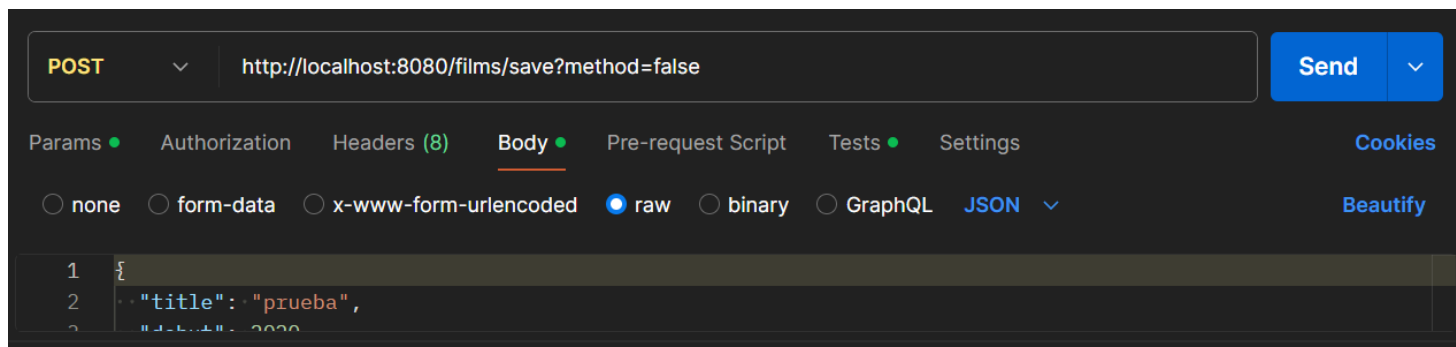


Ejecución Automática de Caso de Test	
Dominio	Proyecto
Aplicación	Ámbito Ejecución
NOMBRE DE LA APLICACIÓN Films	Offline
Descripción	Pre-requisitos
Prueba unitaria para Proyecto Spring Films - Films	
Tester Ejecución	Fecha Ejecución
	09/05/2024

# Paso:	1
Descripción	Create Film 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:

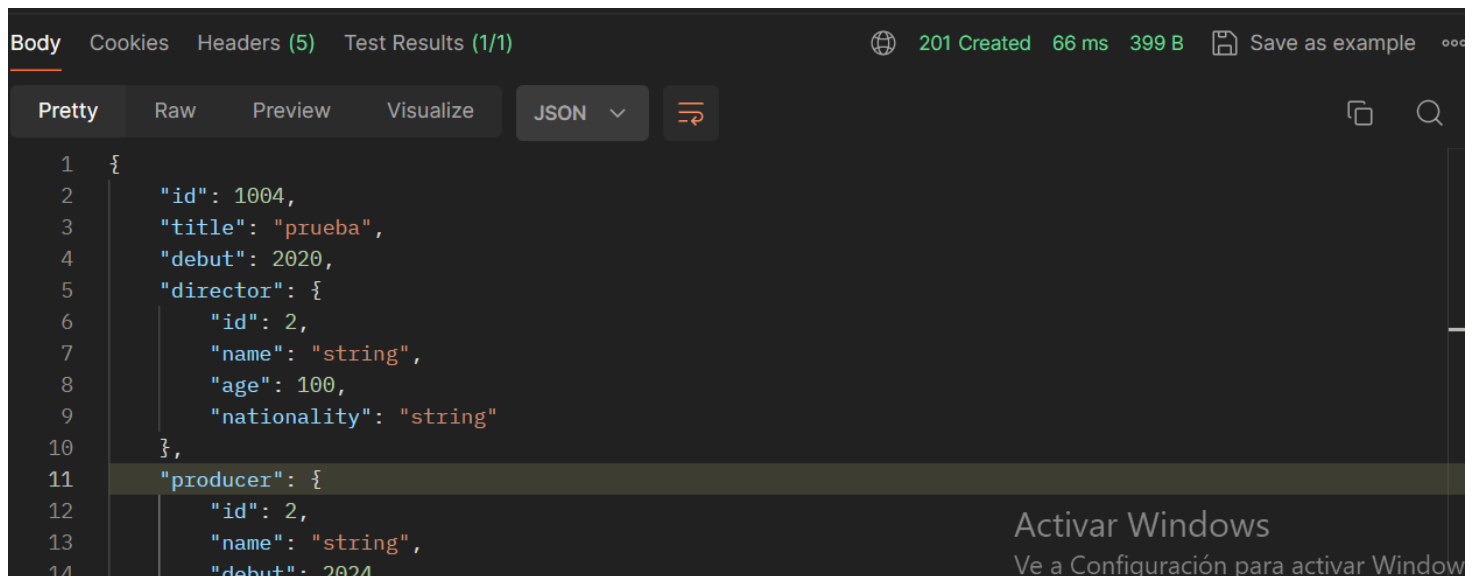
http://localhost:8080/films/save?method=false



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 'Body' tab selected. The JSON response is displayed in a 'Pretty' format. The response structure is as follows:

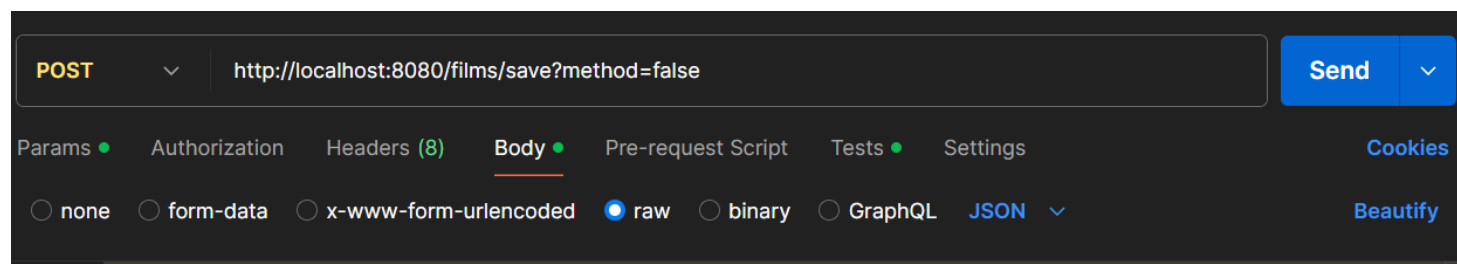
```
1  {
2    "id": 1004,
3    "title": "prueba",
4    "debut": 2020,
5    "director": {
6      "id": 2,
7      "name": "string",
8      "age": 100,
9      "nationality": "string"
10   },
11   "producer": {
12     "id": 2,
13     "name": "string",
14     "debut": 2024
```

At the bottom right of the interface, there is a notification that says 'Activar Windows' and 'Ve a Configuración para activar Windows'.

# Paso:	2
Descripción	Create Film 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:

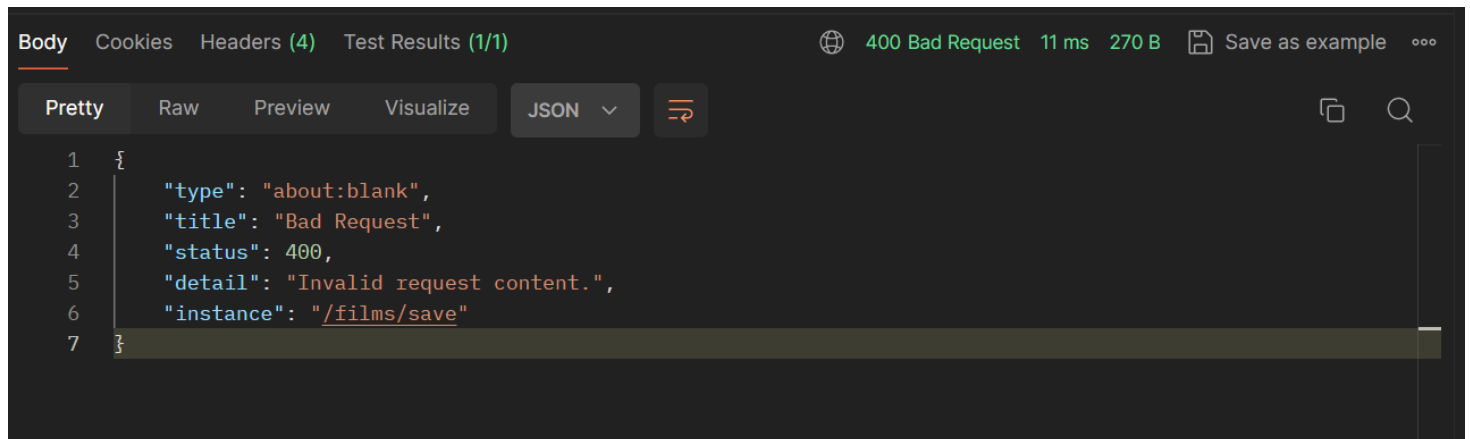
http://localhost:8080/films/save?method=false



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:

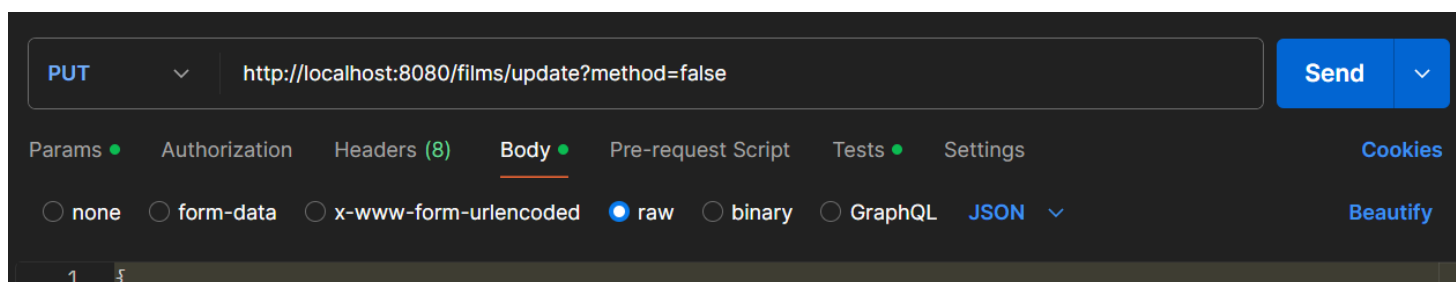
- Body** tab selected.
- Test Results (1/1)** shows a **400 Bad Request** status with a response time of **11 ms** and a size of **270 B**.
- Save as example** button is visible.
- JSON** format is selected for visualization.
- The response body is displayed in a code editor with line numbers 1 through 7:

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

# Paso:	3
Descripción	Update Film 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:

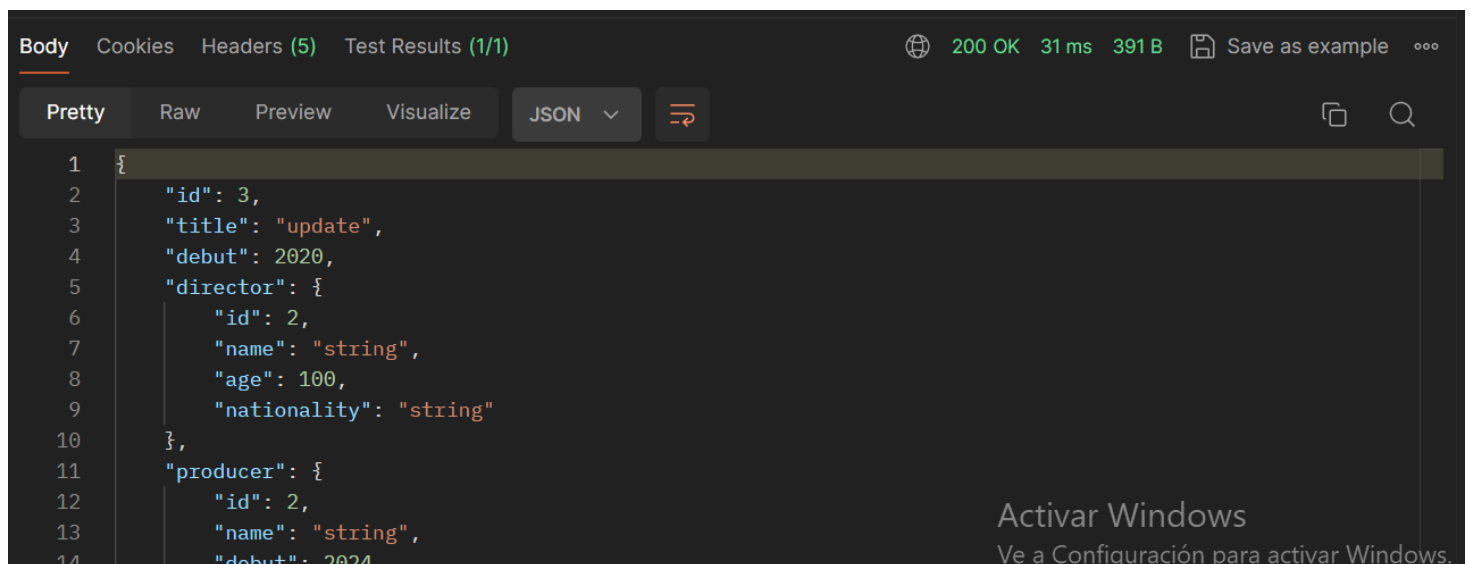
```
http://localhost:8080/films/update?method=false
```



Con el siguiente body

```
{
  "id": 3,
  "title": "update",
  "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"
    }
  ]
}
```

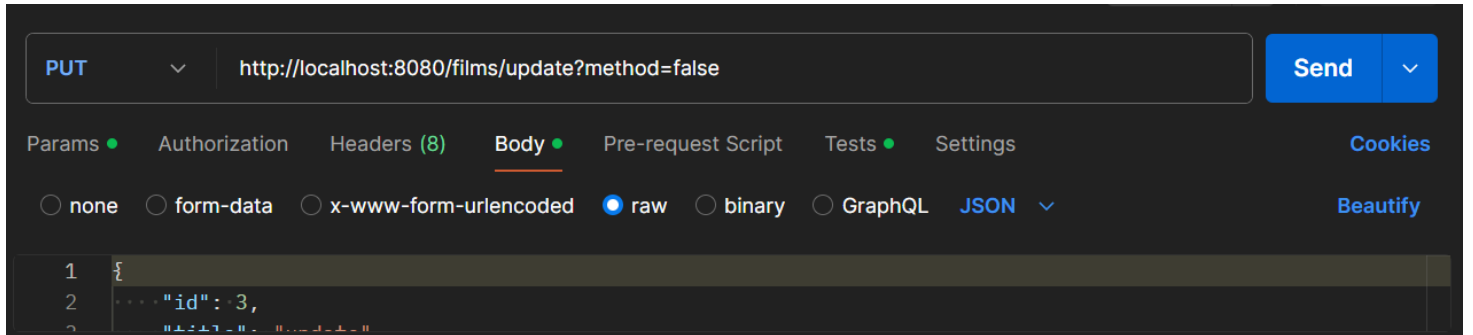
Revisamos la salida:



# Paso:	4
Descripción	Update Film 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:

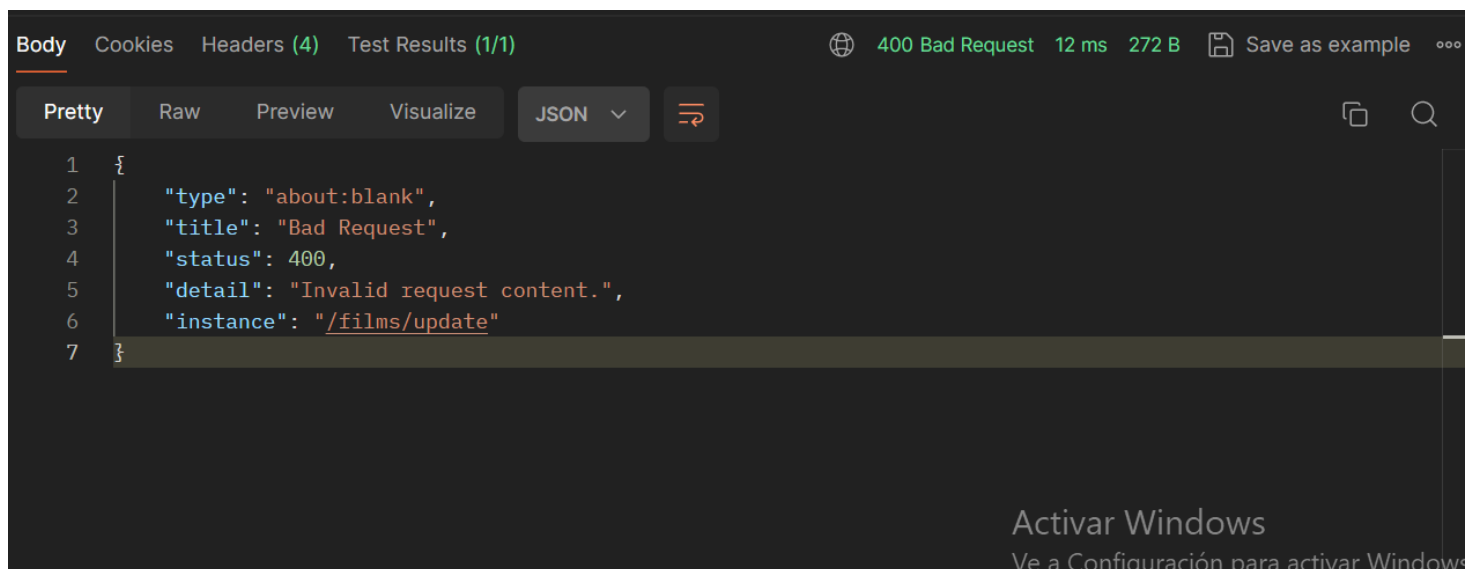
http://localhost:8080/films/update?method=false



Con el siguiente body

```
{
  "id": 3,
  "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 REST client interface with the following elements:

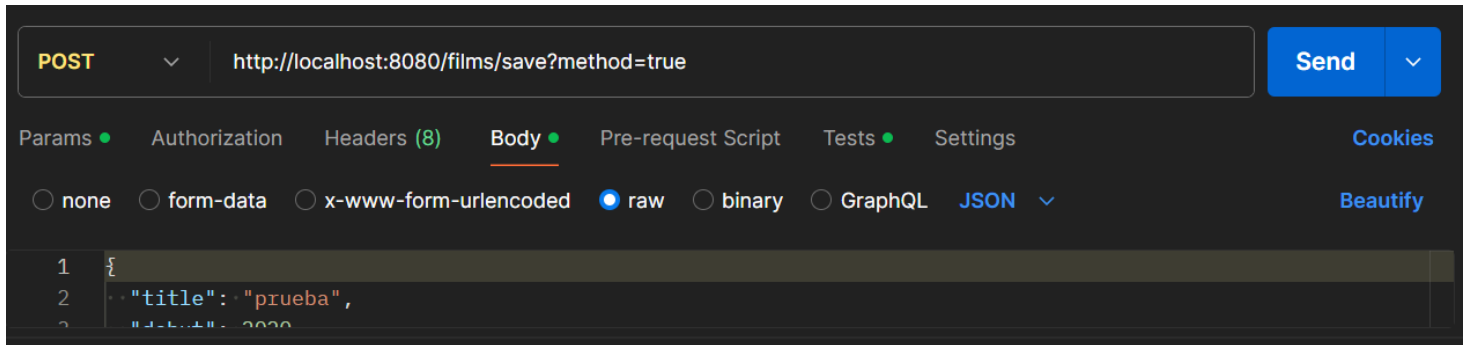
- Body** tab selected.
- Response status: **400 Bad Request**, 12 ms, 272 B.
- Buttons: **Pretty**, **Raw**, **Preview**, **Visualize**, **JSON** (dropdown), and a refresh icon.
- Response body (JSON):

```
1 {
2   "type": "about:blank",
3   "title": "Bad Request",
4   "status": 400,
5   "detail": "Invalid request content.",
6   "instance": "/films/update"
7 }
```
- Bottom right: **Activar Windows** watermark with the text "Ve a Configuración para activar Windows".

# Paso:	5
Descripción	Create Film 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:

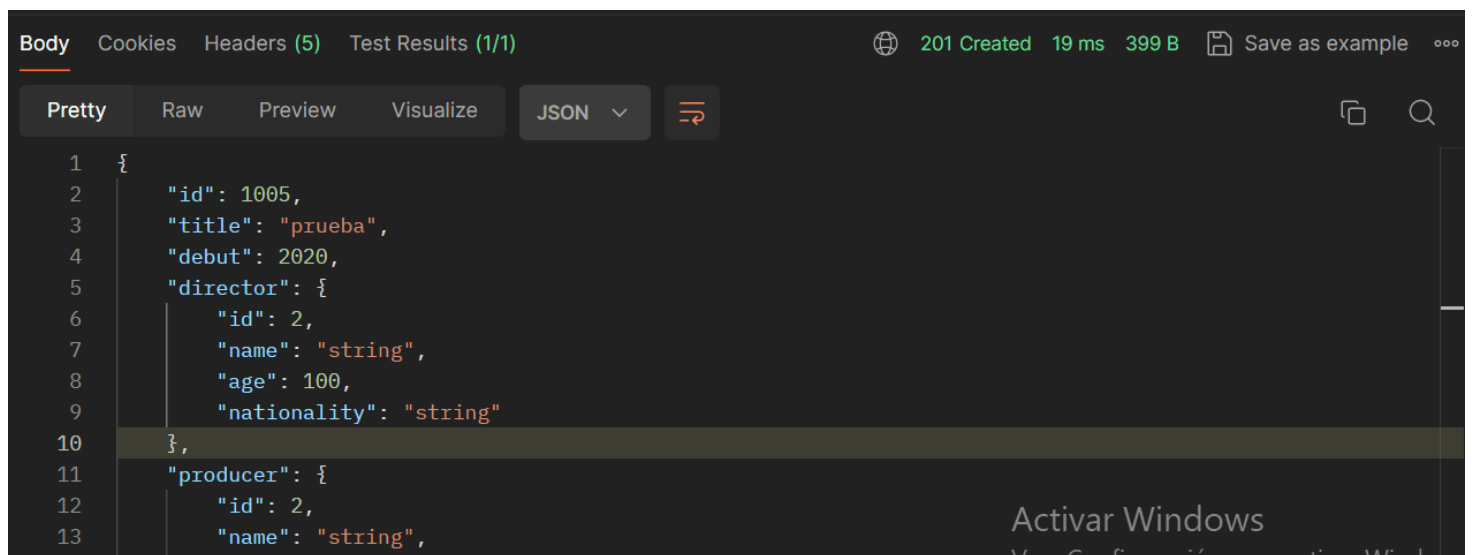
<http://localhost:8080/films/save?method=true>



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:



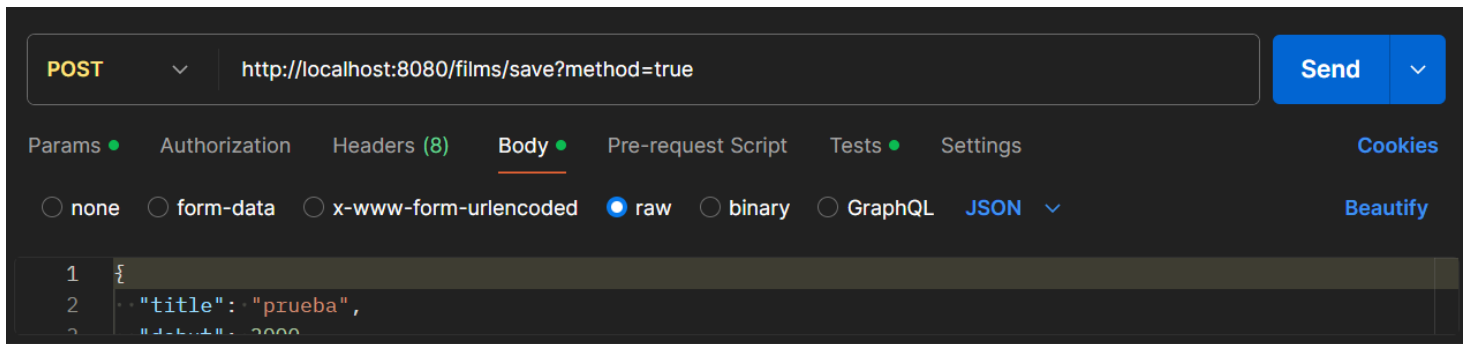
The screenshot shows a REST client interface with the 'Body' tab selected. The JSON response is displayed in a 'Pretty' format. The response structure matches the input body, with the 'actors' array containing one object. The 'id' values in the response (1005, 2, 500) differ from the input (2, 2, 500). A Windows activation watermark is visible in the bottom right corner of the screenshot.

```
1 {
2   "id": 1005,
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",
```

# Paso:	6
Descripción	Create Film 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/films/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:

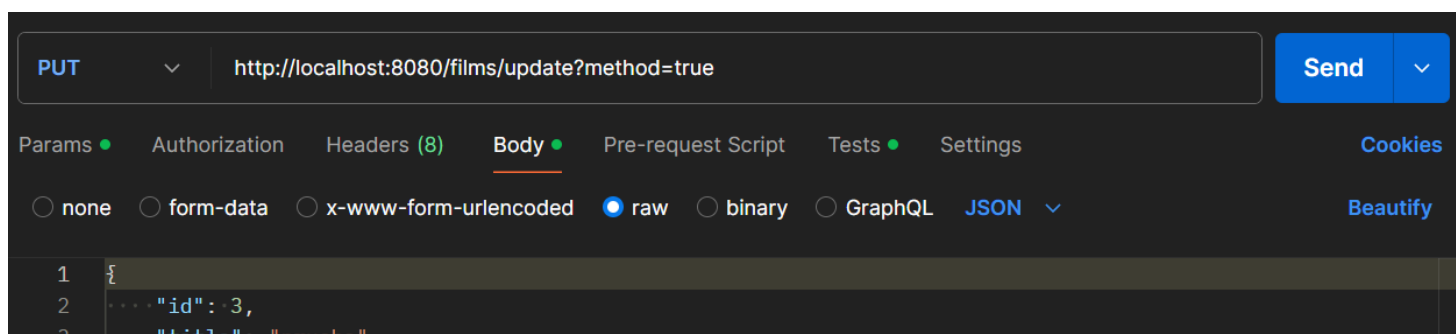
- Body** tab selected.
- Test Results (1/1)** shows a **400 Bad Request** status, **7 ms** duration, and **270 B** size.
- Save as example** button is visible.
- Visualize** tab is selected, showing the response body in **JSON** format.
- The response body is a JSON object with the following structure:

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

# Paso:	7
Descripción	Update Film 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:

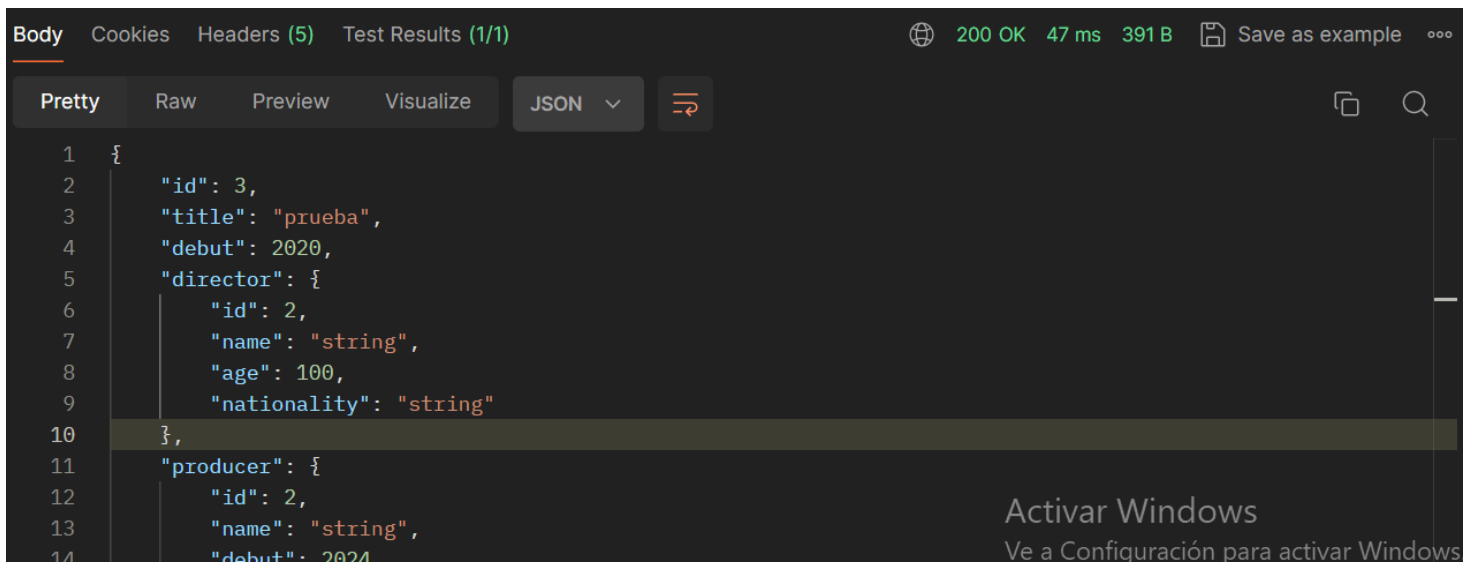
http://localhost:8080/films/update?method=true



Con el siguiente body

```
{
  "id": 3,
  "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:



Body Cookies Headers (5) Test Results (1/1) 200 OK 47 ms 391 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
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
```

Activar Windows
Ve a Configuración para activar Windows

# Paso:	8
Descripción	Update Film 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/films/update?method=true

PUT

http://localhost:8080/films/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": 3,

3

"title": "The Godfather"

Con el siguiente body

```
{
  "id": 3,
  "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 web browser's developer console with the 'Body' tab selected. The console displays a 400 Bad Request error. The error message is as follows:

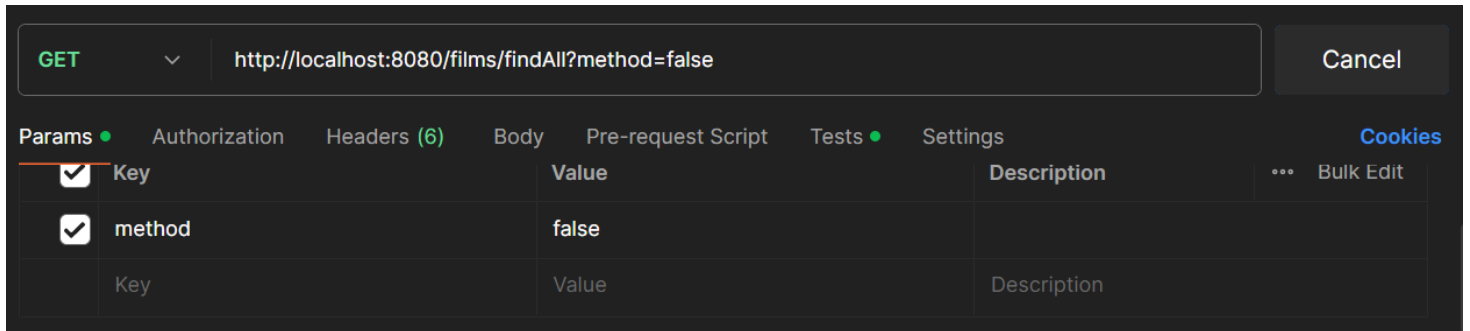
```
{
  "type": "about:blank",
  "title": "Bad Request",
  "status": 400,
  "detail": "Invalid request content.",
  "instance": "/films/update"
}
```

The console also shows the status '400 Bad Request', the time '14 ms', and the size '272 B'. There are buttons for 'Save as example' and 'Pretty' (selected) / 'Raw' / 'Preview' / 'Visualize' / 'JSON' (dropdown) / 'Copy' / 'Search'.

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

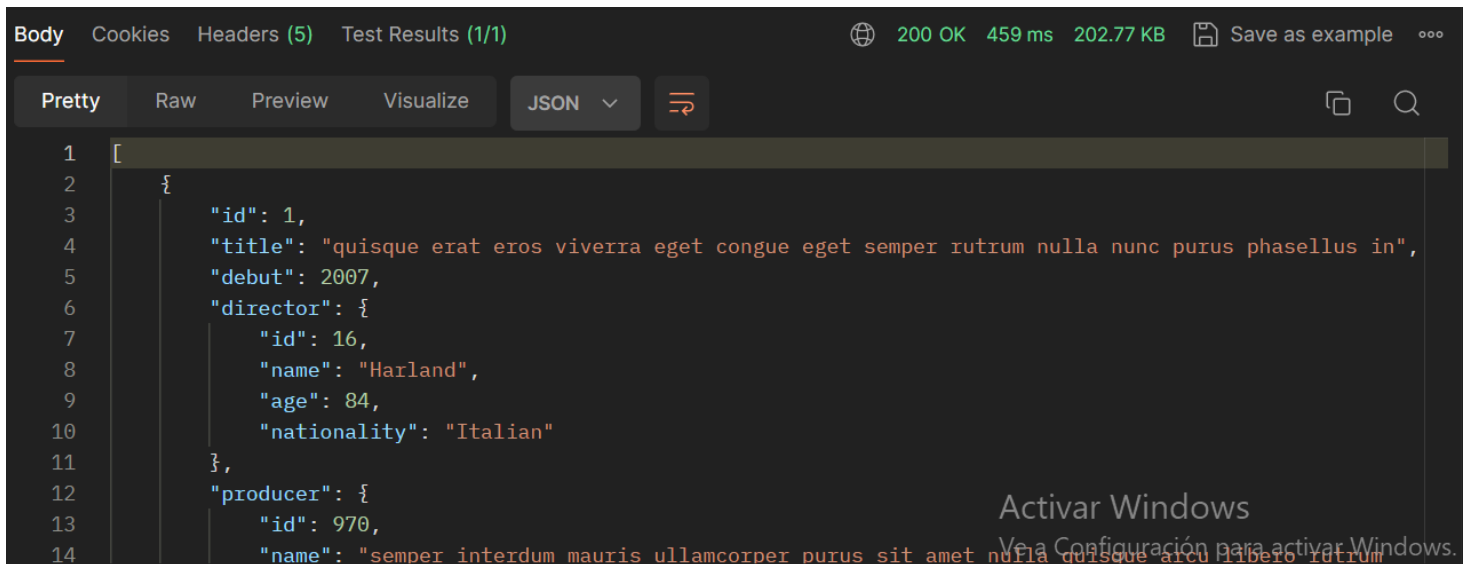
Realizamos una petición REST de tipo GET la siguiente URL:

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



Key	Value	Description
method	false	

Revisamos la salida:



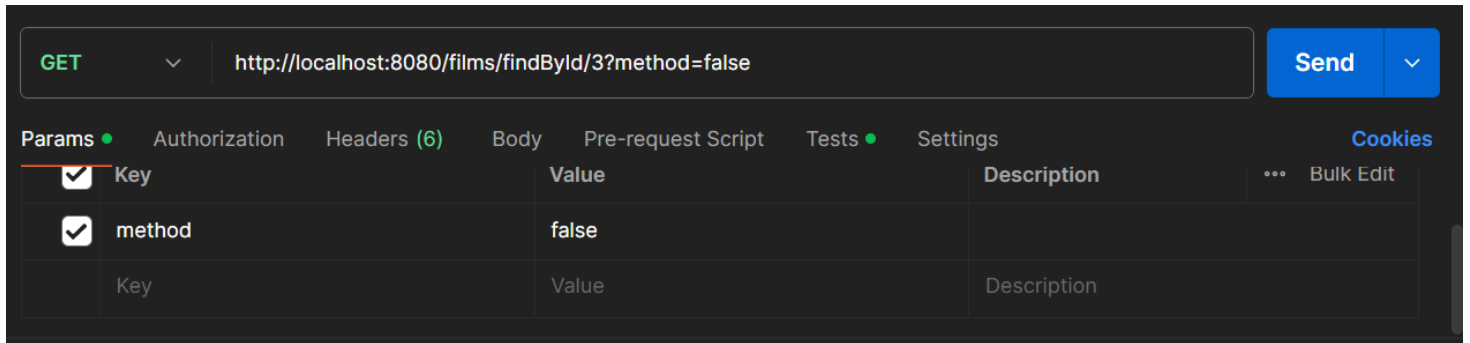
```

1  [
2    {
3      "id": 1,
4      "title": "quisque erat eros viverra eget congue eget semper rutrum nulla nunc purus phasellus in",
5      "debut": 2007,
6      "director": {
7        "id": 16,
8        "name": "Harland",
9        "age": 84,
10       "nationality": "Italian"
11     },
12     "producer": {
13       "id": 970,
14       "name": "semper interdum mauris ullamcorper purus sit amet nulla quisque arcu libero rutrum
  
```

# Paso:	10
Descripción	Get A Film 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/films/findById/3?method=false



GET http://localhost:8080/films/findById/3?method=false Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Key	Value	Description
method	false	

Revisamos la salida:



Body Cookies Headers (5) Test Results (1/1) 200 OK 36 ms 391 B Save as example

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 3,
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

```

# Paso:	11
Descripción	Get A Film 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/films/findById/2000?method=false
```

GET

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

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	false			
	Key	Value	Description		

Revisamos la salida:

Body

Cookies

Headers (5)

Test Results (1/1)

404 Not Found

7 ms

234 B

Save as example

Pretty

Raw

Preview

Visualize

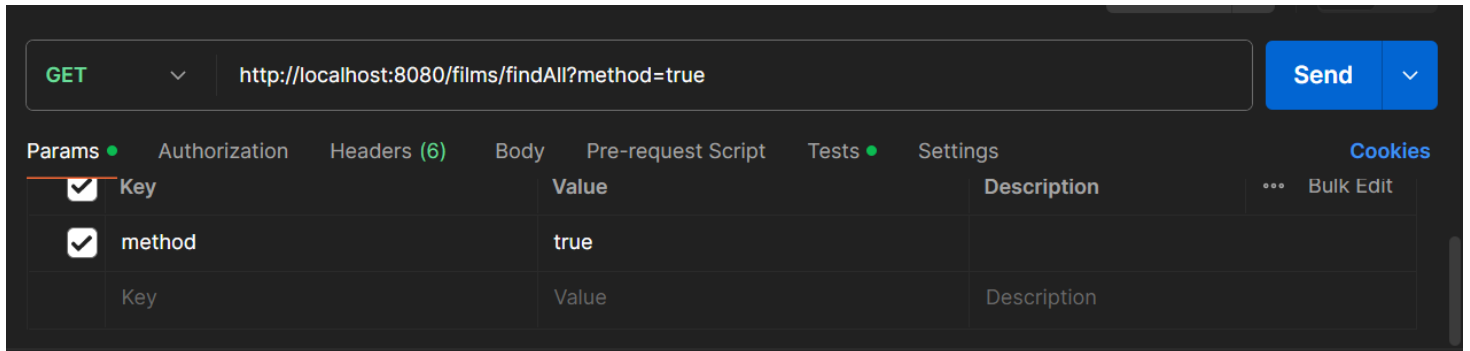
JSON

```
1 {
2   "date": "09/05/2024 16:41:08",
3   "message": "Production not found"
4 }
```

# Paso:	12
Descripción	Get All Films 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/films/findAll?method=true
```



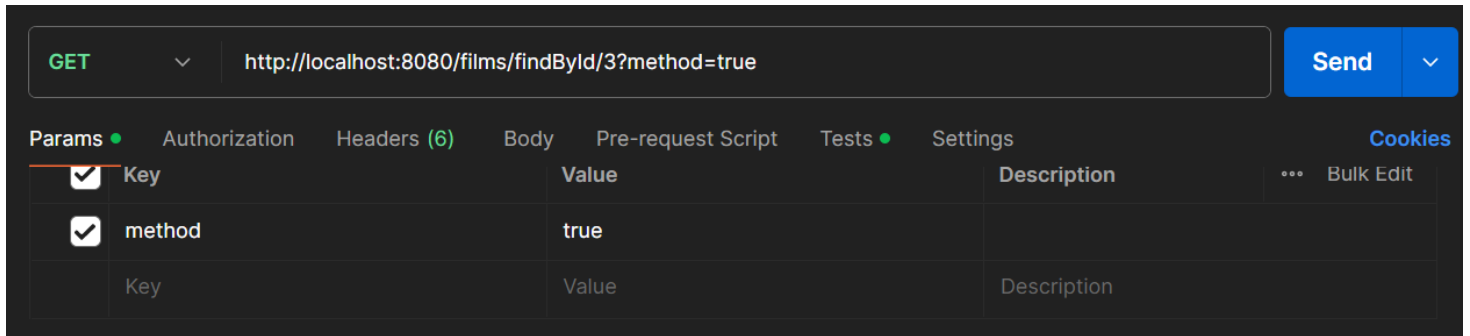
Revisamos la salida:



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

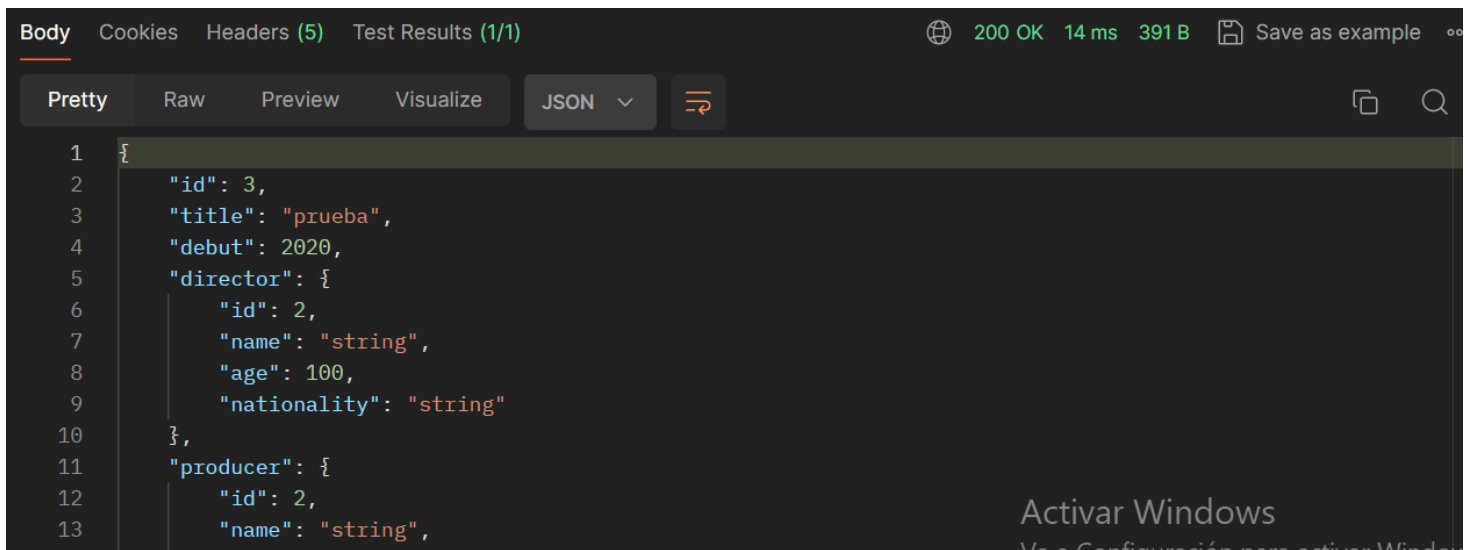
Realizamos una petición REST de tipo GET la siguiente URL:

http://localhost:8080/films/findById/3?method=true



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

Revisamos la salida:



```

1 {
2   "id": 3,
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",

```

# Paso:	14
Descripción	Get A Films 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/films/findById/2000?method=true

GET

http://localhost:8080/films/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

12 ms

234 B

Save as example

Pretty

Raw

Preview

Visualize

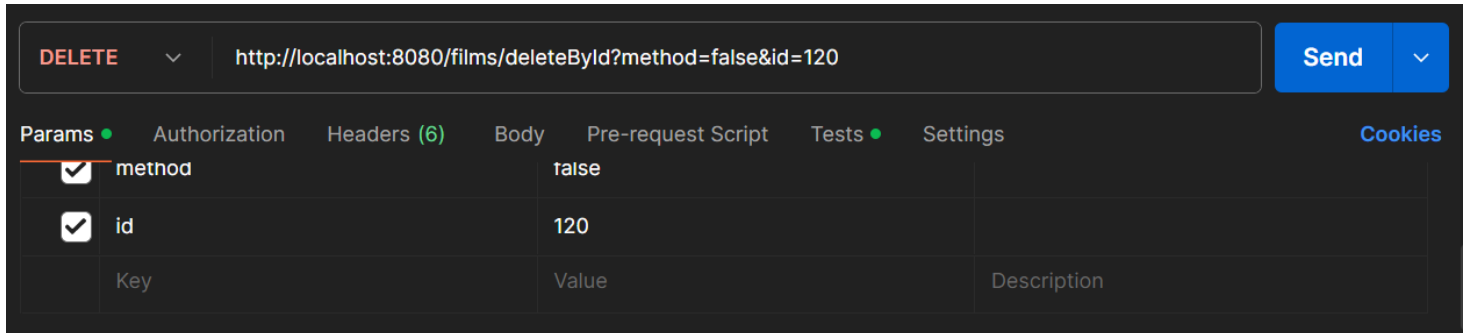
JSON

```
1 {
2   "date": "09/05/2024 16:42:05",
3   "message": "Production not found"
4 }
```

# Paso:	15
Descripción	Delete A Film 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:

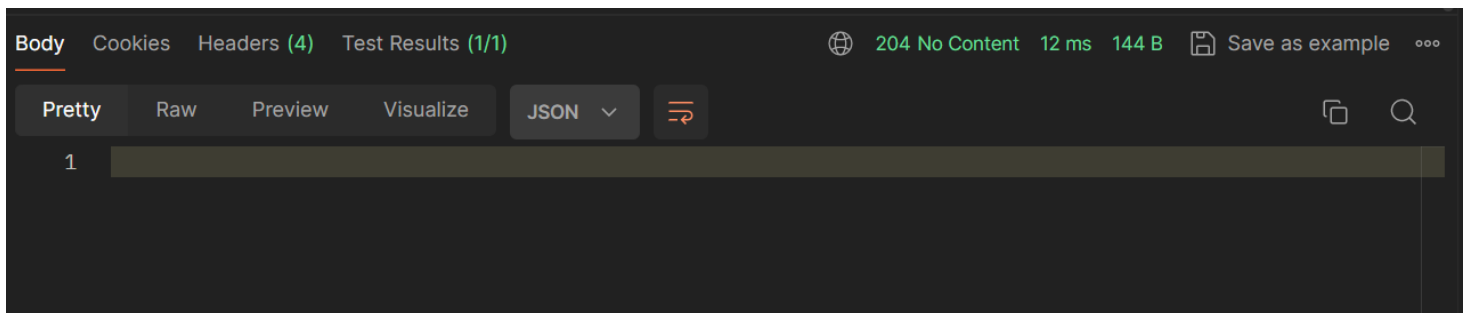
```
http://localhost:8080/films/deleteById?method=false&id=120
```



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

- Method:** DELETE
- URL:** http://localhost:8080/films/deleteById?method=false&id=120
- Params:**
 - method: false
 - id: 120
- Headers:** 6 headers are listed, but none are visible in the screenshot.
- Body:** Empty
- Pre-request Script:** Empty
- Tests:** 0 tests are listed.
- Settings:** No settings are visible.
- Cookies:** No cookies are visible.

Revisamos la salida:



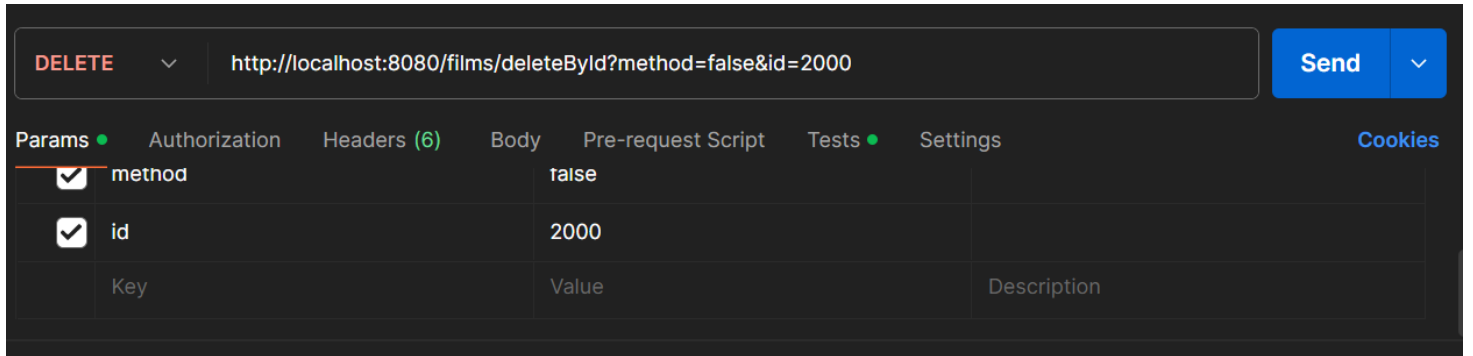
The screenshot shows the response of the DELETE request in a REST client interface:

- Status:** 204 No Content
- Time:** 12 ms
- Size:** 144 B
- Actions:** Save as example, Copy, Search
- Body:** Empty (Pretty view selected)

# Paso:	16
Descripción	Delete A Film 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

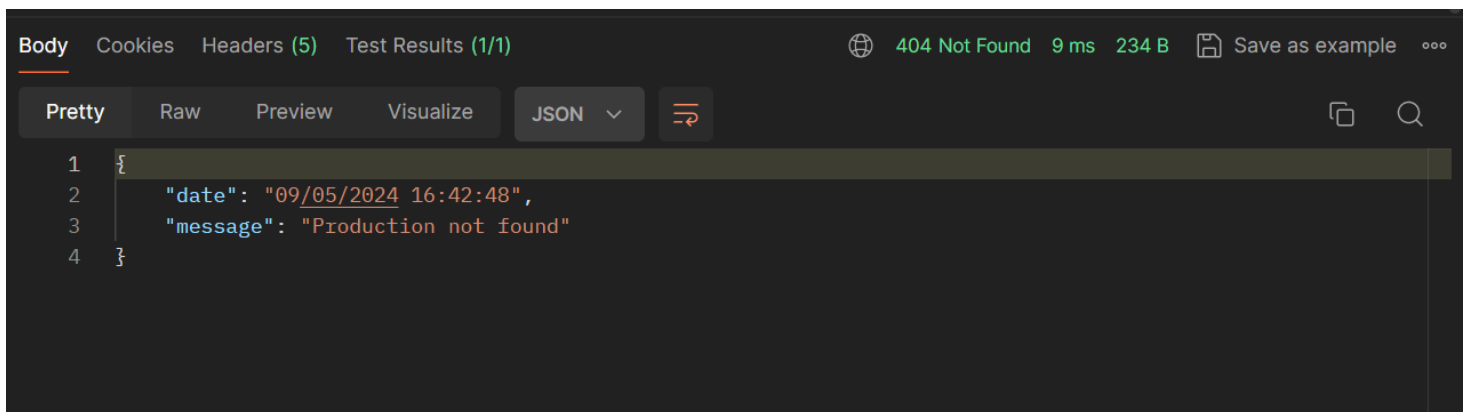
Realizamos una petición REST de tipo DELETE la siguiente URL:

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



The screenshot shows the Postman interface for a DELETE request. The URL bar contains `http://localhost:8080/films/deleteById?method=false&id=2000`. The 'Params' tab is active, showing two parameters: 'method' with value 'false' and 'id' with value '2000'. The 'Send' button is visible on the right.

Revisamos la salida:



The screenshot shows the Postman interface displaying the response body. The status bar indicates a '404 Not Found' response with a time of '9 ms' and a size of '234 B'. The response body is shown in JSON format:

```
{
  "date": "09/05/2024 16:42:48",
  "message": "Production not found"
}
```

# Paso:	17
Descripción	Delete A Film 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/films/deleteById?method=true&id=220
```

DELETE

http://localhost:8080/films/deleteById?method=true&id=220

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

<input checked="" type="checkbox"/>	method	true	
<input checked="" type="checkbox"/>	id	220	
	Key	Value	Description

Revisamos la salida:

Body

Cookies

Headers (4)

Test Results (1/1)

204 No Content

17 ms

144 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

1

# Paso:	18
Descripción	Delete A Film 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/films/deleteById?method=true&id=2000
```

DELETE

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

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

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

Revisamos la salida:

Body

Cookies

Headers (5)

Test Results (1/1)

404 Not Found

11 ms

234 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "date": "09/05/2024 16:43:32",
3   "message": "Production not found"
4 }
```