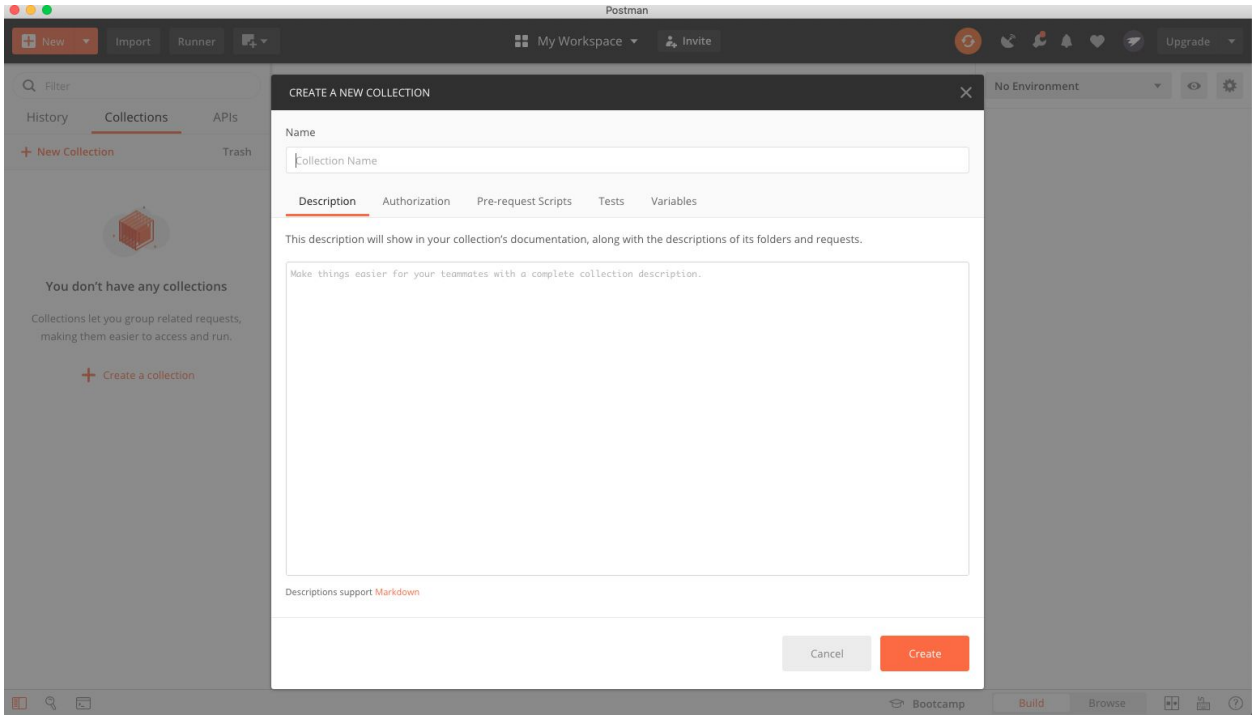
Postman - это инструмент для автоматизации тестирования API с довольно развитой функциональностью.

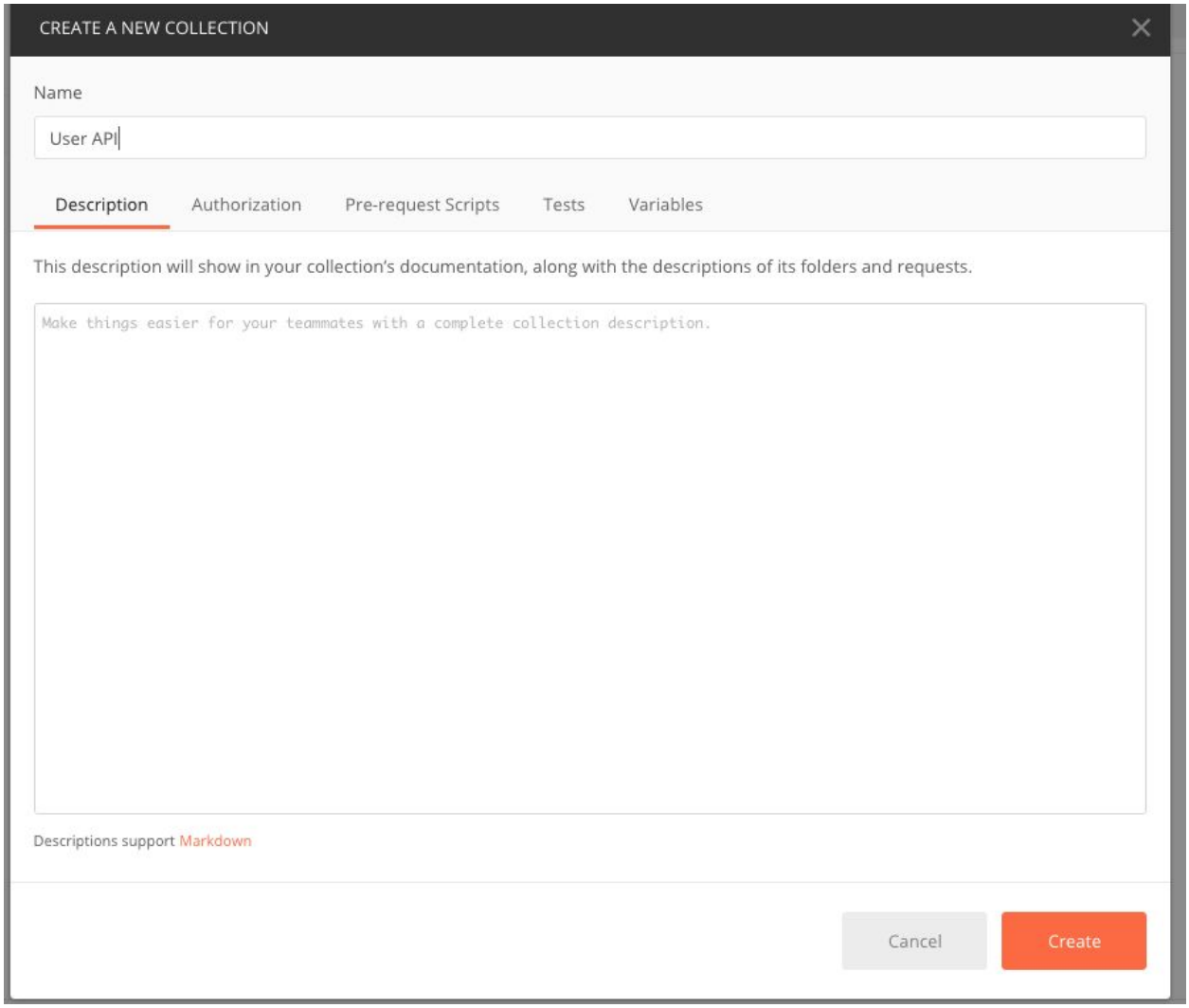
https://www.postman.com/ - здесь можно скачать приложение и запустить.

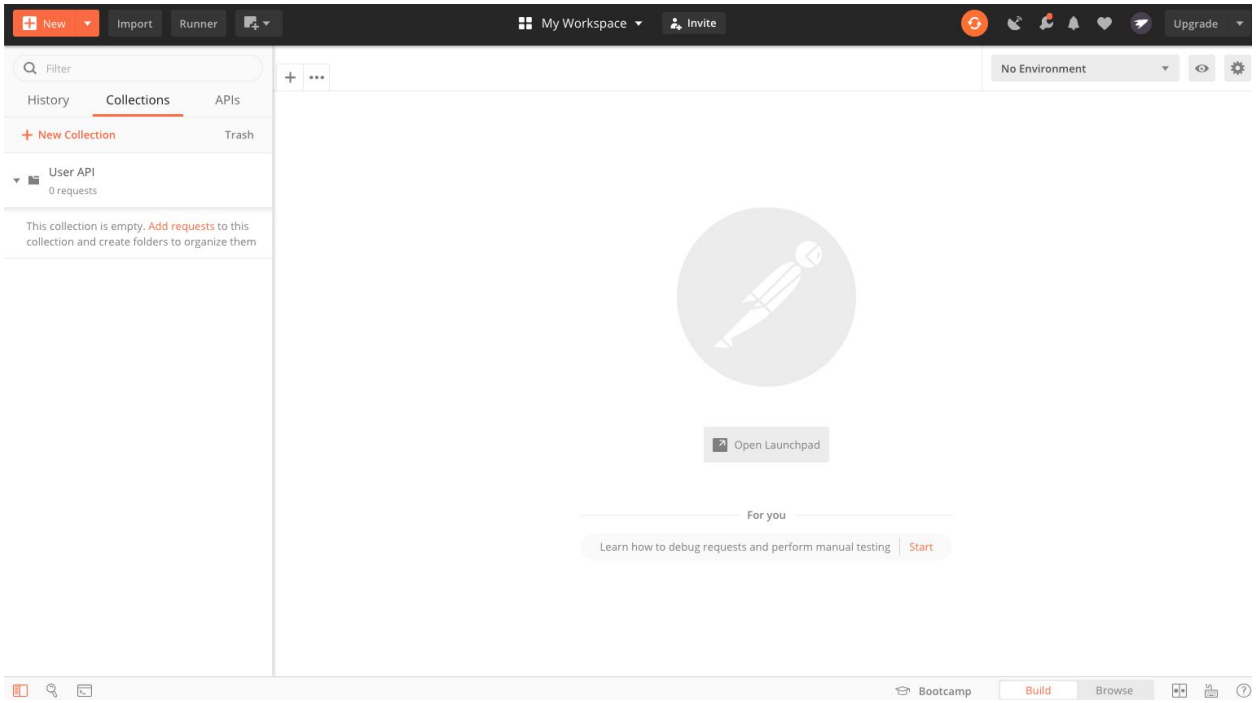
У нас есть какой-то сервис, который реализует CRUD API. Как его можно протестировать?

Давайте создадим коллекцию (это набор запросов, связанных между собой).

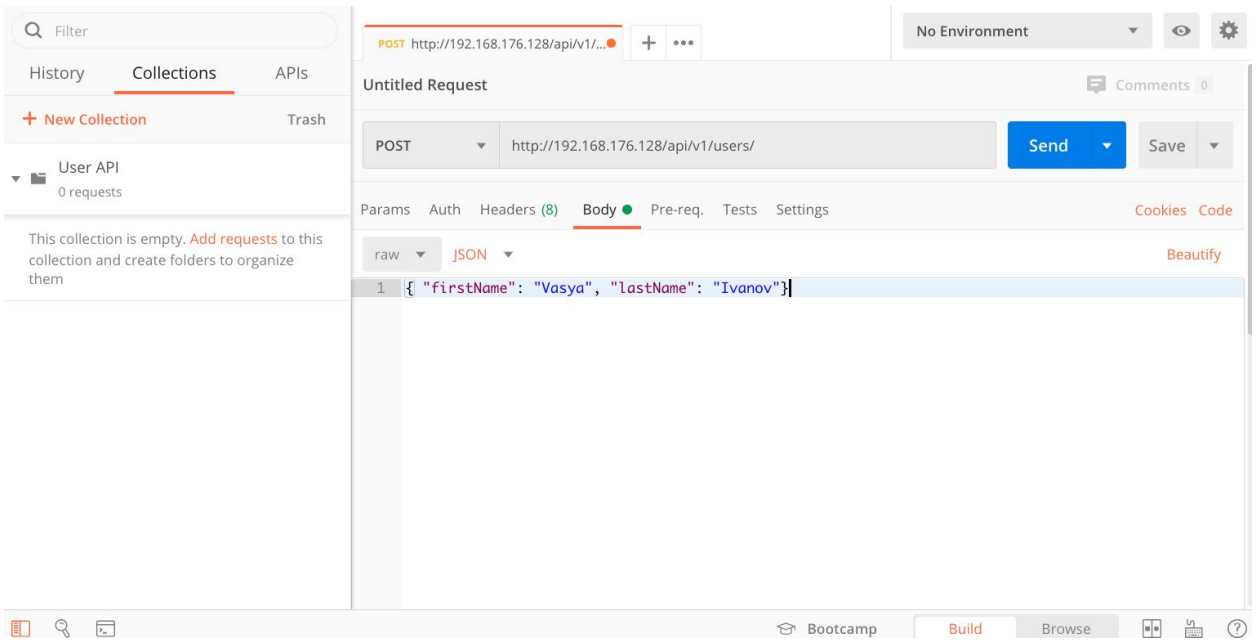
+New Collection




CREATE A NEW COLLECTION
Name
User API
Description
Authorization
Pre-request Scripts
Tests
Variables
This description will show in your collection's documentation, along with the descriptions of its folders and requests.
Make things easier for your teammates with a complete collection description.
Descriptions support Markdown
Cancel
Create

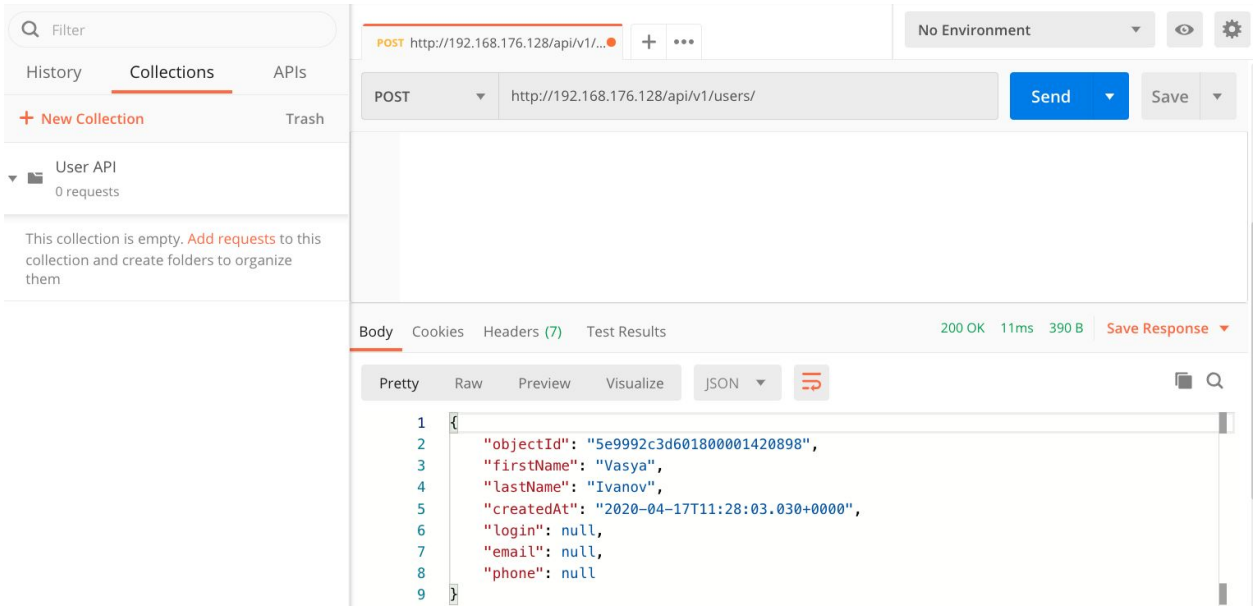


Добавляем новый запрос через “Коллекция” -> “Добавить запрос”, чтобы запрос создался в контексте коллекции.



Нажимем SEND

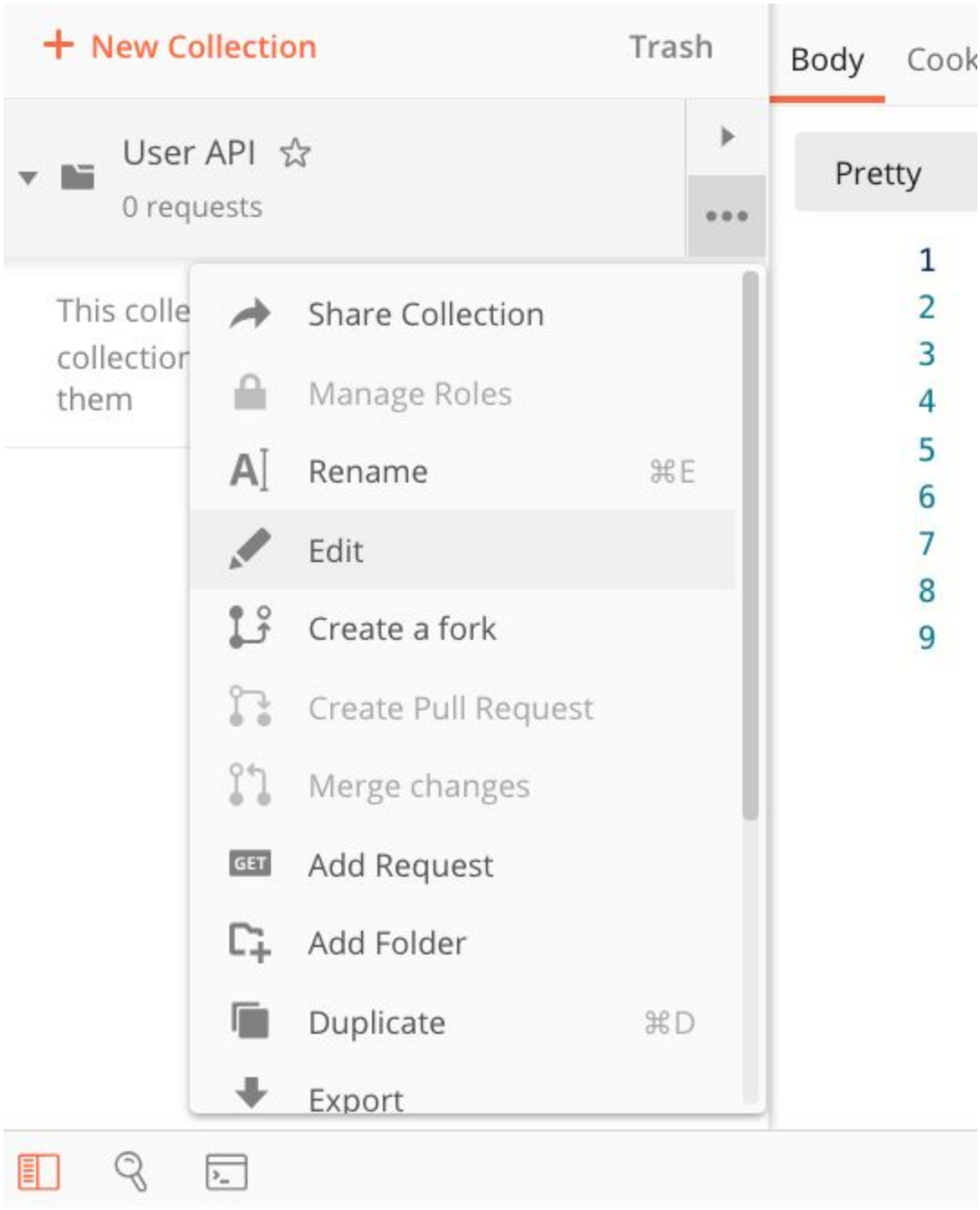
Видим response



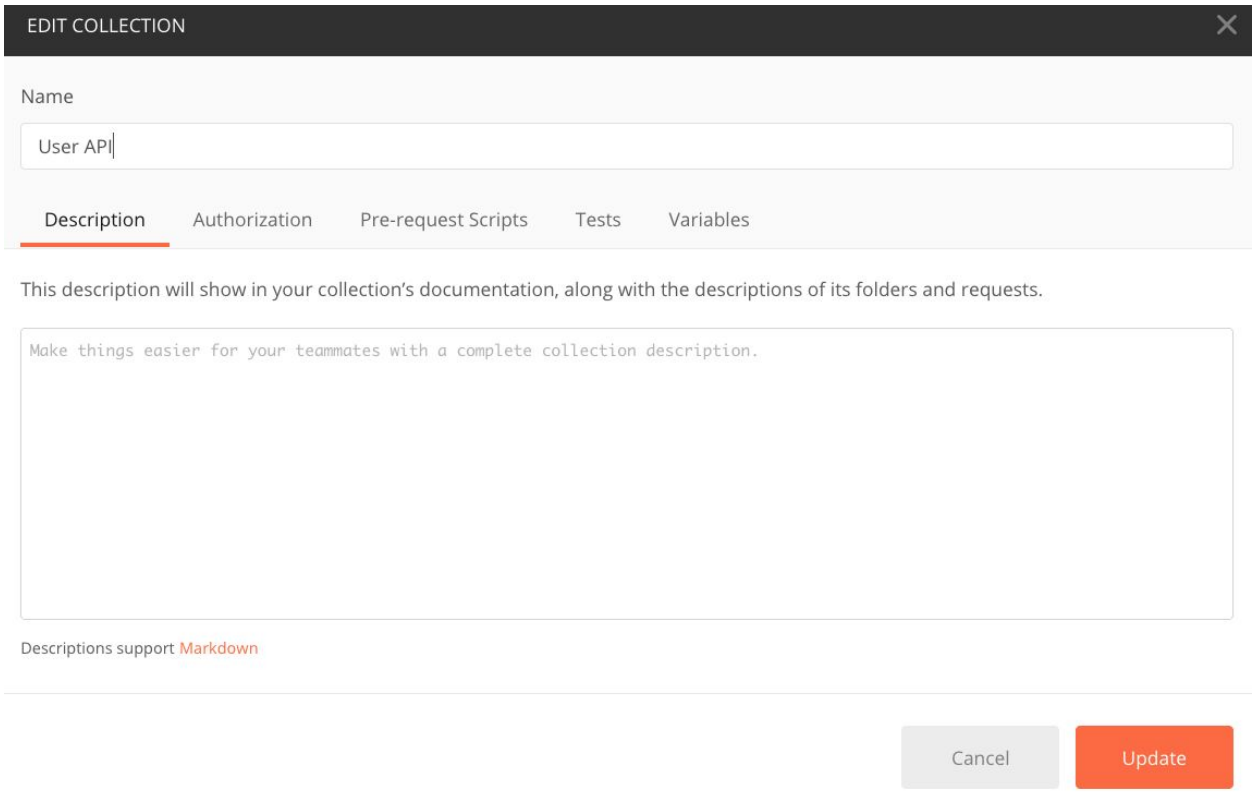
Для того, чтобы проверить получение пользователя, нам нужно сделать запрос GET на /api/v1/users/<идентификатор пользователя>

Можно конечно это сделать руками, но это норм для разовых запросов и плохо, если мы хотим написать переносимый тест

Чтобы сделать chaining запросов (использовать данные из ответа одного запроса в следующем), воспользуемся механизмом переменных коллекции из Postman.

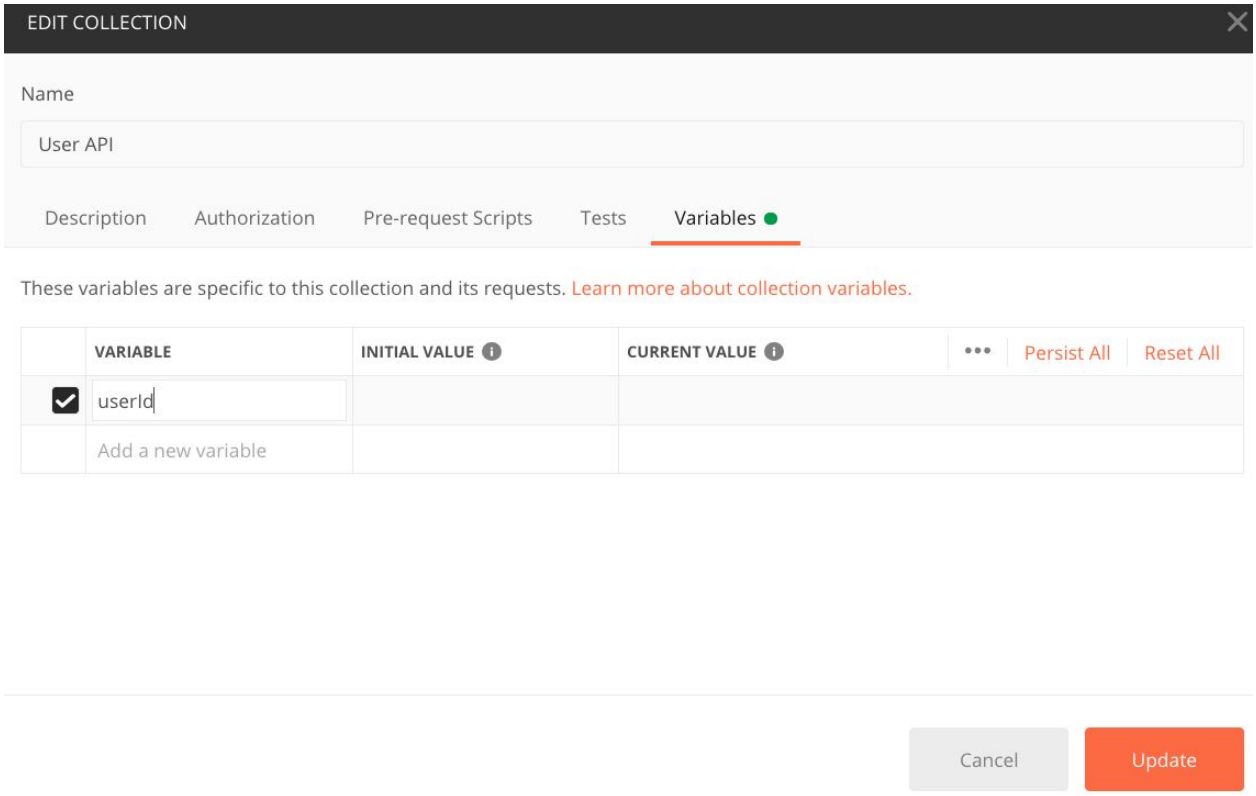

New Collection
Trash
Body
Cook
User API
0 requests
Pretty
1
2
3
4
5
6
7
8
9
This colle
collectior
them
Share Collection
Manage Roles
Rename
⌘E
Edit
Create a fork
Create Pull Request
Merge changes
GET
Add Request
Add Folder
Duplicate
⌘D
Export

Нажием Edit



Открываем вкладку Variables - это будут переменные коллекции

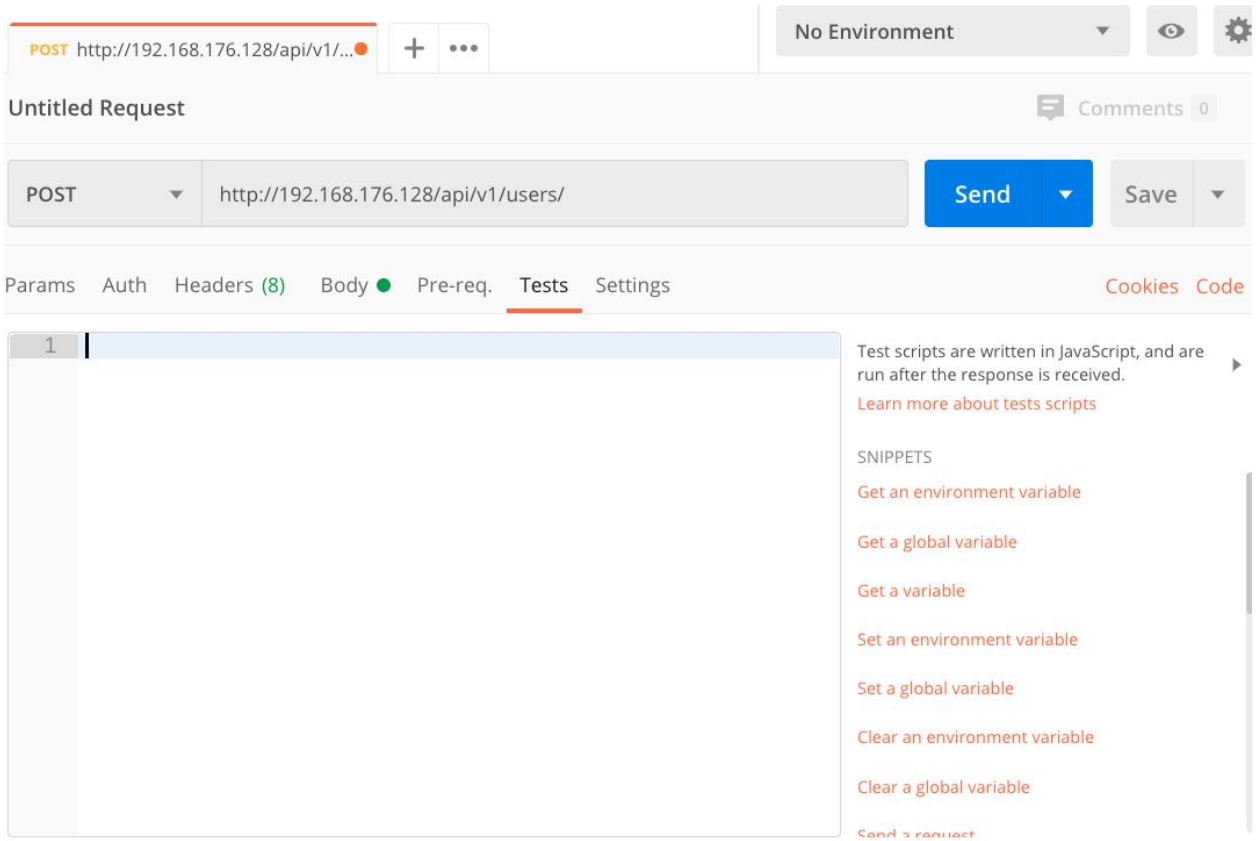
Определяем там переменную userId



Теперь мы можем ее использовать.

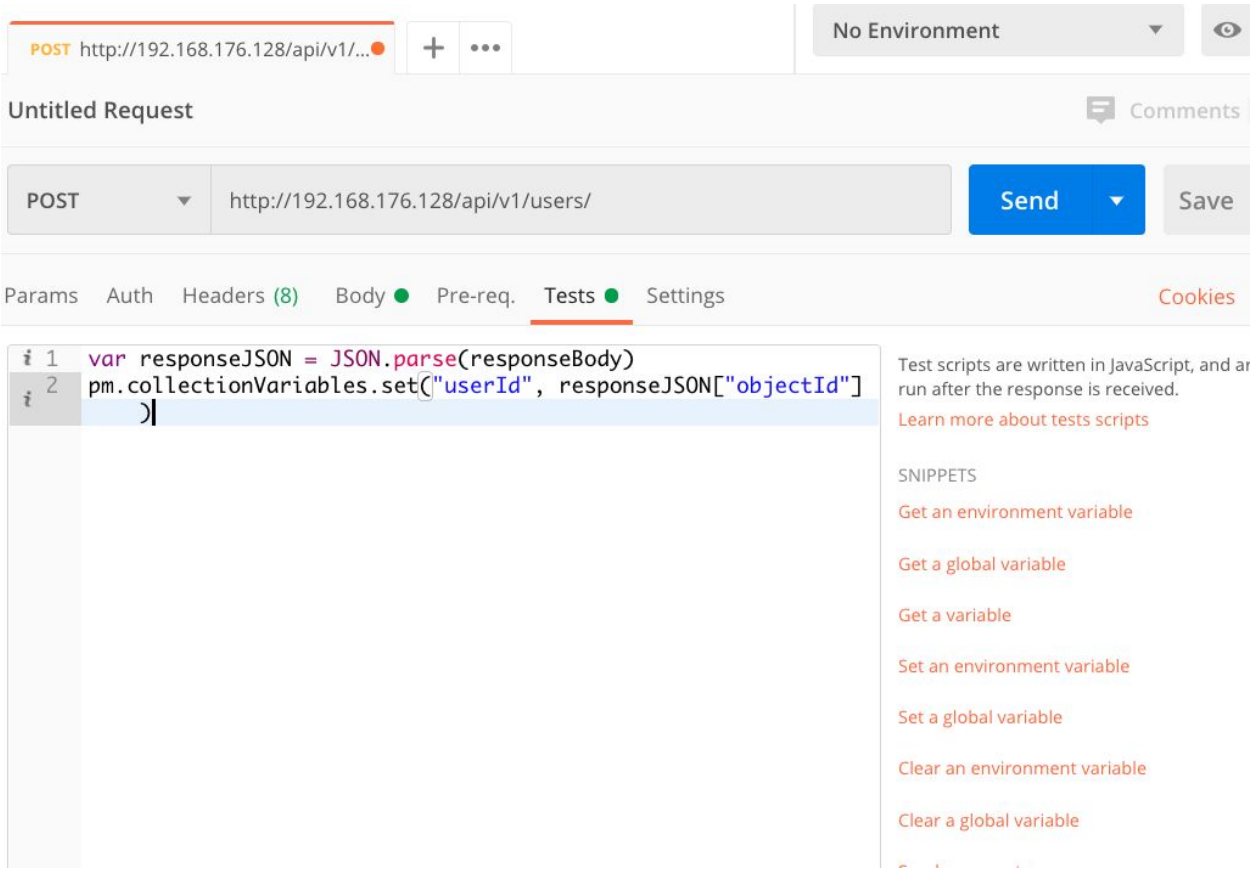
Сначала нужно ее установить.

Во вкладке tests:



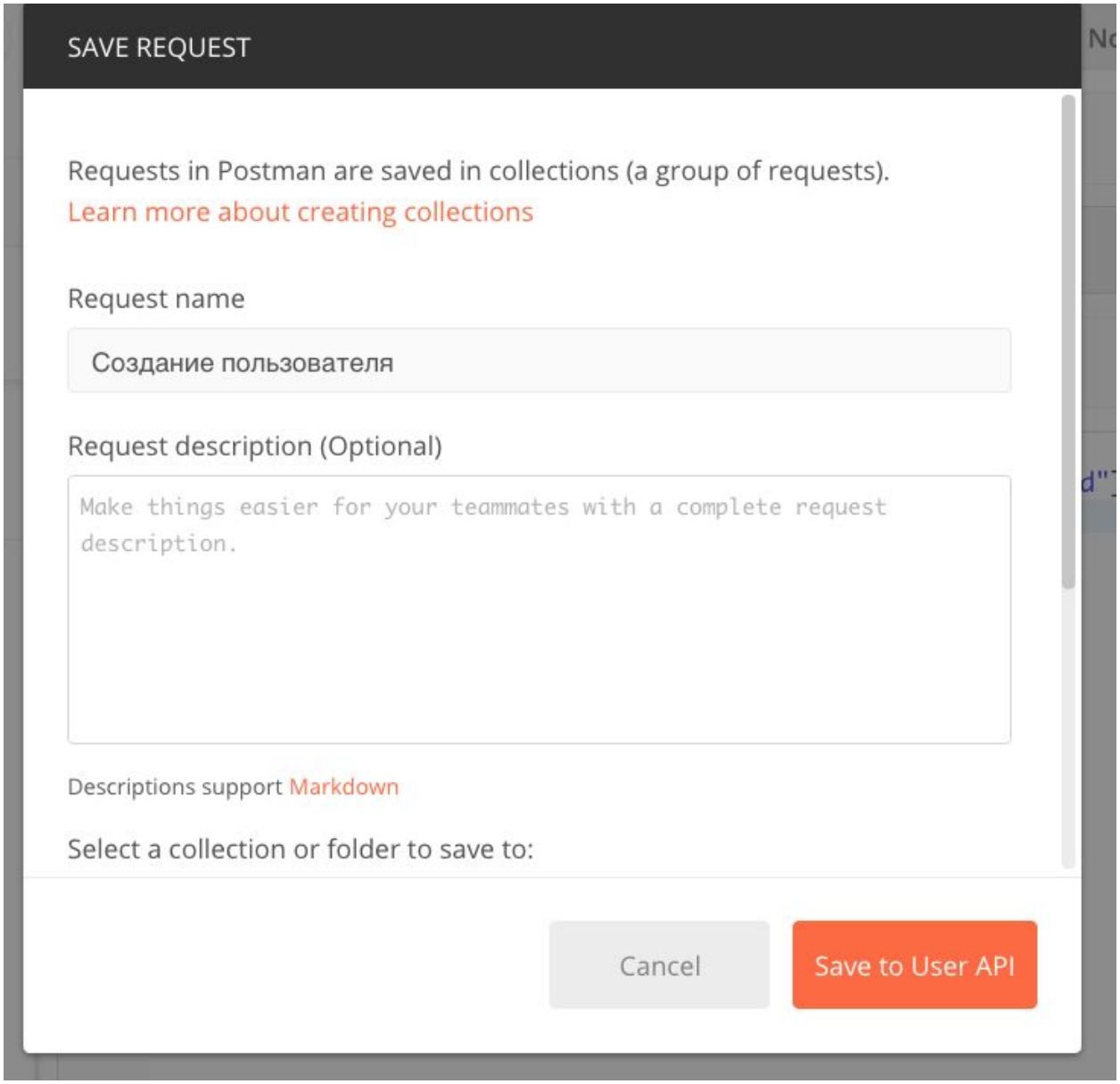
можно как написать тесты на JS. Так и скрипт установки значений

```
var responseJSON = JSON.parse(responseBody)
pm.collectionVariables.set("userId", responseJSON["objectId"])
```

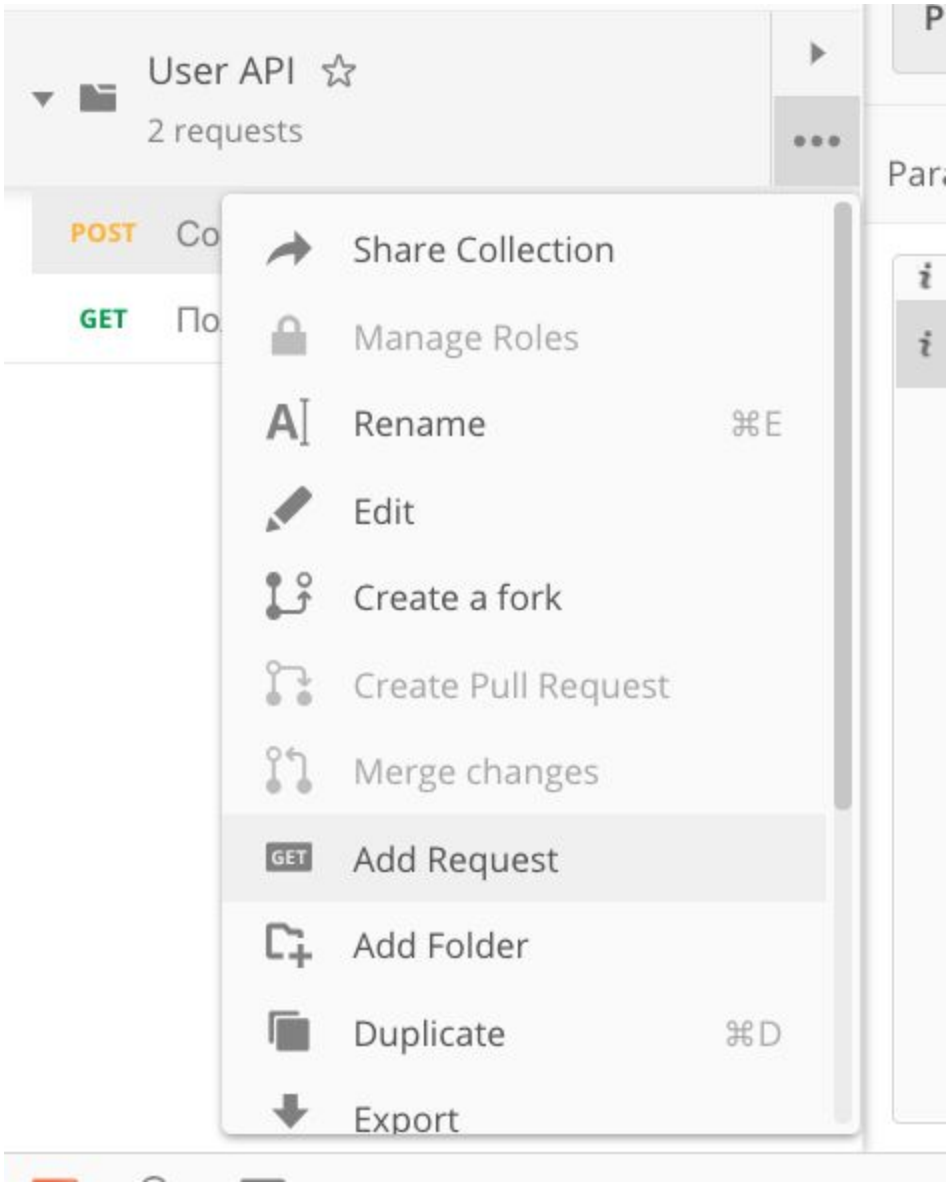


Теперь можем использовать значение этой переменной в следующих запросах

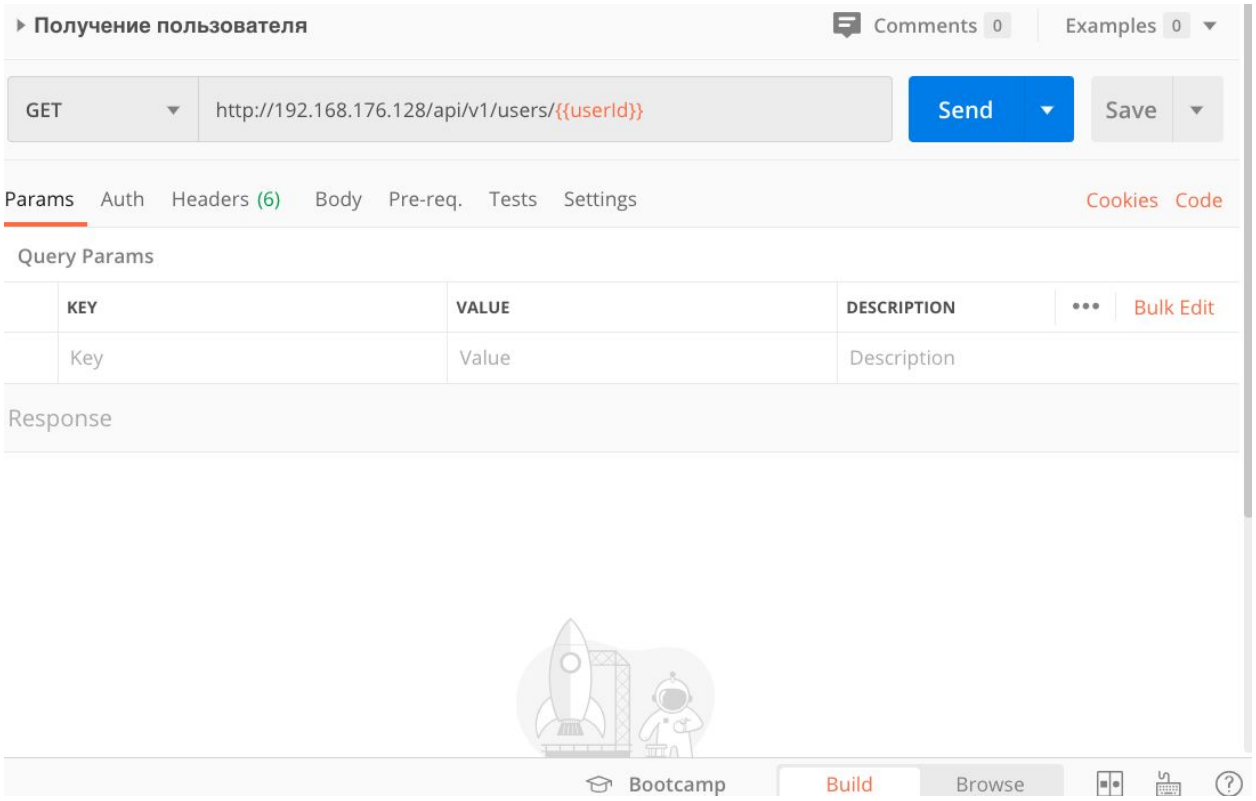
Сохраняем запрос в коллекции

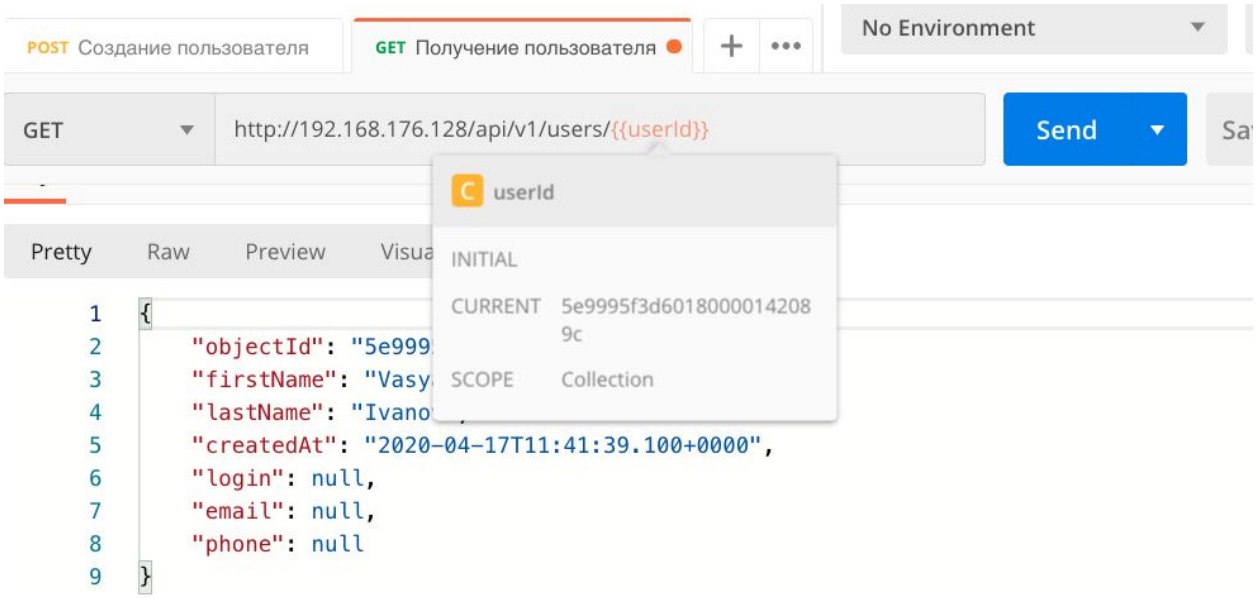


Добавляем запрос “Получение пользователя”

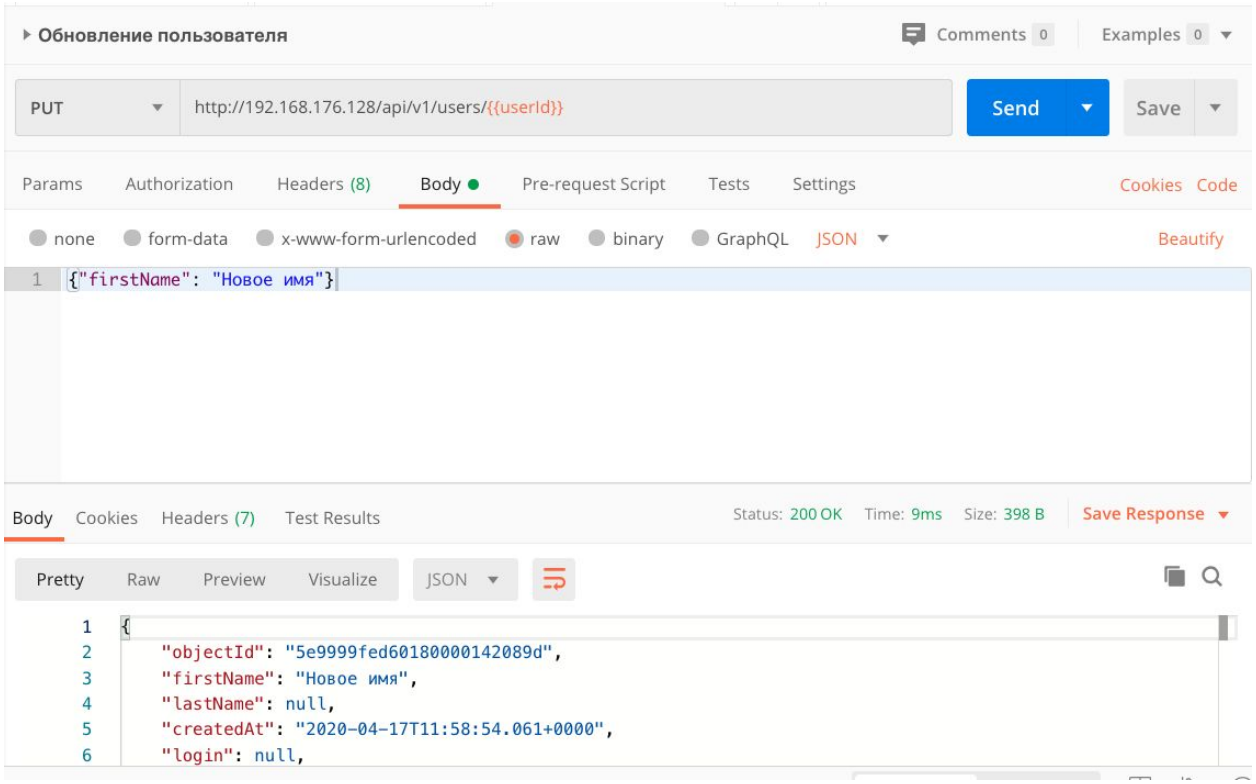


И обращаемся к значению переменной в URL как {{userId}}


Получение пользователя
Comments 0
Examples 0
GET
http://192.168.176.128/api/v1/users/{{userId}}
Send
Save
Params
Auth
Headers (6)
Body
Pre-req.
Tests
Settings
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Response
Bootcamp
Build
Browse

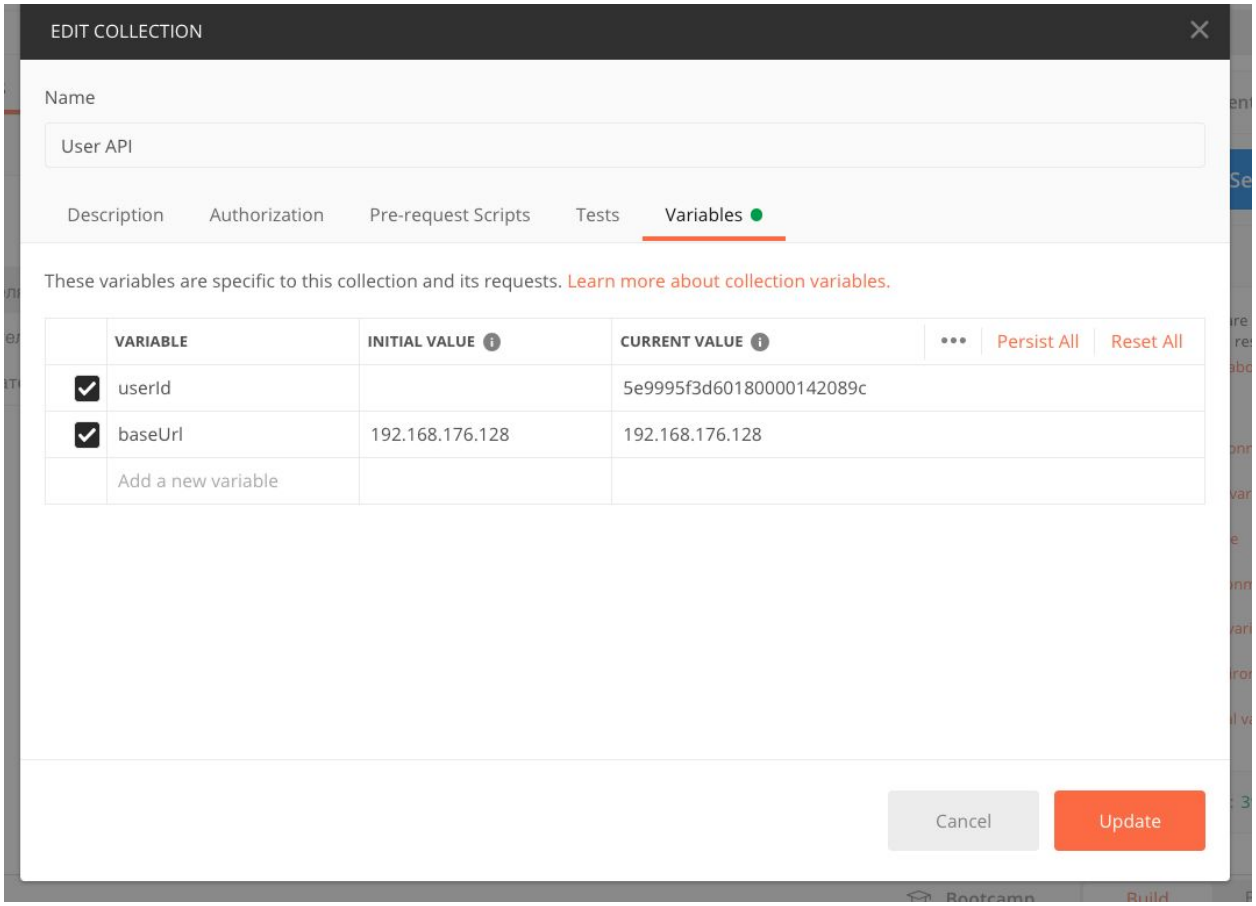


Ну и дальше уже можно удаление и изменений по той же схеме



После внесения изменений не забываем нажать сохранить (чтобы изменения сохранились)

Еще правилом хорошего тона в таких запросах является переменная baseUrl Так чтобы запрос не содержал конкретных ip адресов и доменов



Обновляем все урлы

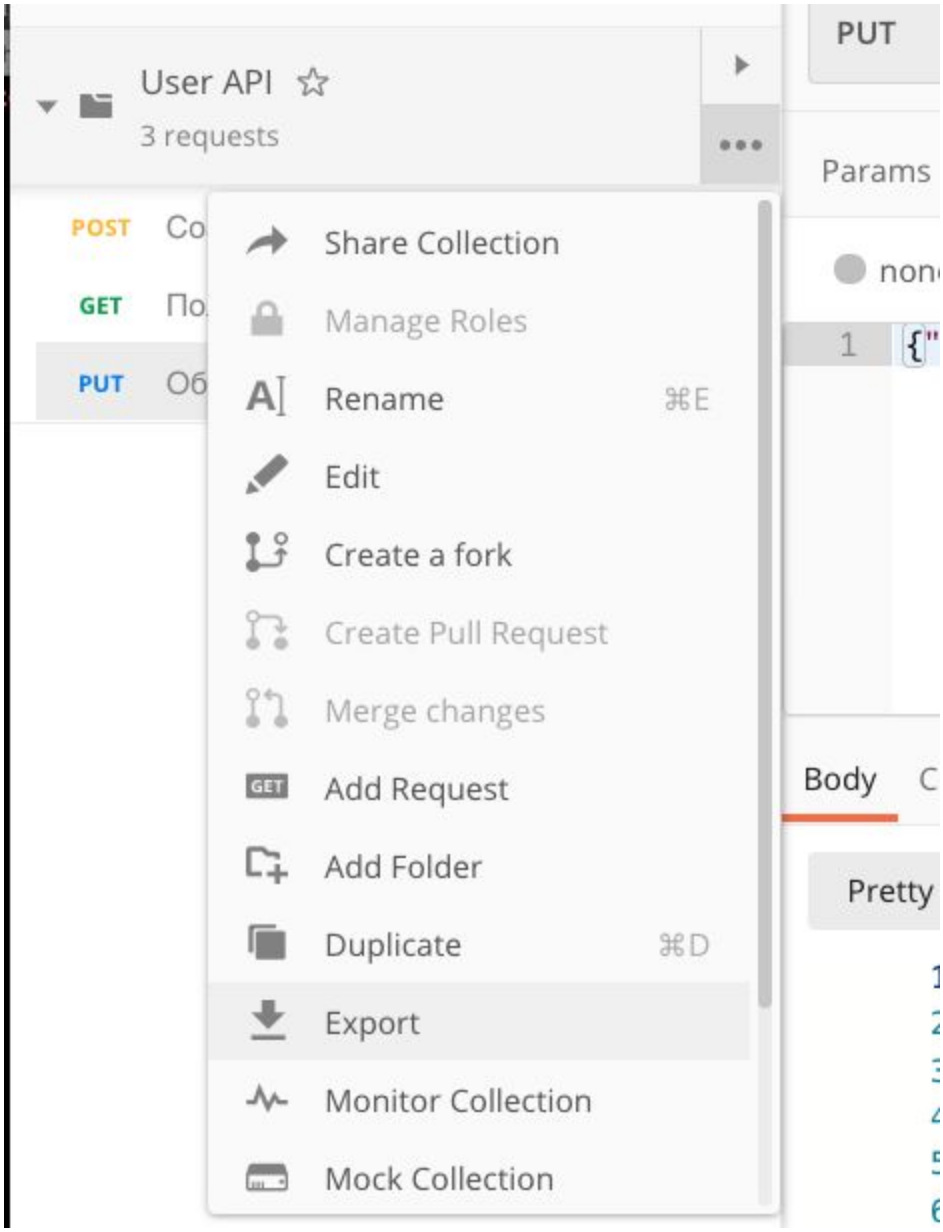
http://{{baseUrl}}/api/v1/users/

http://{{baseUrl}}/api/v1/users/{{userId}}

http://{{baseUrl}}/api/v1/users/{{userId}}

И т.д

Для того, чтобы экспортировать коллекцию в JSON, прежде всего СОХРАНИМ все запросы.



И сохраняем файл в JSON.

Можно проверить, правильно ли все работает с помощью утилиты newman

Устанавливается с помощью brew install newman

```
→  pods git:(master) ✗ newman run ~/Desktop/User\ API.postman_collection.json
newman

User API

→ Создание пользователя
  POST http://192.168.176.128/api/v1/users/ [200 OK, 390B, 49ms]

→ Получение пользователя
  GET http://192.168.176.128/api/v1/users/5e999be5d6018000014208a0 [200 OK, 390B, 9ms]

→ Обновление пользователя
  PUT http://192.168.176.128/api/v1/users/5e999be5d6018000014208a0 [200 OK, 398B, 11ms]
```

	executed	failed
iterations	1	0
requests	3	0
test-scripts	4	0
prerequest-scripts	3	0
assertions	0	0
total run duration: 152ms		
total data received: 491B (approx)		
average response time: 23ms [min: 9ms, max: 49ms, s.d.: 18ms]		

```
→  pods git:(master) ✗
```