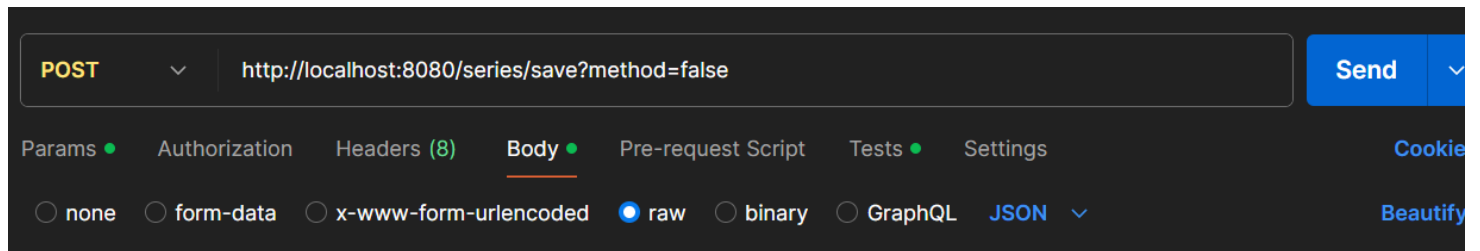


Ejecución Automática de Caso de Test	
Dominio	Proyecto
Aplicación	Ámbito Ejecución
NOMBRE DE LA APLICACIÓN Films	Offline
Descripción	Pre-requisitos
Prueba unitaria para Proyecto Spring Films - Series	
Tester Ejecución	Fecha Ejecución
	09/05/2024

# Paso:	1
Descripción	Create Serie JPA
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 201 Created. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo POST la siguiente URL:

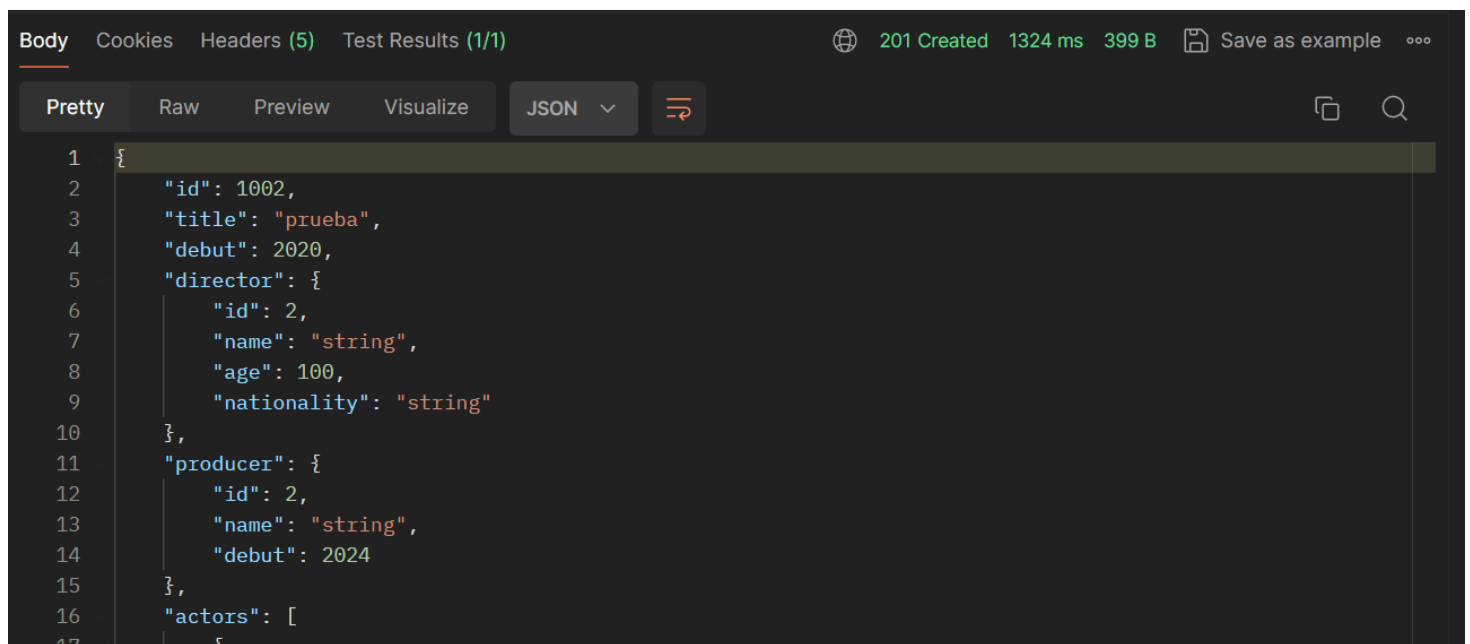
```
http://localhost:8080/series/save?method=false
```



Con el siguiente body

```
{
  "title": "prueba",
  "debut": 2020,
  "director": {
    "id": 2,
    "name": "string",
    "age": 100,
    "nationality": "string"
  },
  "producer": {
    "id": 2,
    "name": "string",
    "debut": 2024
  },
  "actors": [
    {
      "id": 500,
      "name": "string",
      "age": 100,
      "nationality": "string"
    }
  ]
}
```

Revisamos la salida:



The screenshot shows a REST client interface with the following elements:

- Body** tab selected, showing the response body in JSON format.
- JSON** dropdown menu set to **JSON**.
- Save as example** button.
- 201 Created** status, **1324 ms** time, and **399 B** size.
- Test Results (1/1)** tab.
- Headers (5)** tab.
- Cookies** tab.
- Visualize** button.
- Preview** button.
- Raw** button.
- Pretty** button.
- JSON** dropdown menu.
- Copy** button.
- Search** button.

```
1 {
2   "id": 1002,
3   "title": "prueba",
4   "debut": 2020,
5   "director": {
6     "id": 2,
7     "name": "string",
8     "age": 100,
9     "nationality": "string"
10  },
11  "producer": {
12    "id": 2,
13    "name": "string",
14    "debut": 2024
15  },
16  "actors": [
17    {
```

# Paso:	2
Descripción	Create Serie JPA Error
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 400 Bad Request. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo POST la siguiente URL:

http://localhost:8080/series/save?method=false

POST

⌵

http://localhost:8080/series/save?method=false

Send

⌵

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

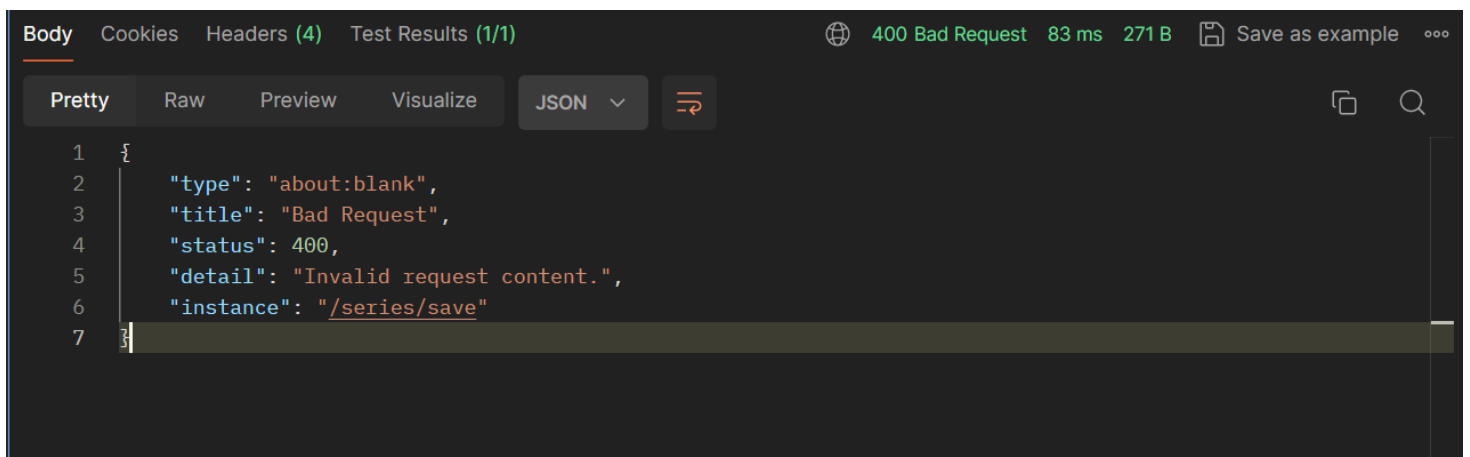
⌵

Beautify

Con el siguiente body

```
{
  "title": "prueba",
  "debut": 3000,
  "director": {
    "id": 2,
    "name": "string",
    "age": 100,
    "nationality": "string"
  },
  "producer": {
    "id": 2,
    "name": "string",
    "debut": 2024
  },
  "actors": [
    {
      "id": 500,
      "name": "string",
      "age": 100,
      "nationality": "string"
    }
  ]
}
```

Revisamos la salida:



The screenshot shows a REST client interface with the following details:

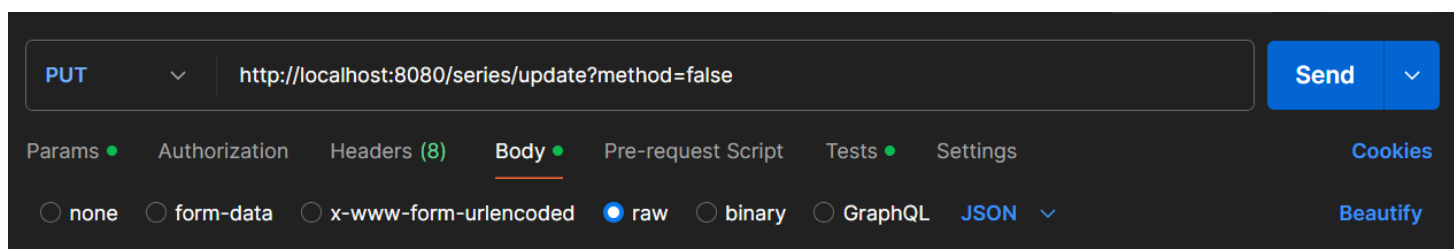
- Body** tab selected.
- Headers (4)** and **Test Results (1/1)** tabs are visible.
- Test Results (1/1)** shows a **400 Bad Request** status, **83 ms** response time, and **271 B** body size.
- Save as example** button is present.
- JSON** tab is selected for the response body.
- The response body is displayed in a code editor with line numbers 1 through 7.

```
1 {
2   "type": "about:blank",
3   "title": "Bad Request",
4   "status": 400,
5   "detail": "Invalid request content.",
6   "instance": "/series/save"
7 }
```

# Paso:	3
Descripción	Update Serie JPA
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 200 OK. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo PUT la siguiente URL:

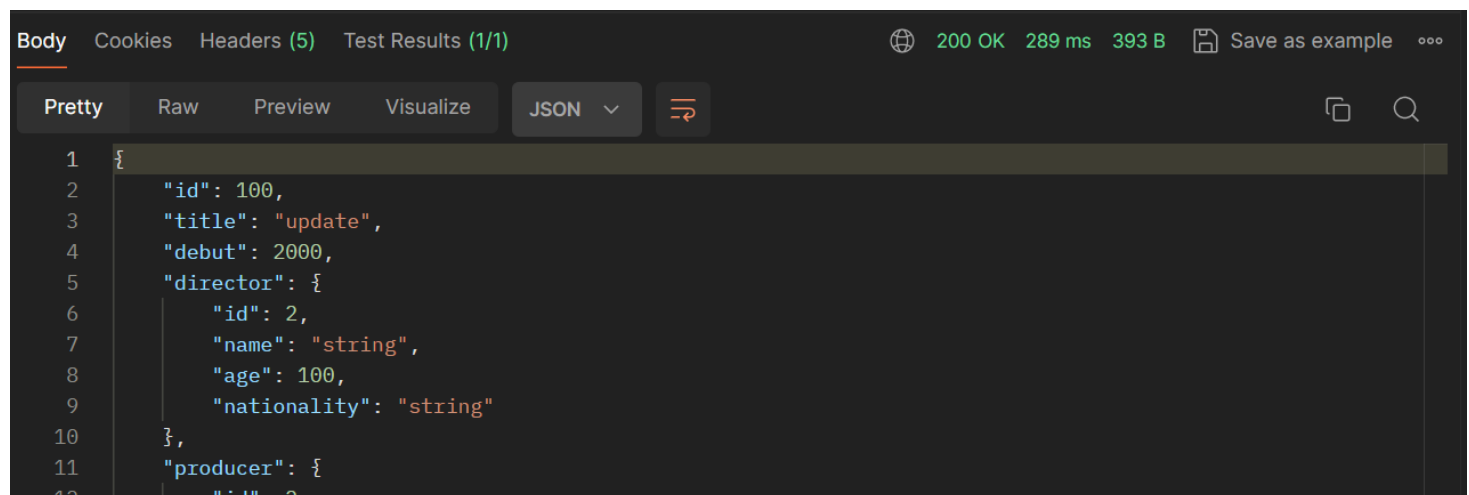
```
http://localhost:8080/series/update?method=false
```



Con el siguiente body

```
{
  "id": 100,
  "title": "update",
  "debut": 2000,
  "director": {
    "id": 2,
    "name": "string",
    "age": 100,
    "nationality": "string"
  },
  "producer": {
    "id": 2,
    "name": "string",
    "debut": 2024
  },
  "actors": [
    {
      "id": 500,
      "name": "string",
      "age": 100,
      "nationality": "string"
    }
  ]
}
```

Revisamos la salida:



Body Cookies Headers (5) Test Results (1/1) 200 OK 289 ms 393 B Save as example

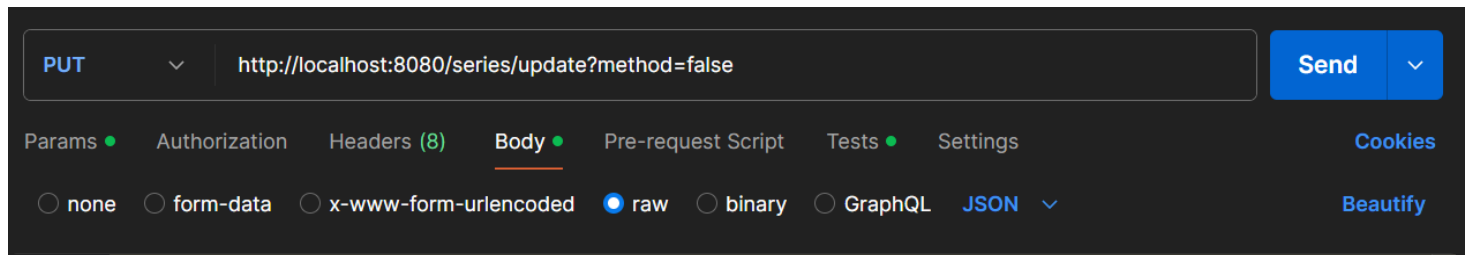
Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 100,
3   "title": "update",
4   "debut": 2000,
5   "director": {
6     "id": 2,
7     "name": "string",
8     "age": 100,
9     "nationality": "string"
10  },
11  "producer": {
12    "id": 2
```

# Paso:	4
Descripción	Update Serie JPA Error
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 400 Bad Request. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo PUT la siguiente URL:

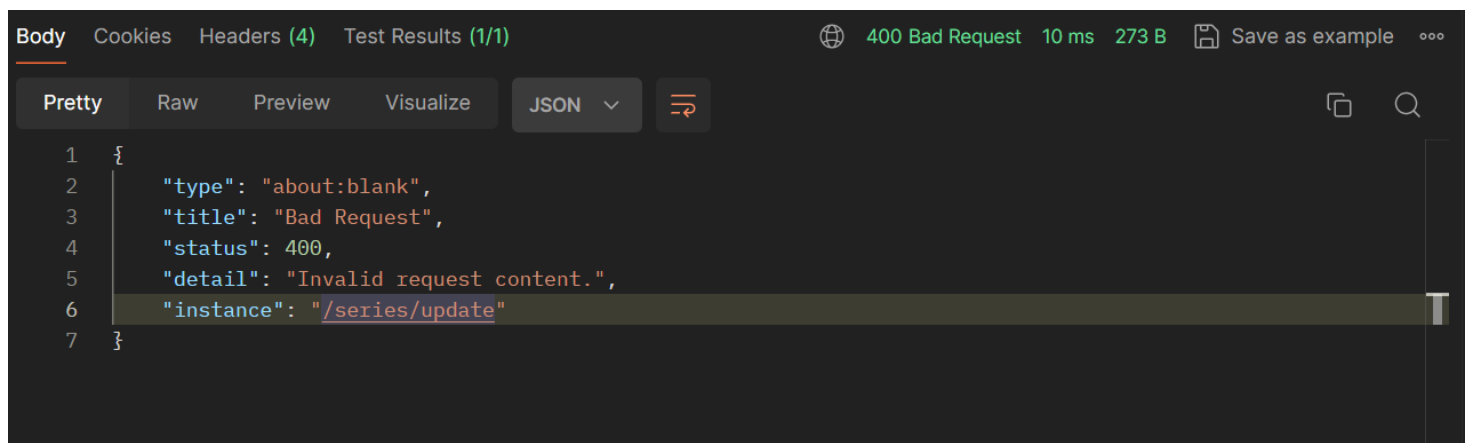
http://localhost:8080/series/update?method=false



Con el siguiente body

```
{
  "id": 100,
  "title": "update",
  "debut": 3000,
  "director": {
    "id": 2,
    "name": "string",
    "age": 100,
    "nationality": "string"
  },
  "producer": {
    "id": 2,
    "name": "string",
    "debut": 2024
  },
  "actors": [
    {
      "id": 500,
      "name": "string",
      "age": 100,
      "nationality": "string"
    }
  ]
}
```

Revisamos la salida:



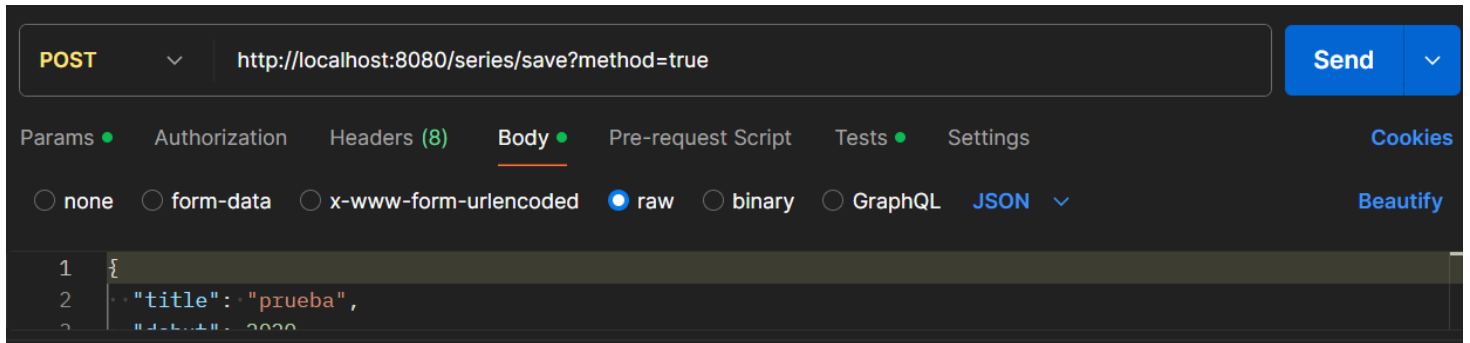
The screenshot shows a web browser's developer console with the 'Body' tab selected. The console displays a 400 Bad Request error. The error message is a JSON object:

```
{
  "type": "about:blank",
  "title": "Bad Request",
  "status": 400,
  "detail": "Invalid request content.",
  "instance": "/series/update"
}
```

# Paso:	5
Descripción	Create Serie Criteria
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 201 Created. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo POST la siguiente URL:

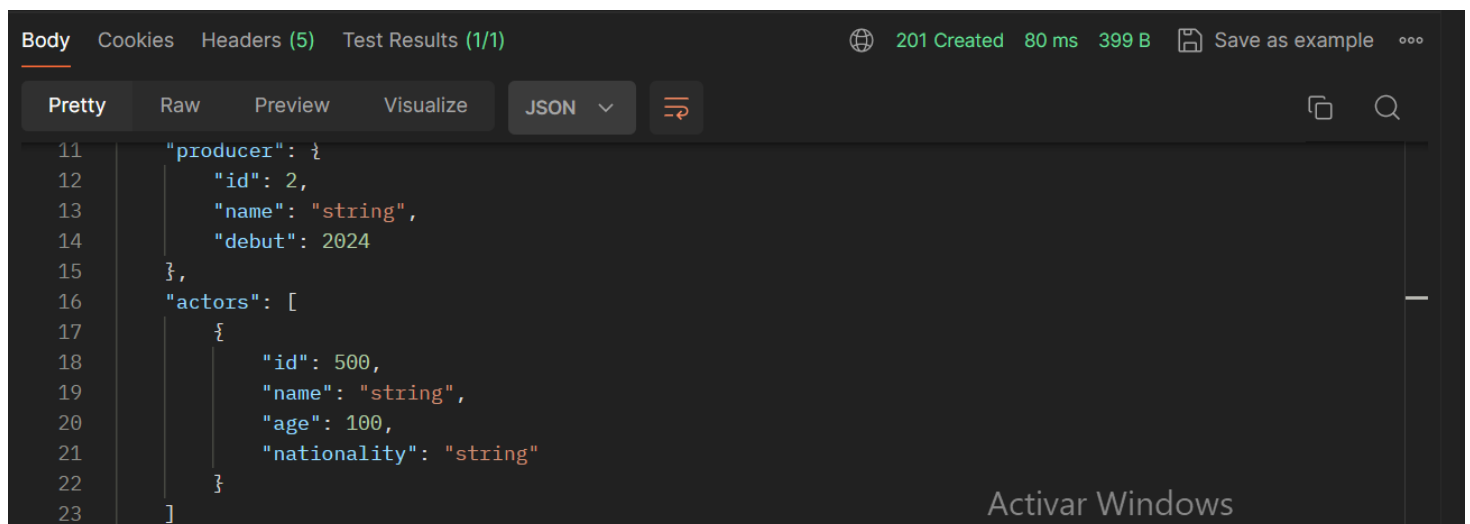
http://localhost:8080/series/save?method=true



Con el siguiente body

```
{
  "title": "prueba",
  "debut": 2020,
  "director": {
    "id": 2,
    "name": "string",
    "age": 100,
    "nationality": "string"
  },
  "producer": {
    "id": 2,
    "name": "string",
    "debut": 2024
  },
  "actors": [
    {
      "id": 500,
      "name": "string",
      "age": 100,
      "nationality": "string"
    }
  ]
}
```

Revisamos la salida:



The screenshot shows a REST client interface with the following elements:

- Body** tab selected, showing the JSON response.
- Headers (5)** and **Test Results (1/1)** tabs are visible.
- 201 Created** status, **80 ms** time, and **399 B** size are displayed.
- Save as example** button and a menu icon (three dots) are present.
- Pretty** tab selected, showing the JSON response in a formatted view.
- Raw**, **Preview**, and **Visualize** tabs are visible.
- JSON** dropdown menu and a refresh icon are present.
- Activar Windows** watermark is visible in the bottom right corner.

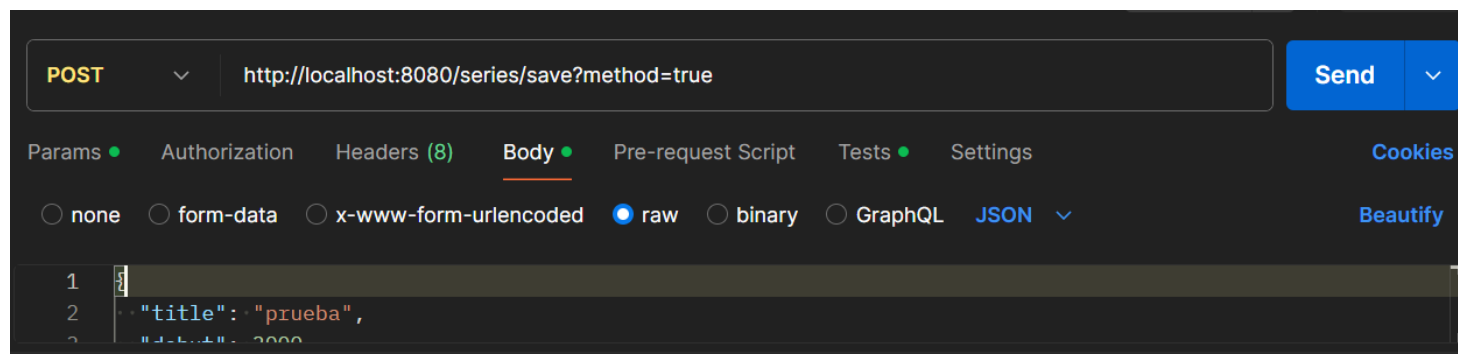
The JSON response body is:

```
11  "producer": {
12    "id": 2,
13    "name": "string",
14    "debut": 2024
15  },
16  "actors": [
17    {
18      "id": 500,
19      "name": "string",
20      "age": 100,
21      "nationality": "string"
22    }
23  ]
```

# Paso:	6
Descripción	Create Serie Criteria Error
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 400 Bad Request. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo POST la siguiente URL:

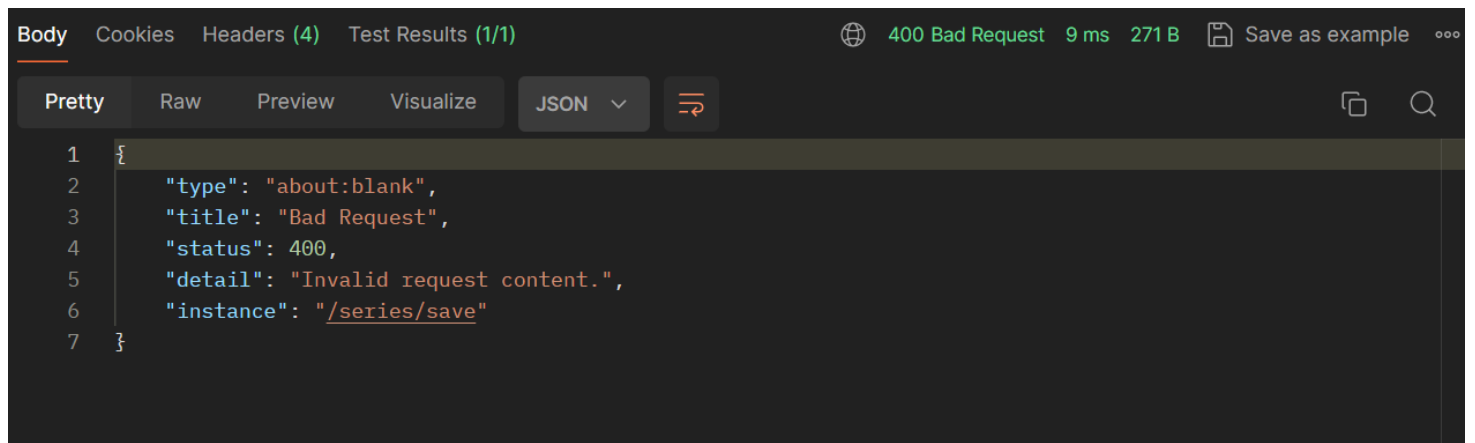
http://localhost:8080/series/save?method=true



Con el siguiente body

```
{
  "title": "prueba",
  "debut": 3000,
  "director": {
    "id": 2,
    "name": "string",
    "age": 100,
    "nationality": "string"
  },
  "producer": {
    "id": 2,
    "name": "string",
    "debut": 2024
  },
  "actors": [
    {
      "id": 500,
      "name": "string",
      "age": 100,
      "nationality": "string"
    }
  ]
}
```

Revisamos la salida:



The screenshot shows a REST client interface with the following details:

- Body** tab selected.
- Test Results (1/1)** shows a **400 Bad Request** status.
- Response time:** 9 ms
- Response size:** 271 B
- Buttons:** Save as example, Copy, Search.
- JSON View:** Pretty, Raw, Preview, Visualize, JSON (selected), and a refresh button.
- Response Body:**

```
1 {
2   "type": "about:blank",
3   "title": "Bad Request",
4   "status": 400,
5   "detail": "Invalid request content.",
6   "instance": "/series/save"
7 }
```

# Paso:	7
Descripción	update Serie Criteria
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 200 OK. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo PUT la siguiente URL:

http://localhost:8080/series/update?method=true

PUT

http://localhost:8080/series/update?method=true

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

{

2

"id": 100,

3

"title": "The Shawshank Redemption"

Con el siguiente body

```
{
  "id": 100,
  "title": "update",
  "debut": 2000,
  "director": {
    "id": 2,
    "name": "string",
    "age": 100,
    "nationality": "string"
  },
  "producer": {
    "id": 2,
    "name": "string",
    "debut": 2024
  },
  "actors": [
    {
      "id": 500,
      "name": "string",
      "age": 100,
      "nationality": "string"
    }
  ]
}
```

Revisamos la salida:



Body Cookies Headers (5) Test Results (1/1) 200 OK 38 ms 393 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 100,
3   "title": "update",
4   "debut": 2000,
5   "director": {
6     "id": 2,
7     "name": "string",
8     "age": 100,
9     "nationality": "string"
10  },
11  "producer": {
12    "id": 2,
13    "name": "string"
```

Activar Windows

# Paso:	8
Descripción	update Serie Criteria Error
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 400 Bad Request. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo PUT la siguiente URL:

http://localhost:8080/series/update?method=true

PUT

http://localhost:8080/series/update?method=true

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

{

2

"id": 100,

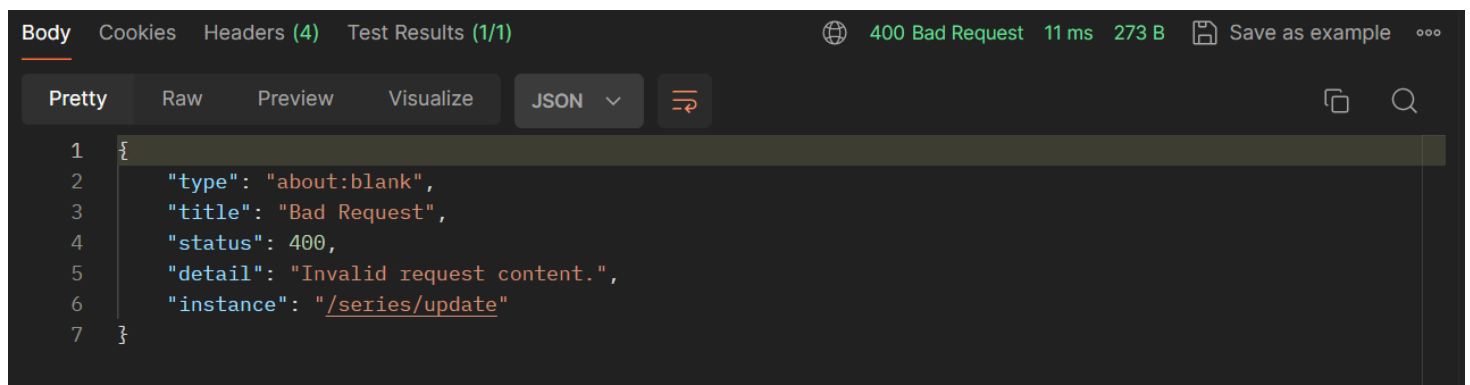
3

"title": "The Godfather",

Con el siguiente body

```
{
  "id": 100,
  "title": "update",
  "debut": 3000,
  "director": {
    "id": 2,
    "name": "string",
    "age": 100,
    "nationality": "string"
  },
  "producer": {
    "id": 2,
    "name": "string",
    "debut": 2024
  },
  "actors": [
    {
      "id": 500,
      "name": "string",
      "age": 100,
      "nationality": "string"
    }
  ]
}
```

Revisamos la salida:

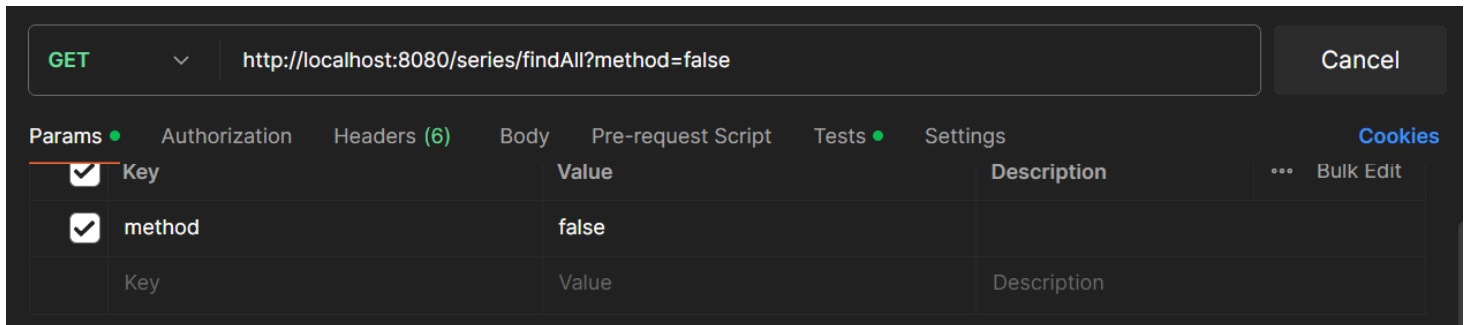


```
1 {
2   "type": "about:blank",
3   "title": "Bad Request",
4   "status": 400,
5   "detail": "Invalid request content.",
6   "instance": "/series/update"
7 }
```

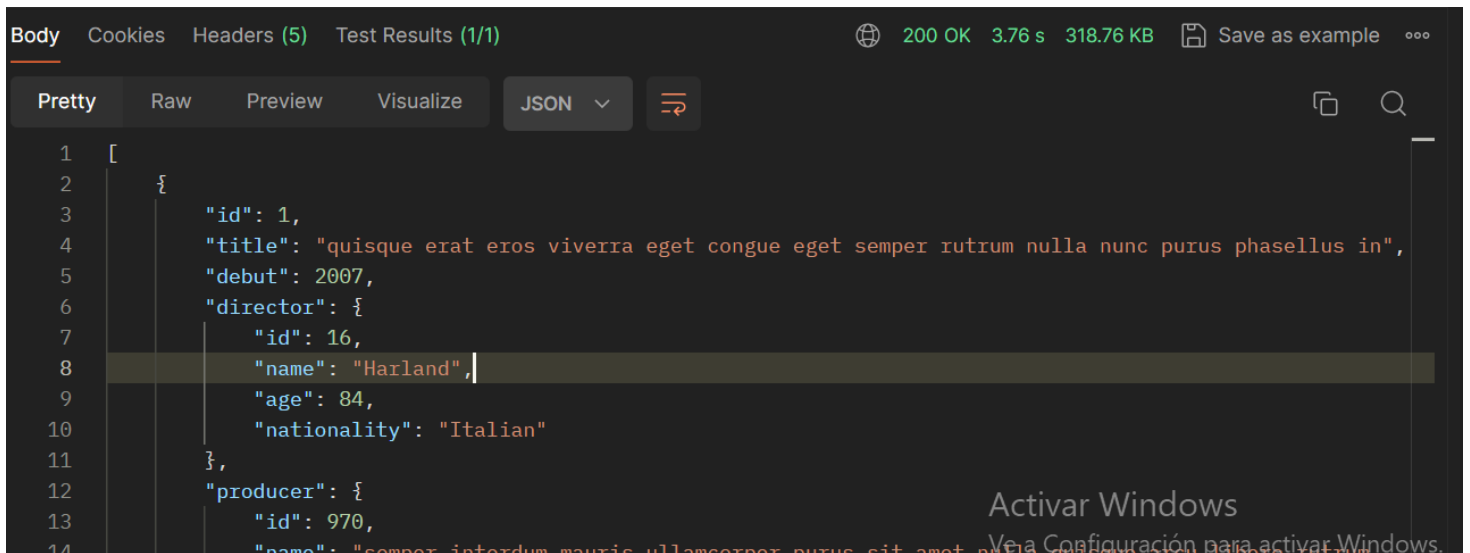
# Paso:	9
Descripción	Get All Series JPA
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 200 OK. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo GET la siguiente URL:

http://localhost:8080/series/findAll?method=false



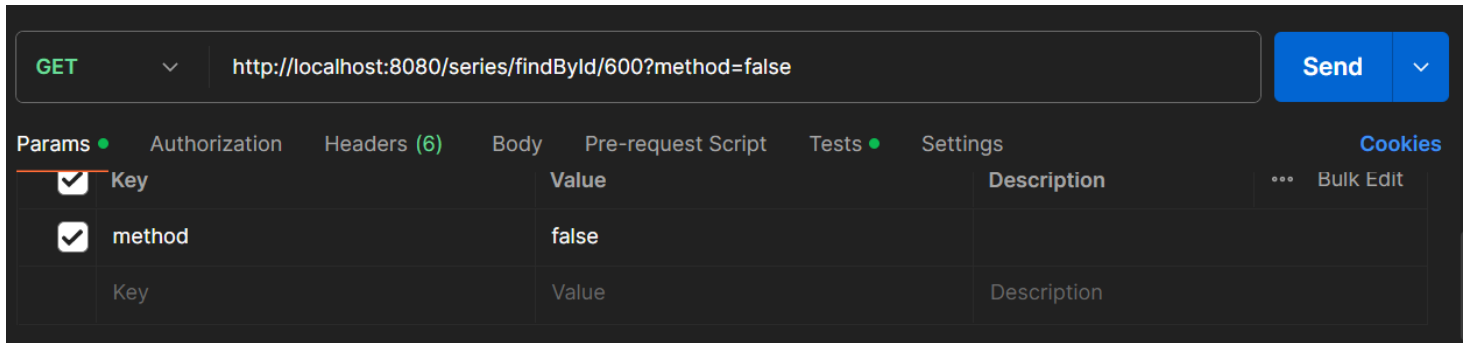
Revisamos la salida:



# Paso:	10
Descripción	Get A Serie JPA
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 200 OK. La operación en la base de datos se completa correctamente

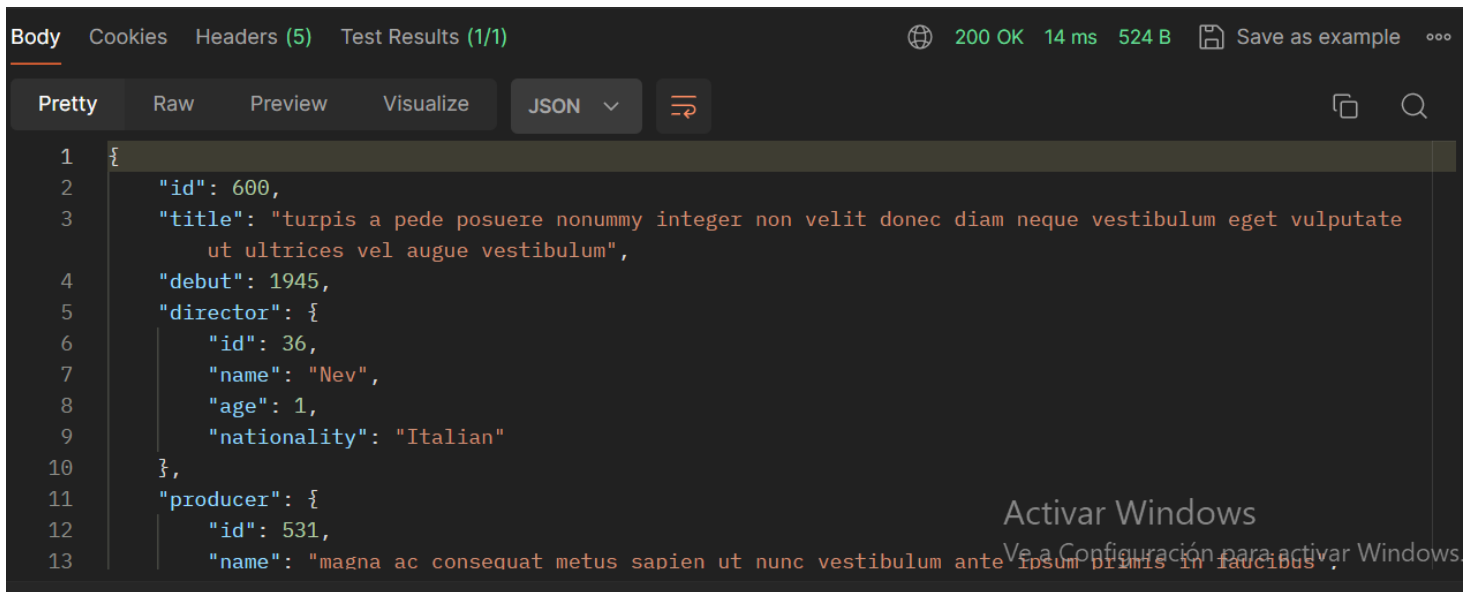
Realizamos una petición REST de tipo GET la siguiente URL:

http://localhost:8080/series/findById/600?method=false



Key	Value	Description
method	false	

Revisamos la salida:



```

1 {
2   "id": 600,
3   "title": "turpis a pede posuere nonummy integer non velit donec diam neque vestibulum eget vulputate
4     ut ultrices vel augue vestibulum",
5   "debut": 1945,
6   "director": {
7     "id": 36,
8     "name": "Nev",
9     "age": 1,
10    "nationality": "Italian"
11  },
12  "producer": {
13    "id": 531,
14    "name": "magna ac consequat metus sapien ut nunc vestibulum ante ipsum primis in faucibus"
15  }
16 }
```

# Paso:	11
Descripción	Get A Serie JPA Error
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 404 Not Found. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo GET la siguiente URL:

```
http://localhost:8080/series/findById/2000?method=false
```

GET

http://localhost:8080/series/findById/2000?method=false

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Key	Value	Description
method	false	
Key	Value	Description

Revisamos la salida:

Body

Cookies

Headers (5)

Test Results (1/1)

404 Not Found

19 ms

234 B

Save as example

Pretty

Raw

Preview

Visualize

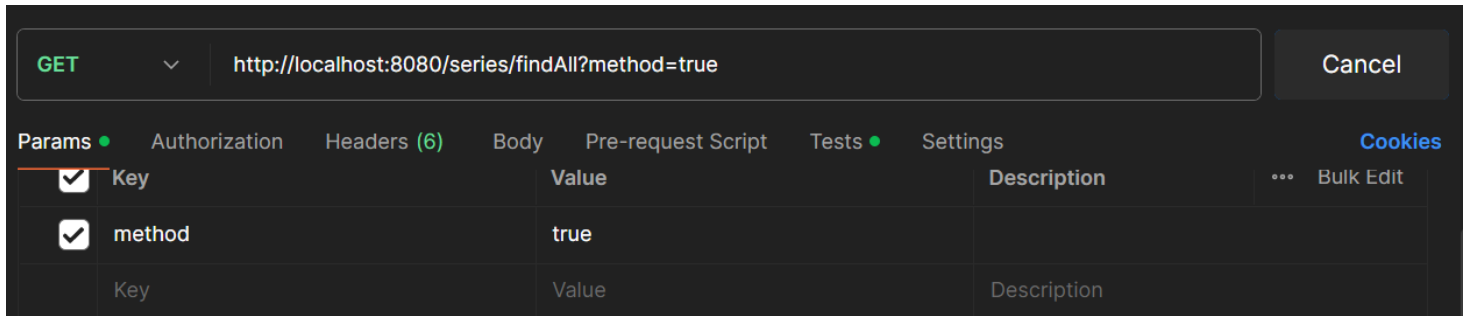
JSON

```
1 {
2   "date": "09/05/2024 16:29:56",
3   "message": "Production not found"
4 }
```

# Paso:	12
Descripción	Get All Series Criteria
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 200 OK. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo GET la siguiente URL:

http://localhost:8080/series/findAll?method=true



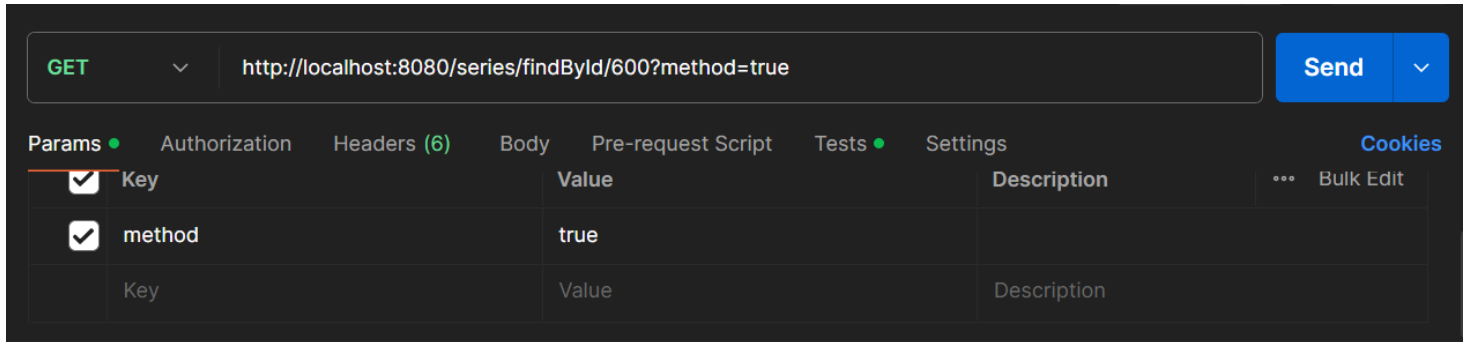
Revisamos la salida:



# Paso:	13
Descripción	Get A Series Criteria
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 200 OK. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo GET la siguiente URL:

http://localhost:8080/series/findById/600?method=true



Key	Value	Description
method	true	
Key	Value	Description

Revisamos la salida:



```

1 {
2   "id": 600,
3   "title": "turpis a pede posuere nonummy integer non velit donec diam neque vestibulum eget vulputate
         ut ultrices vel augue vestibulum",
4   "debut": 1945,
5   "director": {
6     "id": 36,
7     "name": "Nev",
8     "age": 1,
9     "nationality": "Italian"
10  },
11  "producer": {

```

# Paso:	14
Descripción	Get A Series Criteria Error
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 404 Not Found. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo GET la siguiente URL:

http://localhost:8080/series/findById/2000?method=true

GET

http://localhost:8080/series/findById/2000?method=true

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	method	true			
	Key	Value	Description		

Revisamos la salida:

Body

Cookies

Headers (5)

Test Results (1/1)

404 Not Found

8 ms

234 B

Save as example

Pretty

Raw

Preview

Visualize

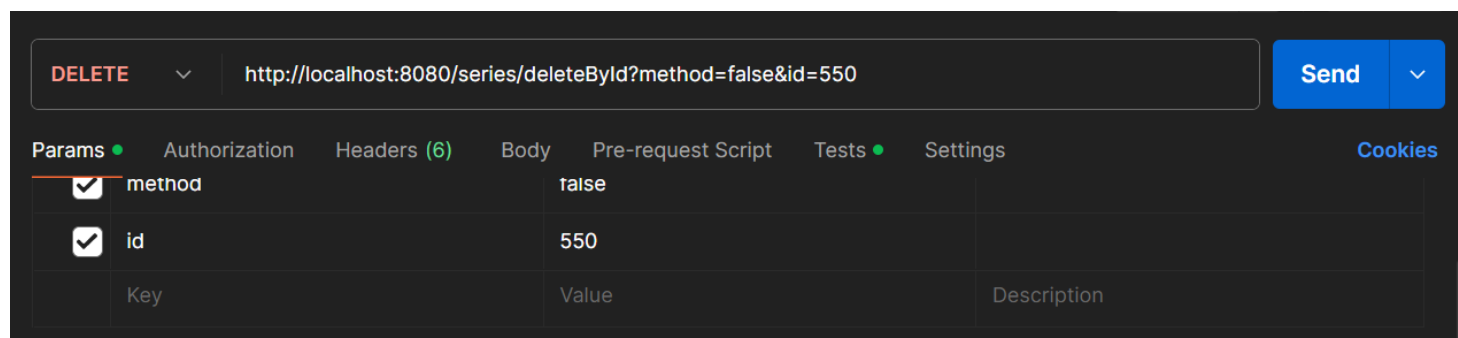
JSON

```
1 {
2   "date": "09/05/2024 16:31:45",
3   "message": "Production not found"
4 }
```

# Paso:	15
Descripción	Delete A Serie JPA
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 204 No Content. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo DELETE la siguiente URL:

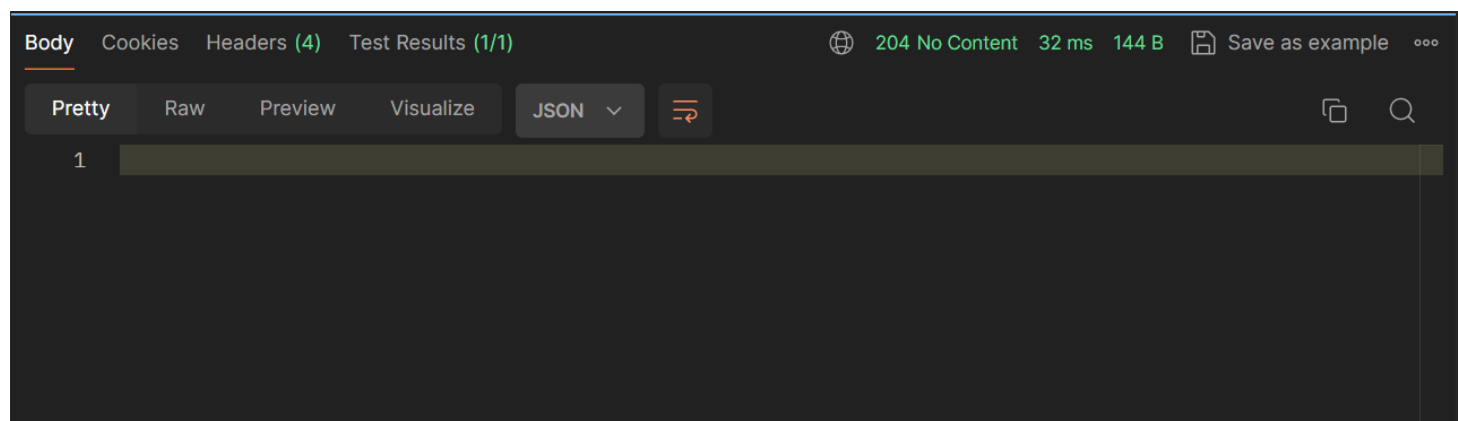
```
http://localhost:8080/series/deleteById?method=false&id=550
```



The screenshot shows a REST client interface with the following configuration:

- Method:** DELETE
- URL:** http://localhost:8080/series/deleteById?method=false&id=550
- Params:**
 - ☒ method: false
 - ☒ id: 550
- Headers:** 6 (not expanded)
- Body:** (empty)
- Pre-request Script:** (empty)
- Tests:** (empty)
- Settings:** (empty)
- Cookies:** (empty)

Revisamos la salida:



The screenshot shows the response of the DELETE request in a REST client interface:

- Status:** 204 No Content
- Time:** 32 ms
- Size:** 144 B
- Body:** (empty, as expected for a 204 response)
- Headers:** 4 (not expanded)
- Test Results:** 1/1 (all tests passed)

# Paso:	16
Descripción	Delete A Serie JPA Error
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 404 Not Found. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo DELETE la siguiente URL:

```
http://localhost:8080/series/deleteById?method=false&id=2000
```

DELETE

http://localhost:8080/series/deleteById?method=false&id=2000

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

<input checked="" type="checkbox"/>	method	false	
<input checked="" type="checkbox"/>	id	2000	
	Key	Value	Description

Revisamos la salida:

Body

Cookies

Headers (5)

Test Results (1/1)

404 Not Found

9 ms

234 B

Save as example

Pretty

Raw

Preview

Visualize

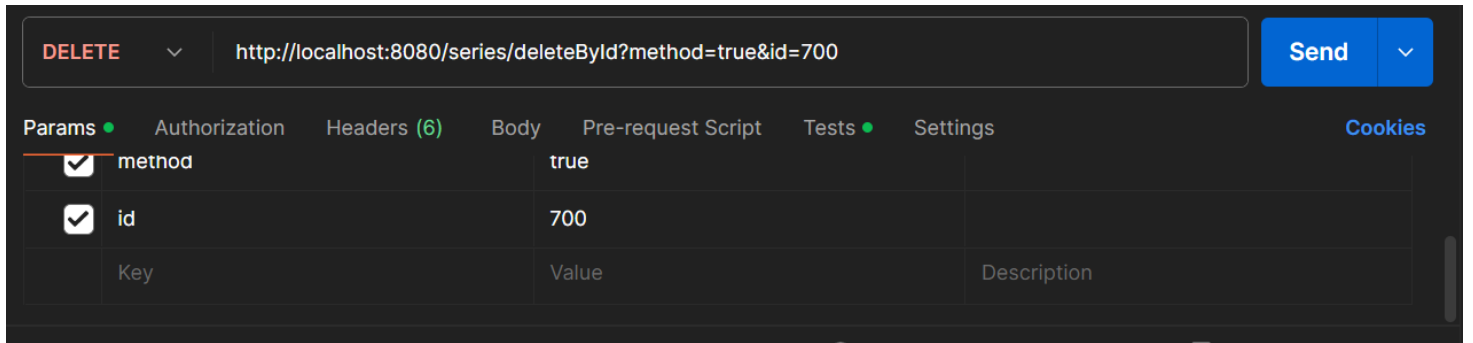
JSON

```
1 {
2   "date": "09/05/2024 16:32:26",
3   "message": "Production not found"
4 }
```

Paso:	17
Descripción	Delete A Serie Criteria
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 204 No Content. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo DELETE la siguiente URL:

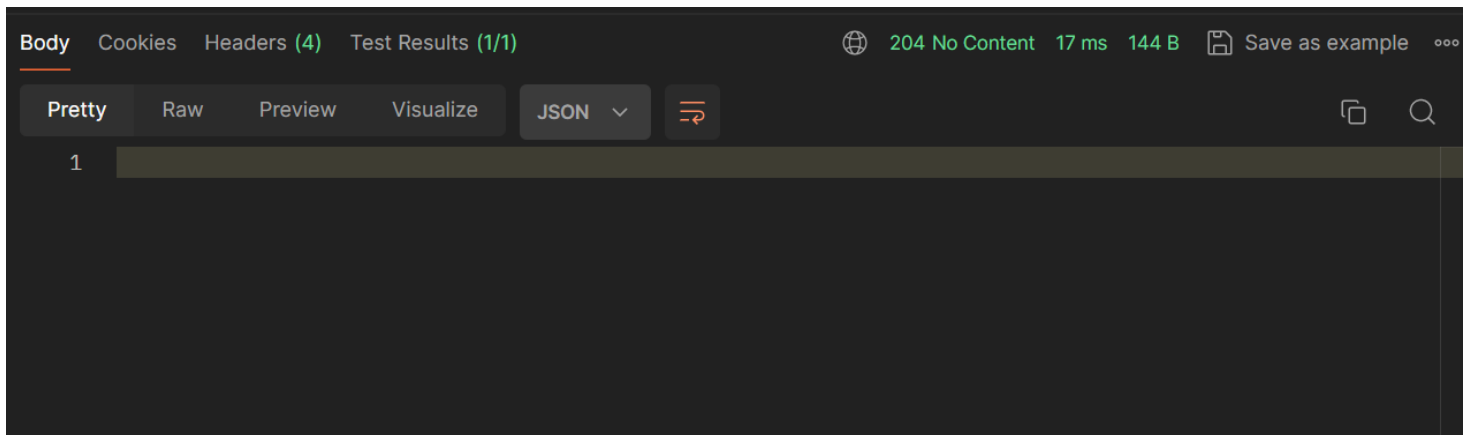
```
http://localhost:8080/series/deleteById?method=true&id=700
```



The screenshot shows a REST client interface with the following configuration:

- Method:** DELETE
- URL:** http://localhost:8080/series/deleteById?method=true&id=700
- Params:**
 - ☒ method: true
 - ☒ id: 700
- Headers:** 6 (not expanded)
- Body:** (empty)
- Pre-request Script:** (empty)
- Tests:** (empty)
- Settings:** (empty)
- Cookies:** (empty)

Revisamos la salida:



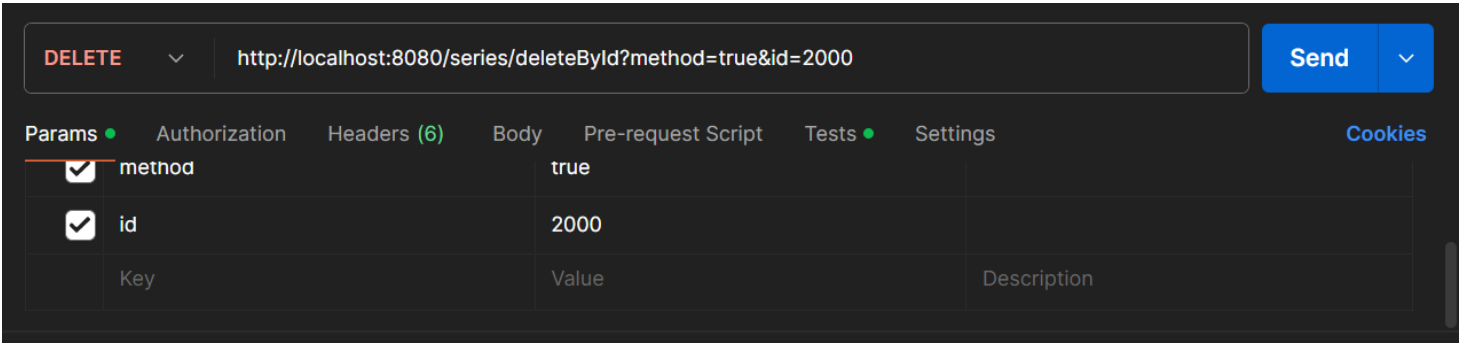
The screenshot shows the response of the DELETE request in a REST client interface:

- Status:** 204 No Content
- Time:** 17 ms
- Size:** 144 B
- Save as example:** (button)
- Body:** (empty, as expected for 204 No Content)
- Headers:** 4 (not expanded)
- Test Results:** 1/1 (not expanded)

# Paso:	18
Descripción	Delete A Serie Criteria Error
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 404 Not Found. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo DELETE la siguiente URL:

```
http://localhost:8080/series/deleteById?method=true&id=2000
```



Revisamos la salida:

