API RESTFUL CON VALIDACIONES

SPRINGBOOT CON LOMBOK

María López Patón

Acceso a Datos

31/01/2025

ÍNDICE

1. LIBRO

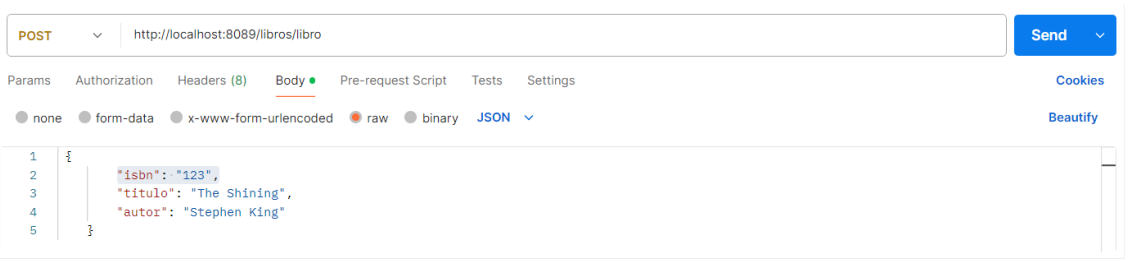
1.1 VALIDACIÓN ISBN

La validación del ISBN, según se ha especificado en el enunciado de la práctica, debe tener el siguiente formato: 000-0-000-00000-0

Código:

```java
@Id
@Size(max = 20)
@Column(name = "isbn", nullable = false, length = 20)
@NotBlank(message = "El  campo ISBN no puede estar vacío")
@Pattern(regexp = "^\\d{3}-\\d{1}-\\d{3}-\\d{5}-\\d{1}$", message = "Formato correcto:  000-0-000-00000-0")
private String isbn;
```

Método POST:



Respuesta:

```json
{
    "timestamp": "2025-01-29T12:14:39.801+00:00",
    "status": 400,
    "error": "Bad Request",
    "message": "Validation failed for object='libro'. Error count: 2",
    "path": "/libros/libro"
}
```

1.2 VALIDACIÓN TÍTULO

El título solo podrá contener caracteres alfanuméricos. En este caso, he añadido a la validación que también pueda contener espacios, para que se asemeje más al posible título de un libro real.

Código:

```java
@Size(max = 200, message = "La longitud máxima del título debe ser de 200 caracteres")
@NotNull(message = "No puede estar vacío")
@Pattern(regexp = "^[A-Za-z0-9 ]{1,200}$", message = "Solo admite caracteres alfanuméricos y espacios")
@Column(name = "titulo", nullable = false, length = 200)
private String titulo;
```

Método POST:

```json
{
    "isbn": "972-2-324-23185-1",
    "titulo": "The Shining%",
    "autor": "Stephen King"
}
```

Respuesta:


```
{
    "timestamp": "2025-01-29T12:21:18.316+00:00",
    "status": 400,
    "error": "Bad Request",
    "message": "Validation failed for object='libro'. Error count: 2",
    "path": "/libros/libro"
}
```

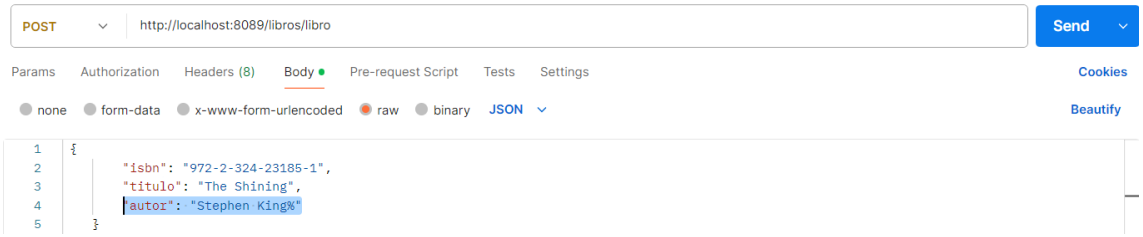

1.3 VALIDACIÓN AUTOR

La validación del autor es igual que la del campo título. En este caso, también he añadido a la validación que se puedan incluir espacios.

Código:

```
@Size(max = 100, message = "La longitud máxima del autor debe ser de 100 caracteres")
@NotNull(message = "No puede estar vacío")
@Pattern(regexp = "^[A-Za-z0-9 ]{1,200}$", message = "Solo admite caracteres alfanuméricos y espacios")
@Column(name = "autor", nullable = false, length = 100)
private String autor;
```

Método POST:

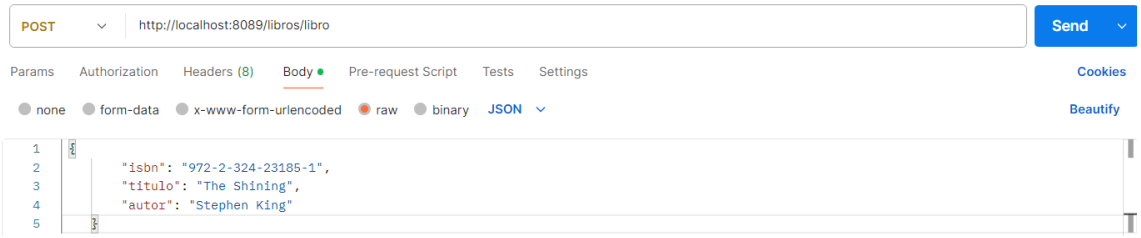


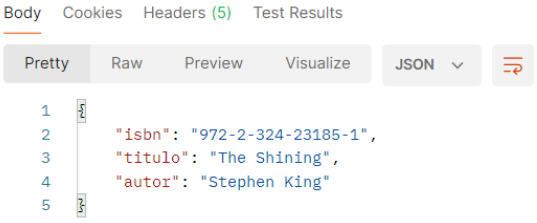
Respuesta:


```
{
    "timestamp": "2025-01-29T12:25:19.282+00:00",
    "status": 400,
    "error": "Bad Request",
    "message": "Validation failed for object='libro'. Error count: 2",
    "path": "/libros/libro"
}
```


1.4 POST CORRECTO DEL OBJETO LIBRO

Una vez mostradas todas las validaciones y la respuesta que genera cada una, este sería el método POST correcto con todos los campos validados y comprobados que son correctos.


POST
http://localhost:8089/libros/libro
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
{
"isbn": "972-2-324-23185-1",
"titulo": "The Shining",
"autor": "Stephen King"
}


Body
Cookies
Headers (5)
Test Results
Pretty
Raw
Preview
Visualize
JSON
{
"isbn": "972-2-324-23185-1",
"titulo": "The Shining",
"autor": "Stephen King"
}

2. USUARIO

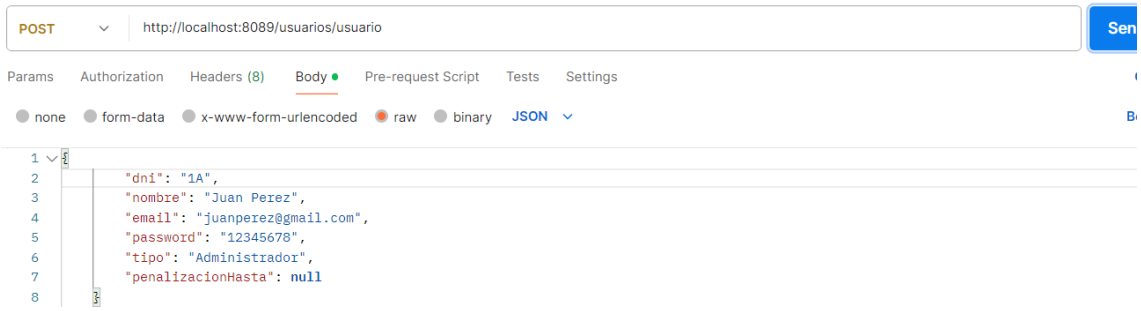
2.1 VALIDACIÓN DNI

El DNI deberá contener 8 números y 1 letra.

Código:

```
@Size(max = 15)
@NotNull(message = "El campo DNI no puede estar vacío")
@Pattern(regexp = "(\\d{8})([A-Z]{1})" , message = "El campo DNI tiene que tener este formato: 00000000X ")
@Column(name = "dni", nullable = false, length = 15)
private String dni;
```

Método POST:



Respuesta:

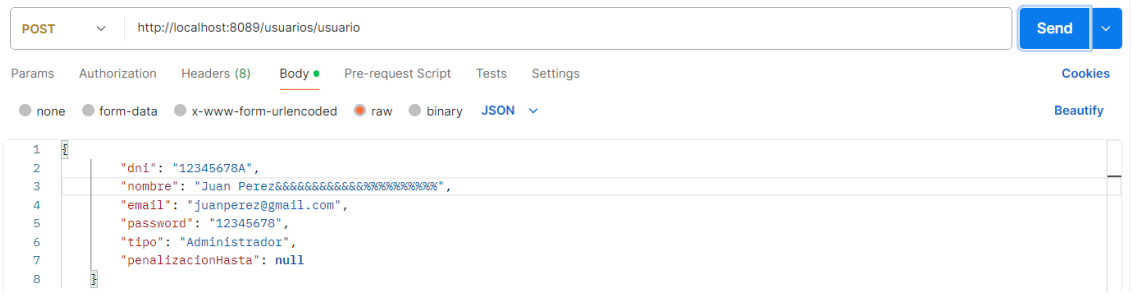
2.2 VALIDACIÓN NOMBRE

El nombre solo permitirá caracteres alfanuméricos

Código:

```
@Size(max = 100)
@NotNull
@Pattern(regexp = "^[A-Za-z0-9 ]{1,200}$", message = "Solo admite caracteres alfanuméricos y espacios")
@Column(name = "nombre", nullable = false, length = 100)
private String nombre;
```

Método POST:



Respuesta:


```
{
    "timestamp": "2025-01-29T12:53:45.539+00:00",
    "status": 400,
    "error": "Bad Request",
    "message": "Validation failed for object='usuario'. Error count: 1",
    "path": "/usuarios/usuario"
}
```

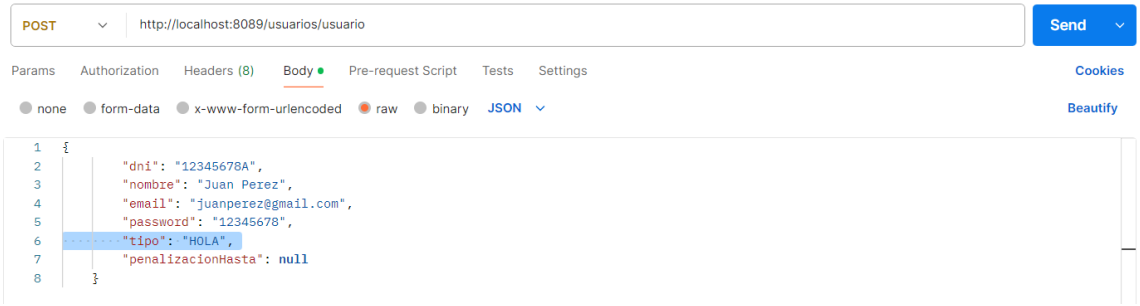

2.3 VALIDACIÓN TIPO

En la validación del tipo solo podrá escribirse Normal o Administrador

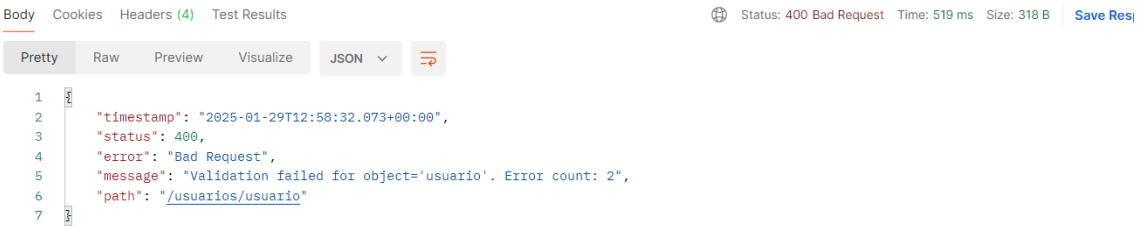
Código:

```
@NotNull(message = "El campo tipo no puede ser nulo")
@Lob
@Column(name = "tipo", nullable = false)
@Pattern(regexp = "^(Normal|Administrador)$", message = "El tipo debe ser uno de los siguientes valores: normal, administrador")
private String tipo;
```

Método POST:



Respuesta:



2.4 VALIDACIÓN EMAIL

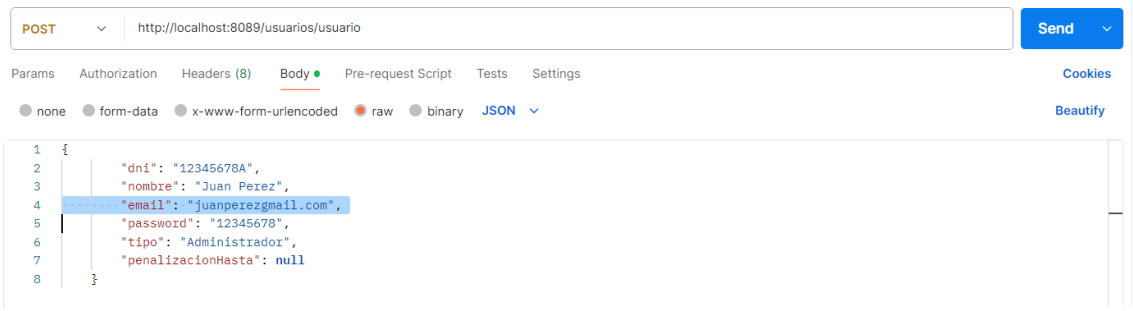
El email deberá tener primero caracteres alfanuméricos (de 1 a 50 caracteres), después un @ obligatorio, entre 1 y 30 letras, un punto obligatorio y finalmente entre 1 y 4 letras.

xxxxxxxxxx@xxxxxxxx.xxxx

Código:

```
@Size(max = 100)
@NotNull(message = "El campo email no puede ser nulo")
@Pattern(regexp = "[A-Za-z0-9]{1,50}@[A-Za-z]{1,30}.[A-Za-z]{1,4}", message = "El email debe tener este formato: xxxxx@xxx.xxx ")
@Column(name = "email", nullable = false, length = 100)
private String email;
```

Método POST:



Respuesta:

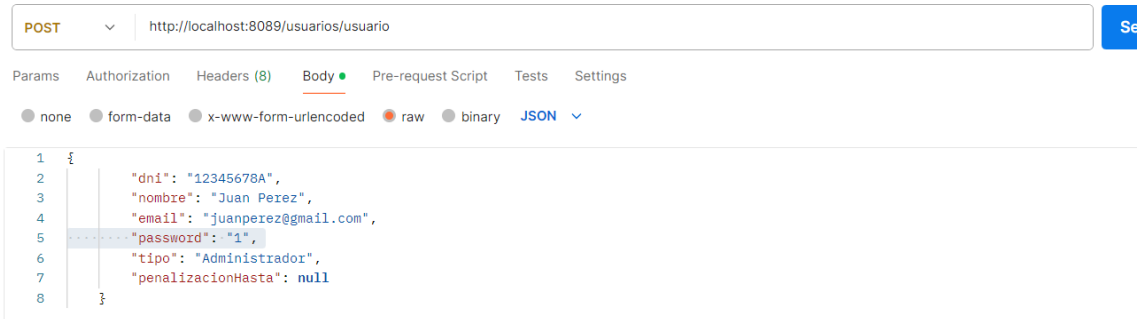
2.5 VALIDACIÓN CONTRASEÑA

La validación de la contraseña puede contener entre 4 y 12 caracteres alfanuméricos.

Código:

```
@Size(max = 255)
@NotNull(message = "El campo password no puede ser nulo")
@Pattern(regexp = "^[A-Za-z0-9]{4,12}$" , message = "La contraseña debe tener entre 4 y 12 caracteres alfanuméricos")
@Column(name = "password", nullable = false)
private String password;
```

Método POST:



Respuesta:

2.6 VALIDACIÓN PENALIZACIÓN

La penalización se podrá escribir como nula.

Método POST:

```
{
    "dni": "12345672A",
    "nombre": "Juan Perez",
    "email": "juanperez@gmail.com",
    "password": "12345678",
    "tipo": "Administrador",
    "penalizacionHasta": null
}
```

2.7 POST CORRECTO DEL OBJETO USUARIO

Una vez mostradas todas las validaciones y la respuesta que genera cada una, este sería el método POST correcto con todos los campos validados y comprobados que son correctos.

```
{
    "dni": "12345672A",
    "nombre": "Juan Perez",
    "email": "juanperez@gmail.com",
    "password": "1234567",
    "tipo": "Administrador",
    "penalizacionHasta": null
}
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 95 ms Size: 313 B

Pretty Raw Preview Visualize JSON

```
{
    "id": 536,
    "dni": "12345672A",
    "nombre": "Juan Perez",
    "email": "juanperez@gmail.com",
    "password": "1234567",
    "tipo": "Administrador",
    "penalizacionHasta": null
}
```

3. EJEMPLAR

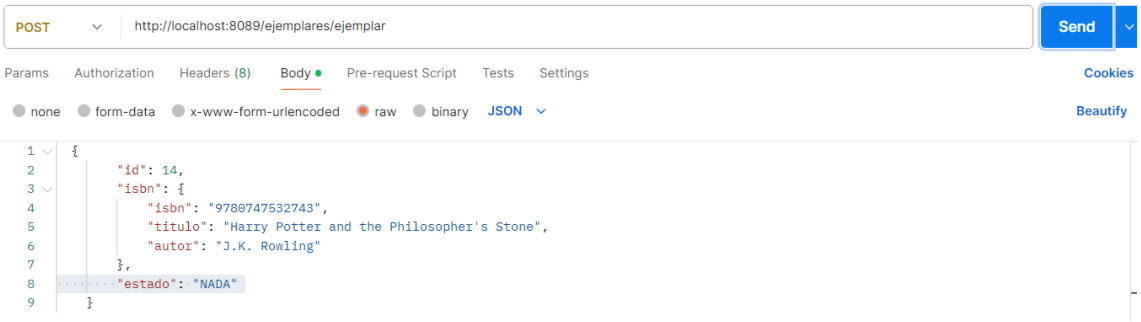
3.1 VALIDACIÓN ESTADO

La validación del estado consiste en comprobar si el estado es Disponible, Dañado o Prestado

Código:

```
@ColumnDefault("'Disponible'")
@Lob
@NotBlank(message = "El campo estado no puede estar vacio")
@Pattern(regexp = "^(Disponible|Prestado|Dañado)$", message = "El estado debe ser uno de los siguientes valores: disponible, prestado, dañado")
@Column(name = "estado")
private String estado;
```

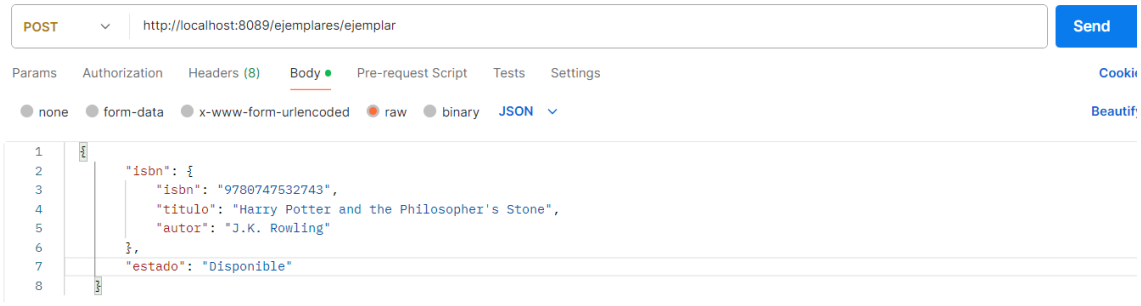
Método POST:



Respuesta:


```
{
    "timestamp": "2025-01-29T12:37:06.207+00:00",
    "status": 400,
    "error": "Bad Request",
    "message": "Validation failed for object='ejemplar'. Error count: 1",
    "path": "/ejemplares/ejemplar"
}
```


3.2 POST CORRECTO DEL OBJETO EJEMPLAR




```
{
    "id": 255,
    "isbn": {
        "isbn": "9780747532743",
        "titulo": "Harry Potter and the Philosopher's Stone",
        "autor": "J.K. Rowling"
    },
    "estado": "Disponible"
}
```