

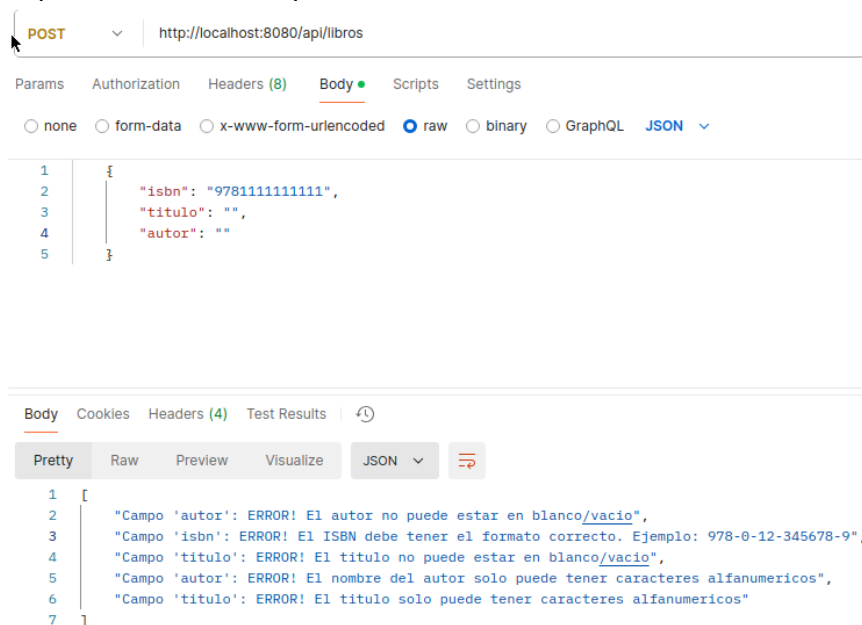
API RESTFUL con CRUD, caché y validaciones

Libros:

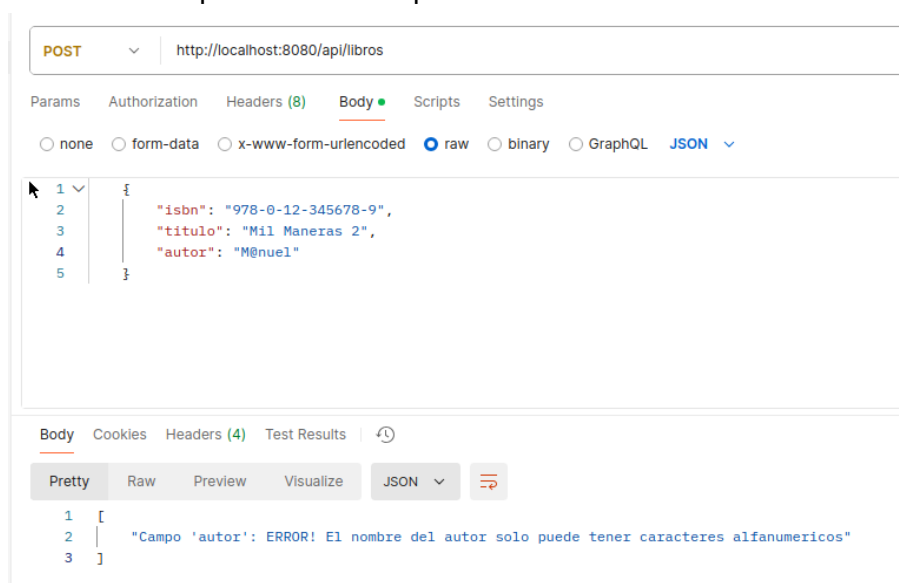
ISBN: formato correcto, ejemplo ISBN-13: 978-0-596-52068-7.

Título y autor: solo caracteres alfanuméricos de longitud máxima del campo de la base de datos.

Si ponemos los 3 campos de manera incorrecta al introducir un libro:



ISBN no sigue el formato correcto por lo que da error. Los campos titulo y autor no pueden estar vacios por lo que tambien da error. Si ponemos el isbn correctamente, el titulo correctamente pero en el autor ponemos un simbolo:



Vemos que da error en el autor ya que solo admite caracteres alfanumericos (igual para titulo).

Ejemplares:

Estado: solo puede contener uno de los siguientes valores: disponible, prestado, dañado.

POST http://localhost:8080/api/ejemplares

Params Authorization Headers (8) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "id": null,
3   "libro": {
4     "isbn": "978-0-12-345678-9",
5     "titulo": "Mil Maneras 2",
6     "autor": "Williams Rey"
7   },
8   "estado": "No disponible"
9 }
10
11
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

```
1
2 "Campo 'estado': ERROR! El ejemplar solo puede ser: Disponible | Prestado | Danado"
3
```

Da error ya que estamos poniendo un estado invalido

POST http://localhost:8080/api/ejemplares

Params Authorization Headers (8) Body Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "id": null,
3   "libro": {
4     "isbn": "978-0-12-345678-9",
5     "titulo": "Mil Maneras 2",
6     "autor": "Williams Rey"
7   },
8   "estado": "Disponible"
9 }
10
11
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 9,
3   "libro": {
4     "isbn": "978-0-12-345678-9",
5     "titulo": "Mil Maneras 2",
6     "autor": "Williams Rey"
7   },
8   "estado": "Disponible"
9 }
```

Si lo introducimos correctamente se añade el ejemplar a la base de datos

Usuarios:

DNI: validar mediante método específico.

Nombre: solo caracteres alfanuméricos de longitud máxima del campo de la base de datos.

Tipo: solo puede contener uno de los siguientes valores: normal, administrador.

Email: Solo emails que acaben en un determinado proveedor de correo, por ejemplo gmail.
Ejemplo de Pattern - ([A-Za-z0-9]{1,50}@gmail.com)

Password: Cadena alfanumérica de entre 4 y 12 caracteres.

Penalización Hasta: por ejemplo aquí sí que admitiría nulos.

The screenshot displays a REST client interface. The top section shows a POST request to the URL `http://localhost:8080/api/usuarios`. The 'Body' tab is selected, showing a JSON object with the following fields: `"dni": "1234568A"`, `"nombre": ""`, `"tipo": "otro"`, `"email": "perez@example.com"`, and `"password": "23A"`. The bottom section shows the response in 'Pretty' format, which is a JSON array of error messages: `[{"Campo": "password", "Mensaje": "ERROR! La contraseña debe ser una cadena de entre 4 y 12 caracteres"}, {"Campo": "nombre", "Mensaje": "ERROR! El nombre no puede estar en blanco/vacio"}, {"Campo": "tipo", "Mensaje": "ERROR! El usuario solo puede ser: normal | administrador"}, {"Campo": "dni", "Mensaje": "ERROR! El dni tiene que contener 8 numeros seguido de 1 letra"}, {"Campo": "password", "Mensaje": "ERROR! La contraseña solo puede tener caracteres alfanumericos"}, {"Campo": "email", "Mensaje": "ERROR! El email tiene que tener un formato correcto. Ejemplo: prueba@gmail.com"}]`

Prueba de los campos incumpliendo las validaciones correspondientes