**Модульное тестирование методов API**

1. Получите список доступных методов API

2. Для одного из методов сформируйте список позитивных и негативных тест кейсов.

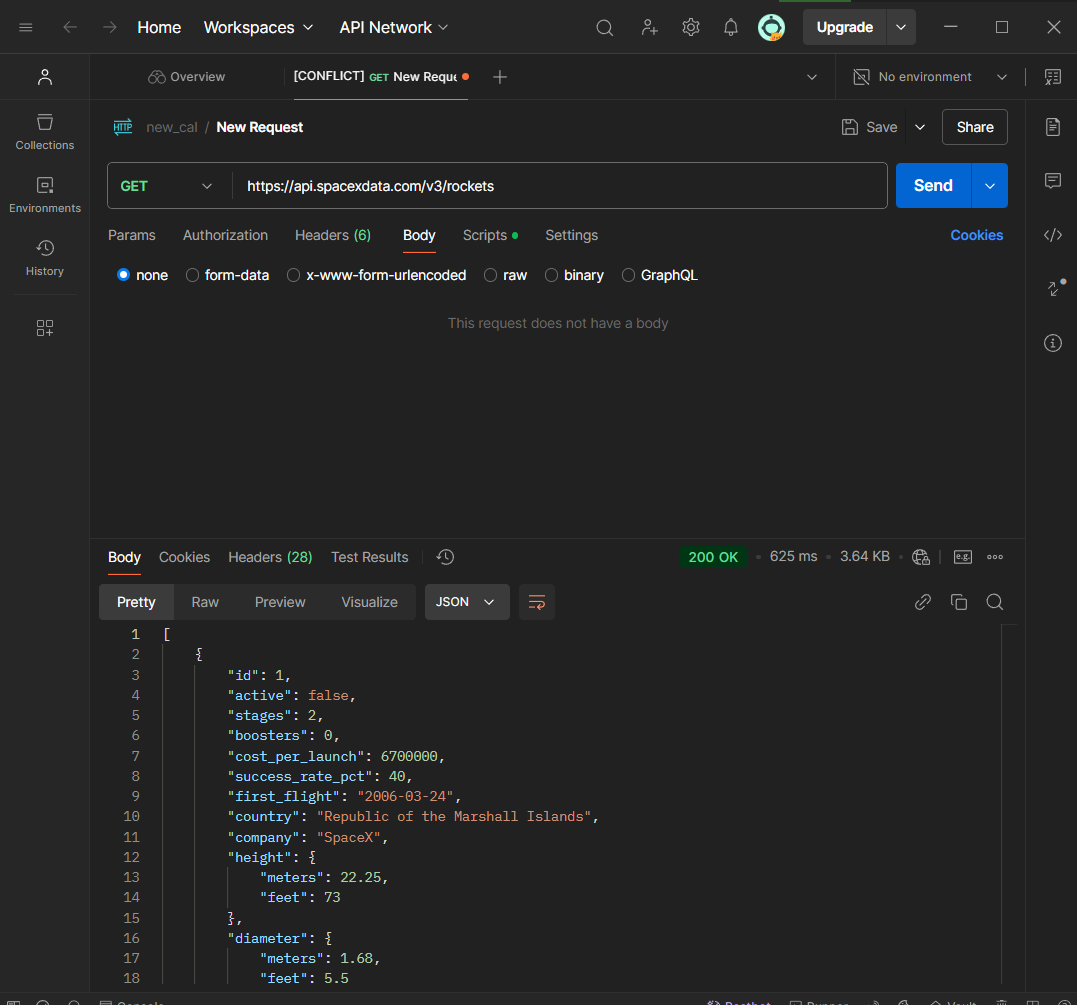
**Позитивные** тест кейсы:

**Тест 1**: получить информацию обо всех ракетах

Метод: GET

URL: <https://api.spacexdata.com/v3/rockets>

Ожидаемый результат: Код: 200, статус: OK, массив ракет.



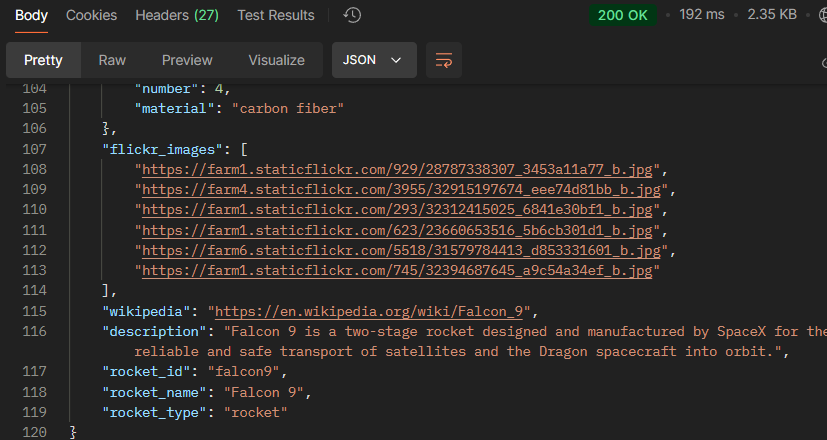
Фактический результат: Код: 200, статус: OK, массив ракет.

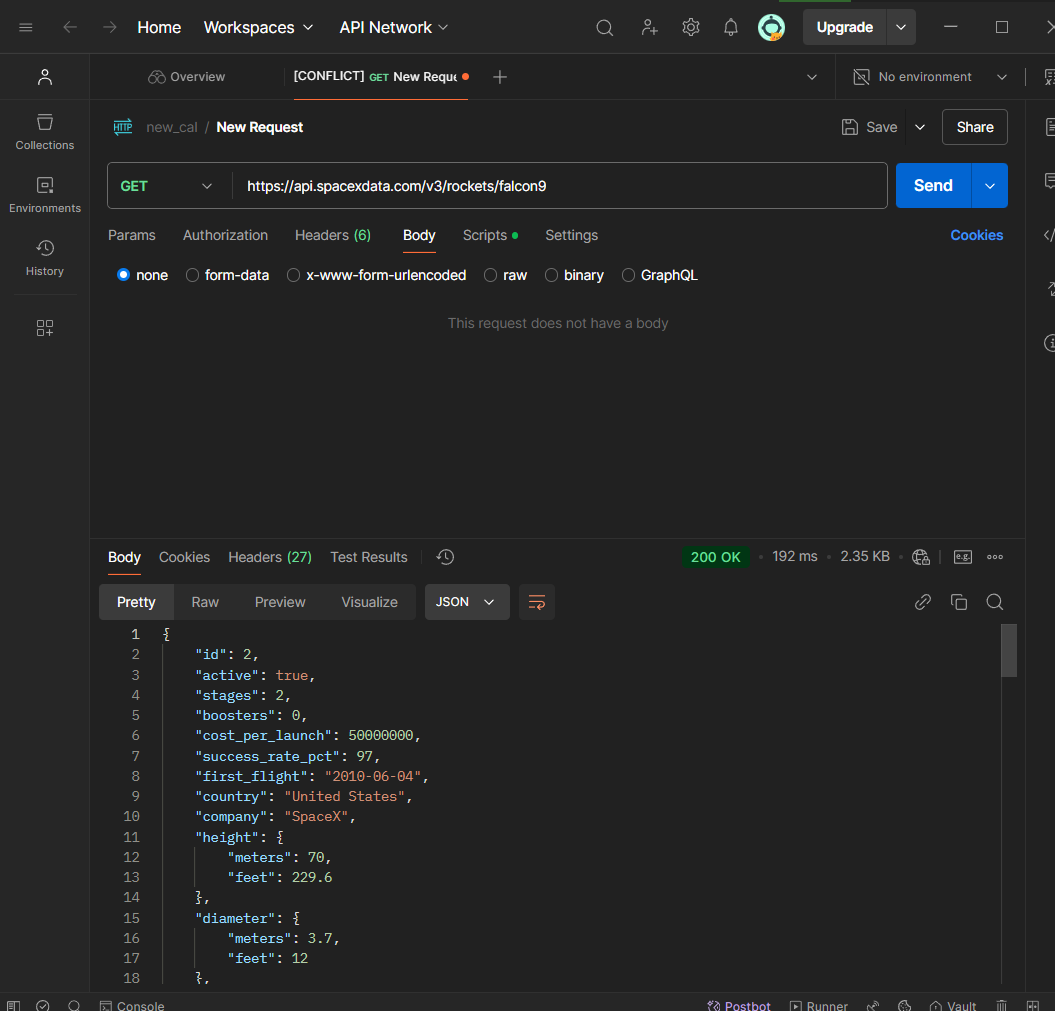
**Тест 2:** Получение ракеты по валидному ID

Метод: GET

URL: <https://api.spacexdata.com/v3/rockets/falcon9>

Ожидаемый результат: Код: 200, статус: OK, единственная ракета.





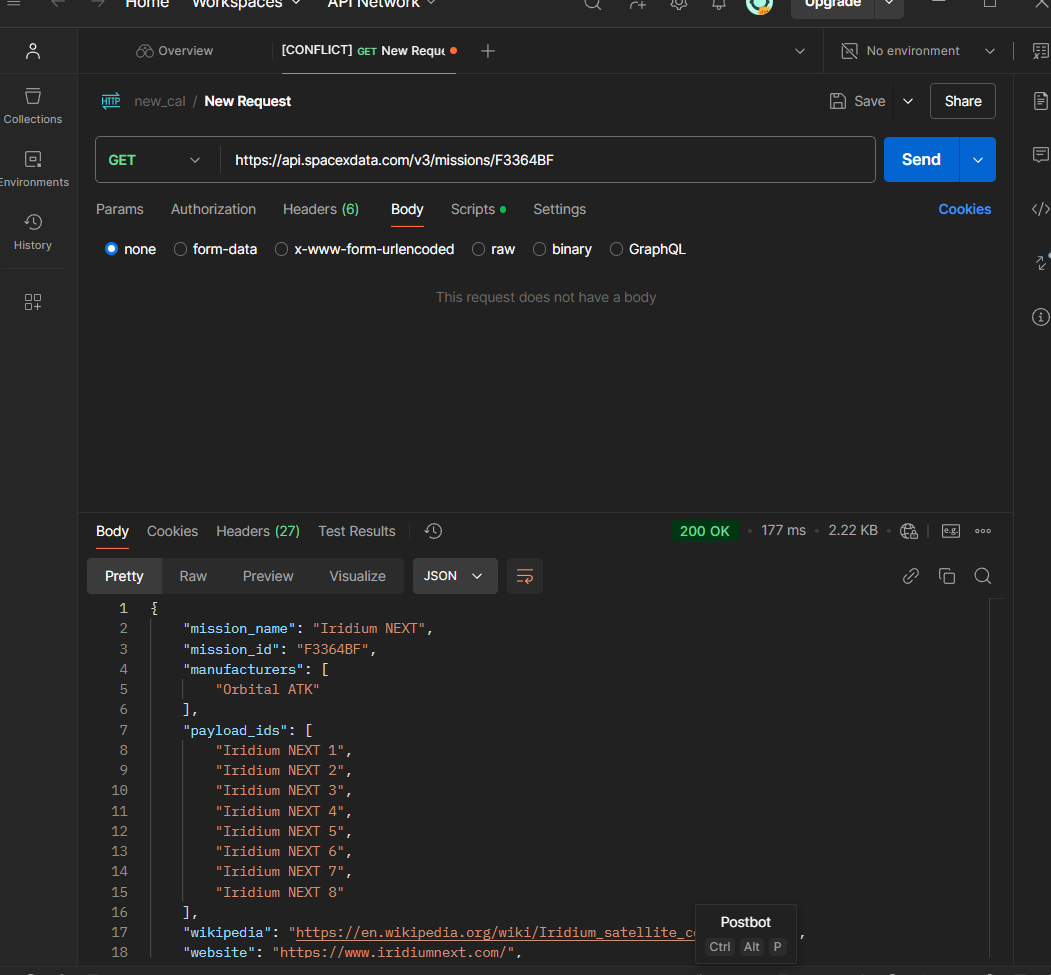
Фактический результат: Ожидаемый результат: Код: 200, статус: OK, единственная ракета.

**Тест 3:** получить миссию по валидному ID

Метод: GET

URL: <https://api.spacexdata.com/v3/missions/F3364BF>

Ожидаемый результат: код 200, статус: OK, единственная миссия.



Фактический результат: код: 200, статус: OK, единственная ракета.

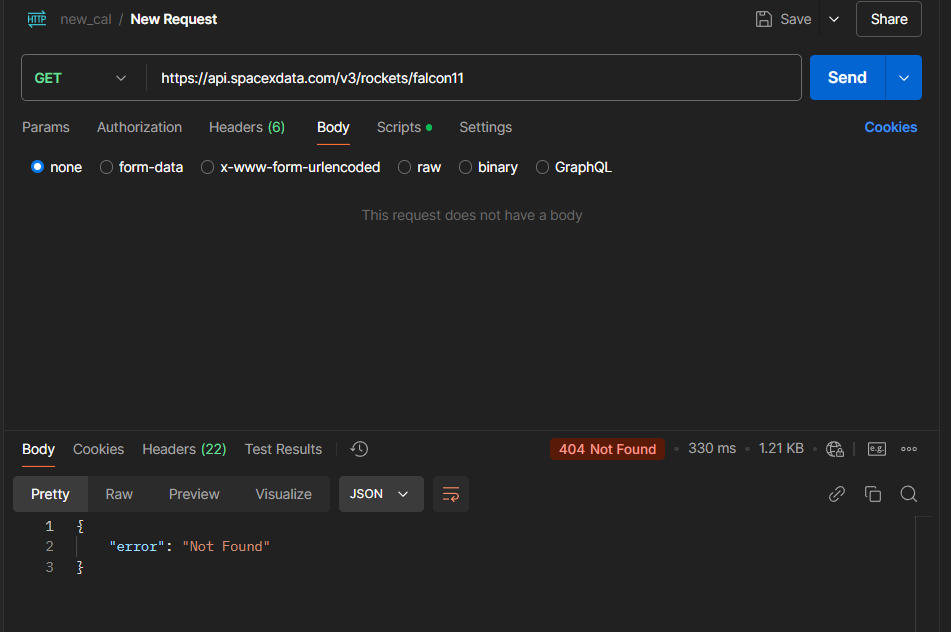
**Негативные** тест кейсы.

**Тест 1:** получить ракету с несуществующим id

Метод: GET

URL: <https://api.spacexdata.com/v3/rockets/falcon11>

Ожидаемый результат: код: 404, статус: Not Found



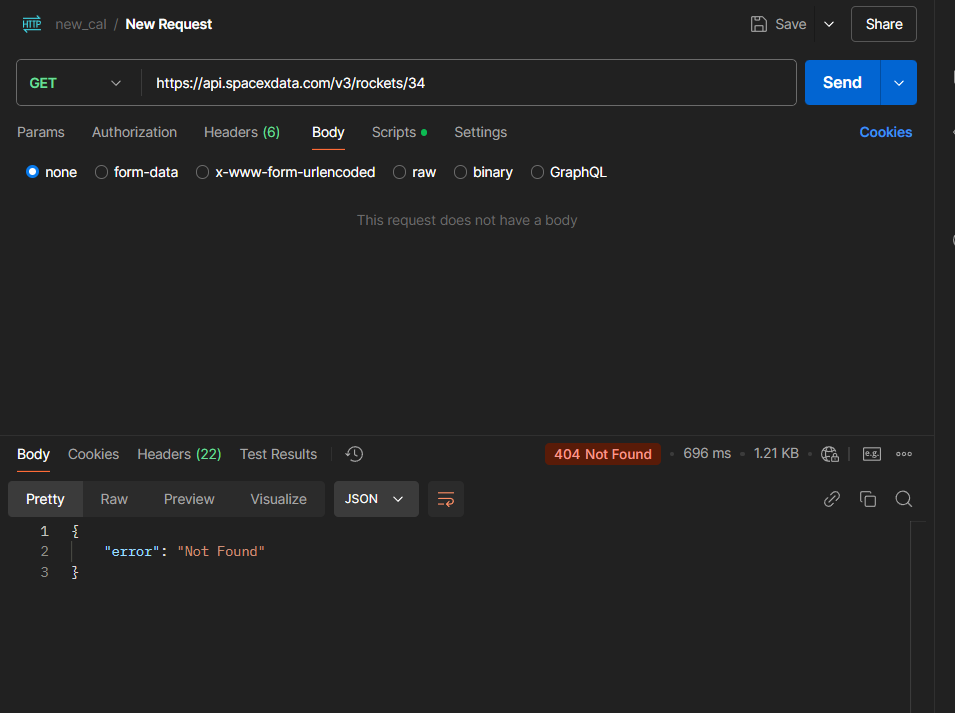
Фактический результат: код: 404, статус: Not Found.

**Тест 2:** получить ракету по не валидному id (число).

Метод: GET

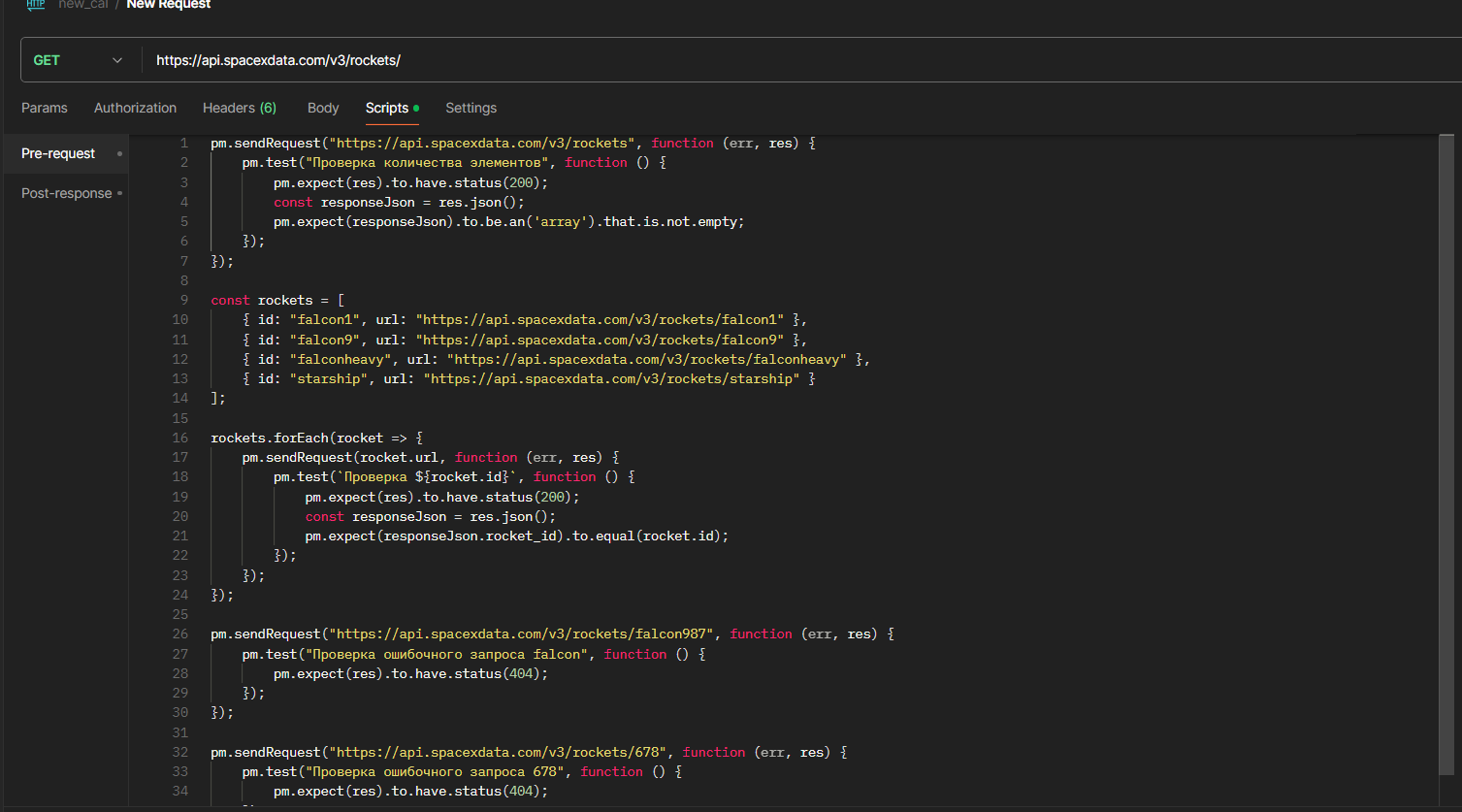
URL: <https://api.spacexdata.com/v3/rockets/falcon11>

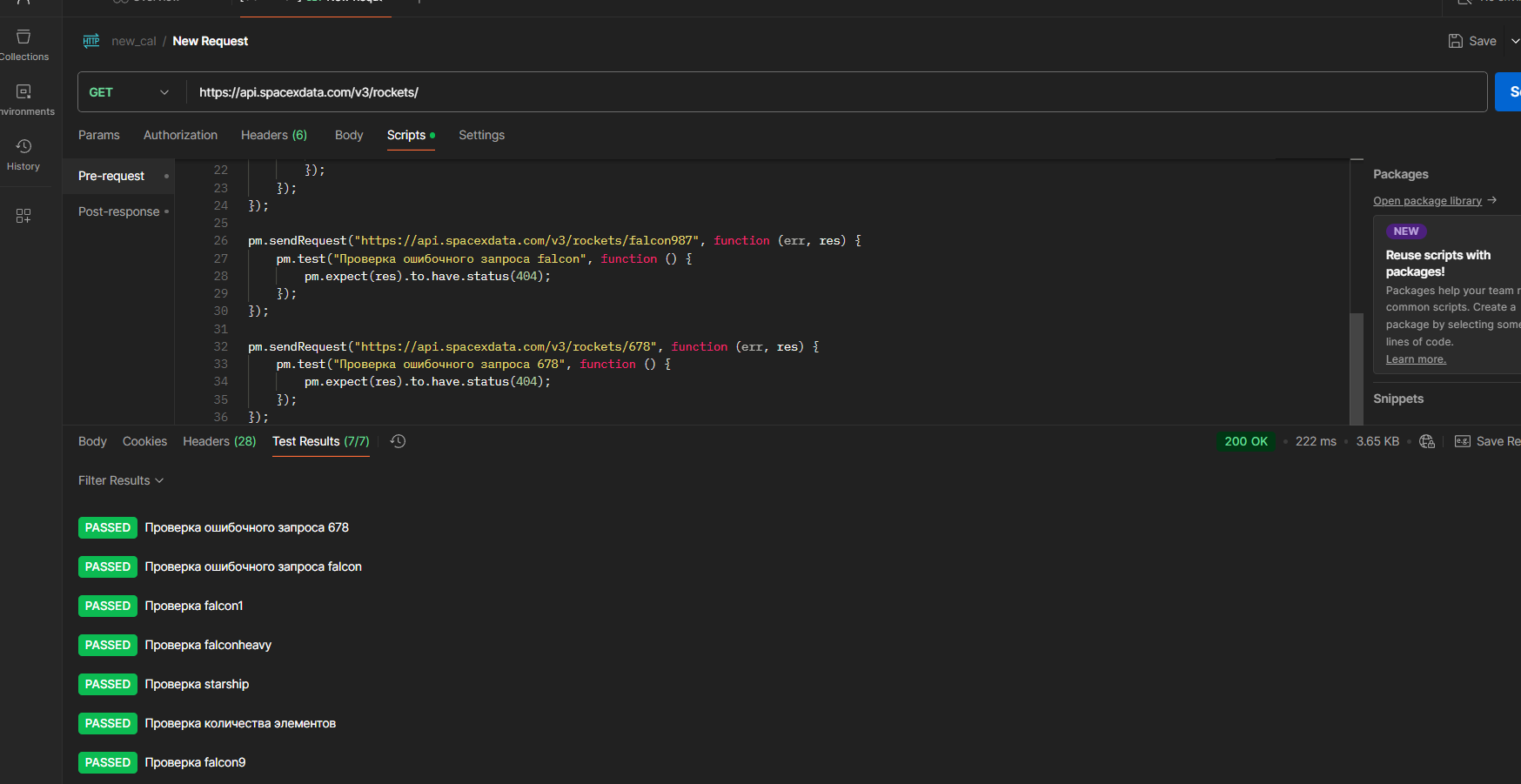
Ожидаемый результат: код: 404, статус: Not Found



Фактический результат: код: 404, статус: Not Found.

3.Напишите автоматизированные тесты для тест-кейсов из вашего списка и выполните их, автотесты можно написать самостоятельно или воспользоваться шаблонами для автотестов API (в Postman или подобных приложениях).

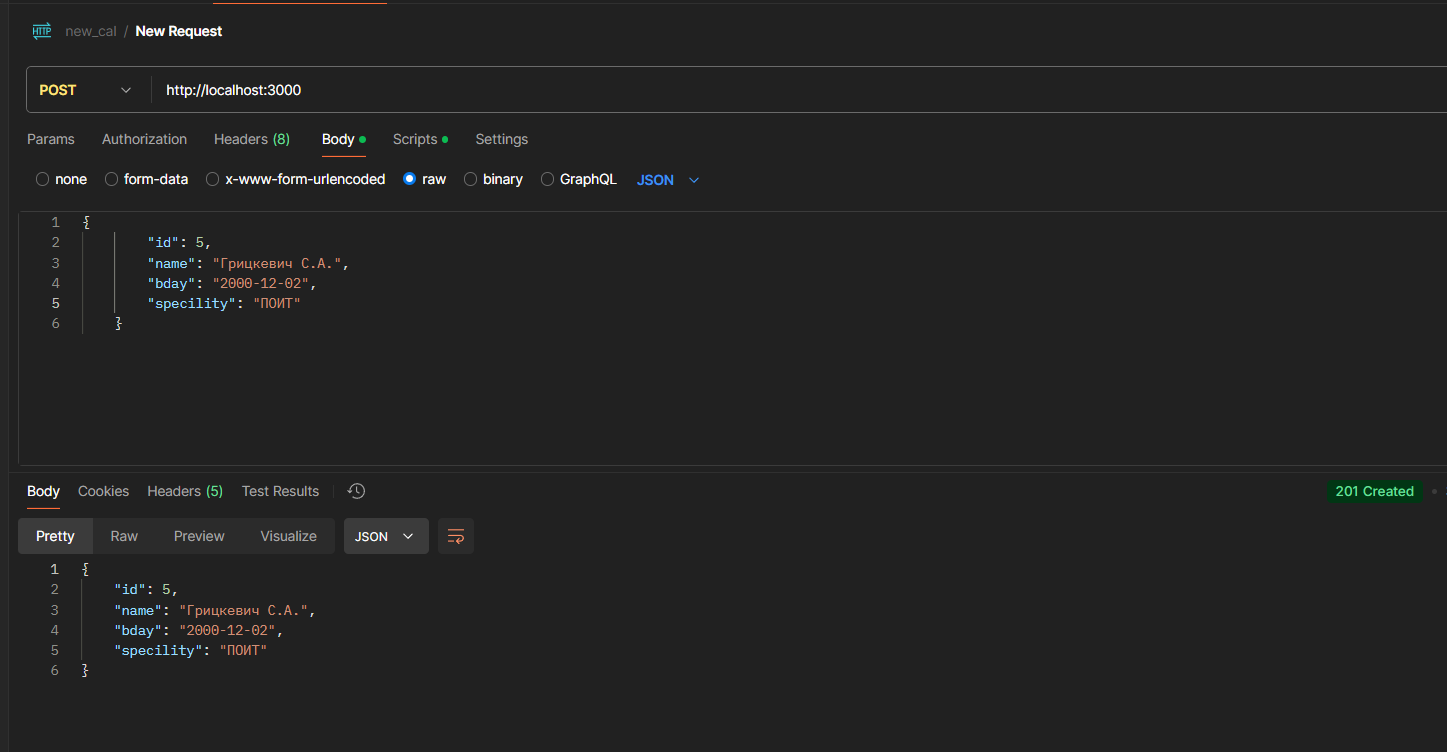




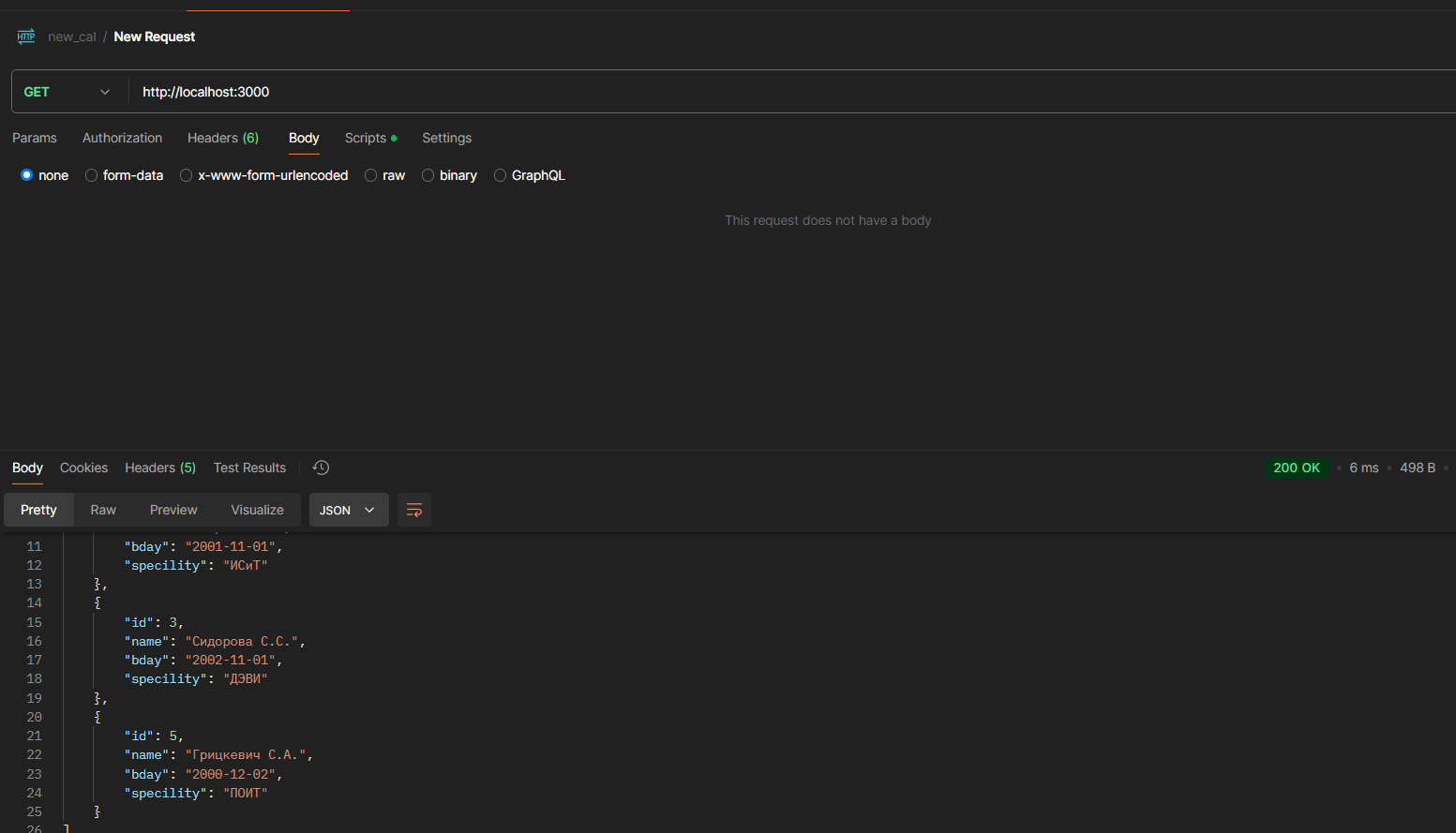
**Интеграционное тестирование методов** Например, можно провести тесты методов CRUD:

4. Протестируйте API для управления пользователями (/users): − Create: Создайте нового пользователя, передав корректный JSON. Проверьте ответ, код состояния (201 Created) и то, что пользователь создан. − Read: Запросите информацию о созданном пользователе, убедитесь в правильности данных. − Update: Обновите данные пользователя (например, email) и проверьте, применились ли изменения. − Delete: Удалите пользователя и убедитесь, что он больше недоступен. Критерии проверки: - коды ответов соответствуют действиям (201 для создания, 200 для получения и обновления, 204 для удаления). - данные обновляются корректно.

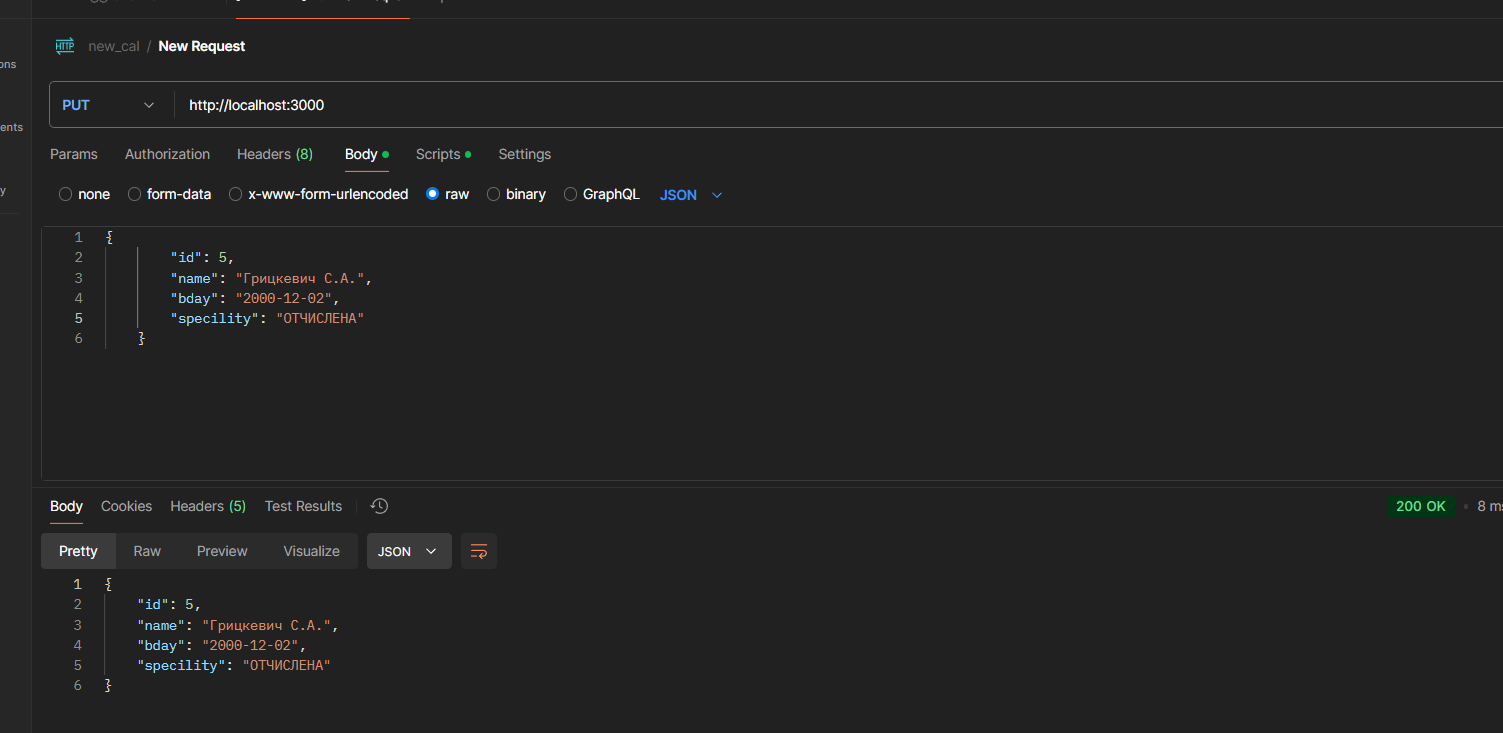
**Create:**

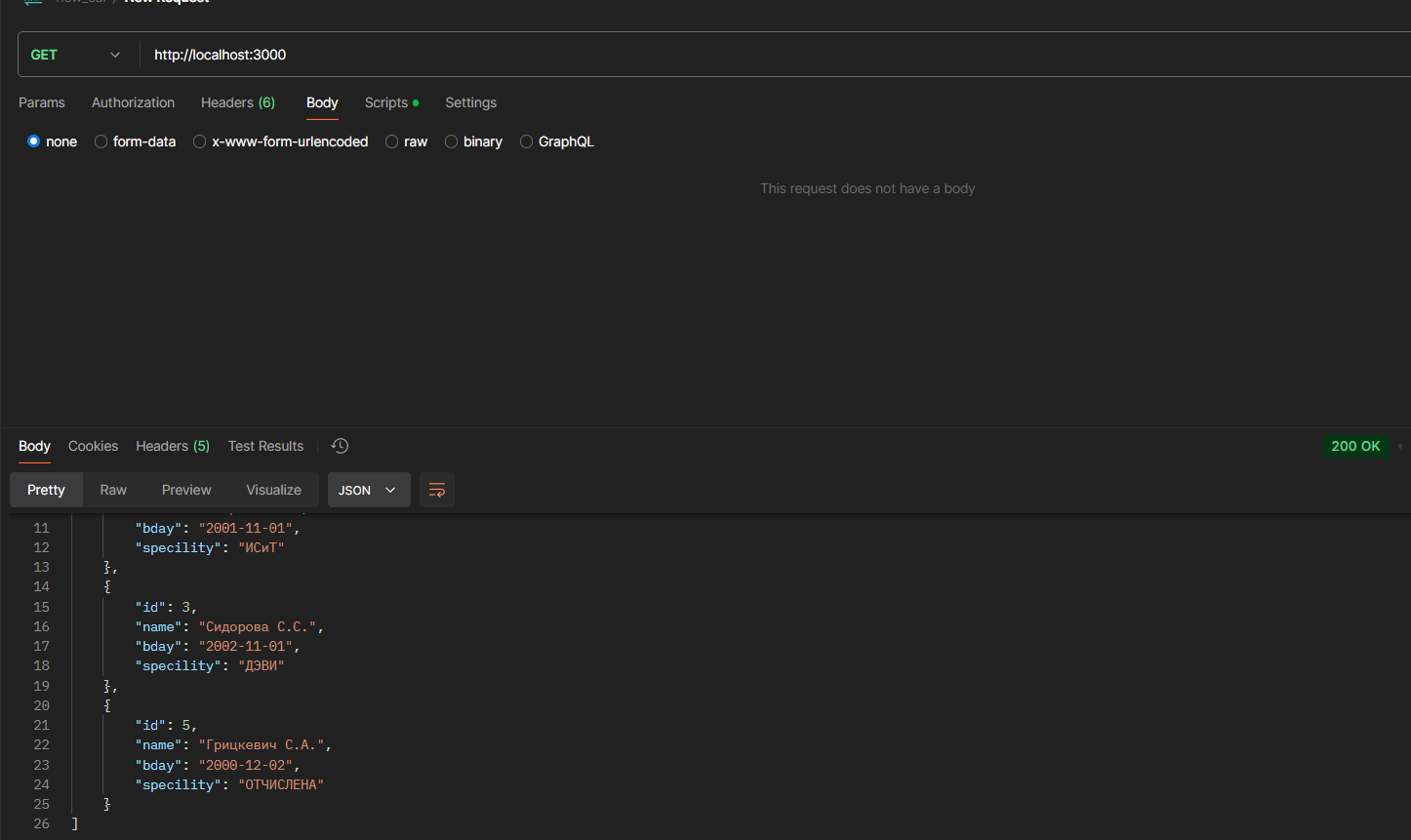


**Read:**

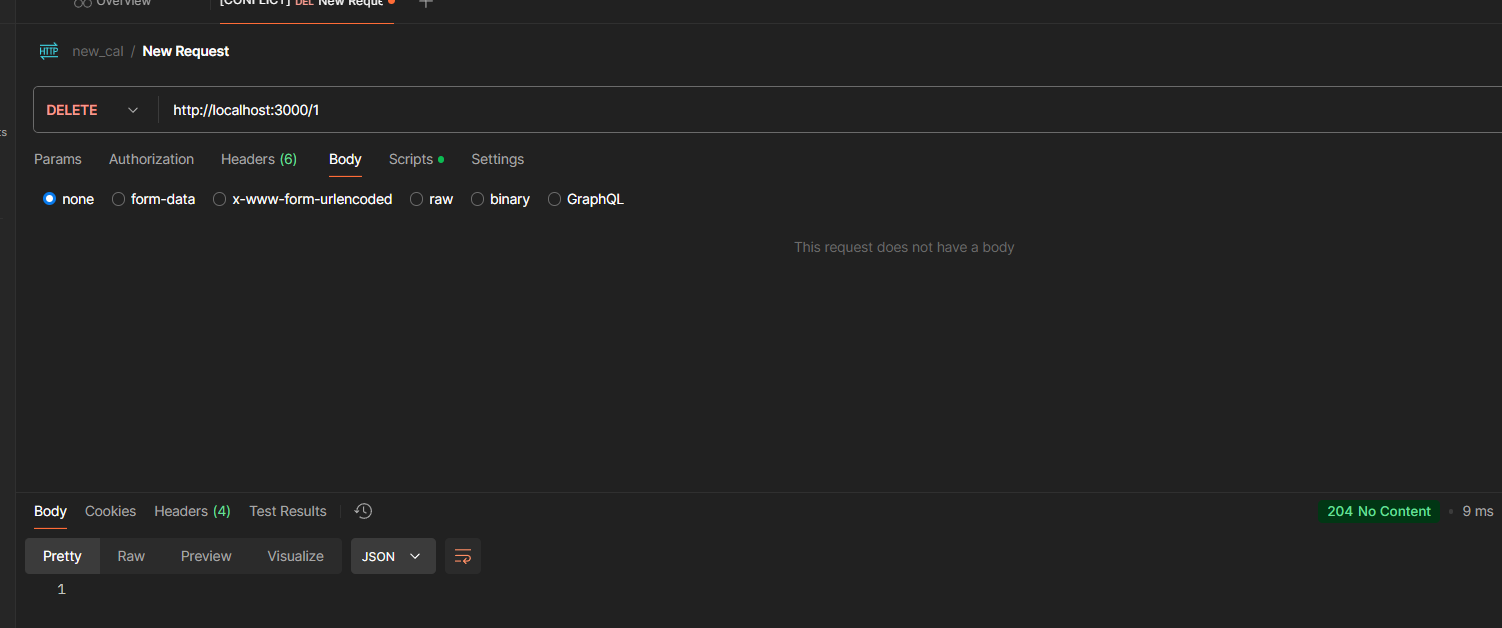


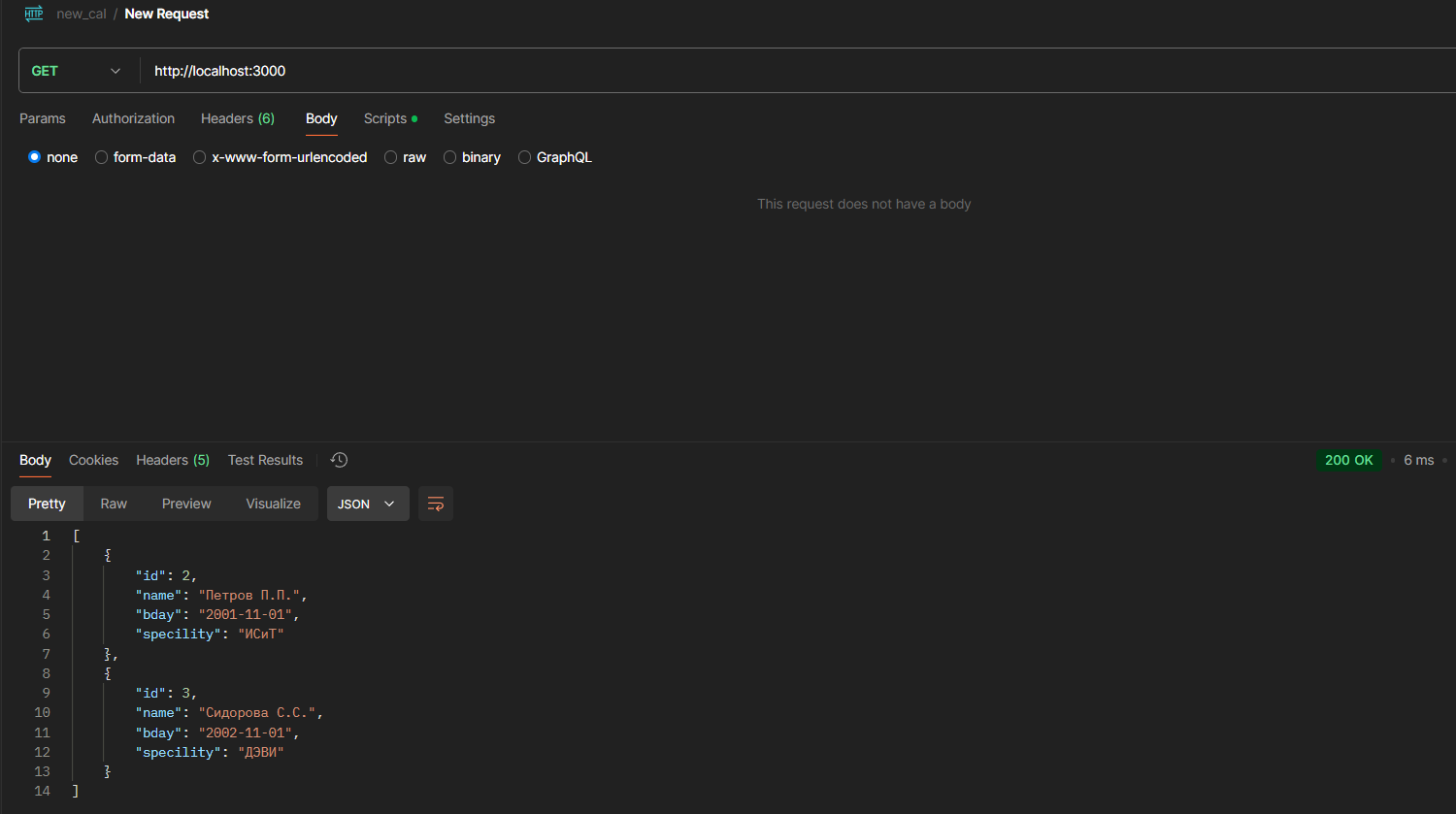
**Update:**

****



**Delete:**

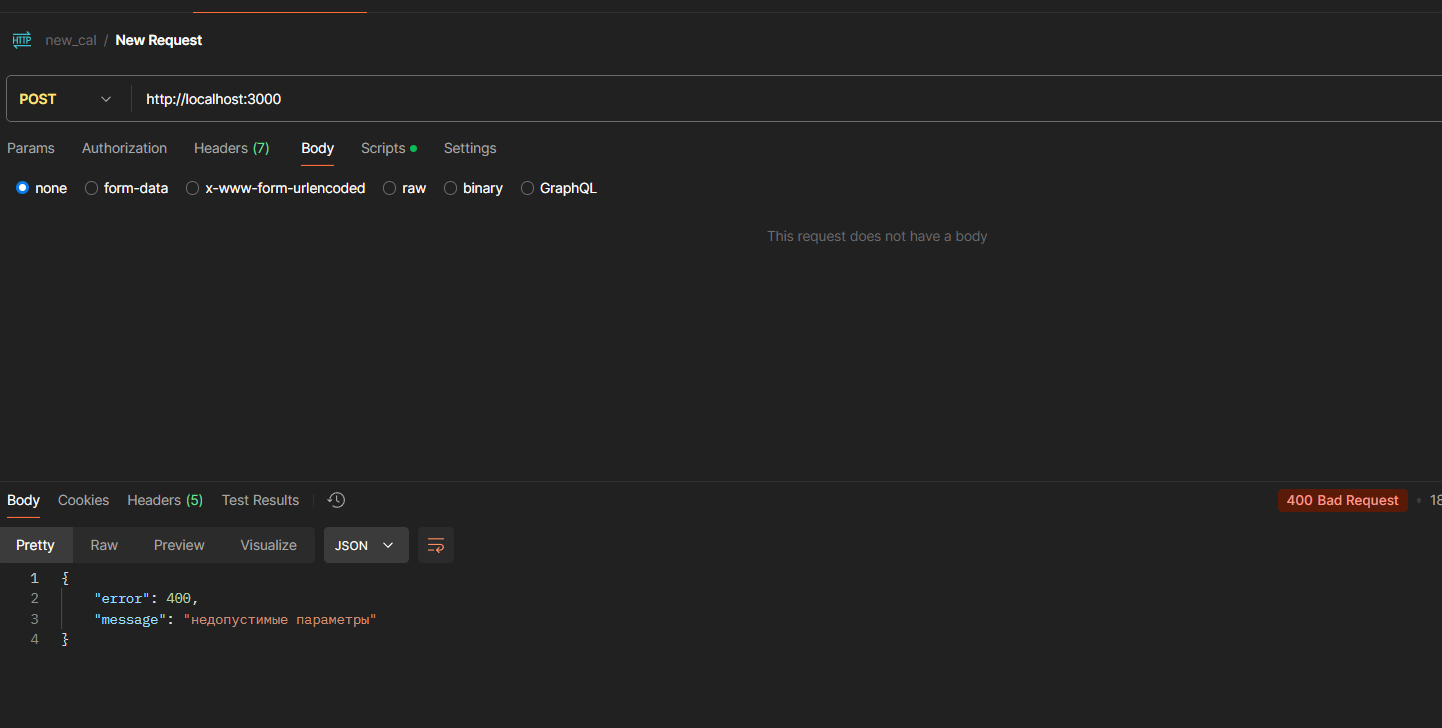
****

****

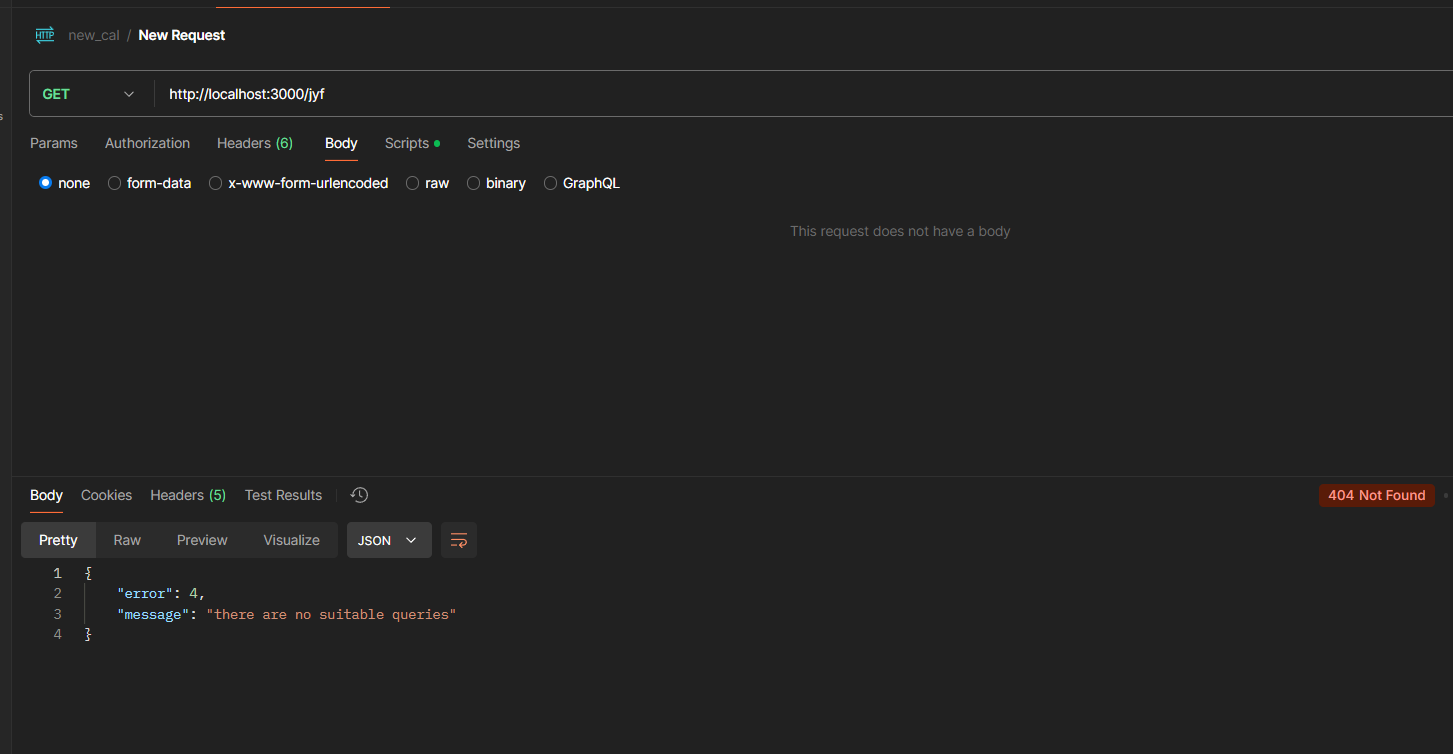
**Тестирование обработки ошибок 5.**

Протестируйте API на случай некорректных запросов, например: − отправьте запрос с пустым телом; − используйте некорректный формат данных (например, строку вместо числа); − попробуйте запрос к несуществующей конечной точке. Критерии проверки: - возвращается код ошибки (например, 400 для неправильного тела, 404 для несуществующей конечной точки).

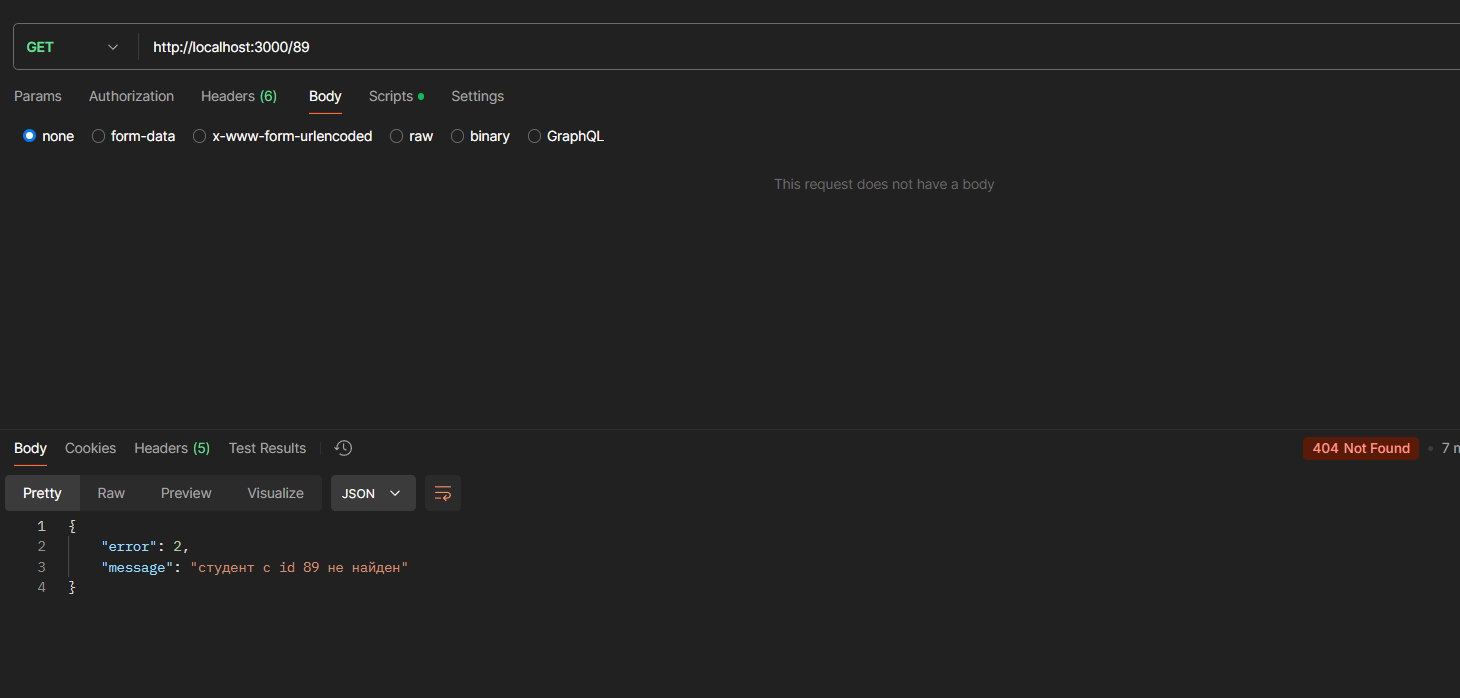
**POST-запрос без тела:**

****

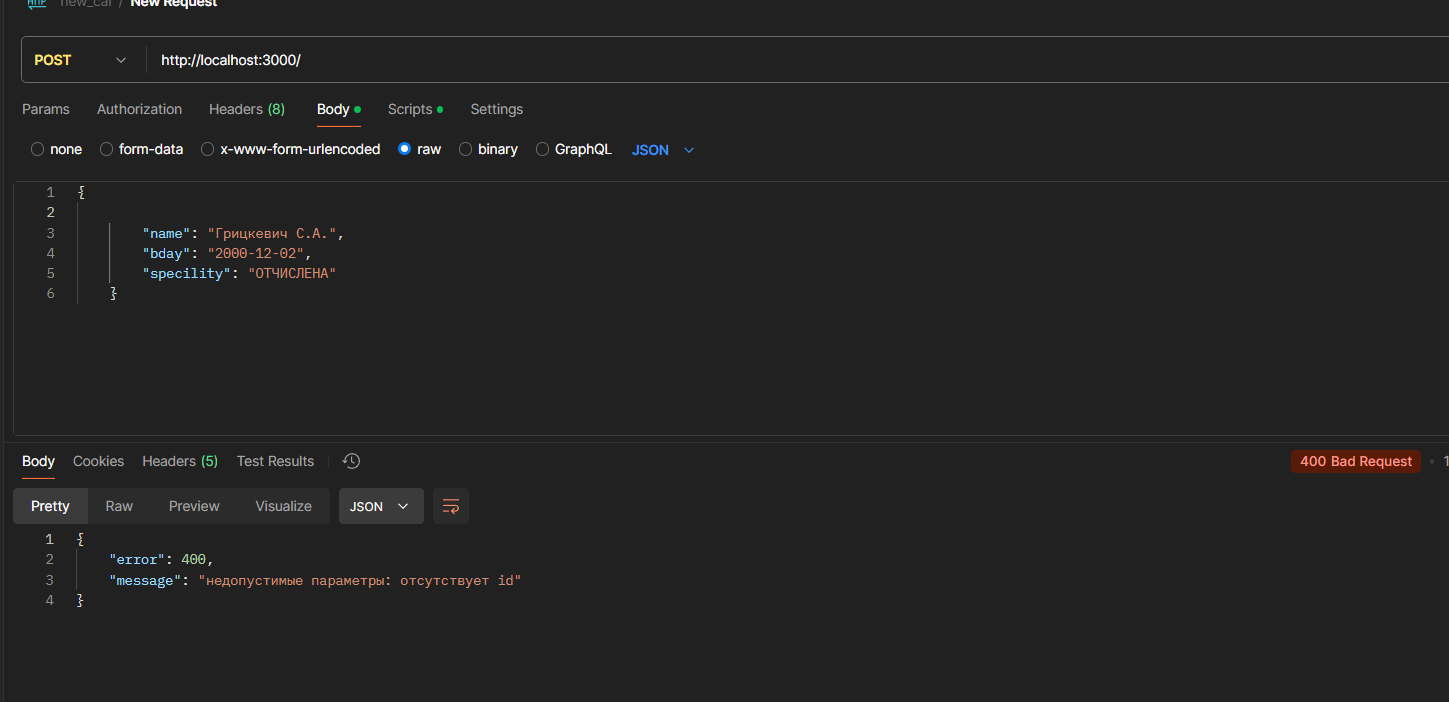
**GET-запрос с некорректным параметром:**



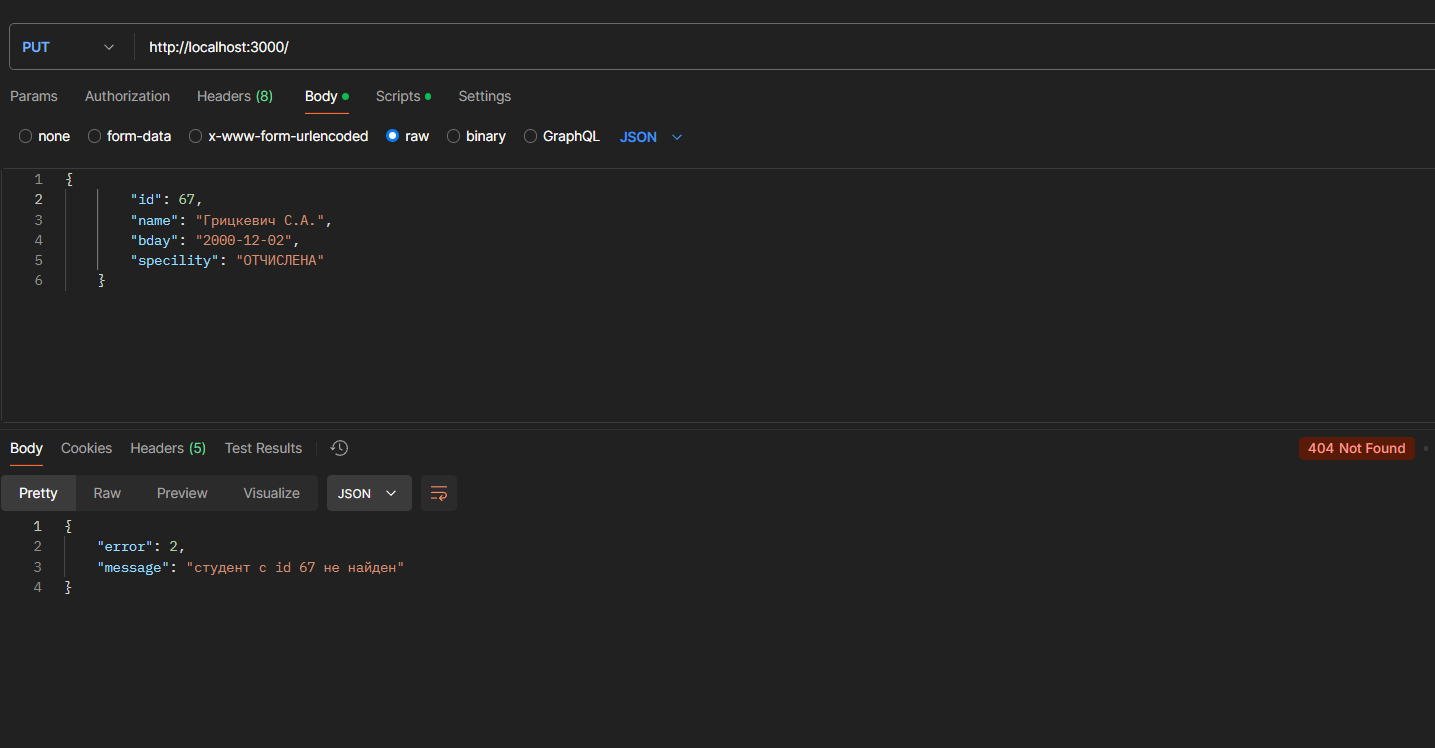
**GET-запрос к несуществующей точке:**

****

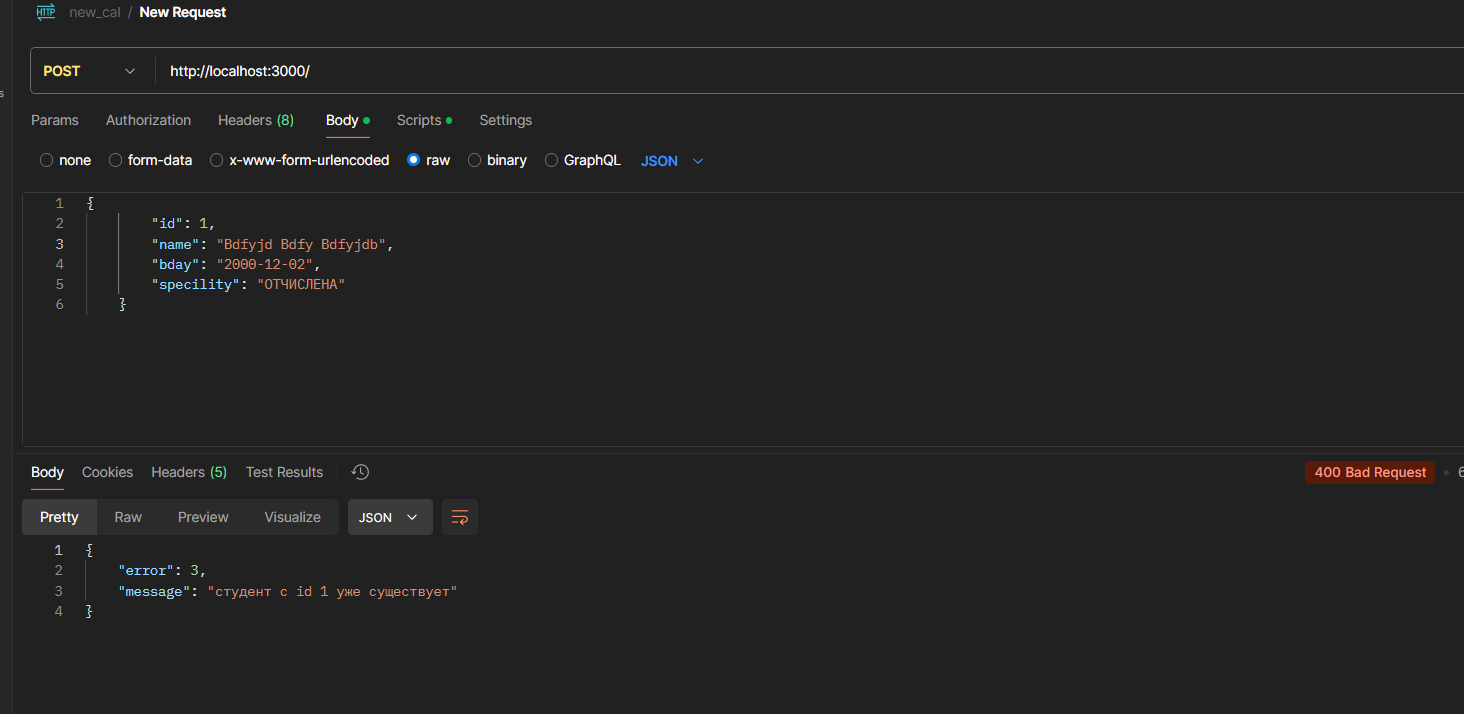
**POST-запрос с неполными данными:**

****

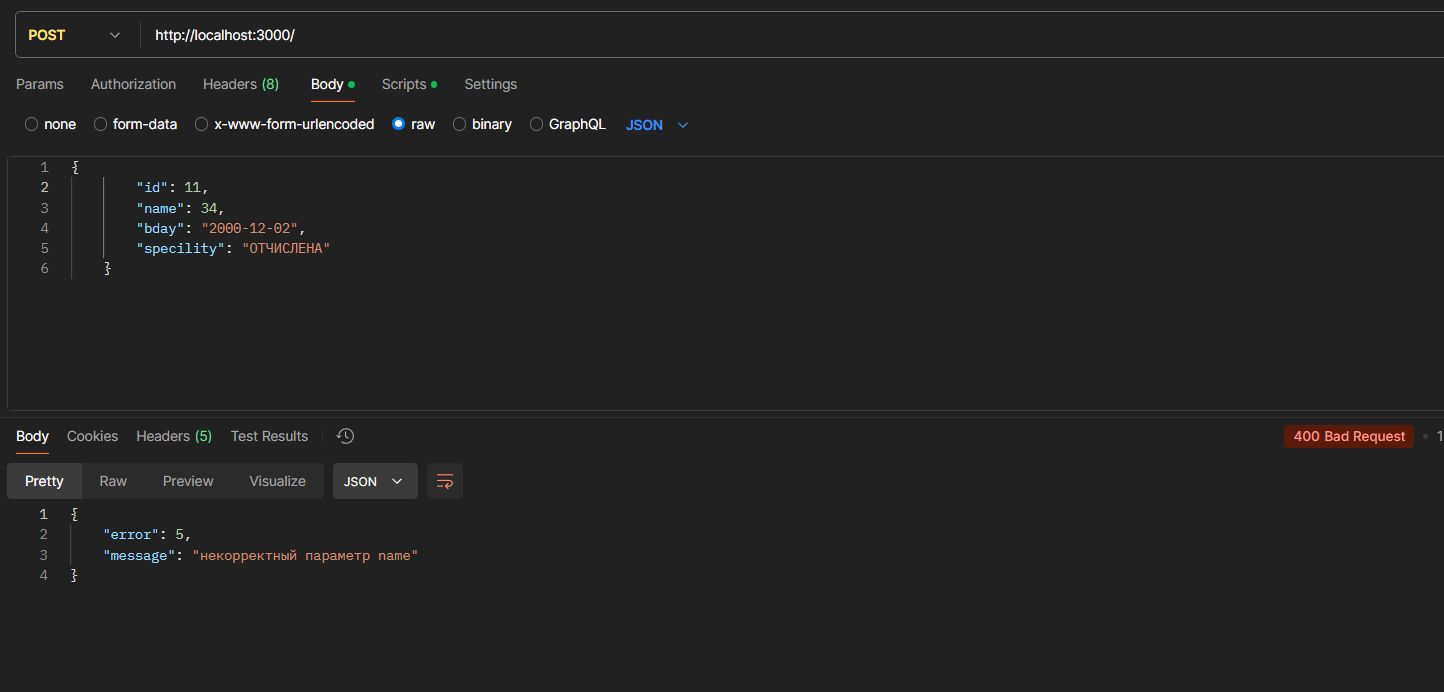
**PUT-запрос к несуществующей точке:**

****

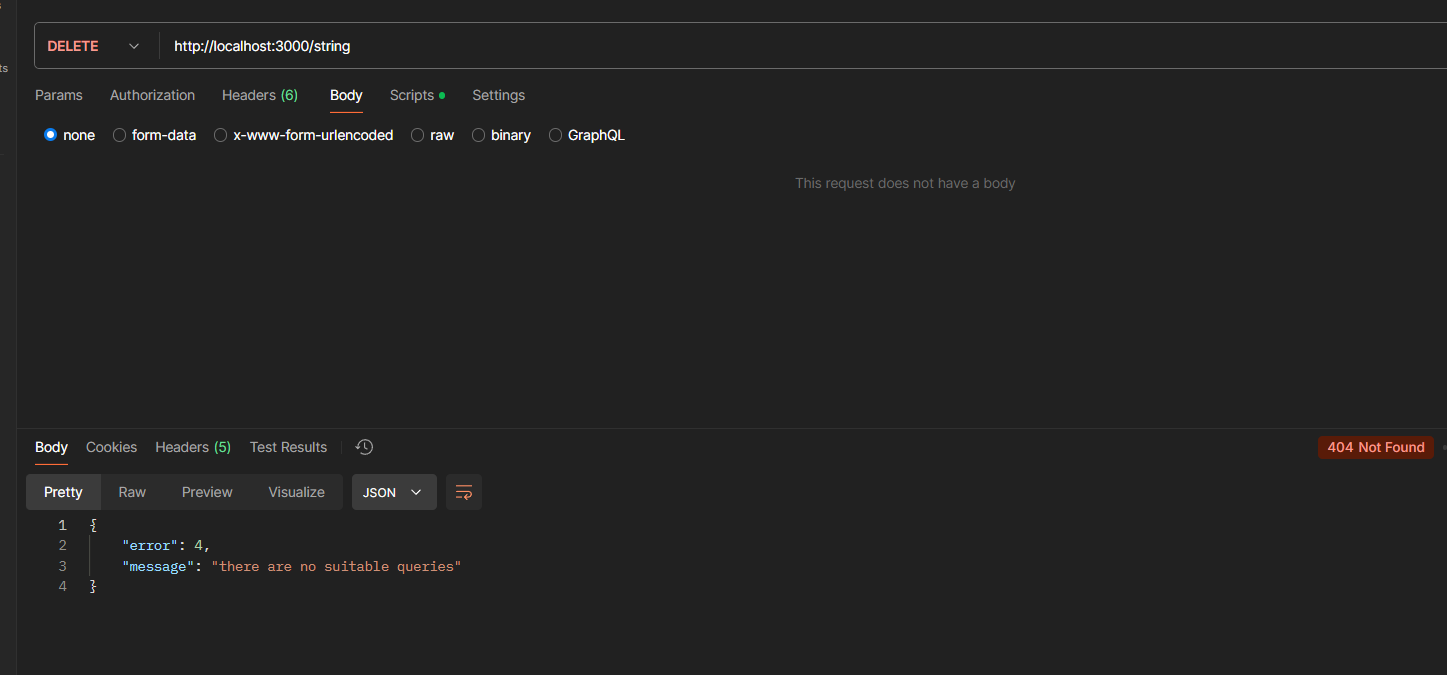
**POST-запрос с некорректными данными:**

****

**POST-запрос с некорректными данными:**

****

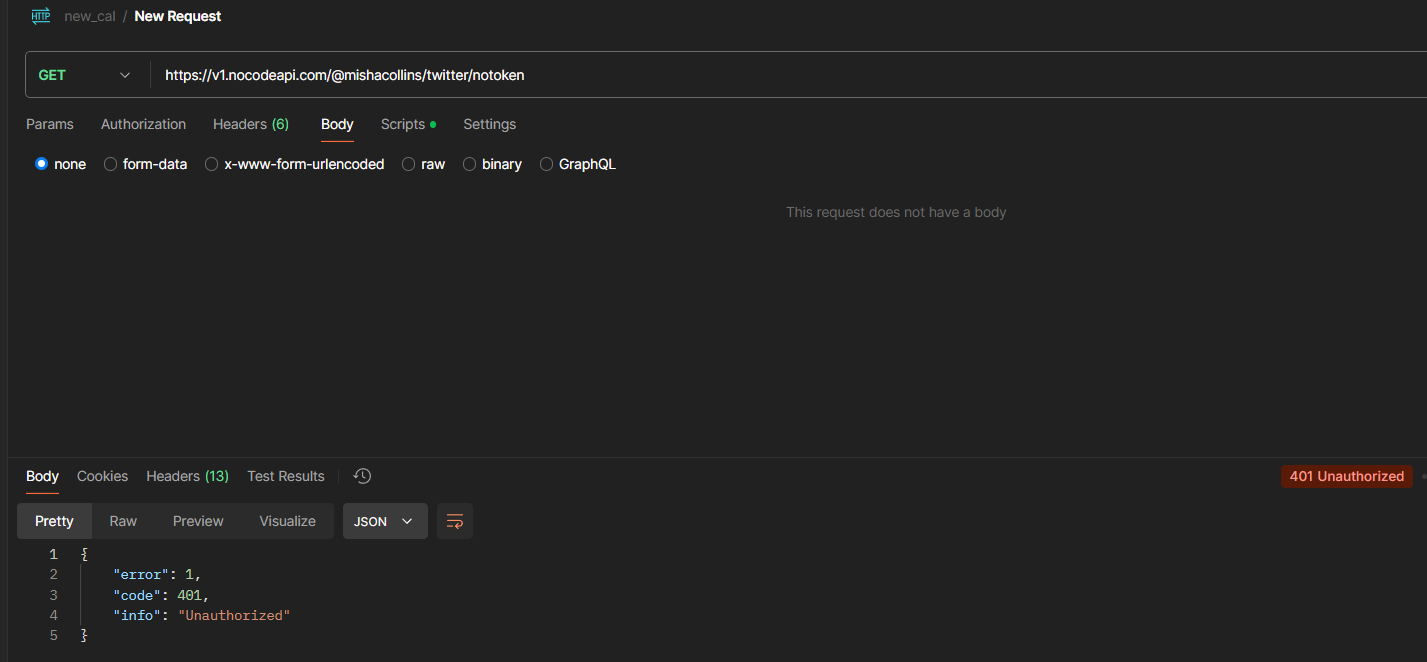
**DELETE-запрос с некорректным параметром:**

****

**Тестирование прав доступа:**

Метод: GET

URL: <https://v1.nocodeapi.com/@mishacollins/twitter/notoken>

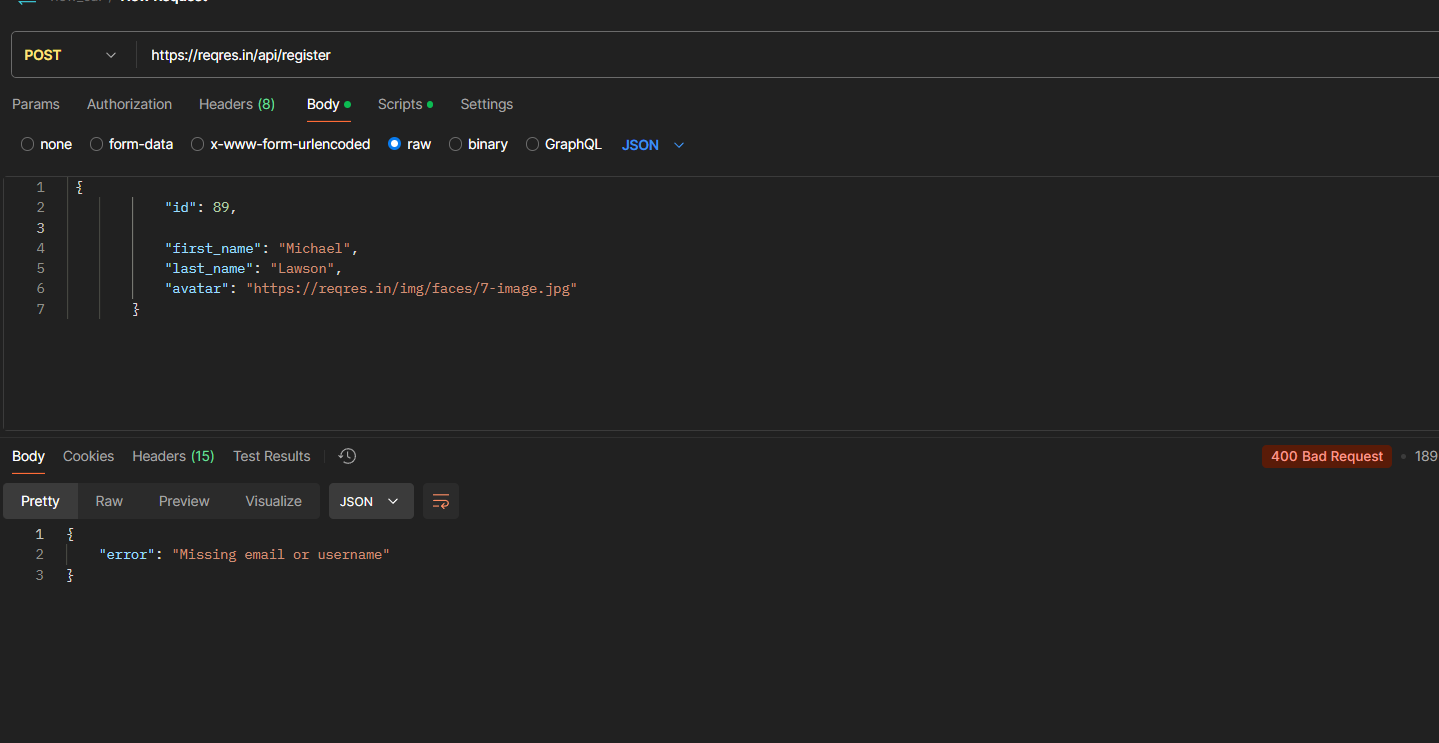


**Тестирование валидации данных**

**Отсутствует обязательное поле:**

**Метод: GET**

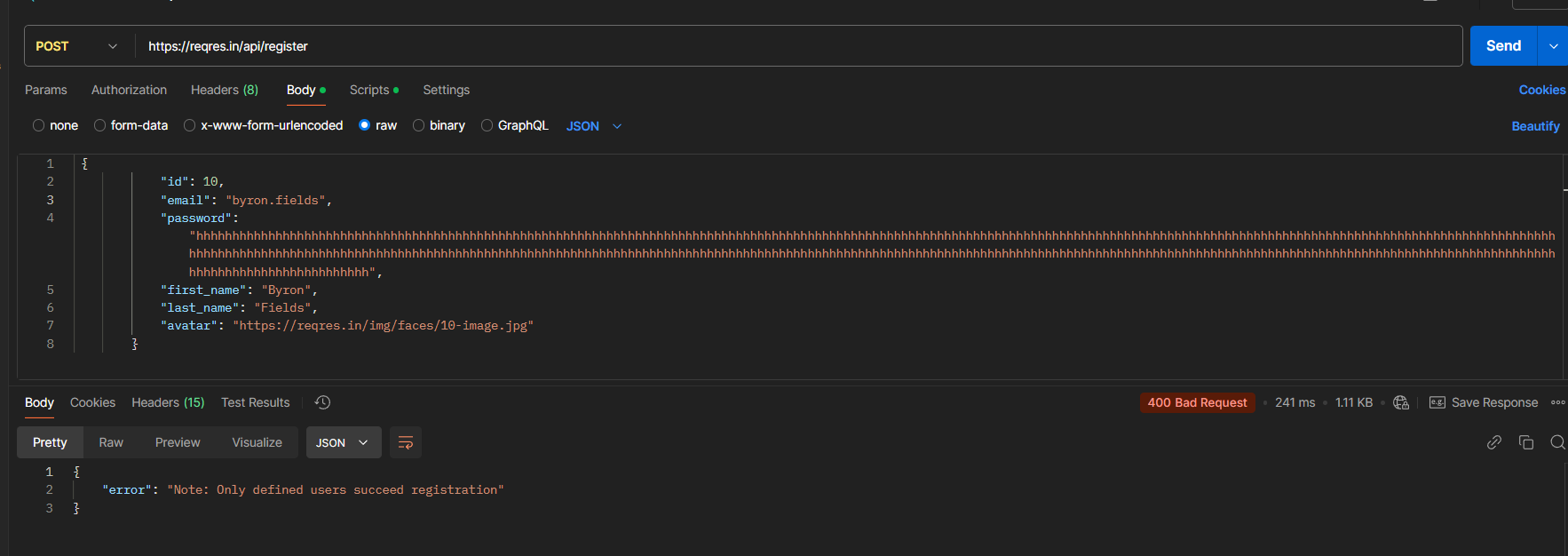
**URL:** [**https://reqres.in/api/register**](https://reqres.in/api/register)



**Слишком длинная строка:**

**Метод: POST**

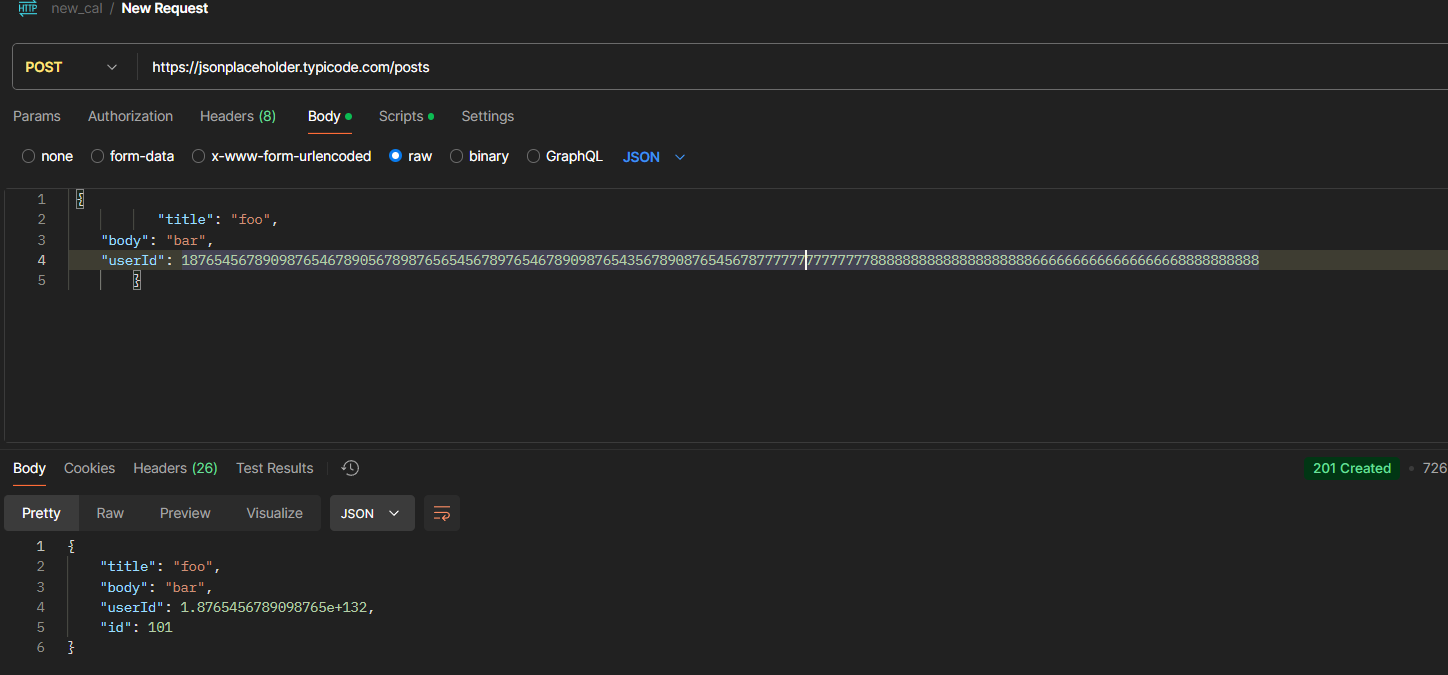
**URL: https://reqres.in/api/register**



**Проверка вхождения числа в диапазон:**

**Метод: POST**

**URL: https://jsonplaceholder.typicode.com/posts**

****

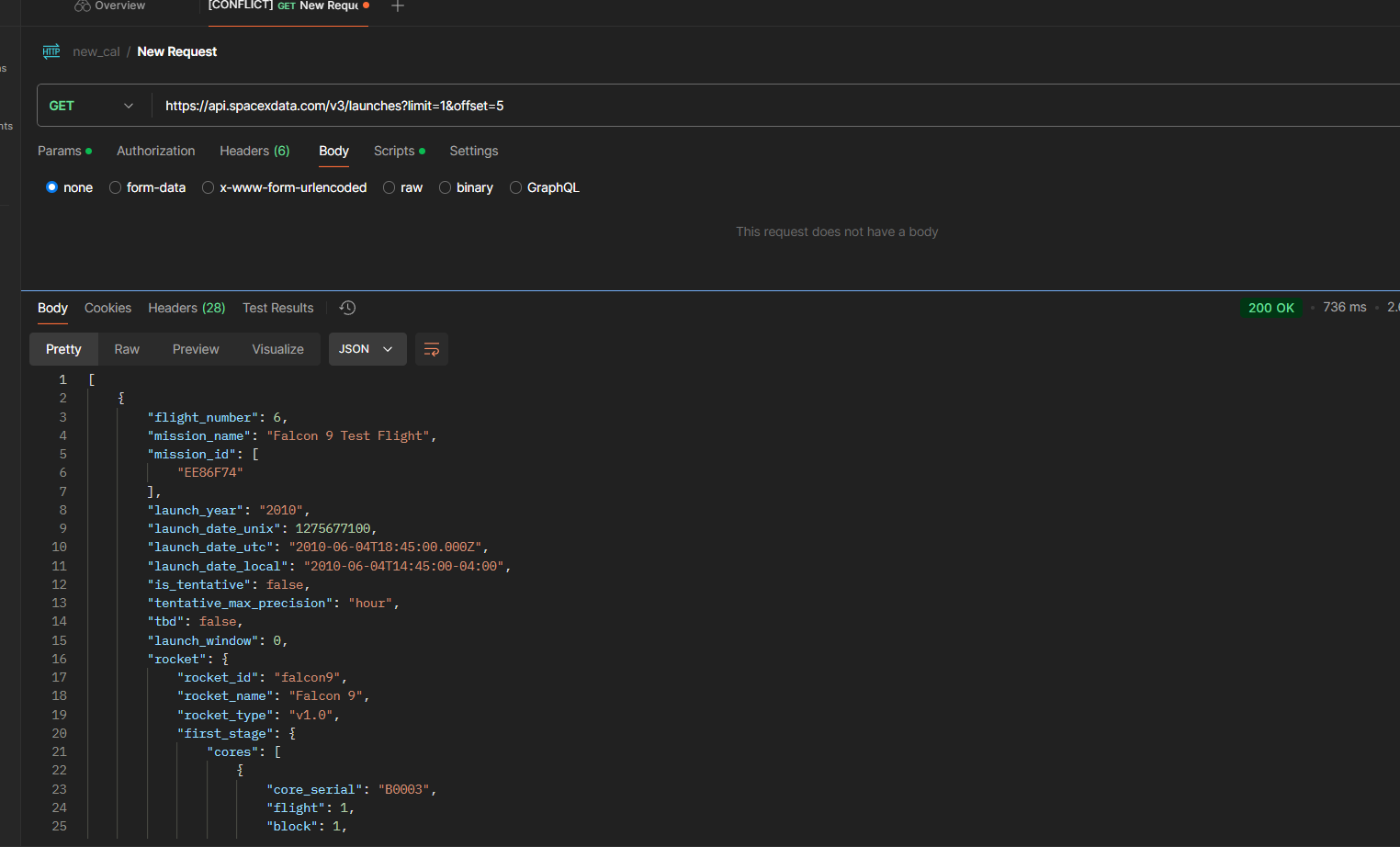
**Тестирование пагинации**

**8. Протестируйте, как API обрабатывает запросы с параметрами пагинации: − запросите список объектов с параметрами page и limit; − убедитесь, что возвращаются только данные указанной страницы; − попробуйте запросить страницу за пределами существующего диапазона.**

**Единственное значение, пропустить 5:**

**Метод: GET**

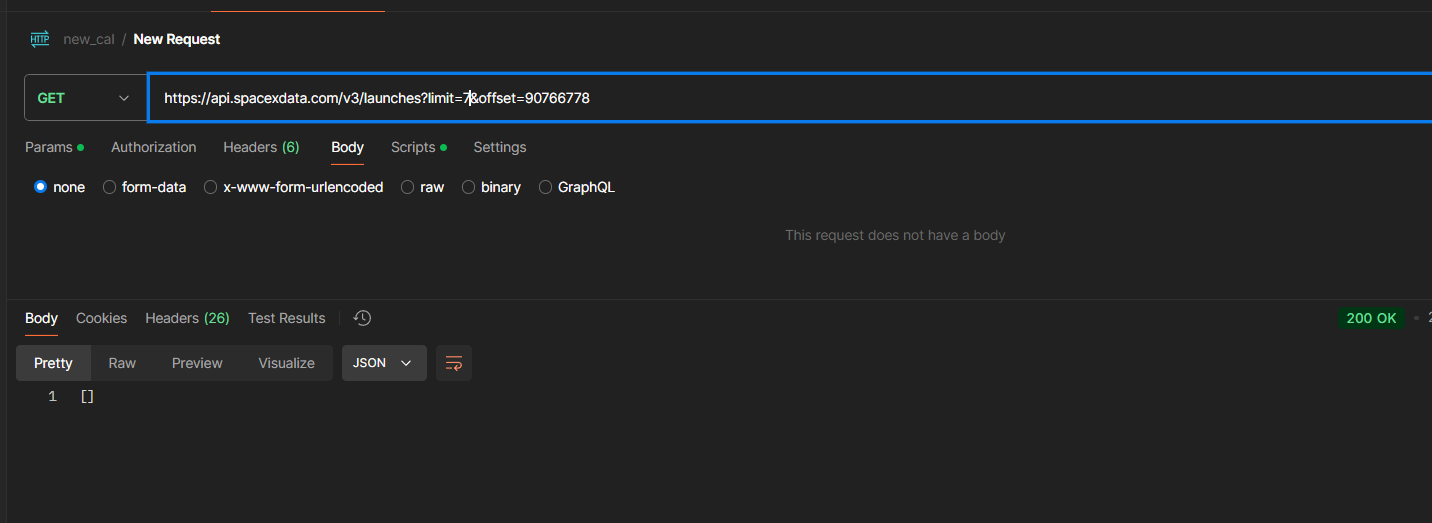
**URL: https://api.spacexdata.com/v3/launches?limit=1&offset=5**



**Много значений, пропустить бесконечно:**

**Метод: GET**

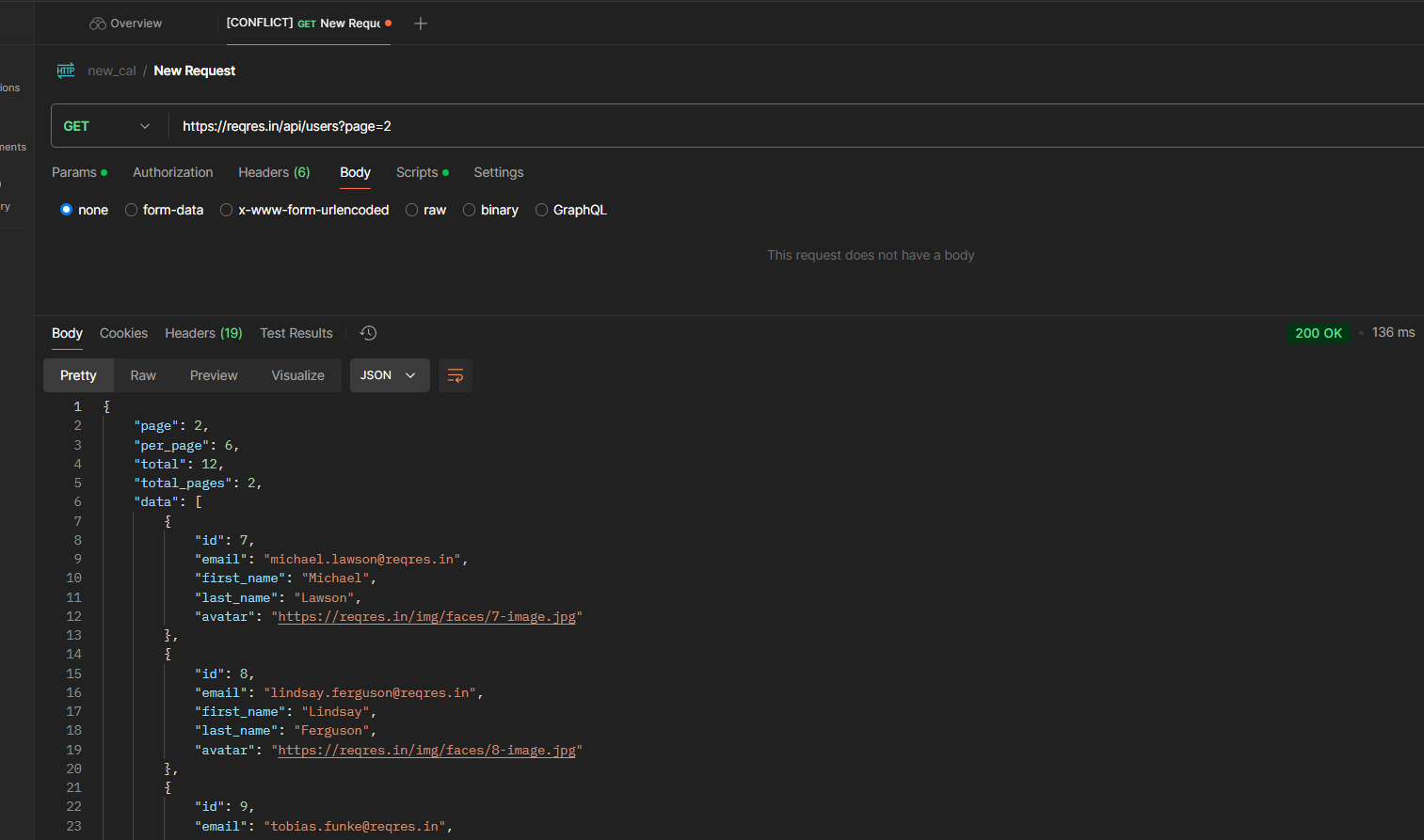
**URL:** [**https://api.spacexdata.com/v3/launches?limit=7&offset=90766778**](https://api.spacexdata.com/v3/launches?limit=7&offset=90766778)



**Проверка страницы:**

**Метод: GET**

**URL: https://reqres.in/api/users?page=2**



**Слишком длинная страница:**

**Метод: GET**

**URL: https://reqres.in/api/users?page=2**

