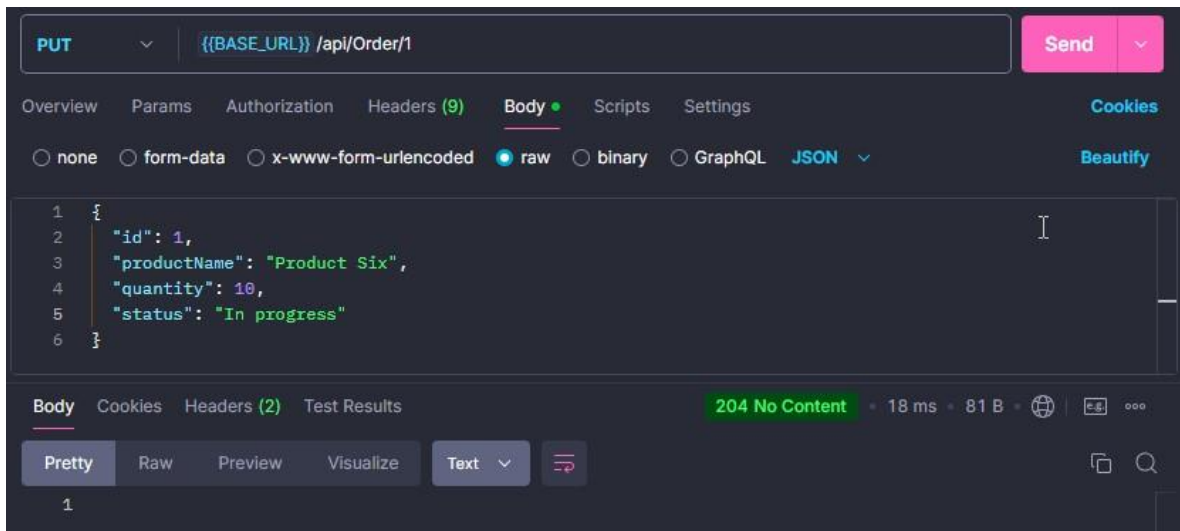


- OC11- Get an order with id=0



BACKEND (C# APIs): Order Processing: (Updated Order- PUT)

- OC12- Update an existing order



- OC13- Update non-existent order

The screenshot shows a REST client interface with a PUT request to `{{BASE_URL}} /api/Order/1000`. The request body is a JSON object:

```
1 {
2   "id": 2000,
3   "productName": "Product Six",
4   "quantity": 10,
5   "status": "In progress"
6 }
```

The response is a **404 Not Found** status with a response time of 30 ms and a body size of 325 B. The response body is a JSON object:

```
1 {
2   "type": "https://tools.ietf.org/html/rfc9110#section-15.5.6",
3   "title": "Not Found",
4   "status": 404,
5   "traceId": "00-359401681ffd3f0b840587d4bc676996-b15d14b06f1c176e-00"
6 }
```

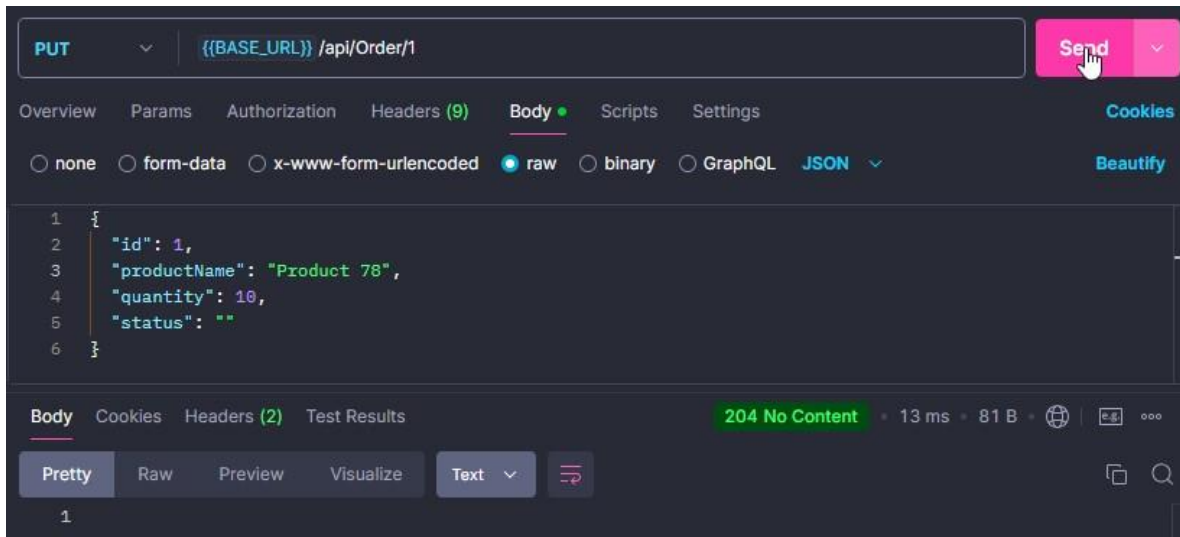
- OC14- Update an order with negative quantity

The screenshot shows a REST client interface with a PUT request to `{{BASE_URL}} /api/Order/1`. The request body is a JSON object:

```
1 {
2   "id": 1,
3   "productName": "Product Six",
4   "quantity": -10,
5   "status": "In progress"
6 }
```

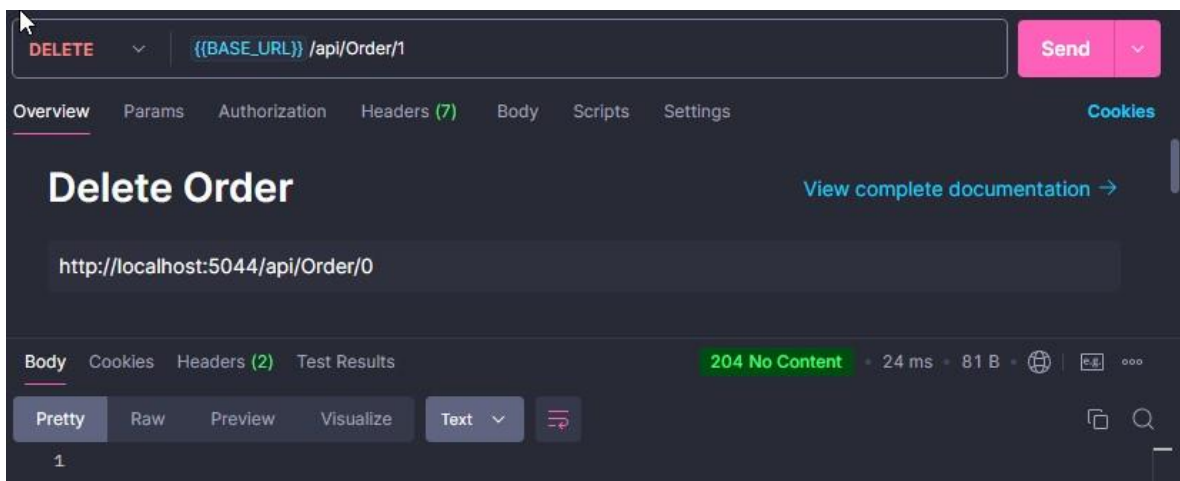
The response is a **204 No Content** status with a response time of 12 ms and a body size of 81 B. The response body is empty.

- OC15- Update an order with missing fields



BACKEND (C# APIs): Order Processing: (Delete Order- DELETE)

- OC16- Delete an existing order



- OC17- Delete a non-existing order

The screenshot shows a REST client interface with a DELETE request to `{{BASE_URL}} /api/Order/1000`. The response is a **404 Not Found** with a status of 3 ms and 325 B. The response body is displayed in JSON format:

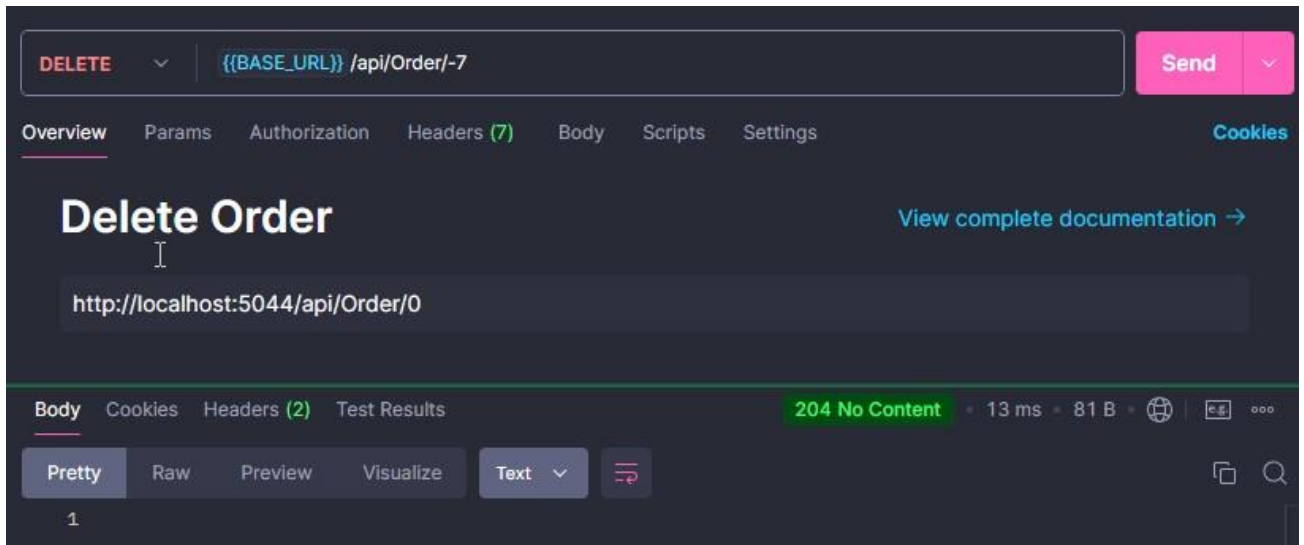
```
1 {
2   "type": "https://tools.ietf.org/html/rfc9110#section-15.5.5",
3   "title": "Not Found",
4   "status": 404,
5   "traceId": "00-482ba38509b7321c474035f3b795d559-09689ff73c5e8d60-00"
6 }
```

- OC18- Delete an order without specifying an ID

The screenshot shows a REST client interface with a DELETE request to `{{BASE_URL}} /api/Order/`. The response is a **400 Bad Request** with a status of 24 ms and 403 B. The response body is displayed in JSON format:

```
1 {
2   "type": "https://tools.ietf.org/html/rfc9110#section-15.5.1",
3   "title": "One or more validation errors occurred.",
4   "status": 400,
5   "errors": {
6     "id": [
7       "The value ' ' is invalid."
8     ]
9   },
10  "traceId": "00-1a621056a80859d951de286e83e6cff5-73a2bf8603823911-00"
11 }
```

- OC19- Delete an order with invalid negative ID



- OC20- Delete an order with ID set to zero

