

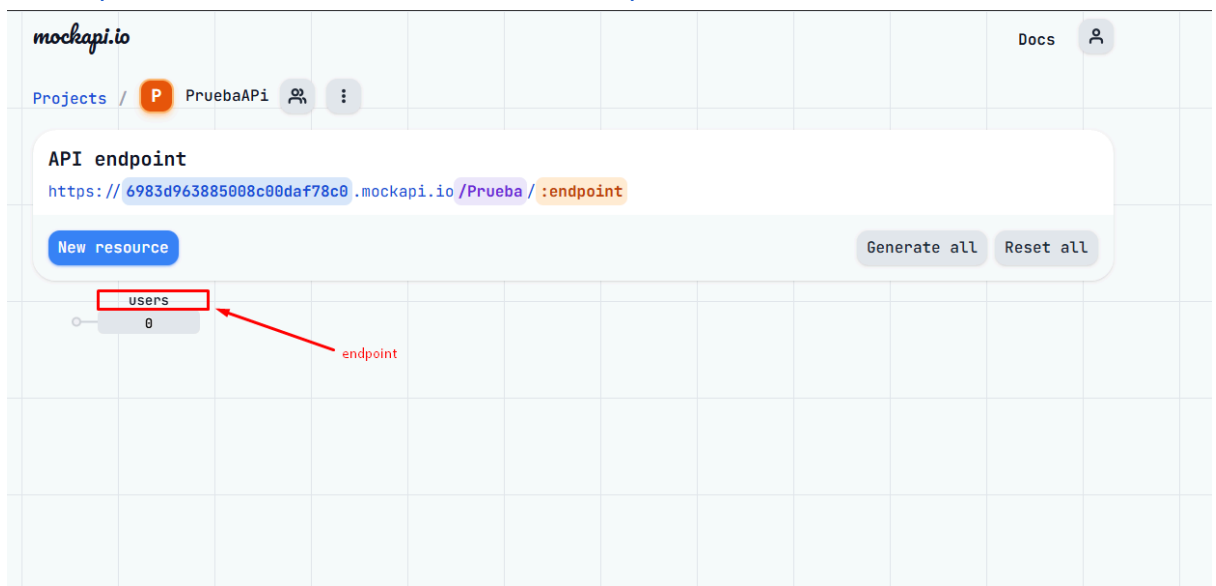
Para el desarrollo de la prueba de la API, inicialmente se tenía previsto utilizar la API pública <https://reqres.in/api/>

. Sin embargo, durante la ejecución de los casos de prueba se identificó una limitación en su funcionamiento: aunque la petición POST se ejecutaba correctamente y retornaba un identificador (id) en la respuesta, dicho registro no era persistente en la API. Como consecuencia, al realizar posteriormente la petición GET utilizando el id generado en el POST, la respuesta retornaba un JSON vacío, ya que el recurso creado no se encontraba almacenado.

Debido a esta limitación, no fue posible validar correctamente el flujo completo de creación y consulta de un usuario, tal como lo requería el ejercicio. Para dar solución a esta situación y cumplir con los objetivos de la prueba, se procedió a utilizar una nueva API, la cual sí permitió la correcta persistencia de los datos y la ejecución exitosa de las peticiones POST y GET, permitiendo consultar el mismo registro previamente creado.

A continuación, se adjuntan las evidencias correspondientes a las peticiones realizadas durante la prueba, las cuales incluyen: POST, GET, PUT y DELETE.

API: <https://6983d963885008c00daf78c0.mockapi.io/Prueba/users>



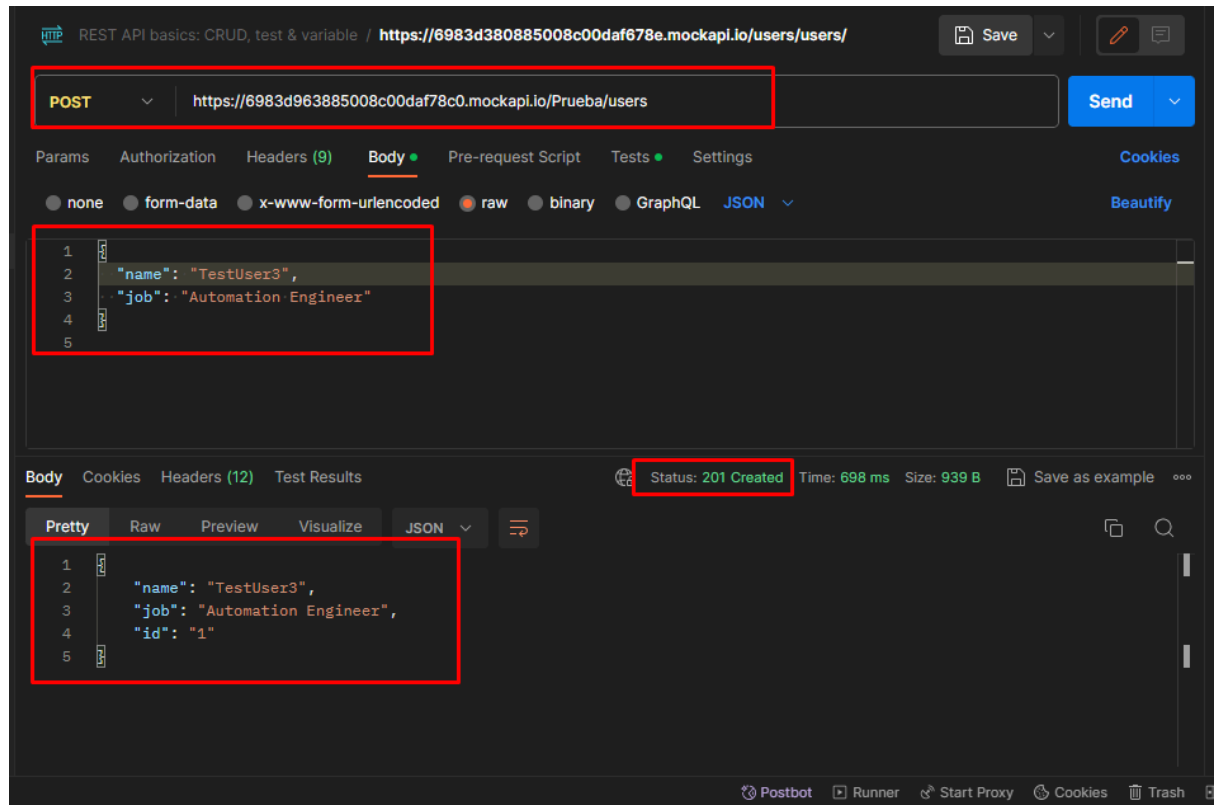
Caso 1: Ejecución petición tipo Post a la API.

Resultado esperado: Se espera que, al realizar la petición de tipo POST mediante la herramienta Postman, la API procese correctamente la solicitud y retorne un código de estado HTTP 201 . Asimismo, se espera que se genere un nuevo registro en el sistema a partir de la siguiente estructura JSON enviada en el cuerpo de la petición:

```
{
  "name": "TestUser3",
  "job": "Automation Engineer"
}
```

Como resultado, la respuesta debe incluir un identificador único (id), el cual permitirá consultar, actualizar o eliminar posteriormente el registro creado, validando así la correcta creación y persistencia del recurso en la API.

Evidencias:



Resultado actual: Se ejecutó correctamente una petición de tipo POST al endpoint /Prueba/users mediante la herramienta Postman, enviando en el cuerpo de la solicitud la siguiente estructura JSON:

```
{
  "name": "TestUser3",
  "job": "Automation Engineer"
}
```

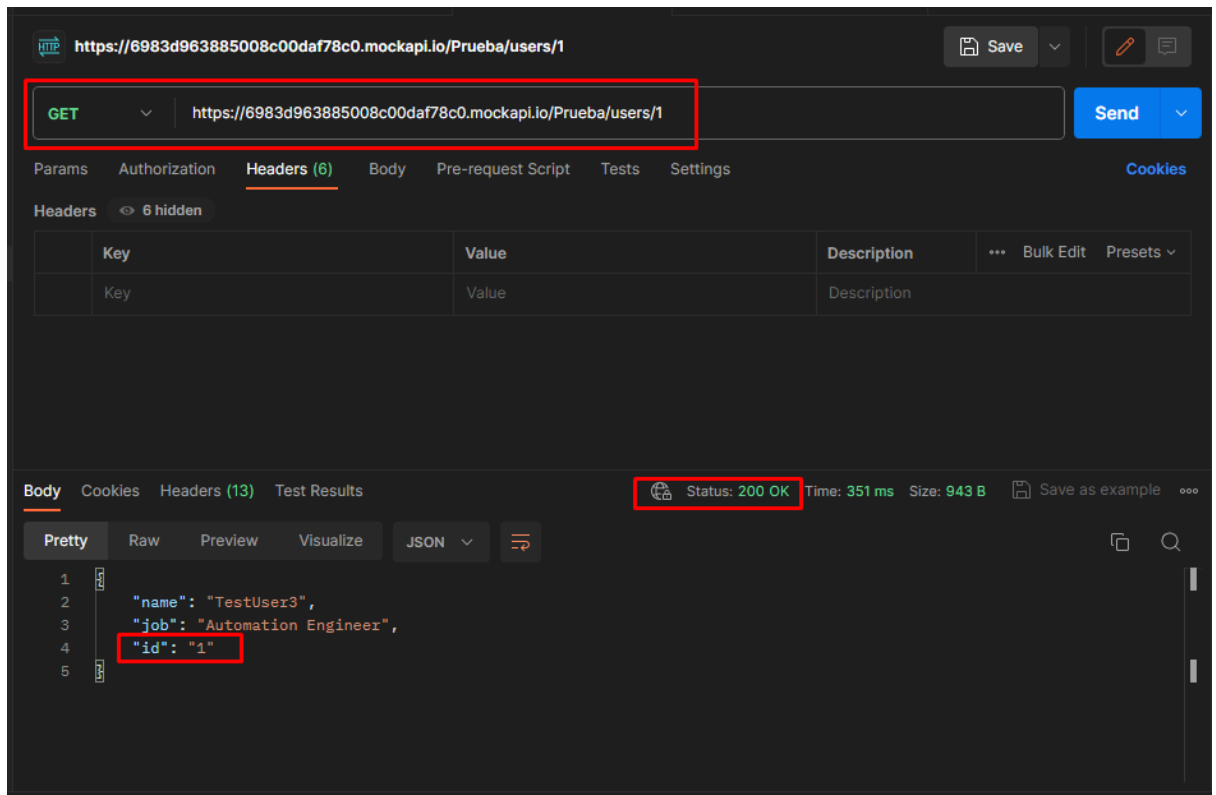
La API respondió con un código de estado HTTP 201, lo que confirma que el recurso fue creado de manera exitosa. En el cuerpo de la respuesta se retornó un objeto en formato JSON que incluye los datos enviados junto con un identificador único (id: "1").

Caso 2: Ejecución petición tipo GET al endpoint "/users".

Resultado esperado: Se espera que una vez capturado el id que se creó en la petición anterior, se pueda realizar la petición tipo GET al endpoint "/users"

añadiendo el id, de tal forma que al enviar la petición, se obtenga como respuesta el registro creado previamente con los mismos datos insertados.

Evidencias:



Resultado actual: Se ejecutó correctamente una petición GET al endpoint /Prueba/users/1, obteniendo una respuesta exitosa con un código de estado HTTP 200 (OK), lo que confirma que el recurso solicitado se encuentra disponible y accesible en la API.

En el cuerpo de la respuesta se retorna un objeto en formato JSON con la información correspondiente al usuario consultado, evidenciando que los datos que se visualizan son los mismos que fueron previamente insertados en la petición de tipo POST.

name: "TestUser3"

job: "Automation Engineer"

id: "1"

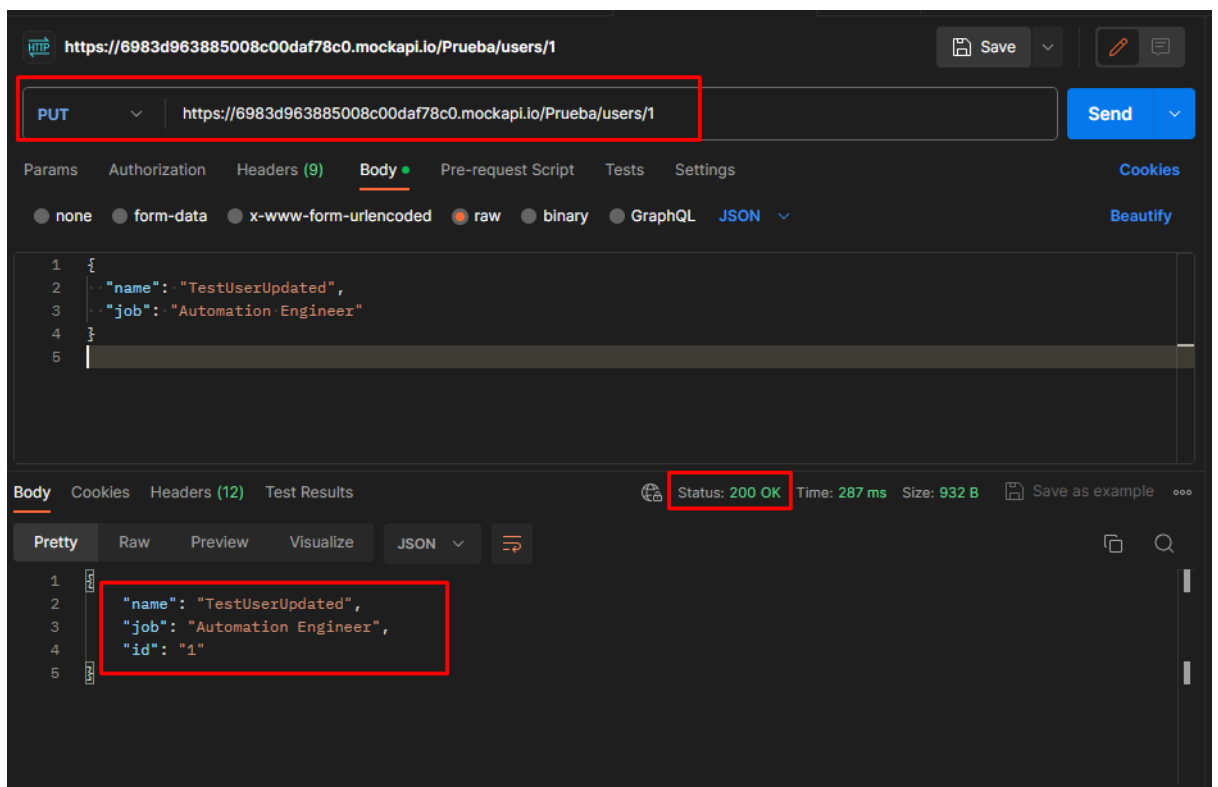
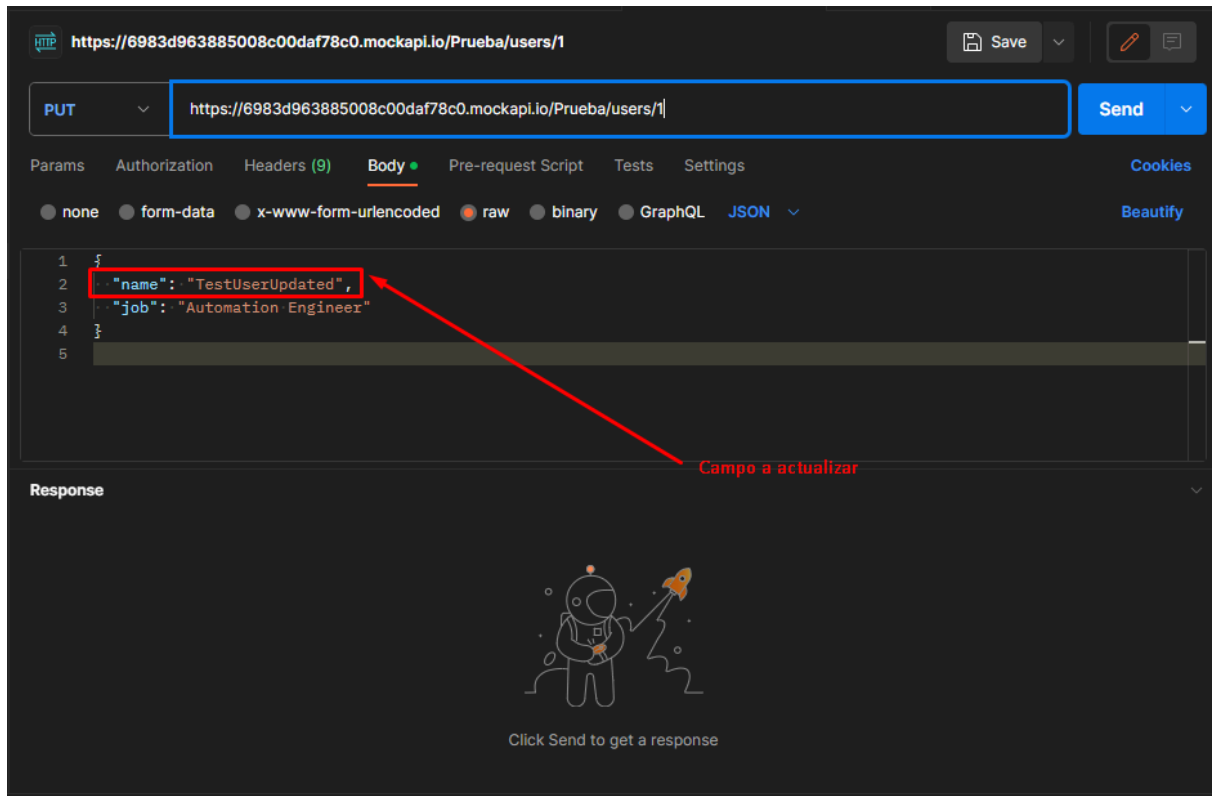
Caso 3: Ejecutar petición tipo PUT al endpoint /users.

Resultado esperado: Al ejecutar una petición de tipo PUT al endpoint correspondiente al usuario previamente creado /Prueba/users/1, se espera que la API procese correctamente la solicitud y actualice la información del recurso indicado.

La respuesta debe retornar un código de estado HTTP 200, confirmando que la actualización del registro fue realizada de manera exitosa.

Adicionalmente, se espera que al realizar una petición GET posterior utilizando el mismo identificador, la información retornada corresponda a los valores actualizados, lo que valida la correcta persistencia de los cambios realizados mediante la operación PUT.

Evidencias:



Resultado actual: Se ejecutó correctamente una petición de tipo PUT al endpoint `/Prueba/users/1`, enviando en el cuerpo de la solicitud un JSON con los campos actualizados del usuario.

La API respondió con un código de estado HTTP 200 (OK), lo que indica que la solicitud fue procesada exitosamente. En el cuerpo de la respuesta se retornó un objeto en formato JSON que refleja los datos actualizados del usuario, manteniendo el mismo identificador (id) y mostrando los nuevos valores enviados en la petición:

name: "TestUserUpdated"

job: "Automation Engineer"

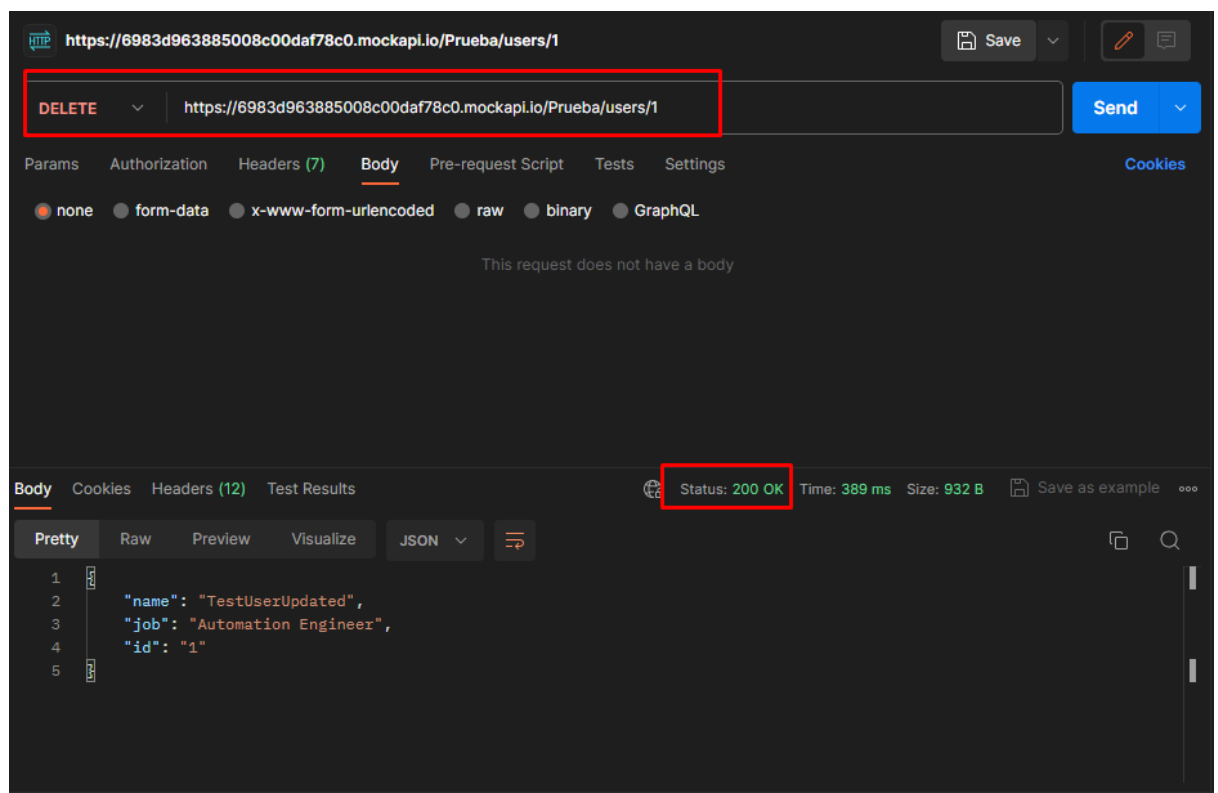
id: "1"

Caso 4: Ejecución petición tipo DELETE.

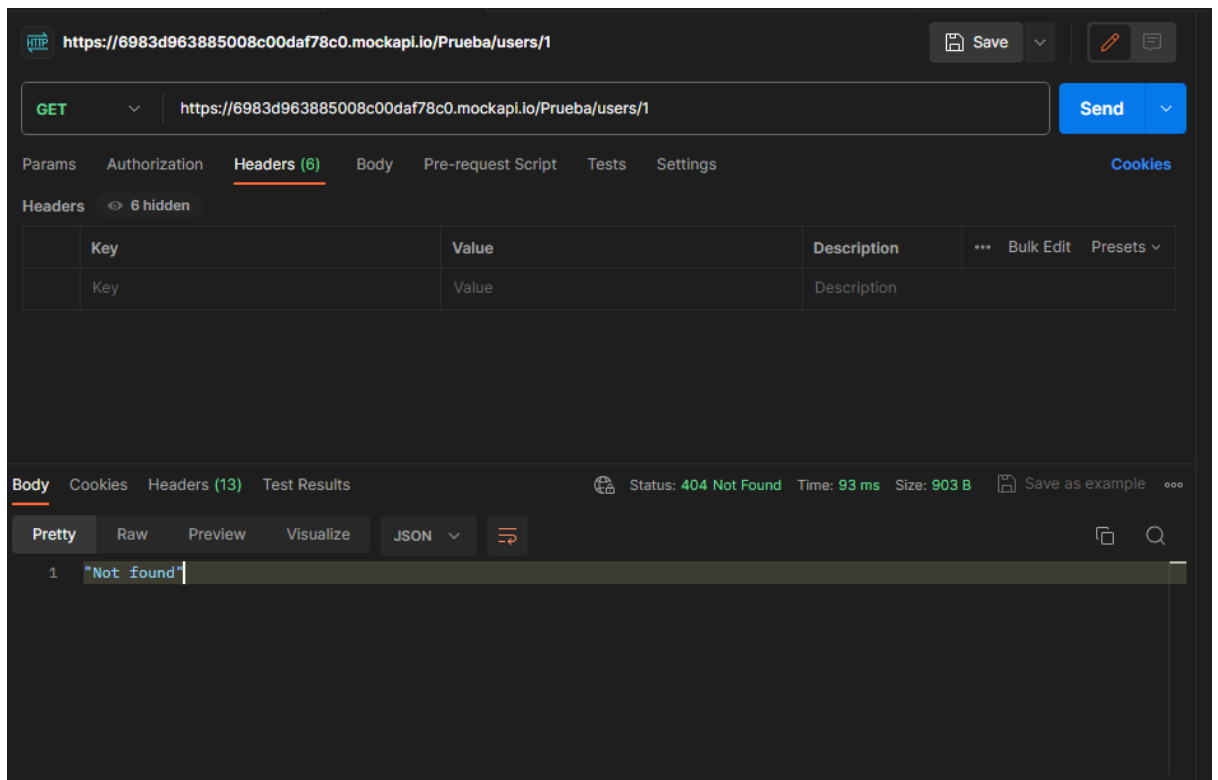
Resultado esperado: Al ejecutar una petición DELETE sobre el usuario existente con id = 1, se espera que la API elimine el recurso indicado y retorne un código de estado HTTP 200 , confirmando la eliminación exitosa.

Asimismo, se espera que al intentar consultar nuevamente el usuario mediante una petición GET con el mismo id, la API retorne un error 404 (Not Found) o un cuerpo vacío, evidenciando que el registro ya no existe en el sistema y que la operación de eliminación fue efectiva.

Evidencias:



Ejecución petición Get después de ejecutar la petición DELETE al mismo endpoint.



Resultado Actual: Se ejecutó correctamente una petición de tipo DELETE al endpoint `/Prueba/users/1`, correspondiente a un usuario previamente existente en la API.

La API respondió con un código de estado HTTP 200, lo que indica que la solicitud fue procesada exitosamente y que el recurso fue eliminado de manera correcta.

Posteriormente al ejecutar una petición GET utilizando el mismo identificador (id: 1), el sistema no retorna información del usuario, validando que el registro fue eliminado correctamente.