***PRUEBAS DE LAS VISTAS DESARROLLADAS EN DJANGO***

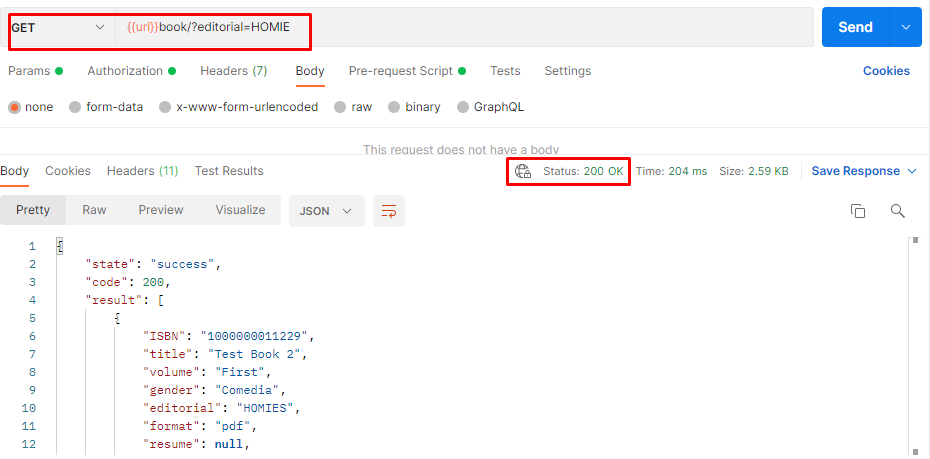
**HAPPY PATH**

**HEROKU**

**BOOK**

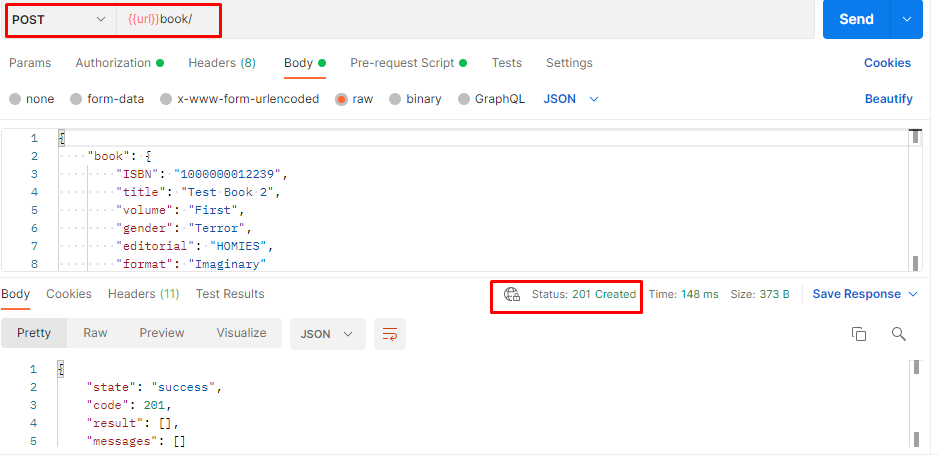
1. Get Book Endpoint.

Se envía el ISBN como Request en el Body y se obtiene un status: 200 OK en el Response con toda la información del libro.



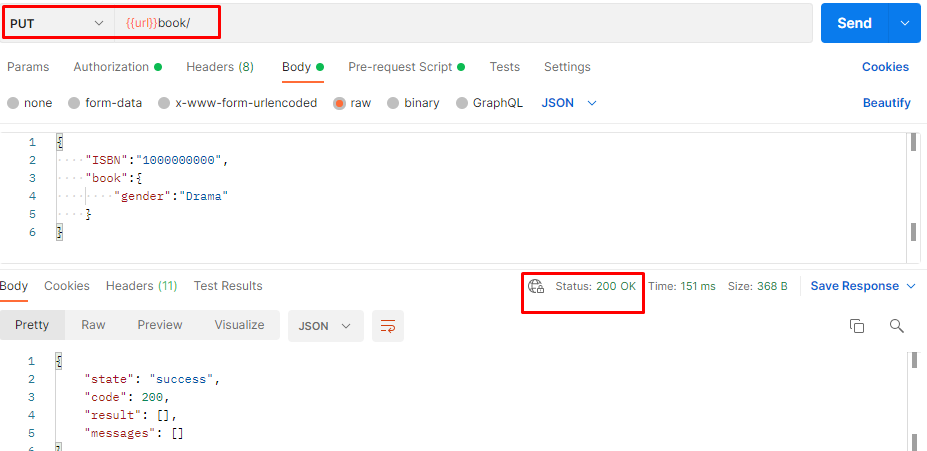
2. POST Create Book Endpoint

Se envía la información de creación del libro en el Body y se obtiene un status: 201 Created en el Response.



3. PUT Modify Book Endpoint

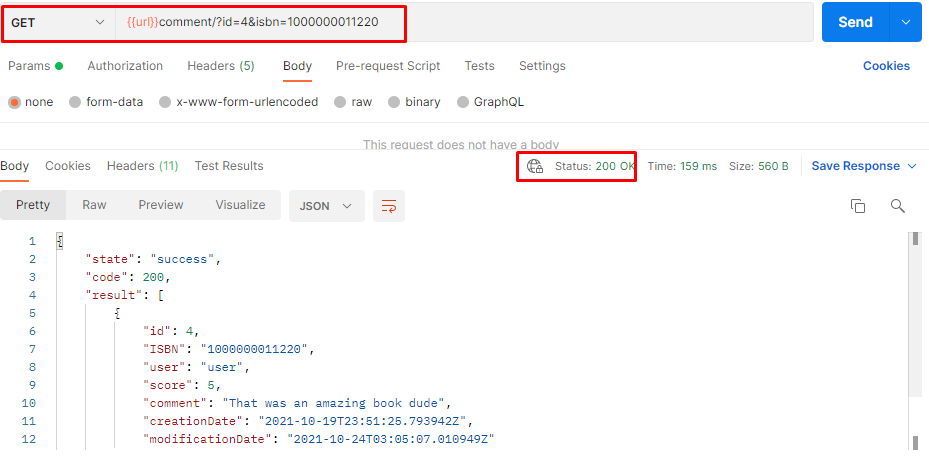
Se envía actualización del genéro de libro identificado según su ISBN y se obtiene un response de 200 OK con state “success”.



**COMMENT**

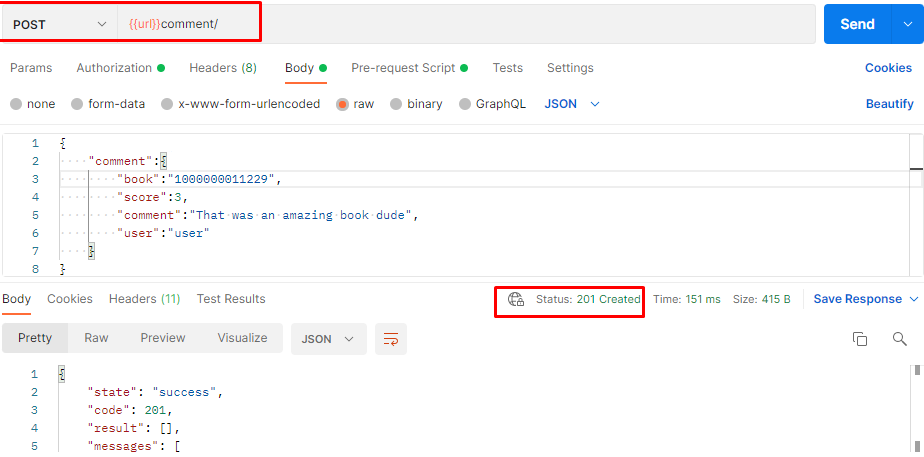
1. GET Comment Endpoint

Se envía un GET Request con el id del libro y el ISBN como parámetros en el endpoint a probar y se obtiene un 200 ok más el comentario que tiene el libro.



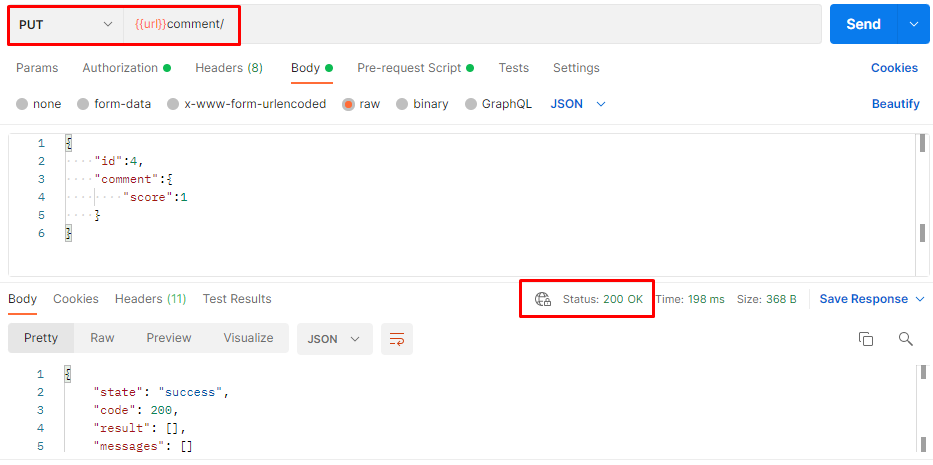
2. POST Create Comment Endpoint

Se envía un POST Request para crear un comentario de un libro y se obtiene un response de 200 ok.



3. PUT Update Comment Endpoint

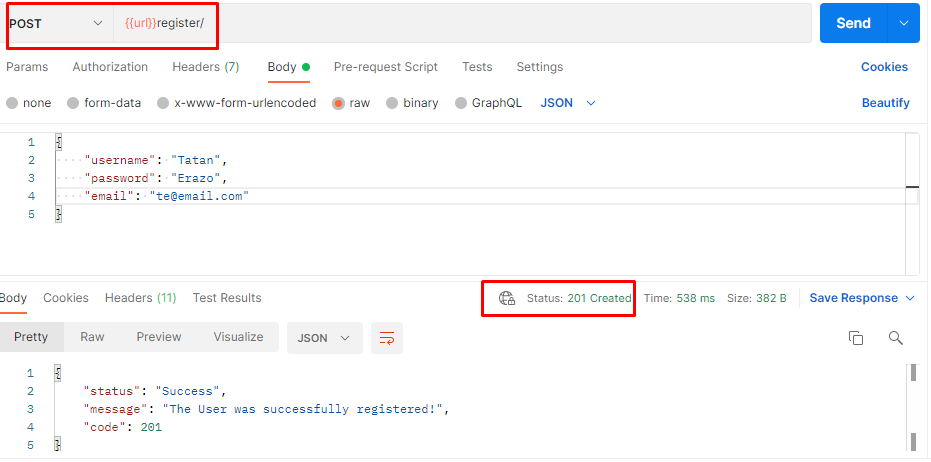
Se envía un PUT Request para actualizar los comentarios de un libro identificado con su Id. Se obtiene un Response 200 ok.



**USER**

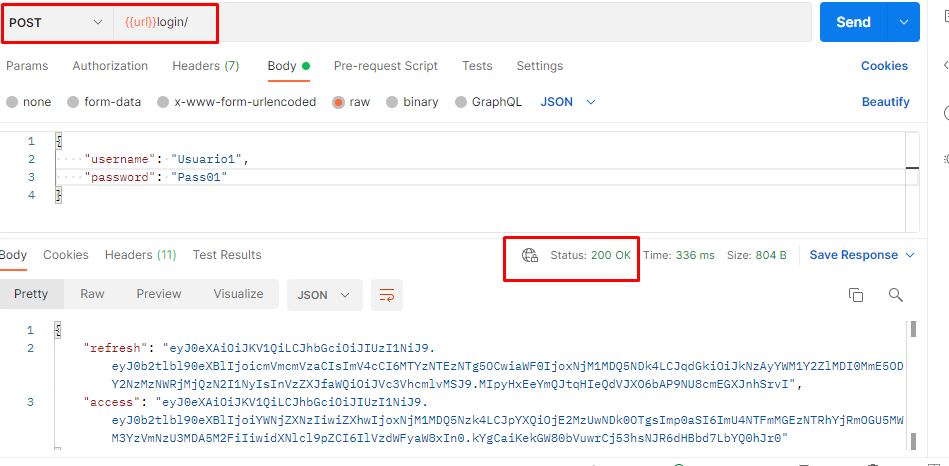
1. POST Register Endpoint.

Se envía un POST request para que un usuario se registre en la e\_libroteka y se obtiene un status: 201 Created y un código de registro.



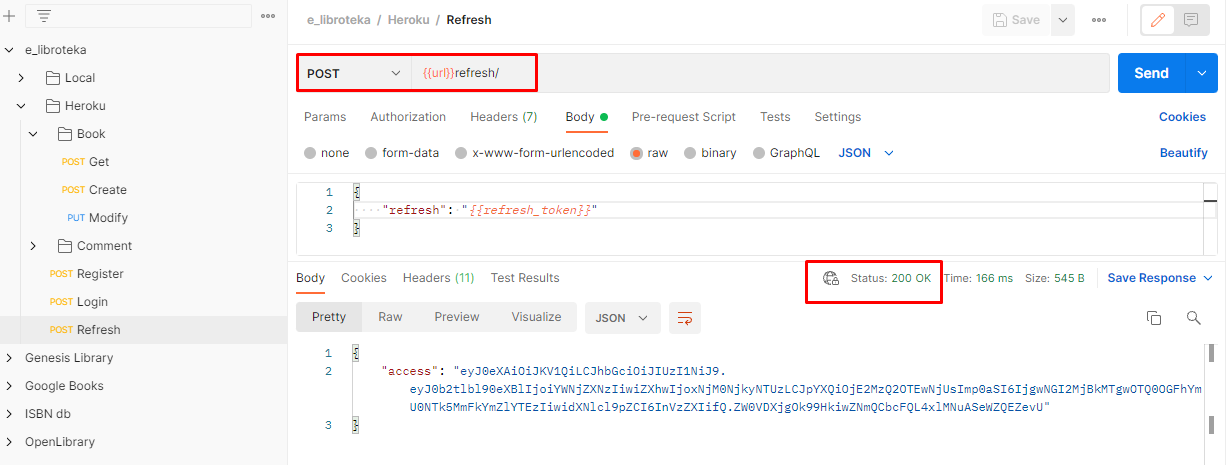
2. POST Login Endpoint.

Se envía un POST request para que un usuario se loguee con el username y password en la página web de la e\_libroteka y se obtiene un status: 200 OK.



3. POST Refresh Endpoint

Se envía un POST Request para refresca el token que autoriza los cambios en Heroku. Se obtine un response 200 OK con la clave de acceso de jwt token.

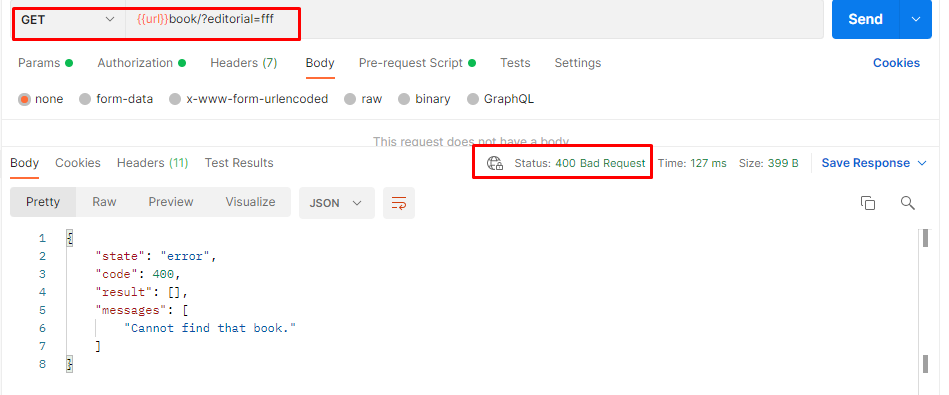


**PRUEBAS DE ERROR HANDLING**

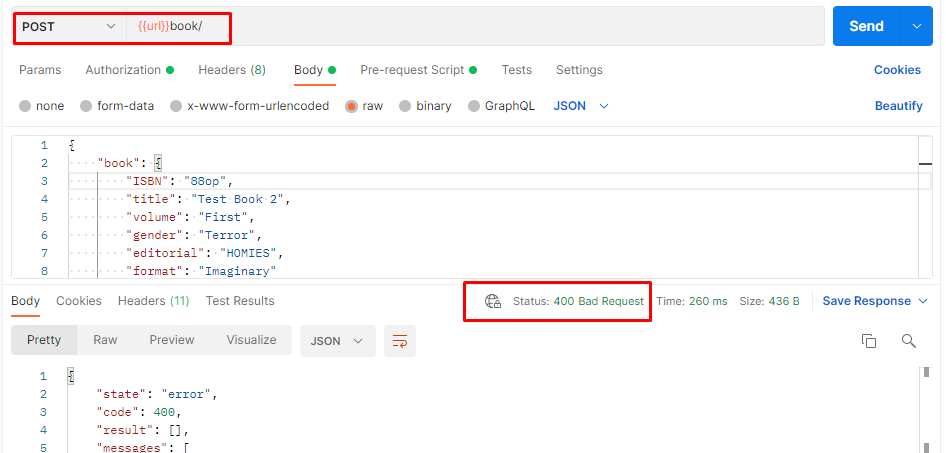
**BOOK**

1. POST Get Book Endpoint.

Se envía un parámetro inválido de editorial en el endpoint a probar y se obtiene un status: 400 – bad request

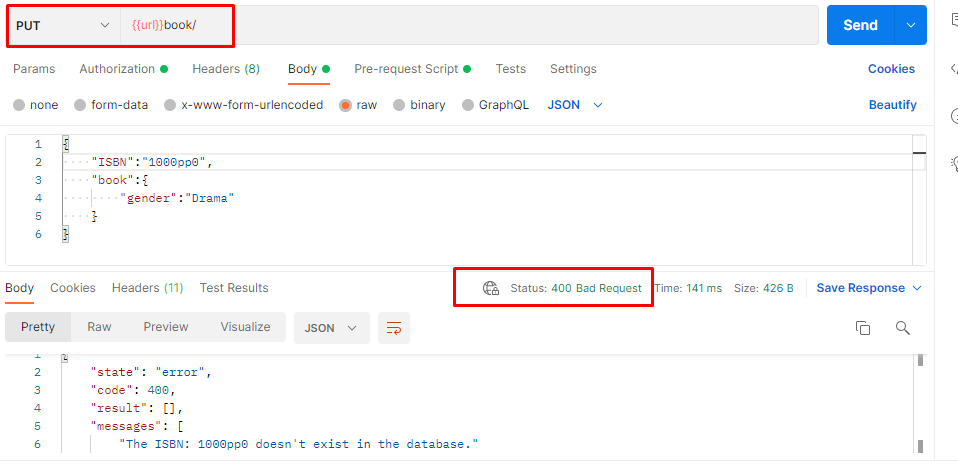
2. Post Create Book Endpoint

Se envía la información de creación del libro en el Body con un ISBN no existente y se obtiene un status: 400 Bad Request en el Response.



3. PUT Modify Book Endpoint

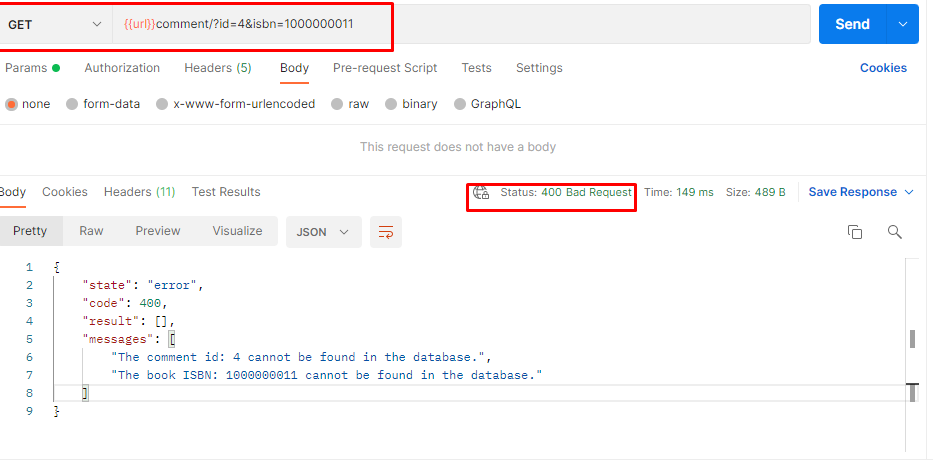
Se envía actualización del genéro con un ISBN inválido y se obtiene un response de 400.



**COMMENT**

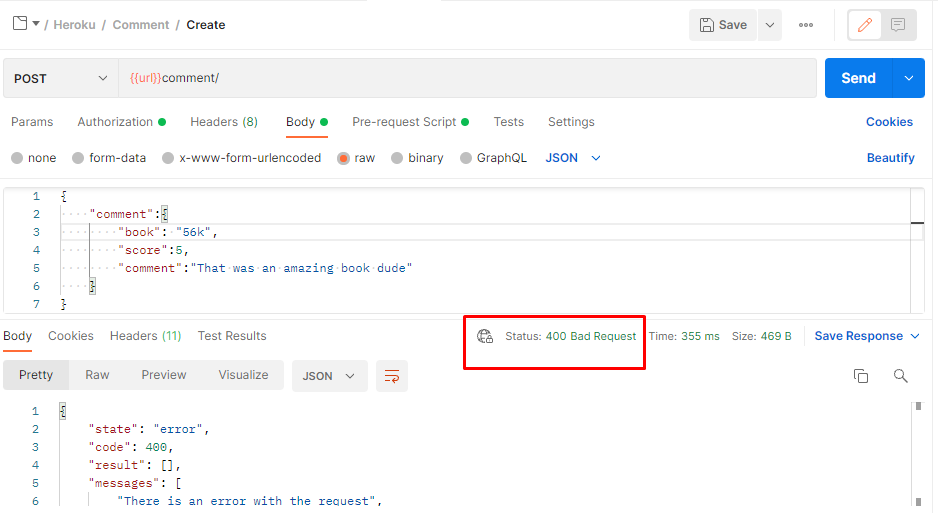
1. GET Comment Endpoint

Se envía un GET Request con un id y también ISBN invalidos como parámetros en el endpoint a probar y se obtiene un 400 Bad Request.



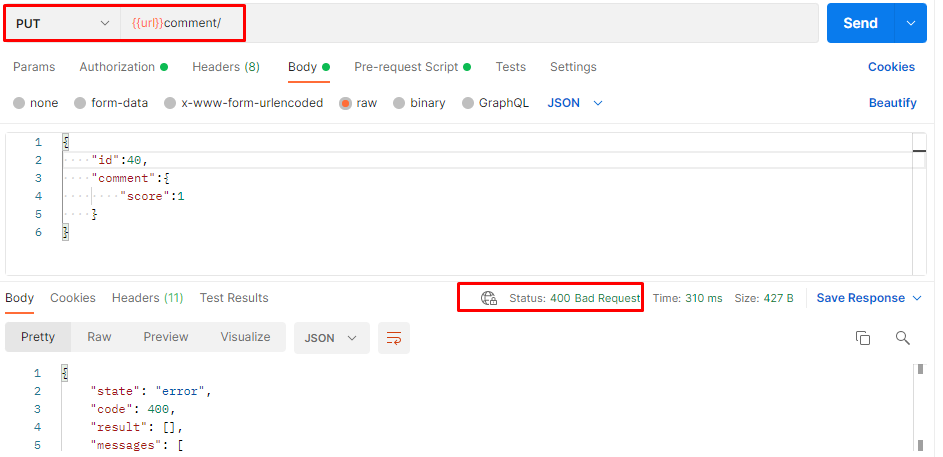
2. POST Create Comment Endpoint

Se envía un POST Request para crear un comentario de un libro sobre un ISBN no válido (no existente) y se obtiene un response de 400 bad request.



3. PUT Update Comment Endpoint

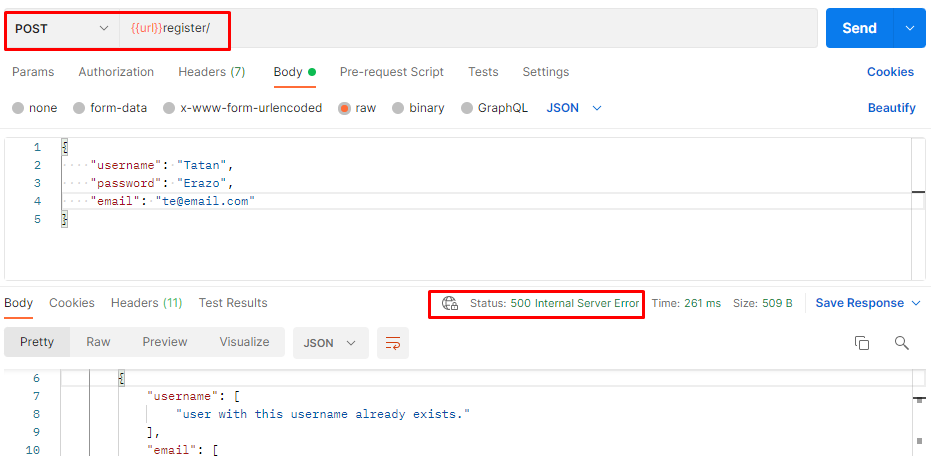
Se envía un PUT Request para actualizar los comentarios de un libro identificado con un Id invalido. Se obtiene un Response400 Bad Request.



**USER**

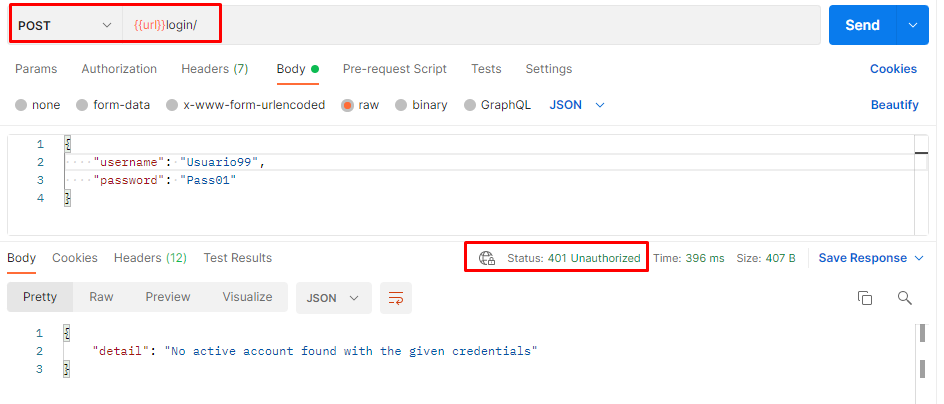
1. POST Register Endpoint.

Se envía un POST request con el username y email de un usuario ya existente para que se registre de nuevo en la e\_libroteka y se obtiene un status: 500 Internal Server Error con un mensaje que indica: “there is an error with the request. User with this username already exists”.



2. POST Login Endpoint.

Se envía un POST request para que un usuario se loguee con un username y password invalidos en la página web de la e\_libroteka y se obtiene un status: 401 Unauthorized con un mensaje que indica “No active account found with the given credentials”.



4. POST Refresh Endpoint

Se envía un POST Request para refrescar el token que autoriza los cambios en Heroku. Se obtiene un response 401 Unauthorized con un mensaje que dice: “token not valid”.

