

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 - Actors	
Tester Ejecución	Fecha Ejecución
	09/05/2024

# Paso:	1
Descripción	Create Actor 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/actors/save?method=false

POST

http://localhost:8080/actors/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

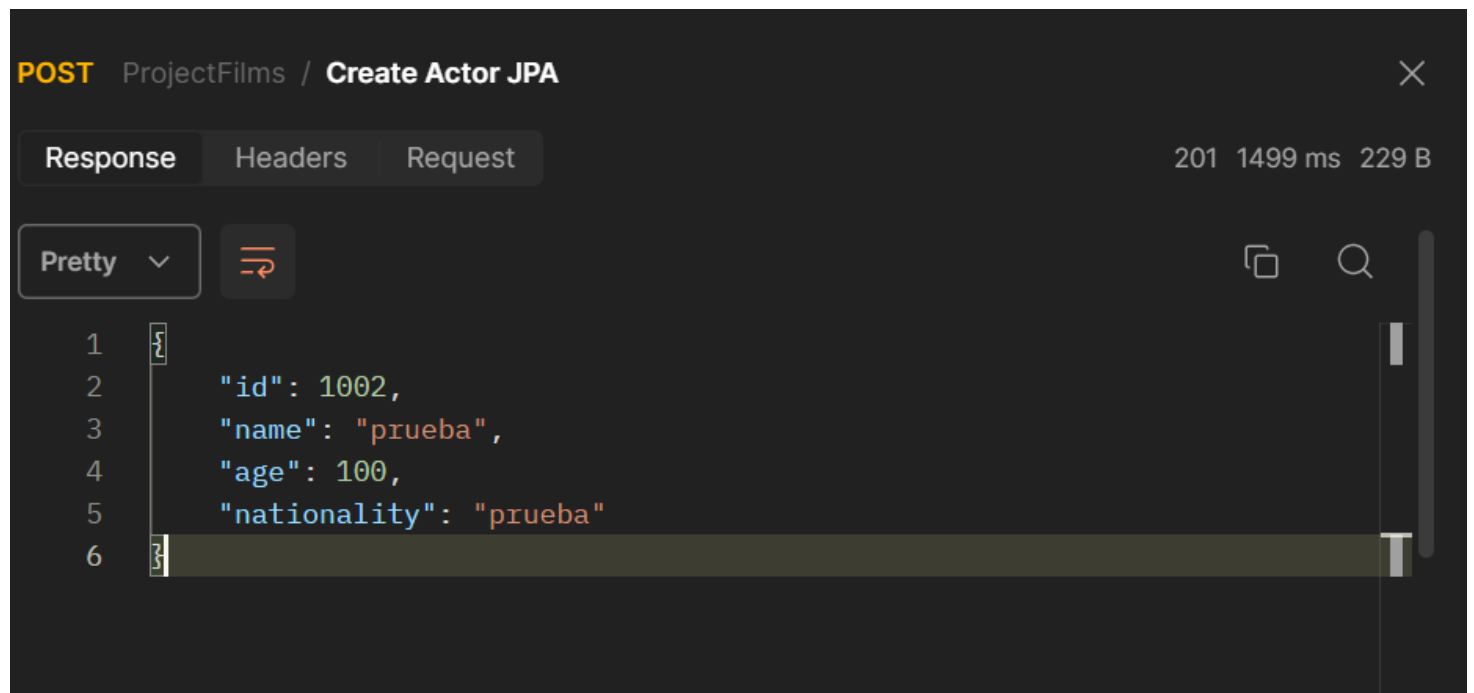
```

1 {
2   "name": "prueba",
3   "age": 100,
4   "nationality": "prueba"
5 }
```

Con el siguiente body

```
{
  "id": 1002,
  "name": "prueba",
  "age": 100,
  "nationality": "prueba"
}
```

Revisamos la salida:



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

- Method:** POST
- Path:** ProjectFilms / Create Actor JPA
- Response Tab:** Selected, showing status 201, time 1499 ms, and size 229 B.
- Format:** Pretty (indicated by a dropdown menu).
- Response Body:**

```
1 {
2   "id": 1002,
3   "name": "prueba",
4   "age": 100,
5   "nationality": "prueba"
6 }
```

# Paso:	2
Descripción	Create Actor 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/actors/save?method=false

POST

http://localhost:8080/actors/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

1

{

2

· "name": "prueba",

3

· "age": 200,

4

· "nationality": "prueba"

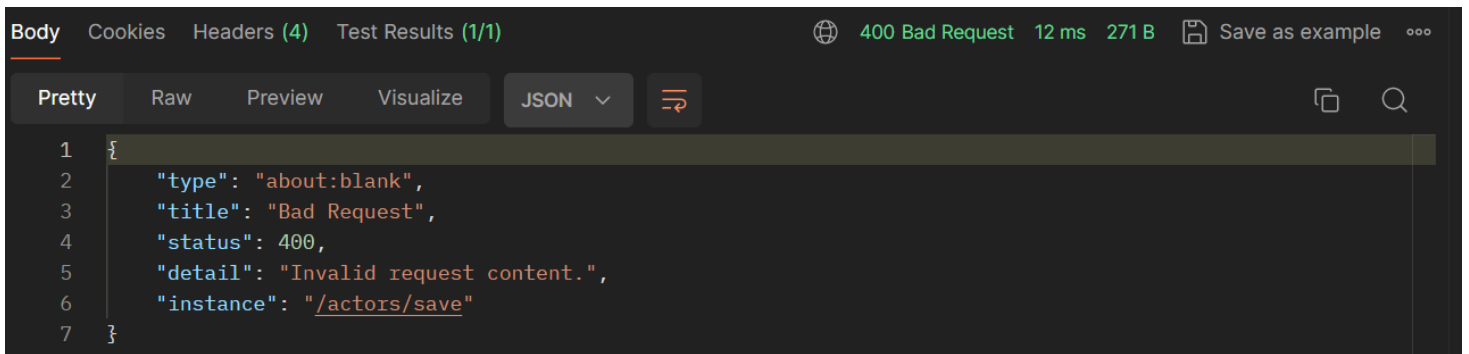
5

}

Con el siguiente body

```
{  
  "name": "prueba",  
  "age": 200,  
  "nationality": "prueba"  
}
```

Revisamos la salida:

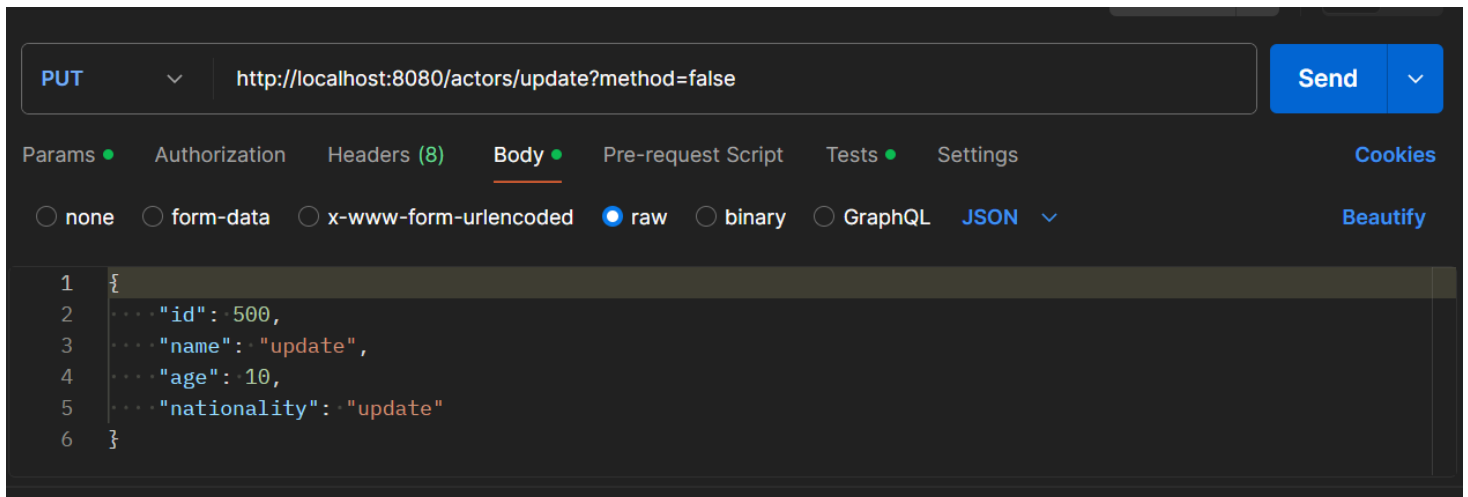


```
Body Cookies Headers (4) Test Results (1/1) 400 Bad Request 12 ms 271 B Save as example  
Pretty Raw Preview Visualize JSON  
1 {  
2   "type": "about:blank",  
3   "title": "Bad Request",  
4   "status": 400,  
5   "detail": "Invalid request content.",  
6   "instance": "/actors/save"  
7 }
```

# Paso:	3
Descripción	Update Actor 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:

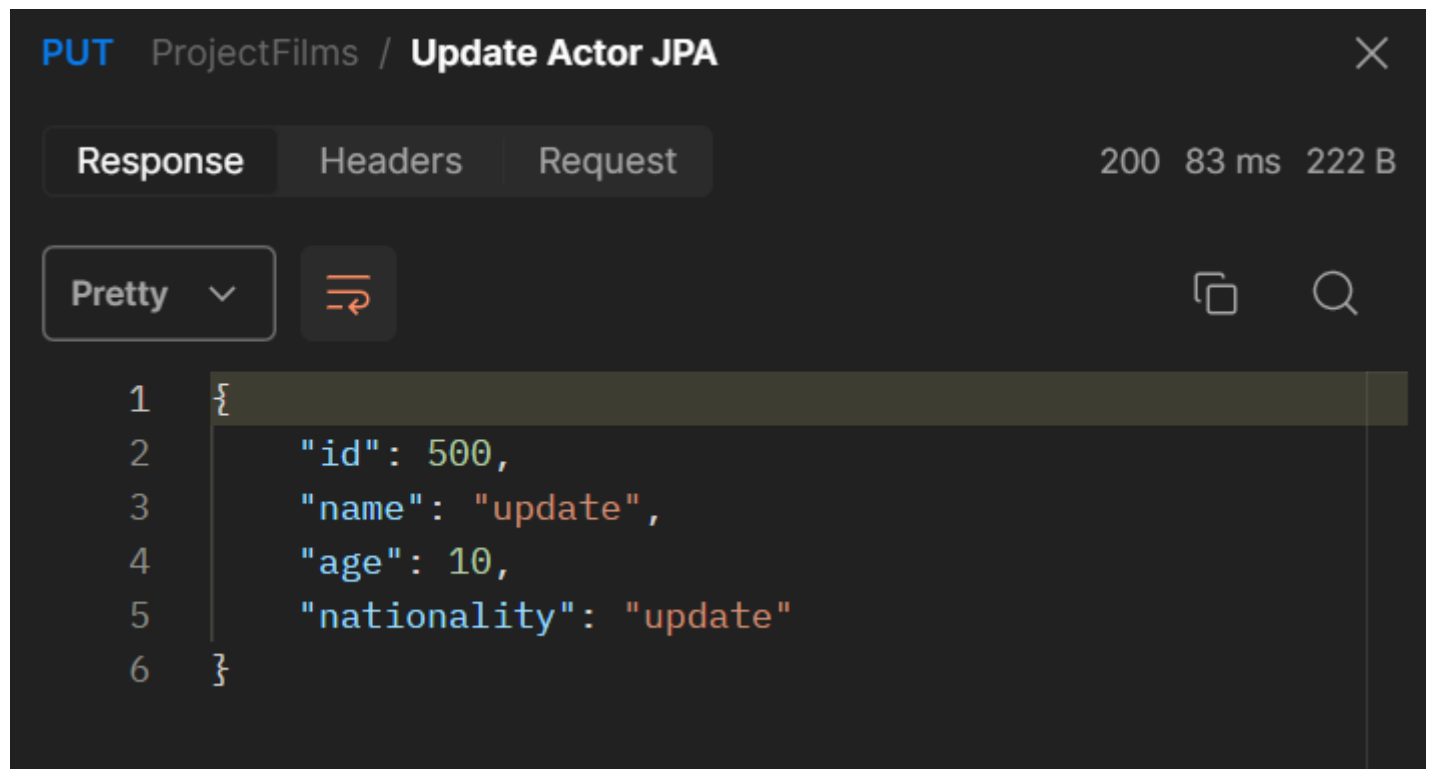
```
http://localhost:8080/actors/update?method=false
```



Con el siguiente body

```
{  
  "id": 500,  
  "name": "update",  
  "age": 10,  
  "nationality": "update"  
}
```

Revisamos la salida:



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

- Method:** PUT
- Path:** ProjectFilms / Update Actor JPA
- Response Tab:** Selected, showing status 200, time 83 ms, and size 222 B.
- Format:** Pretty (indicated by a dropdown menu).
- Response Body:**

```
1 {  
2   "id": 500,  
3   "name": "update",  
4   "age": 10,  
5   "nationality": "update"  
6 }
```

# Paso:	4
Descripción	Update Actor 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/actors/update?method=false

PUT

http://localhost:8080/actors/update?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

```

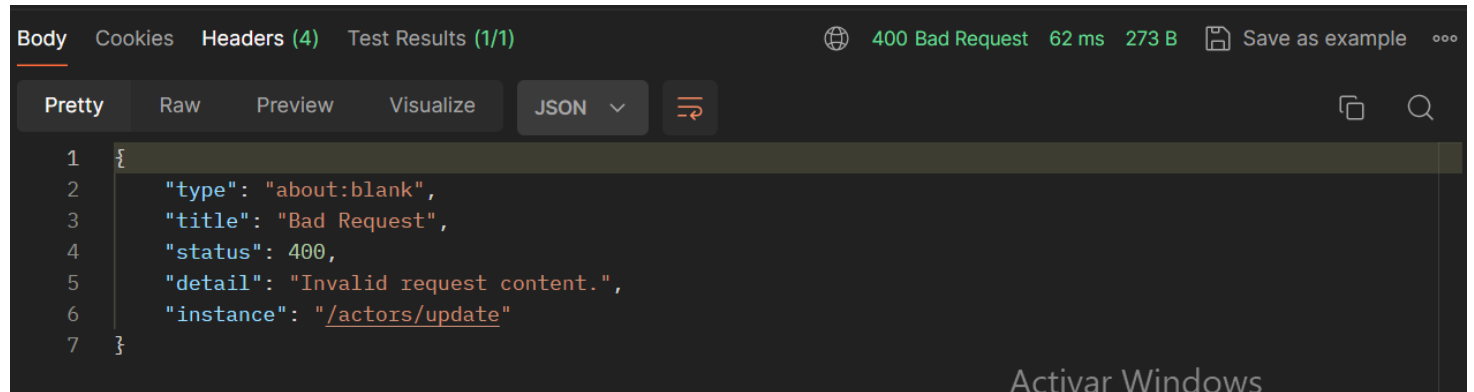
1  {
2    "id": 500,
3    "name": "update",
4    "age": 200,
5    "nationality": "update"
6  }

```


Con el siguiente body

```
{
  "id": 500,
  "name": "update",
  "age": 200,
  "nationality": "update"
}
```

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 with the following JSON response body:

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

The console also shows the status '400 Bad Request', a response time of '62 ms', and a size of '273 B'. There are buttons for 'Save as example' and a search icon. The text 'Activar Windows' is visible in the bottom right corner of the console area.

# Paso:	5
Descripción	Create Actor 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/actors/save?method=true

POST

http://localhost:8080/actors/save?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

"name": "prueba",

3

"age": 100,

4

"nationality": "prueba"

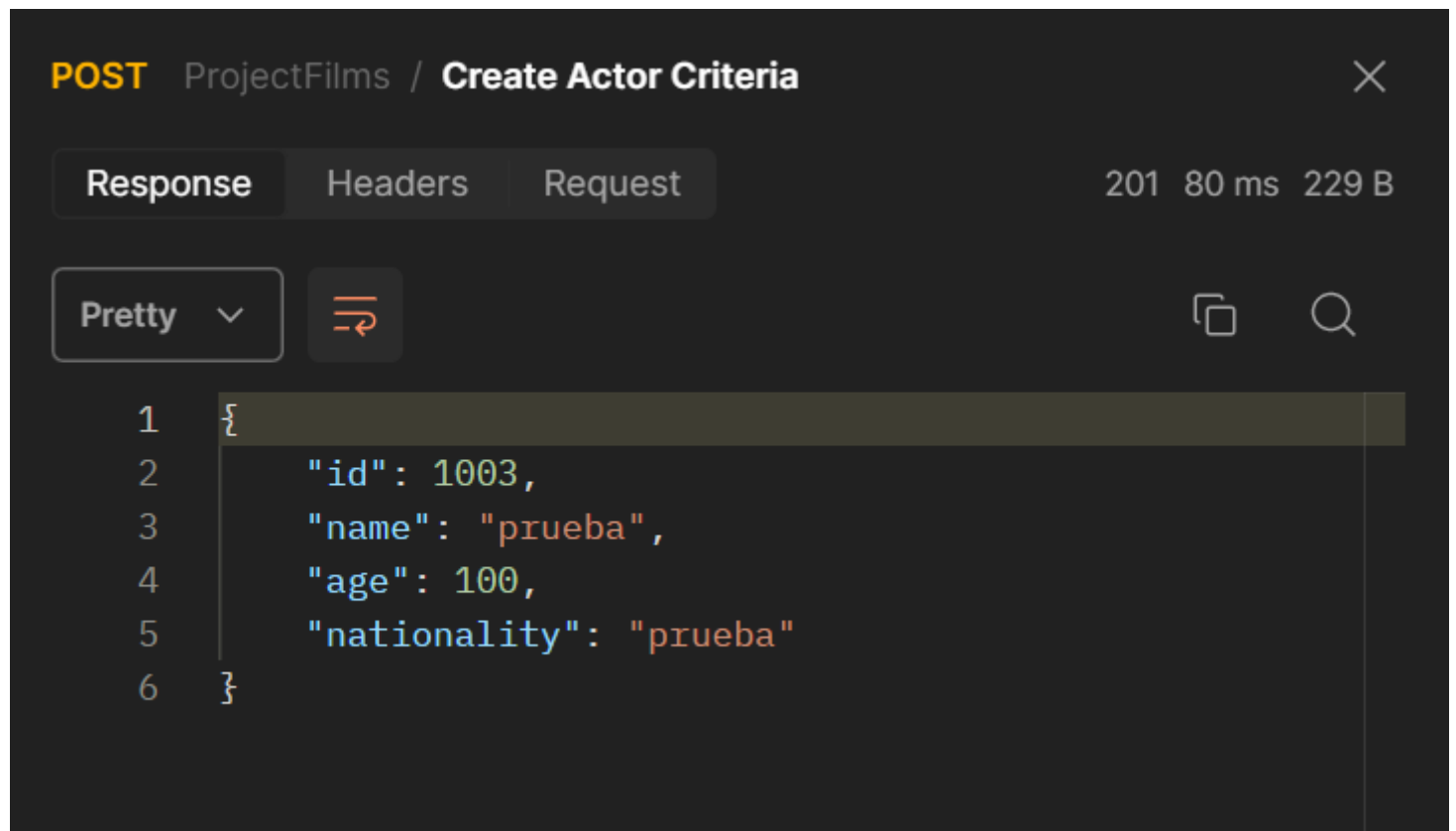
5

}

Con el siguiente body

```
{  
  "name": "prueba",  
  "age": 100,  
  "nationality": "prueba"  
}
```

Revisamos la salida:



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

- Method:** POST
- Path:** ProjectFilms / Create Actor Criteria
- Response Tab:** Selected, showing a status of 201, 80 ms, and 229 B.
- Format:** Pretty (indicated by a dropdown menu).
- Response Body:**

```
1 {  
2   "id": 1003,  
3   "name": "prueba",  
4   "age": 100,  
5   "nationality": "prueba"  
6 }
```

# Paso:	6
Descripción	Create Actor 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/actors/save?method=true

POST

http://localhost:8080/actors/save?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

"name": "prueba",

3

"age": 2000,

4

"nationality": "prueba"

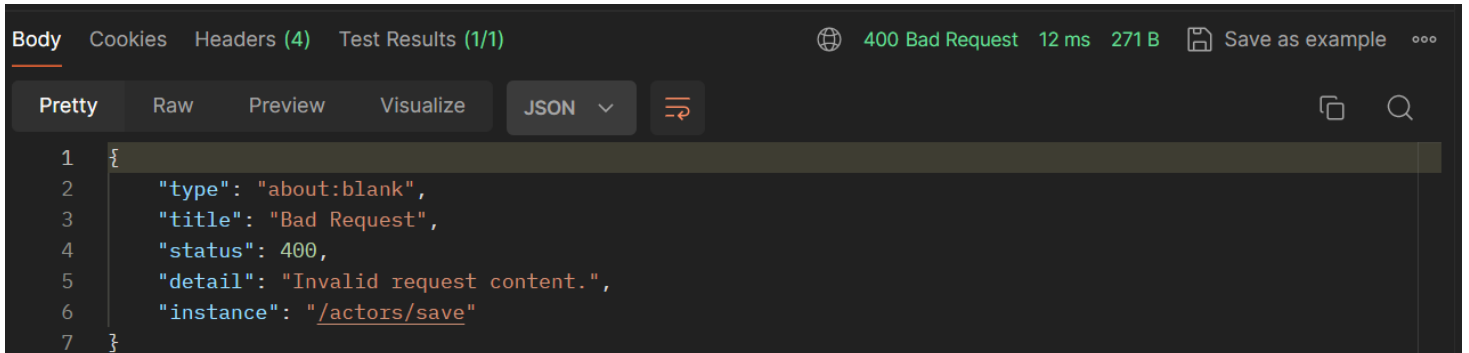
5

}

Con el siguiente body

```
{  
  "name": "prueba",  
  "age": 2000,  
  "nationality": "prueba"  
}
```

Revisamos la salida:



```
Body Cookies Headers (4) Test Results (1/1) 400 Bad Request 12 ms 271 B Save as example  
Pretty Raw Preview Visualize JSON  
1 {  
2   "type": "about:blank",  
3   "title": "Bad Request",  
4   "status": 400,  
5   "detail": "Invalid request content.",  
6   "instance": "/actors/save"  
7 }
```

# Paso:	7
Descripción	Update Actor 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/actors/update?method=true

PUT

http://localhost:8080/actors/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

```

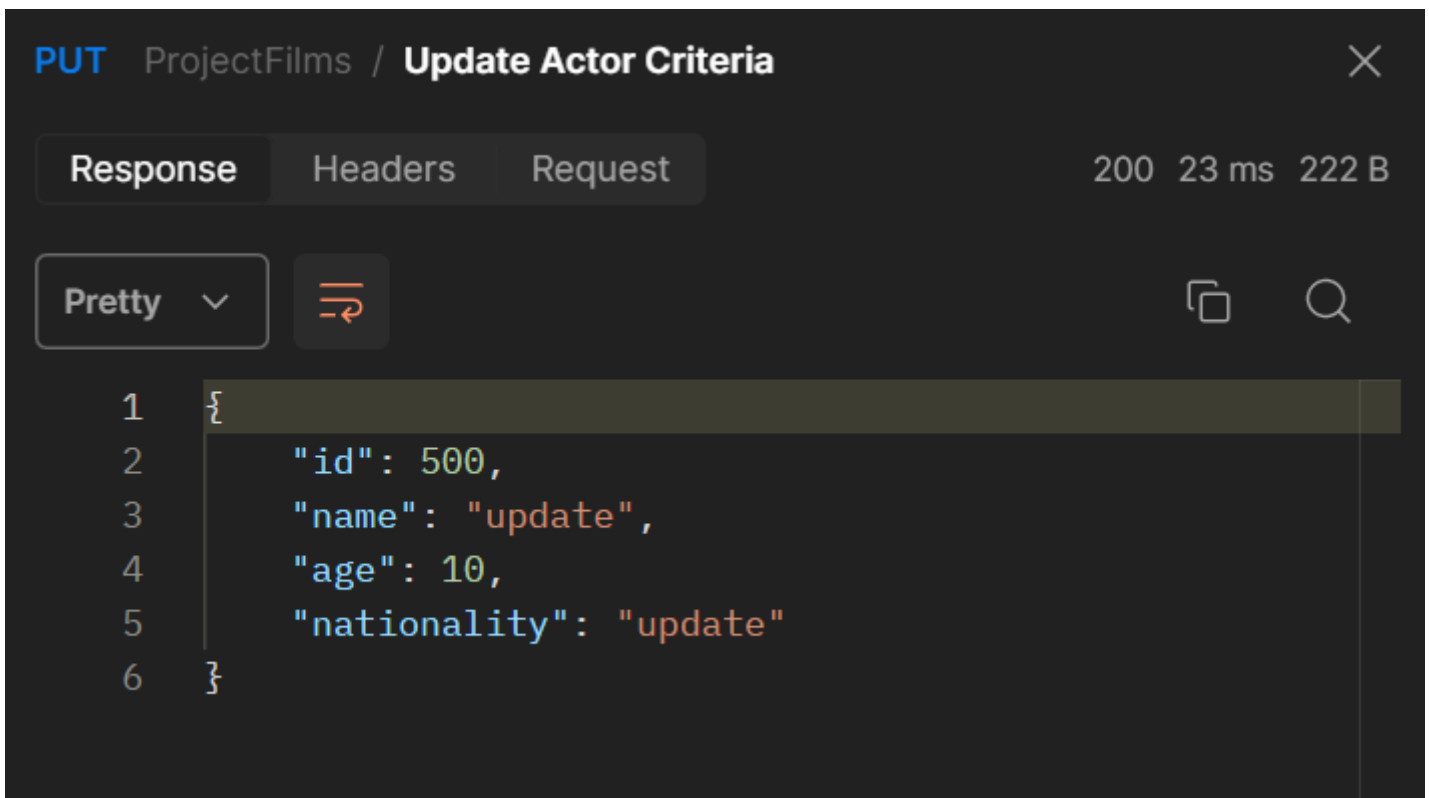
1 {
2   ... "id": 500,
3   ... "name": "update",
4   ... "age": 10,
5   ... "nationality": "update"
6 }

```

Con el siguiente body

```
{  
  "id": 500,  
  "name": "update",  
  "age": 10,  
  "nationality": "update"  
}
```

Revisamos la salida:



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

- Method:** PUT
- Path:** ProjectFilms / Update Actor Criteria
- Response Status:** 200
- Response Time:** 23 ms
- Response Size:** 222 B
- Response Format:** Pretty (indicated by a dropdown menu)
- Response Content:**

```
1 {  
2   "id": 500,  
3   "name": "update",  
4   "age": 10,  
5   "nationality": "update"  
6 }
```

# Paso:	8
Descripción	Update Actor 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/actors/update?method=true

PUT

http://localhost:8080/actors/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

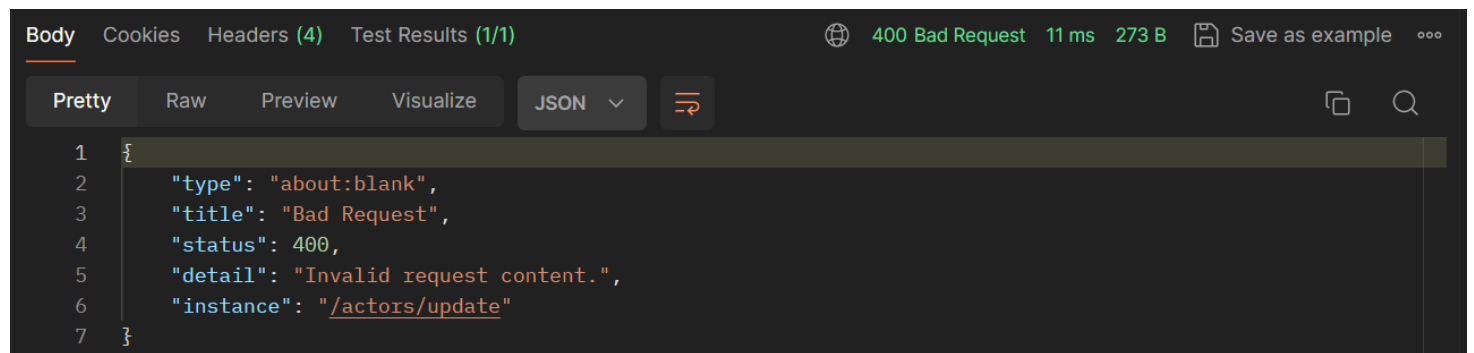
```

1 {
2   ... "id": 500,
3   ... "name": "update",
4   ... "age": 200,
5   ... "nationality": "update"
6 }
```


Con el siguiente body

```
{  
  "id": 500,  
  "name": "update",  
  "age": 200,  
  "nationality": "update"  
}
```

Revisamos la salida:



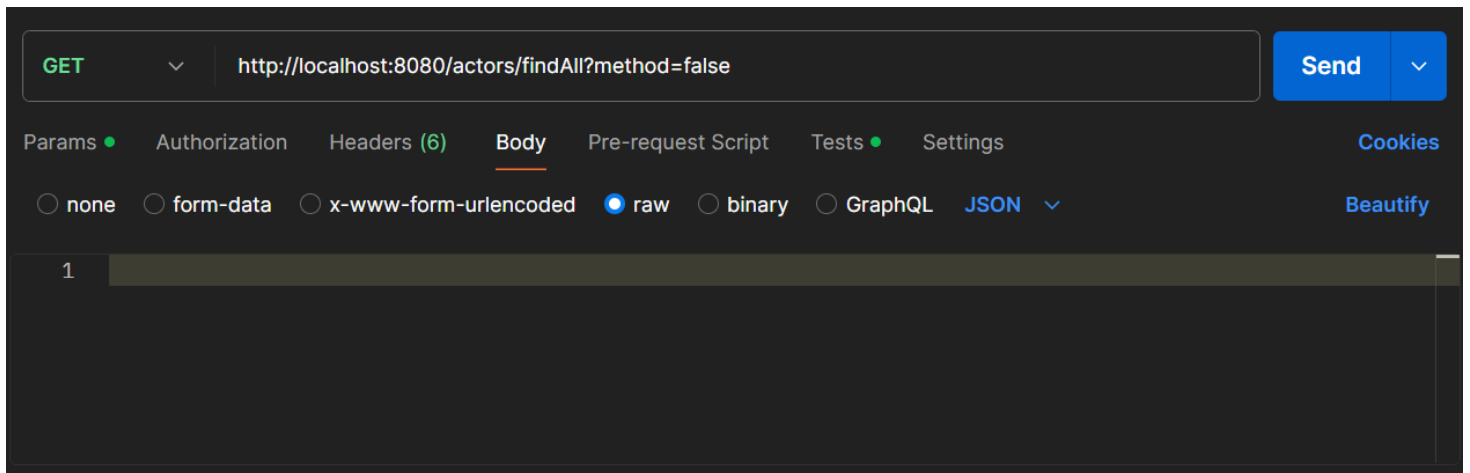
The screenshot shows a web browser's developer console with the 'Body' tab selected. The console displays a 400 Bad Request error. The error message is: "Invalid request content." The error details are: "type": "about:blank", "title": "Bad Request", "status": 400, "detail": "Invalid request content.", "instance": "/actors/update". The error is shown in a dark theme with a light gray border.

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

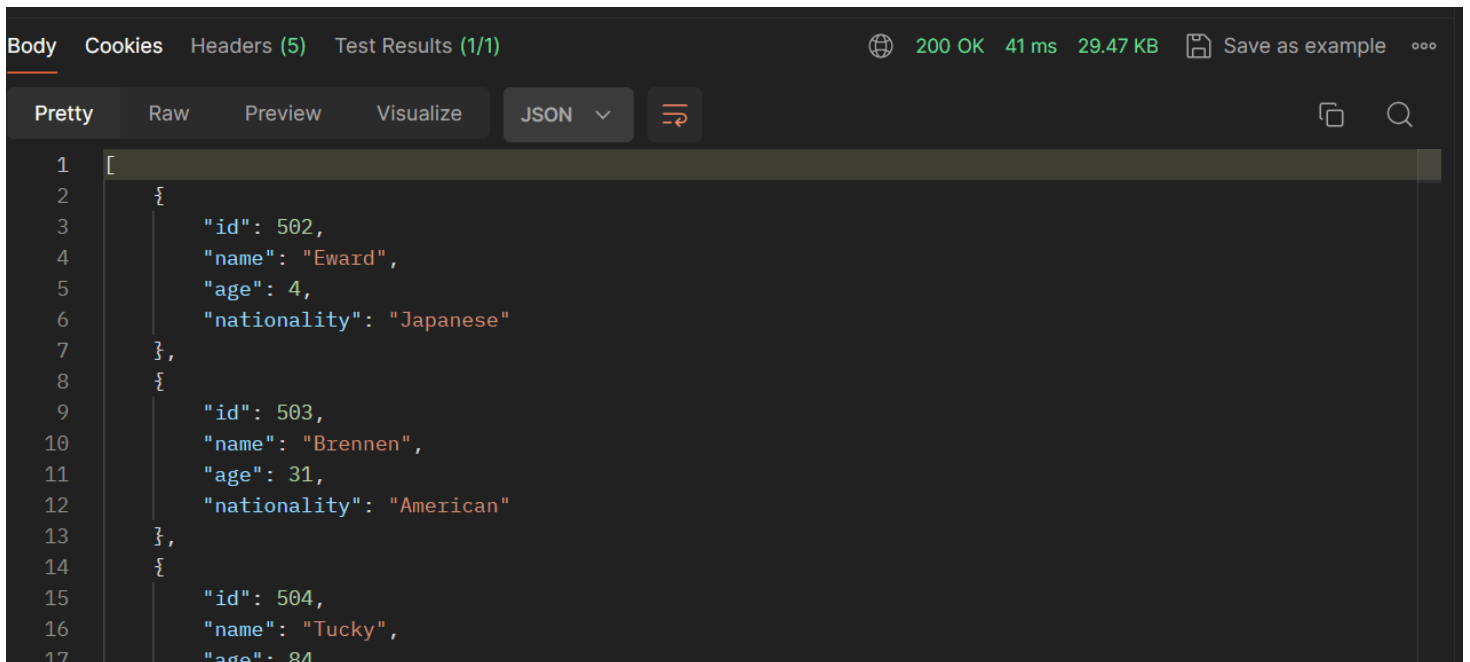
# Paso:	9
Descripción	Get All Actors JPA
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 200 OK. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo la GET siguiente URL:

```
http://localhost:8080/actors/findAll?method=false
```



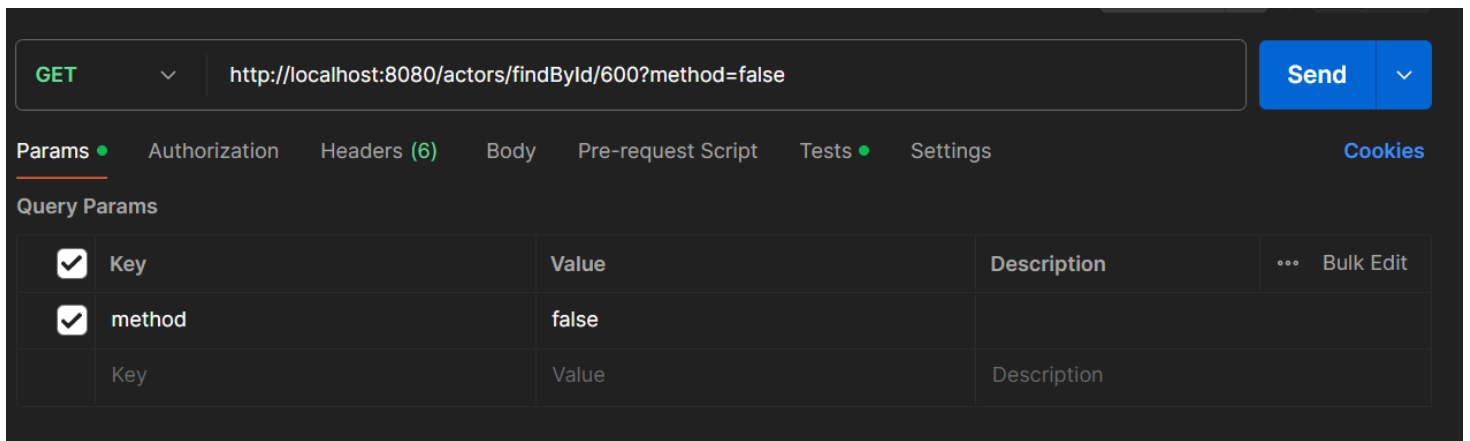
Revisamos la salida:



# Paso:	10
Descripción	Get An Actor 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/actors/findByld/600?method=false



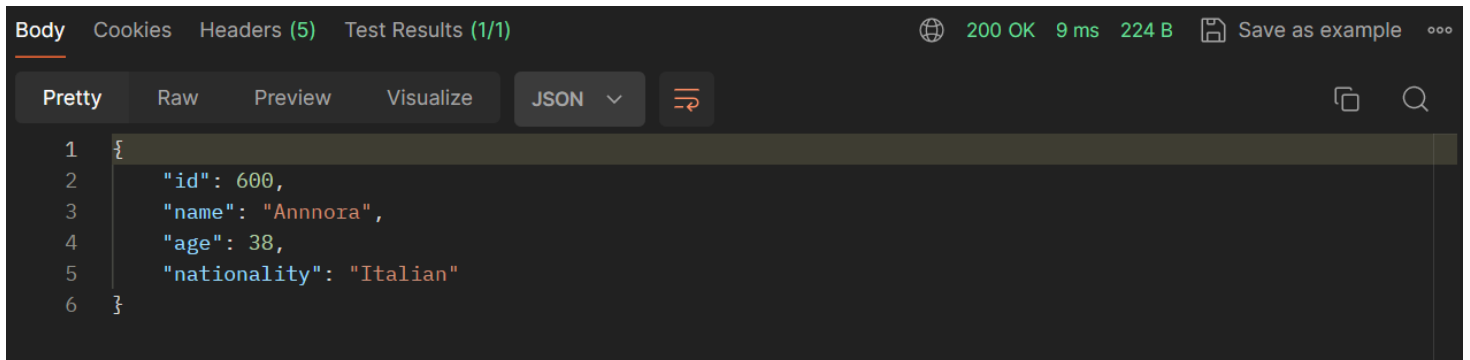
GET ⌵ http://localhost:8080/actors/findByld/600?method=false Send ⌵

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

Query Params

<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) 200 OK 9 ms 224 B Save as example ⋮

Pretty Raw Preview Visualize JSON ⌵ ⌵

```

1 {
2   "id": 600,
3   "name": "Annora",
4   "age": 38,
5   "nationality": "Italian"
6 }
```

# Paso:	11
Descripción	Get An Actor 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 la GET siguiente URL:

```
http://localhost:8080/actors/findByld/2000?method=false
```

GET

http://localhost:8080/actors/findByld/2000?method=false

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

<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

9 ms

230 B

Save as example

Pretty

Raw

Preview

Visualize

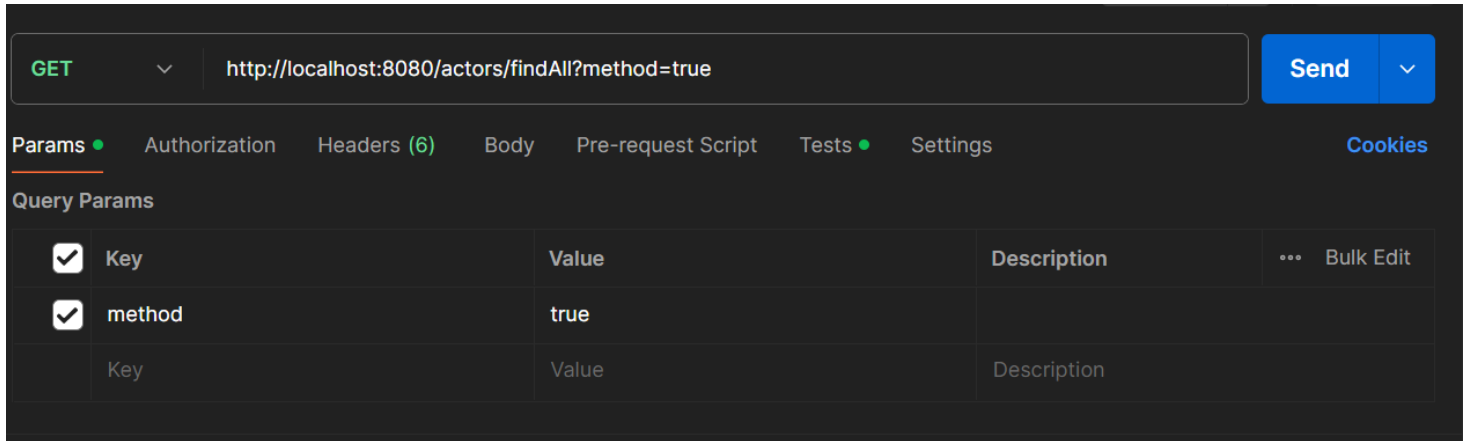
JSON

```
1 {
2   "date": "09/05/2024 15:22:00",
3   "message": "Person not found"
4 }
```

# Paso:	12
Descripción	Get All Actors 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/actors/findAll?method=true



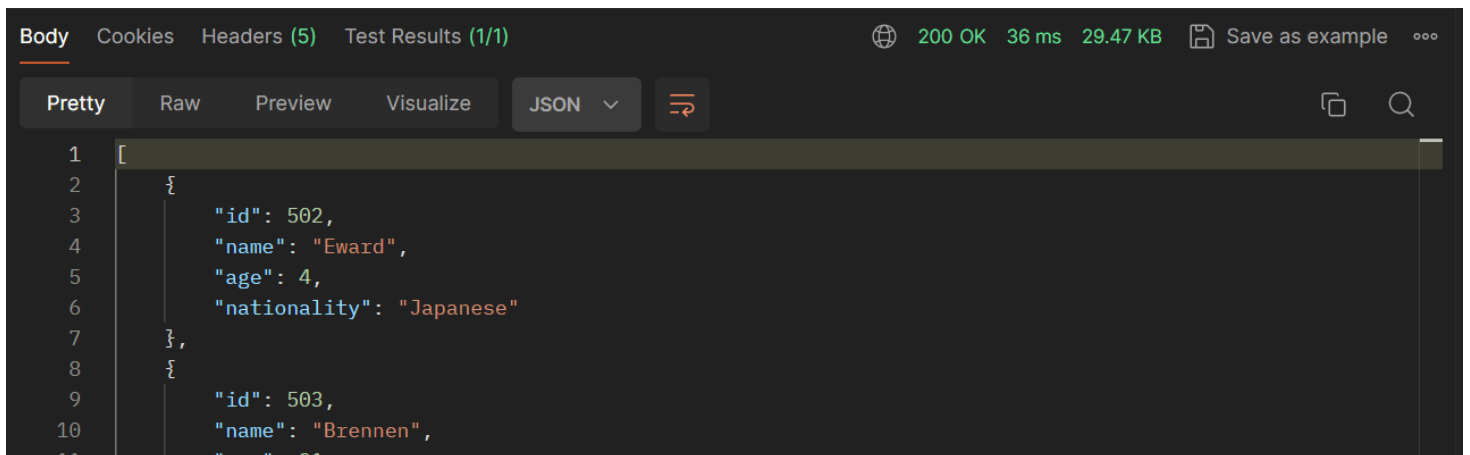
GET http://localhost:8080/actors/findAll?method=true Send

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

Query Params

<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) 200 OK 36 ms 29.47 KB Save as example

Pretty Raw Preview Visualize JSON ⌵ ⌵

```

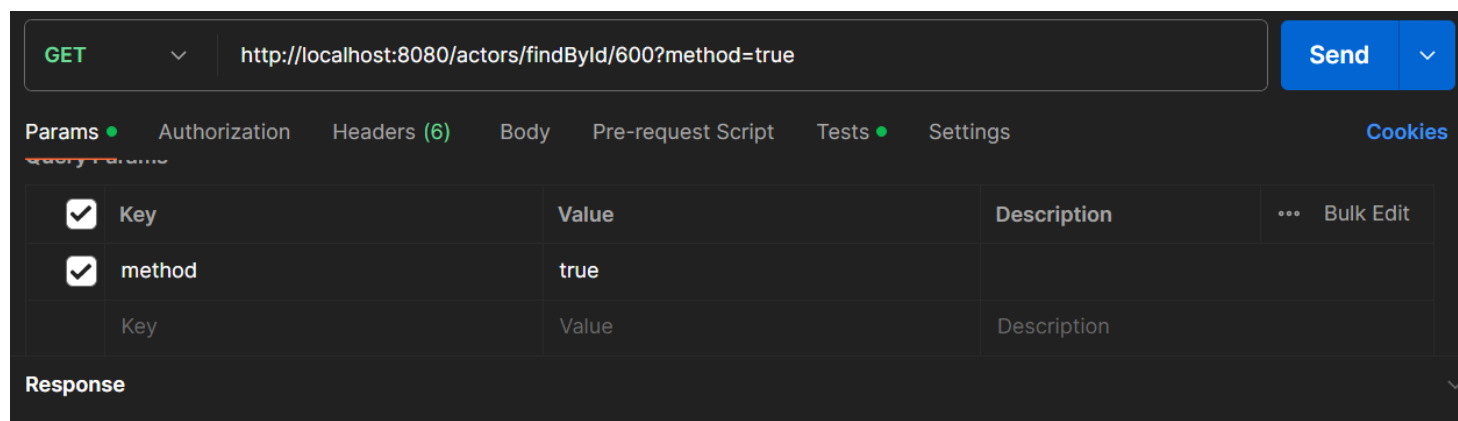
1  [
2    {
3      "id": 502,
4      "name": "Eward",
5      "age": 4,
6      "nationality": "Japanese"
7    },
8    {
9      "id": 503,
10     "name": "Brennen",
11     "age": 21,

```

# Paso:	13
Descripción	Get An Actor 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/actors/findByld/600?method=true
```



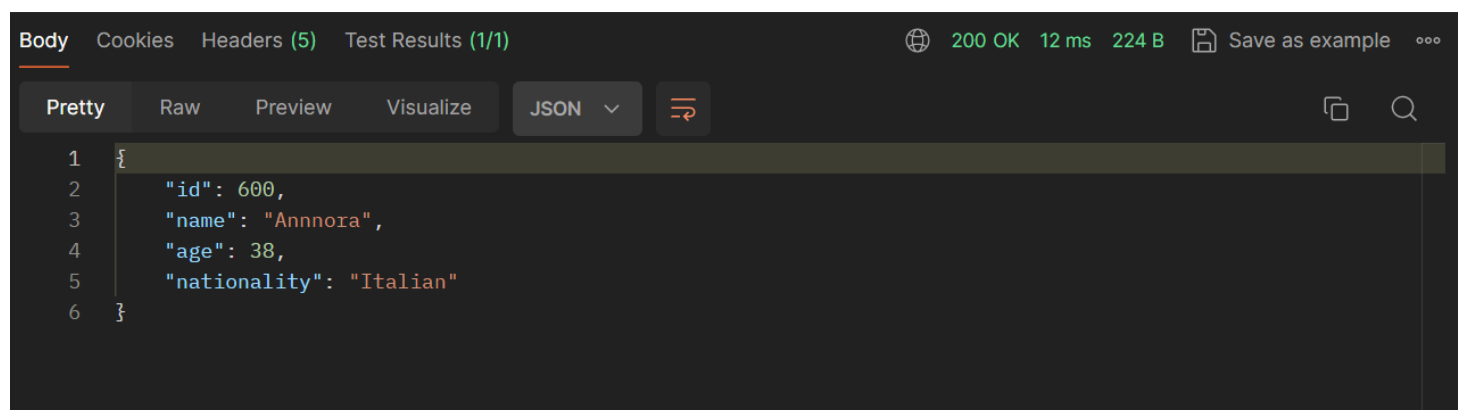
GET ▼ http://localhost:8080/actors/findByld/600?method=true Send ▼

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		

Response ▼

Revisamos la salida:



Body Cookies Headers (5) Test Results (1/1) 200 OK 12 ms 224 B Save as example ...

Pretty Raw Preview Visualize JSON ▼ ≡ 📄 🔍

```

1 {
2   "id": 600,
3   "name": "Annnora",
4   "age": 38,
5   "nationality": "Italian"
6 }
```

# Paso:	14
Descripción	Get An Actor 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/actors/findByld/2000?method=true

GET

http://localhost:8080/actors/findByld/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		

Response

Revisamos la salida:

Body

Cookies

Headers (5)

Test Results (1/1)

404 Not Found

16 ms

230 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "date": "09/05/2024 15:27:15",
3   "message": "Person not found"
4 }
```

# Paso:	15
Descripción	Delete An Actor 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/actors/deleteById?method=false&id=500

DELETE

http://localhost:8080/actors/deleteById?method=false&id=500

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

<input checked="" type="checkbox"/>	method	false	
<input checked="" type="checkbox"/>	id	500	
	Key	Value	Description

Revisamos la salida:

DELETE

ProjectFilms / Delete An Actor JPA

×

Response

Headers

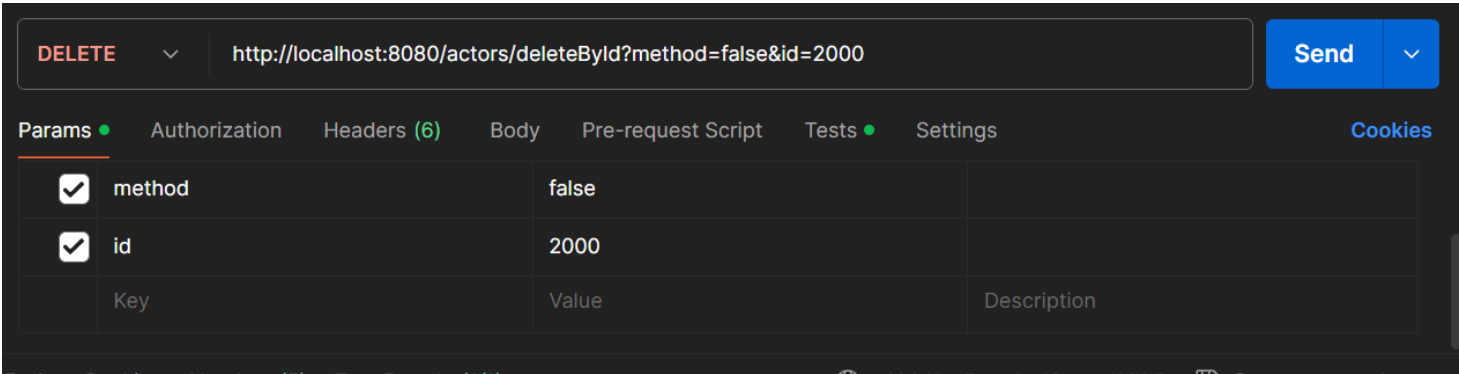
Request

204 37 ms 144 B

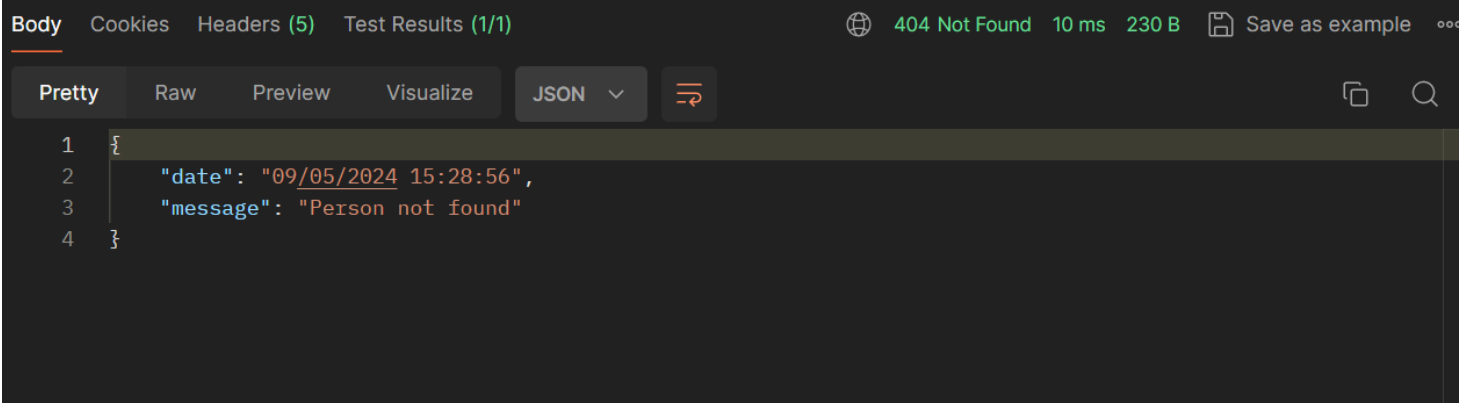
# Paso:	16
Descripción	Delete An Actor JPA Error
Resultado Esperado	La operación se completa correctamente obteniendo respuesta HTTP 404 Not Found. La operación en la base de datos se completa correctamente

Realizamos una petición REST de tipo DELETE la siguiente URL:

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



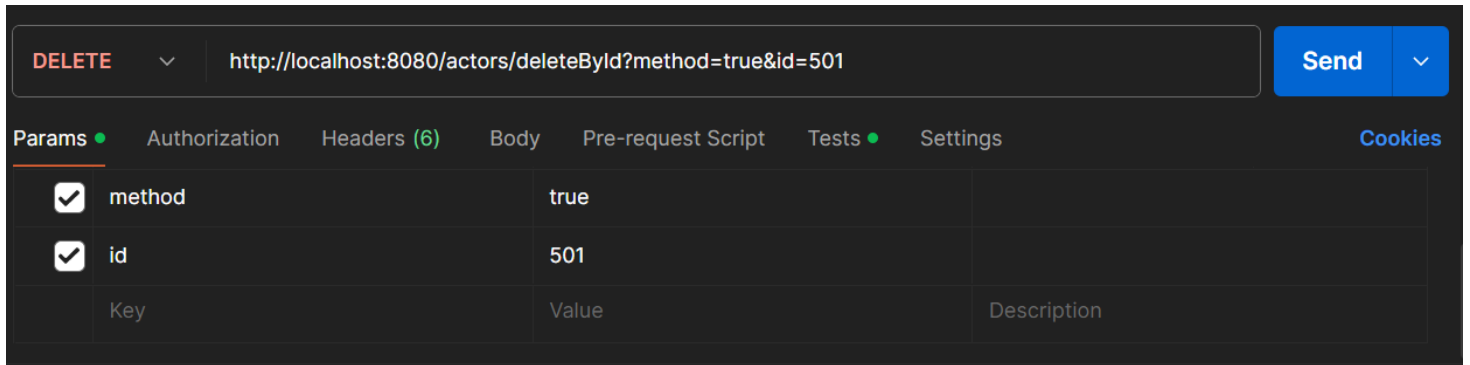
Revisamos la salida:



# Paso:	17
Descripción	Delete An Actor 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:

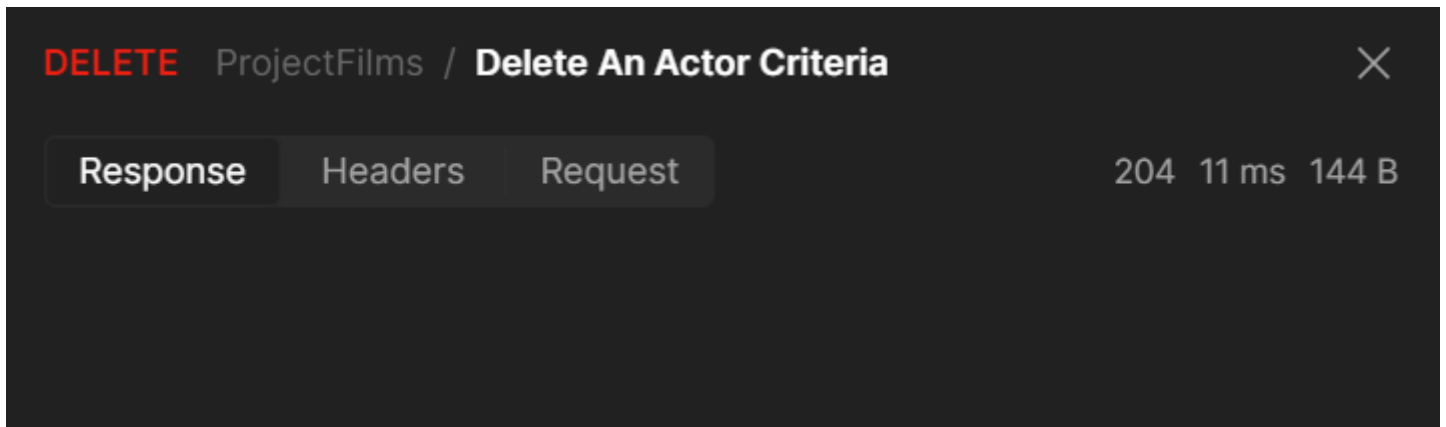
`http://localhost:8080/actors/deleteById?method=true&id=501`



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

- Method:** DELETE
- URL:** `http://localhost:8080/actors/deleteById?method=true&id=501`
- Params:**
 - ☒ **method**: true
 - ☒ **id**: 501
- Headers (6):** (Visible but not detailed in the image)
- Body:** (Empty)
- Pre-request Script:** (Empty)
- Tests:** (Empty)
- Settings:** (Visible but not detailed in the image)
- Cookies:** (Visible but not detailed in the image)

Revisamos la salida:



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

- Method:** DELETE
- Path:** ProjectFilms / Delete An Actor Criteria
- Response:** 204 11 ms 144 B
- Headers:** (Visible but not detailed in the image)
- Request:** (Visible but not detailed in the image)

# Paso:	18
Descripción	Delete An Actor 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/actors/deleteByld?method=true&id=2000
```

DELETE

http://localhost:8080/directors/deleteByld?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

14 ms

230 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "date": "09/05/2024 15:30:23",
3   "message": "Person not found"
4 }
```