Security Testing of Mass Assignment Vulnerabilities in RESTful APIs Hands on

Mariano Ceccato
mariano.ceccato@univr.it

Davide Corradini, Michele Pasqua

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

1. Nominal & Error testing

Obtain case study

```
docker pull davidecorradini94/bookstore
```

Run the case study

- Run the bookstore API

```
docker run -p 8080:8080 davidecorradini94/bookstore
```

- Check the running docker containers

```
docker ps
```

Open-API specification

Postman https://www.postman.com/

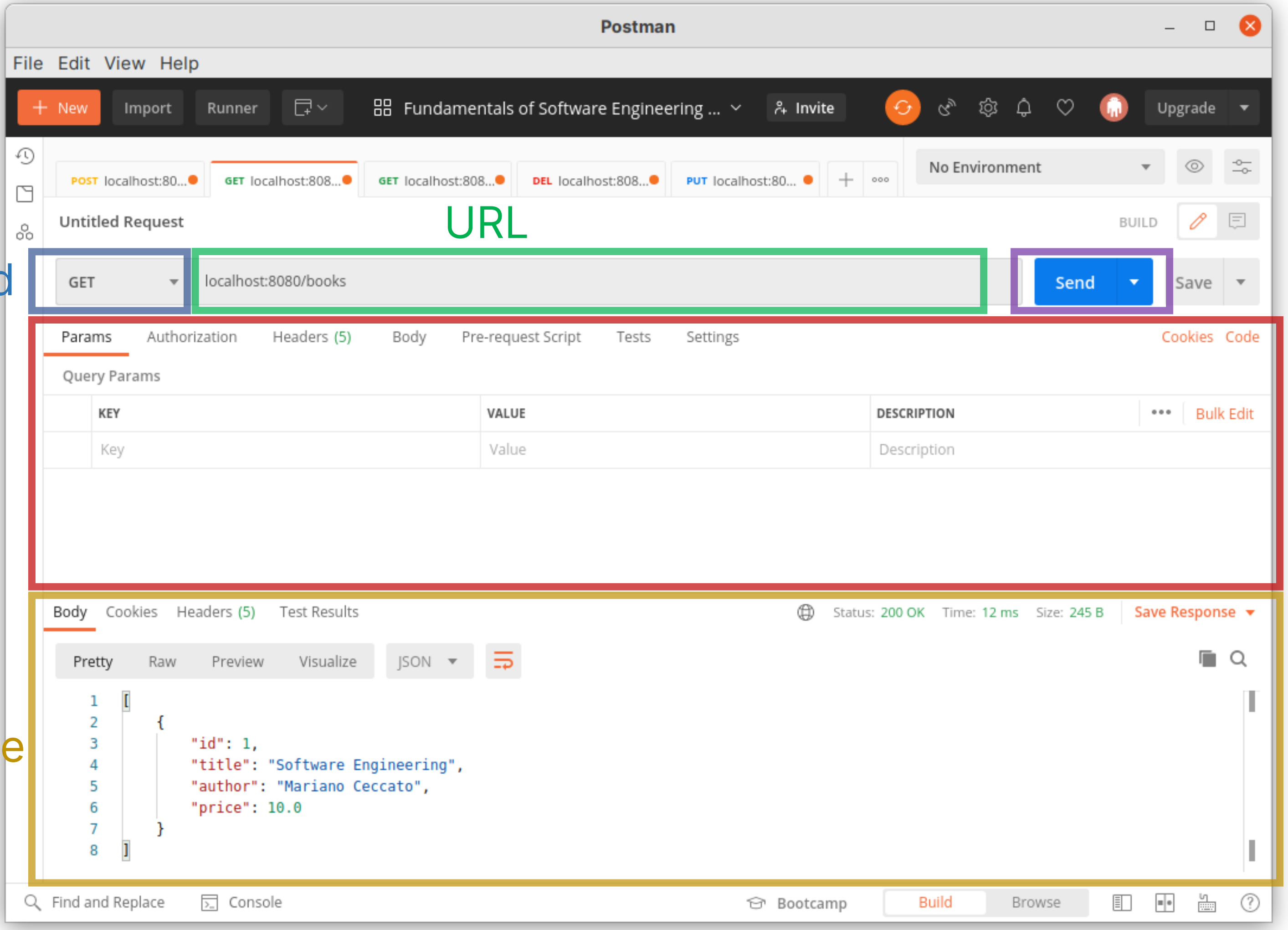
Method

URL

Send request button

Input parameters

Response



Untitled Request

GET localhost:8080/books

Query Params

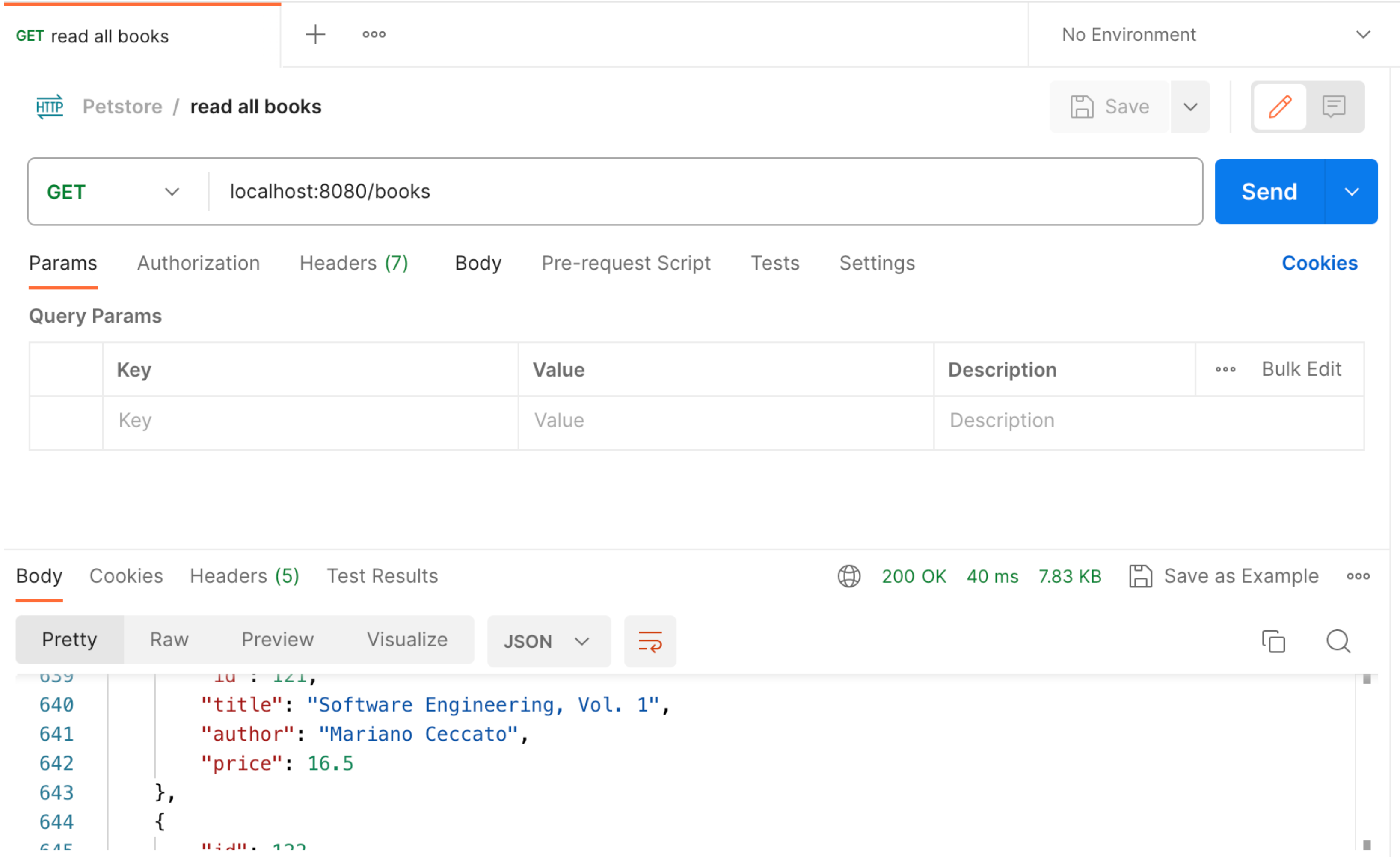
KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 12 ms Size: 245 B

```
[
    {
        "id": 1,
        "title": "Software Engineering",
        "author": "Mariano Ceccato",
        "price": 10.0
    }
]
```

List of books

- Point your browser to
  - http://localhost:8080/books
- or

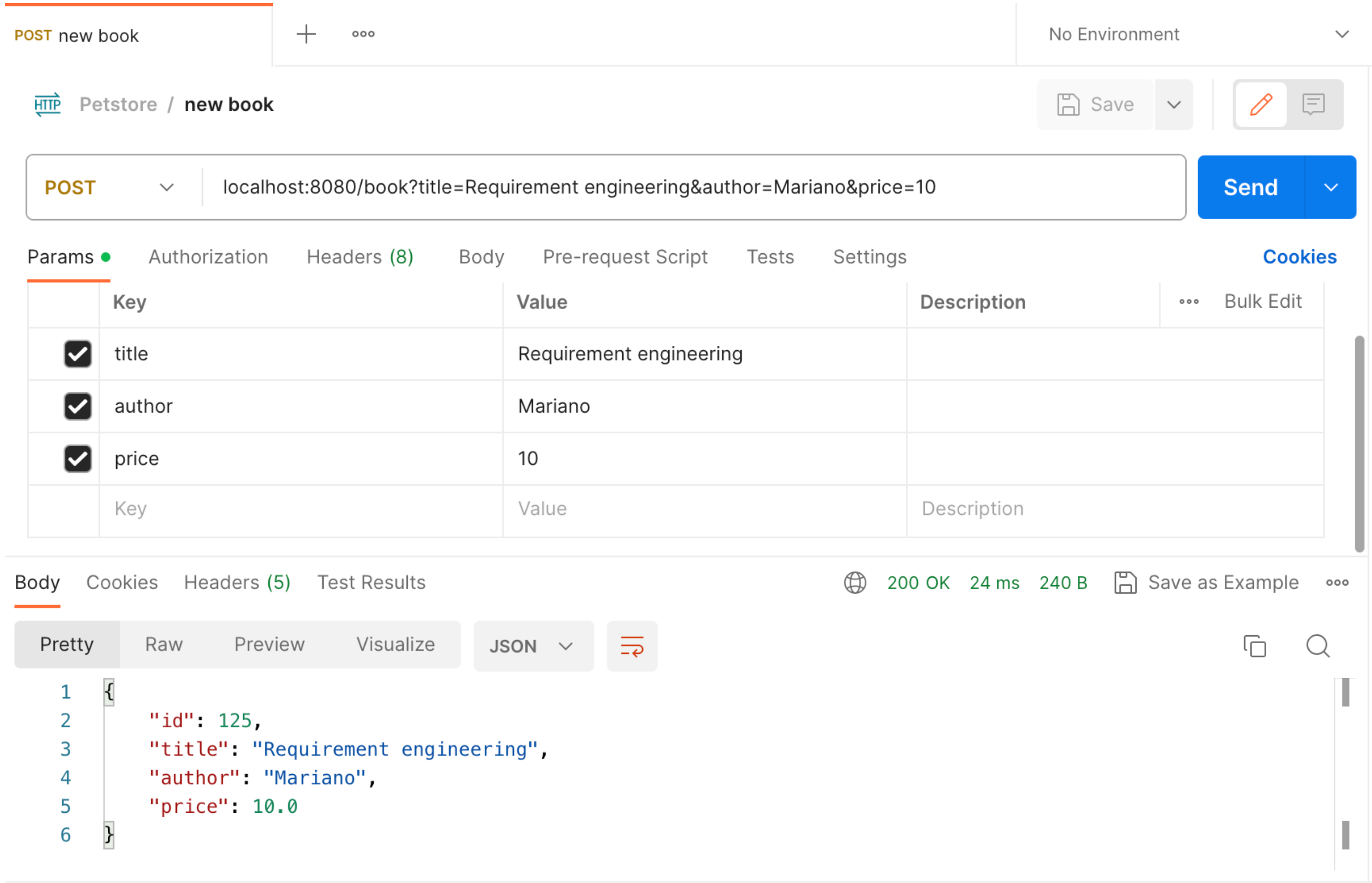


GET localhost:8080/books

200 OK 40 ms 7.83 KB

```
        "id": 121,
        "title": "Software Engineering, Vol. 1",
        "author": "Mariano Ceccato",
        "price": 16.5
    },
    {
        "id": 122
```

Add a new book



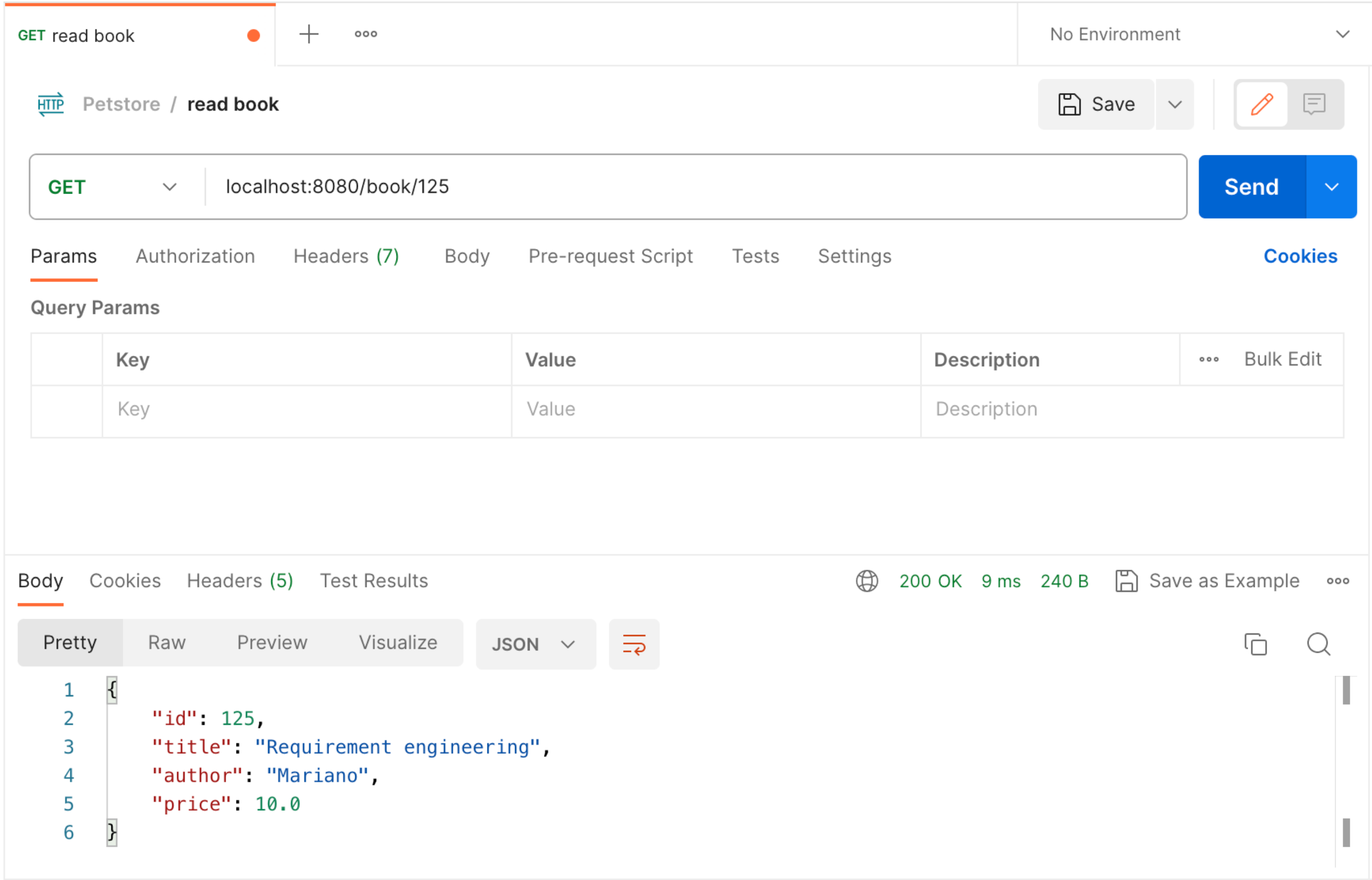
POST localhost:8080/book?title=Requirement engineering&author=Mariano&price=10

	Key	Value	Description
☑	title	Requirement engineering	
☑	author	Mariano	
☑	price	10	
	Key	Value	Description

200 OK 24 ms 240 B

```
{
    "id": 125,
    "title": "Requirement engineering",
    "author": "Mariano",
    "price": 10.0
}
```

Read a book



GET localhost:8080/book/125

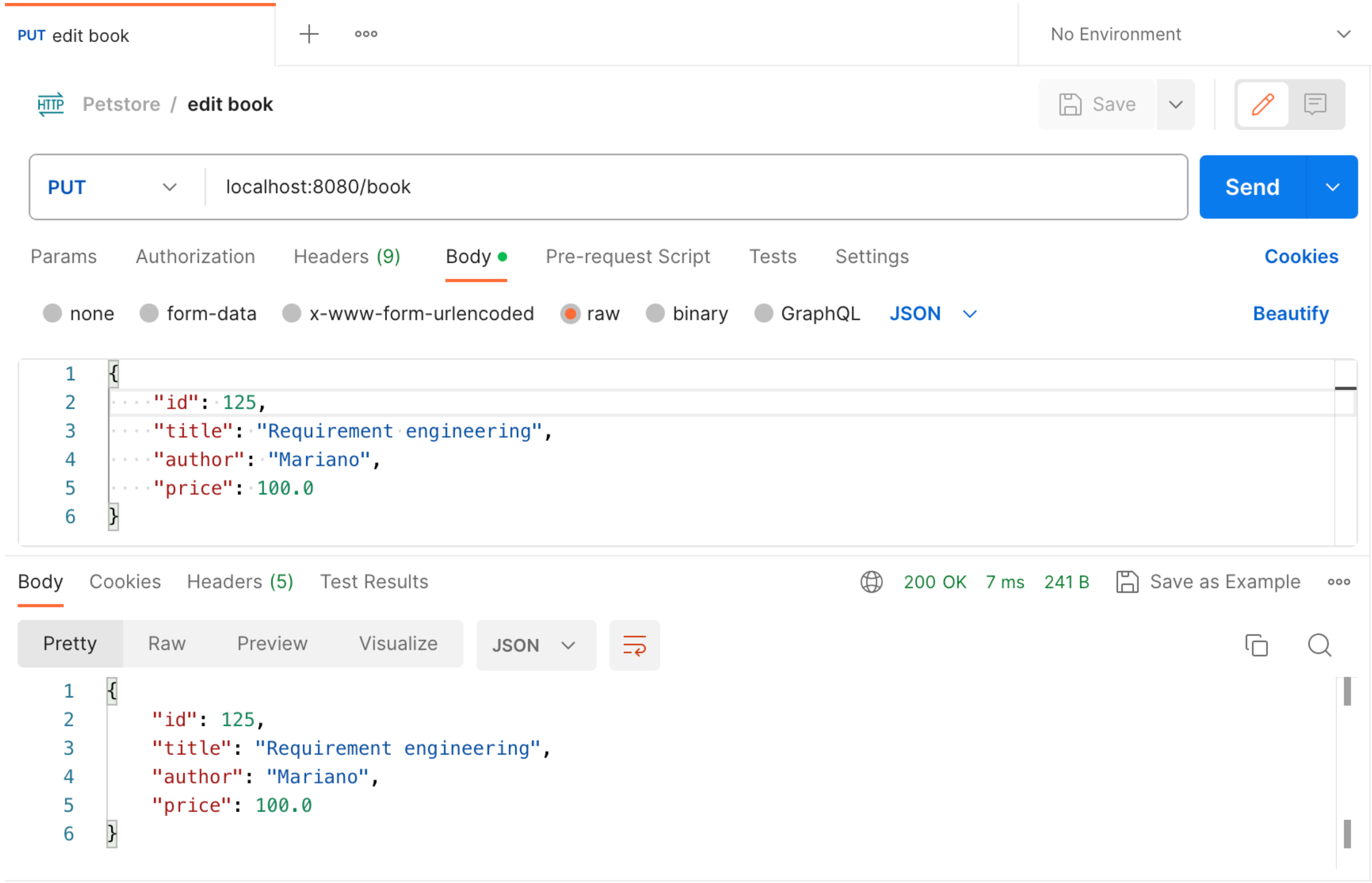
Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

200 OK 9 ms 240 B

```
{
    "id": 125,
    "title": "Requirement engineering",
    "author": "Mariano",
    "price": 10.0
}
```

Update a book



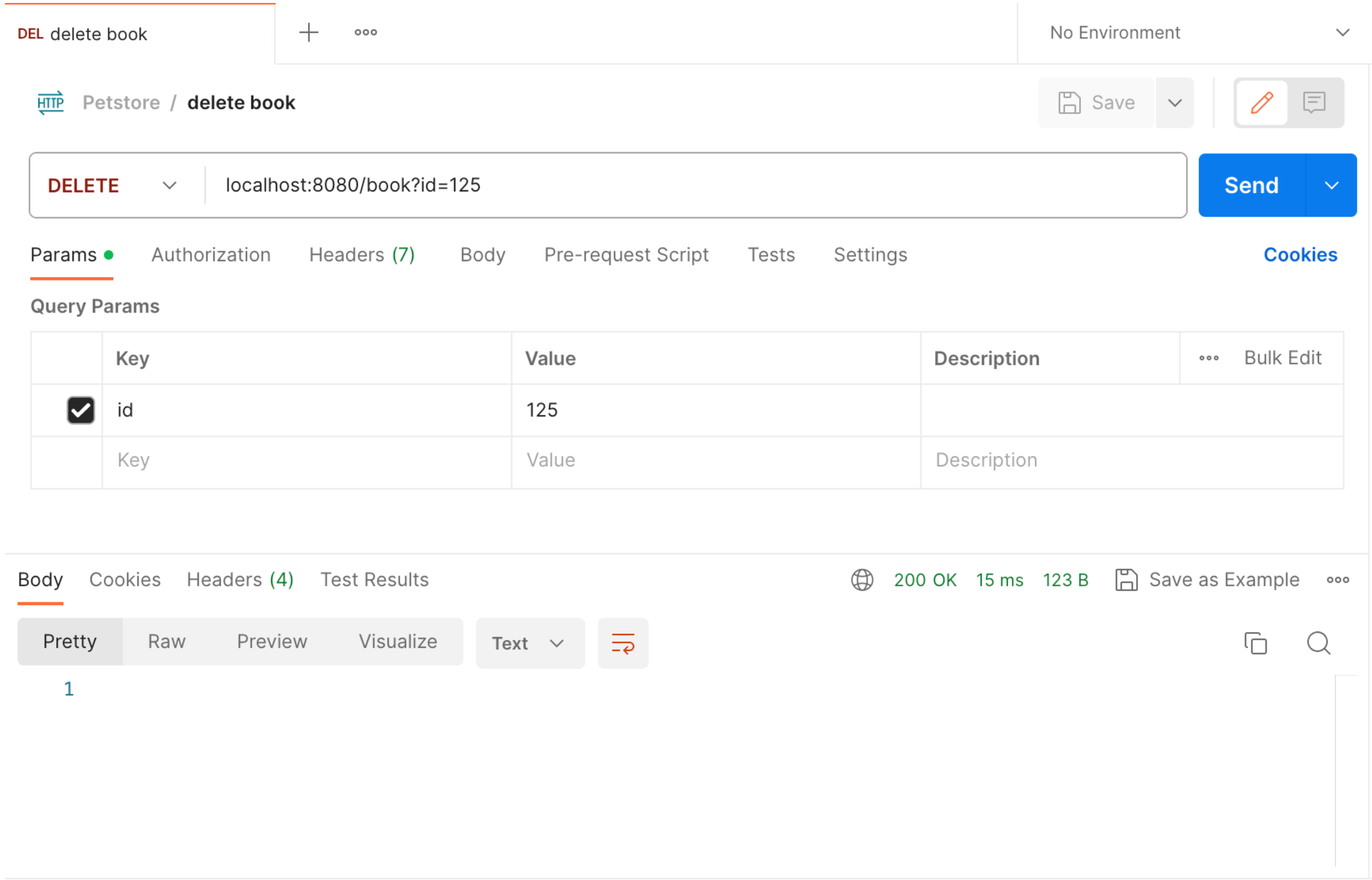
PUT localhost:8080/book

```
{
    "id": 125,
    "title": "Requirement engineering",
    "author": "Mariano",
    "price": 100.0
}
```

200 OK 7 ms 241 B

```
{
    "id": 125,
    "title": "Requirement engineering",
    "author": "Mariano",
    "price": 100.0
}
```

Delete a book



DELETE localhost:8080/book?id=125

Query Params

	Key	Value	Description
☑	id	125	
	Key	Value	Description

200 OK 15 ms 123 B

```
1
```

Obtain RestTestGen

- Clone the official public source code repository

```
git clone https://github.com/SeUniVr/RestTestGen
cd RestTestGen
```

```
docker build -t rtg .
```

With docker

```
sudo chmod +x gradlew
./gradlew build
```

With gradle on linix/mac

```
./gradlew.bat build
```

With gradle on Windows

Run RestTestGen

- Edit the configuration in file rtg-config.yml

```
apiUnderTest: bookstore
strategyClassName: NominalAndErrorStrategy
```

```
docker run -v ./:/app --network="host" rtg
```

Output

- **CoverageReports**: according to several metrics

- **Report**: interaction sequences
  - NominalFuzzer
  - ErrorFuzzer

- **REST-assured**: estarnally runnable test cases
  - NominalFuzzer
  - ErrorFuzzer

2. Security testing

Obtain case study

```
docker pull davidecorradini94/vampi-vuln
```

Run the case study

- Run the bookstore API

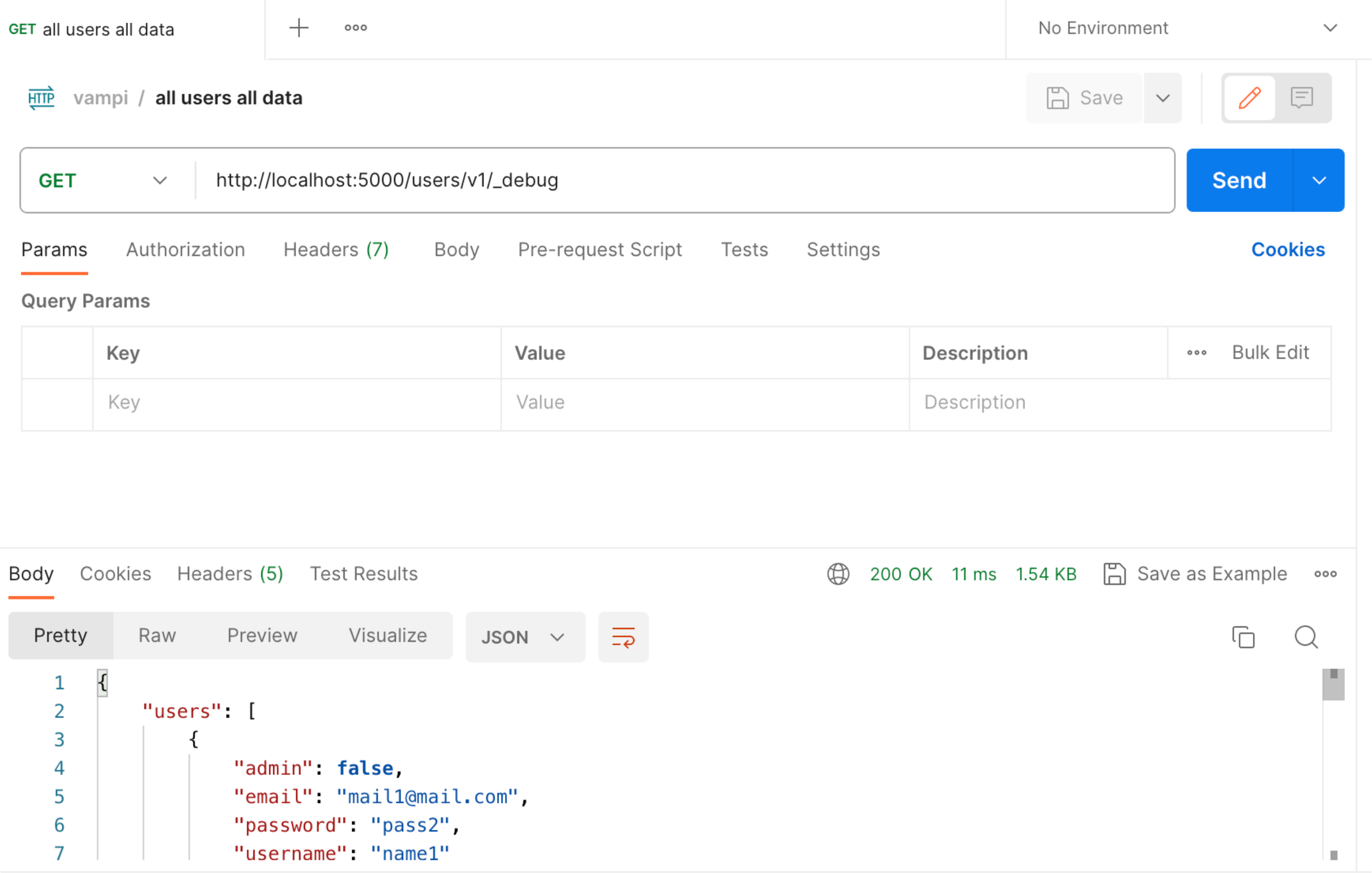
```
docker run -p 5000:5000 davidecorradini94/vampi-vuln
```

- Check the running docker containers

```
docker ps
```

Open-API specification

Read all the users



GET http://localhost:5000/users/v1/_debug

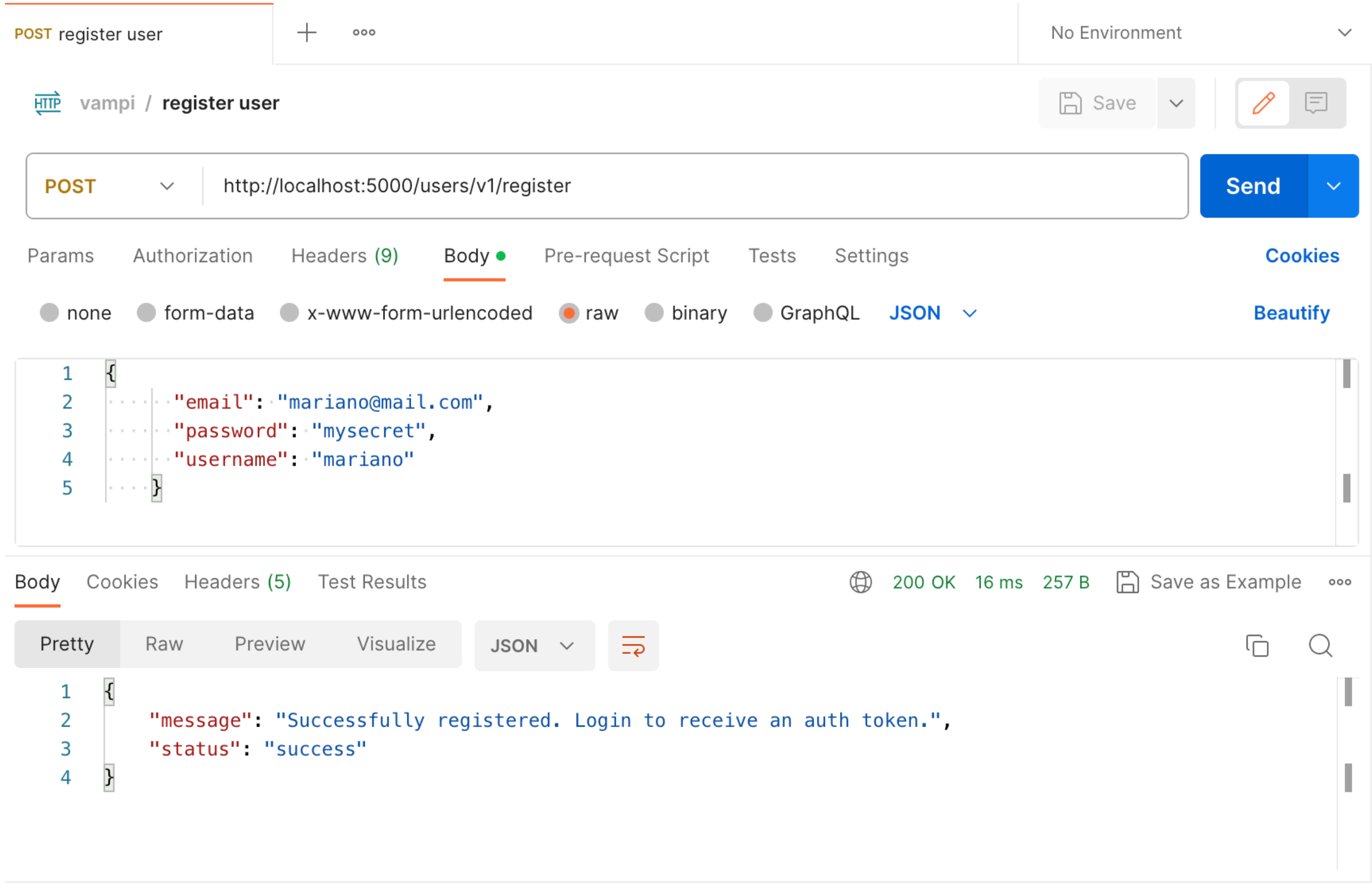
Query Params

Key	Value	Description
Key	Value	Description

200 OK 11 ms 1.54 KB

```
{
    "users": [
        {
            "admin": false,
            "email": "mail1@mail.com",
            "password": "pass2",
            "username": "name1"
```

Create new user



POST http://localhost:5000/users/v1/register

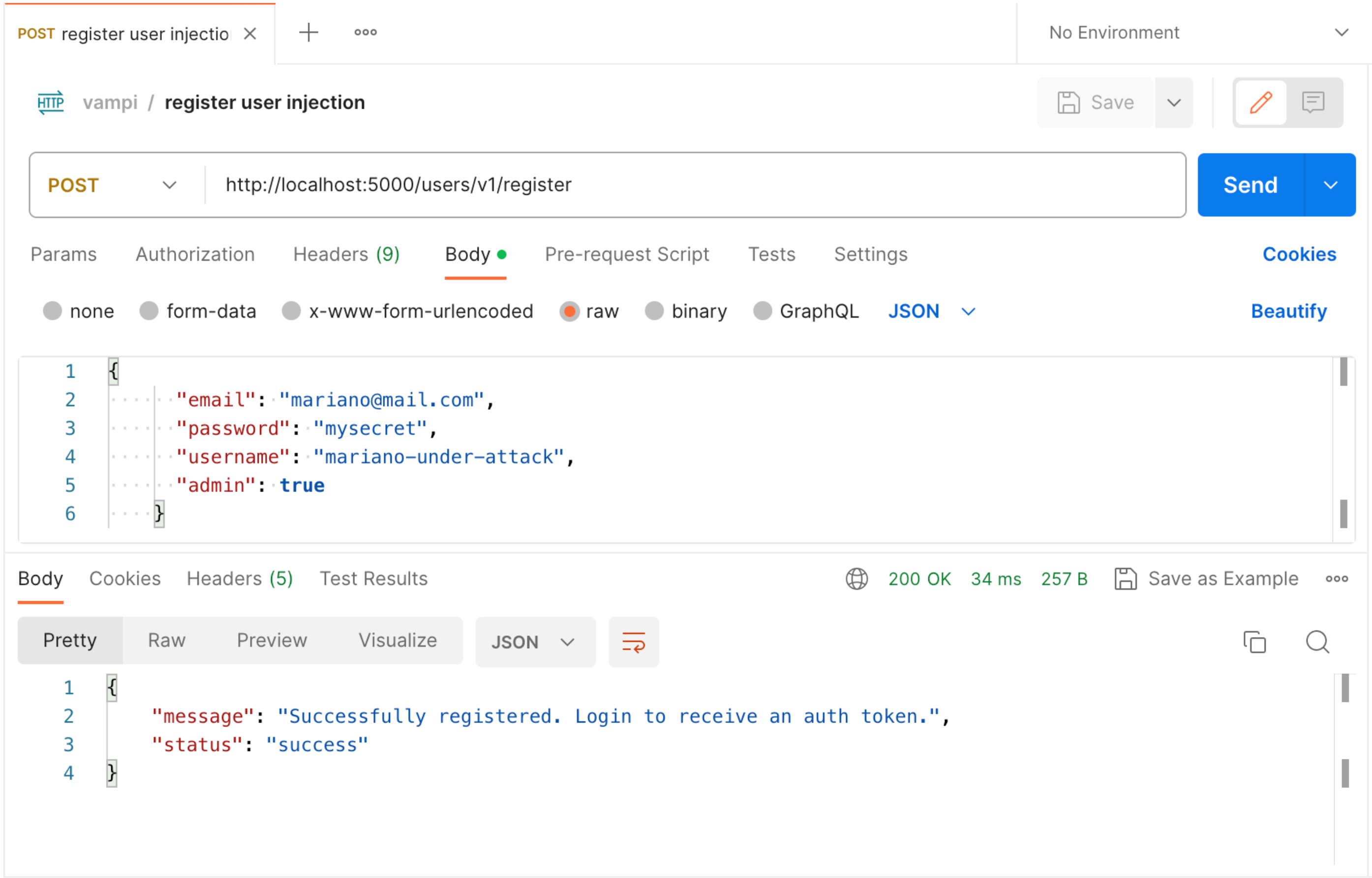
Body: ○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary ○ GraphQL JSON

```
{
    "email": "mariano@mail.com",
    "password": "mysecret",
    "username": "mariano"
    }
```

Body — 200 OK 16 ms 257 B — JSON

```
{
    "message": "Successfully registered. Login to receive an auth token.",
    "status": "success"
}
```

Create user with mass assignment



POST http://localhost:5000/users/v1/register

```
{
    "email": "mariano@mail.com",
    "password": "mysecret",
    "username": "mariano-under-attack",
    "admin": true
    }
```

```
{
    "message": "Successfully registered. Login to receive an auth token.",
    "status": "success"
}
```

Run RestTestGen

- Edit the configuration in file rtg-config.yml

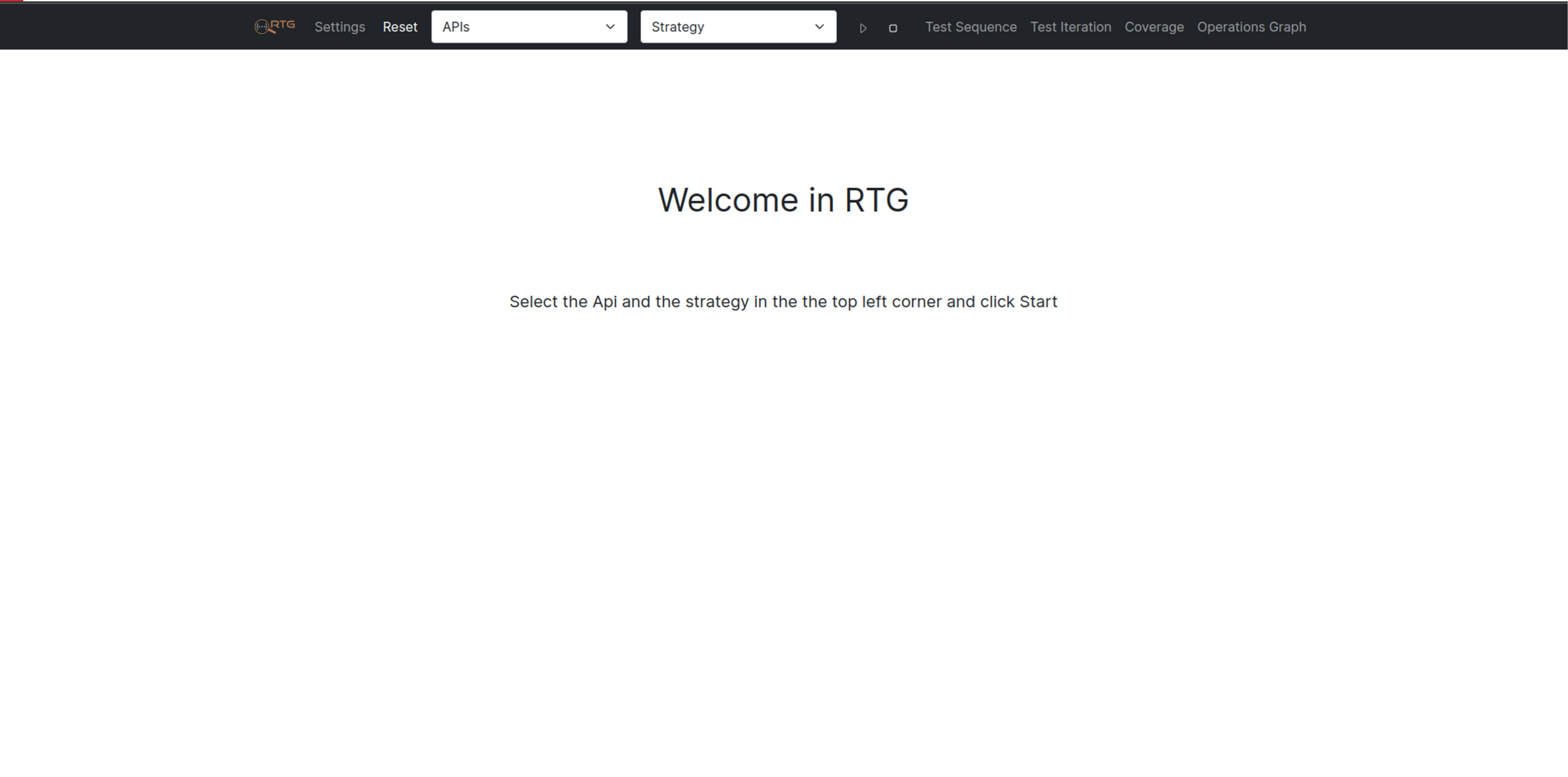
```
apiUnderTest: vampi
strategyClassName: MassAssignmentSecurityTestingStrategy
```

```
docker run -v ./:/app --network="host" rtg
```

Output

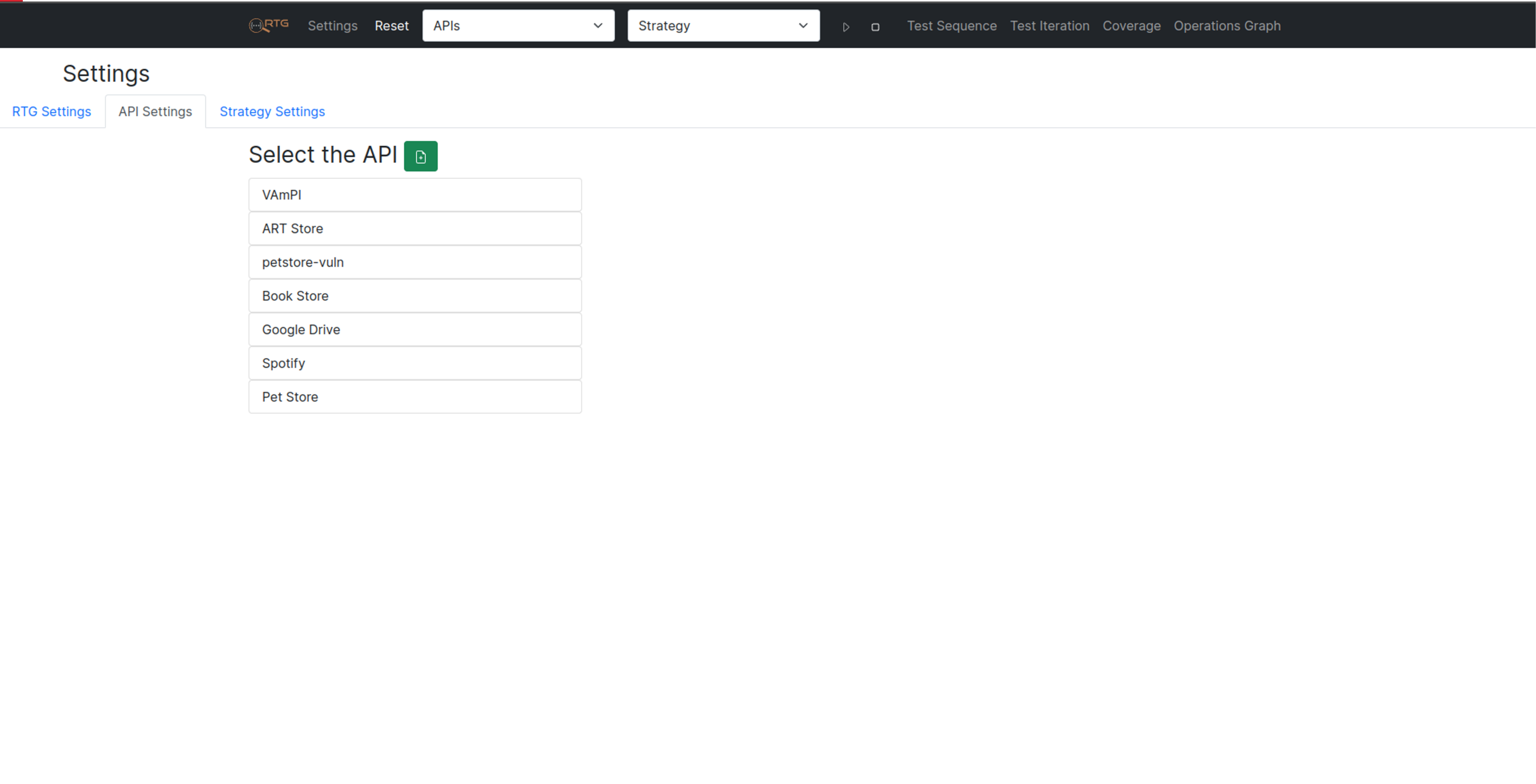
- **CRUDgroups**: result of clustering by resource type
- **Report**: interaction sequences
  - NominalFuzzer+MassAssignmentFuzzer+UserInstantiated

New feature: GUI



Welcome in RTG

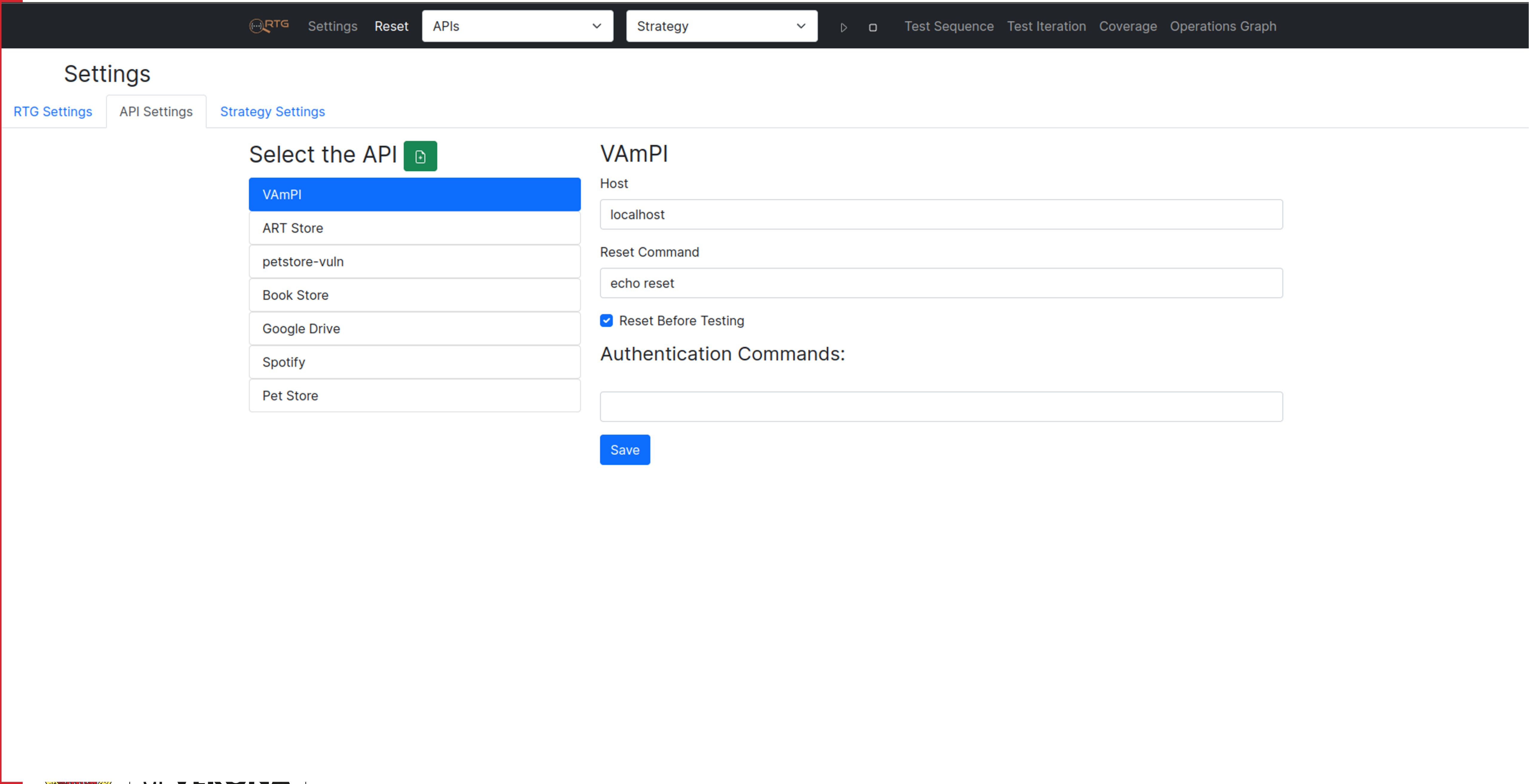
Select the Api and the strategy in the the top left corner and click Start



Settings

Select the API

- VAmPI
- ART Store
- petstore-vuln
- Book Store
- Google Drive
- Spotify
- Pet Store



Settings

Select the API

- VAmPI
- ART Store
- petstore-vuln
- Book Store
- Google Drive
- Spotify
- Pet Store

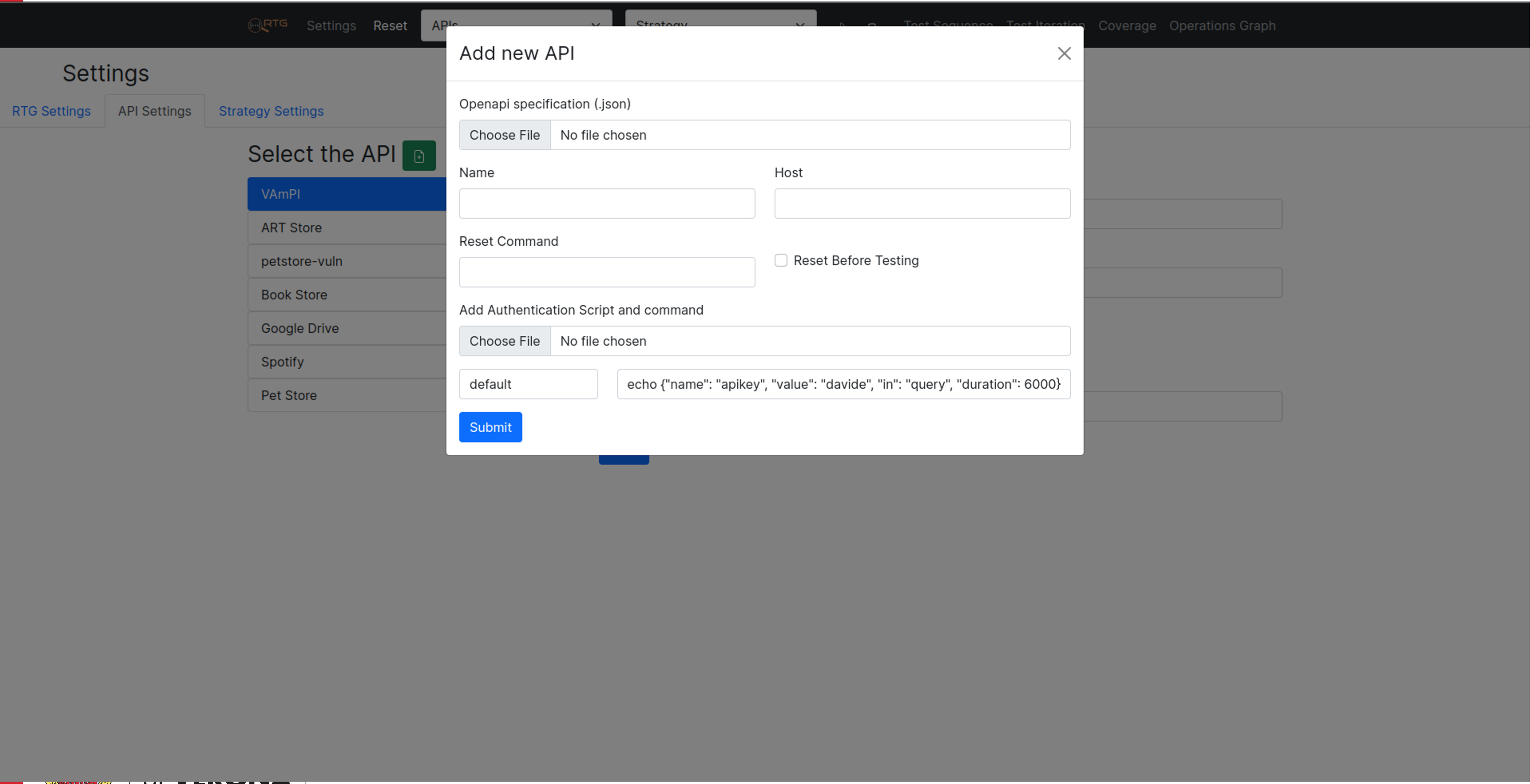
VAmPI

Host: localhost

Reset Command: echo reset

☑ Reset Before Testing

Authentication Commands:



Add new API

Openapi specification (.json)

Choose File No file chosen

Name

Host

Reset Command

□ Reset Before Testing

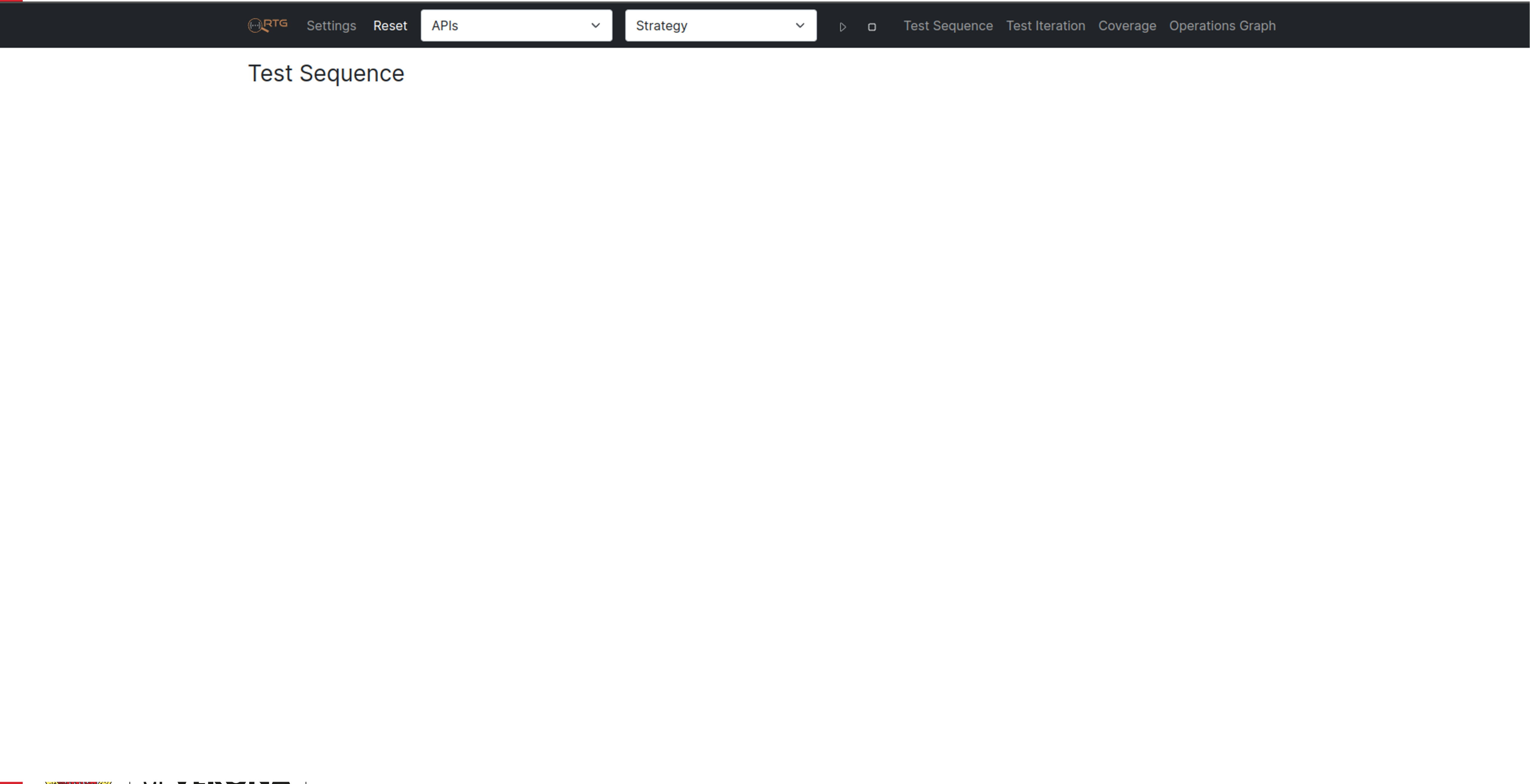
Add Authentication Script and command

Choose File No file chosen

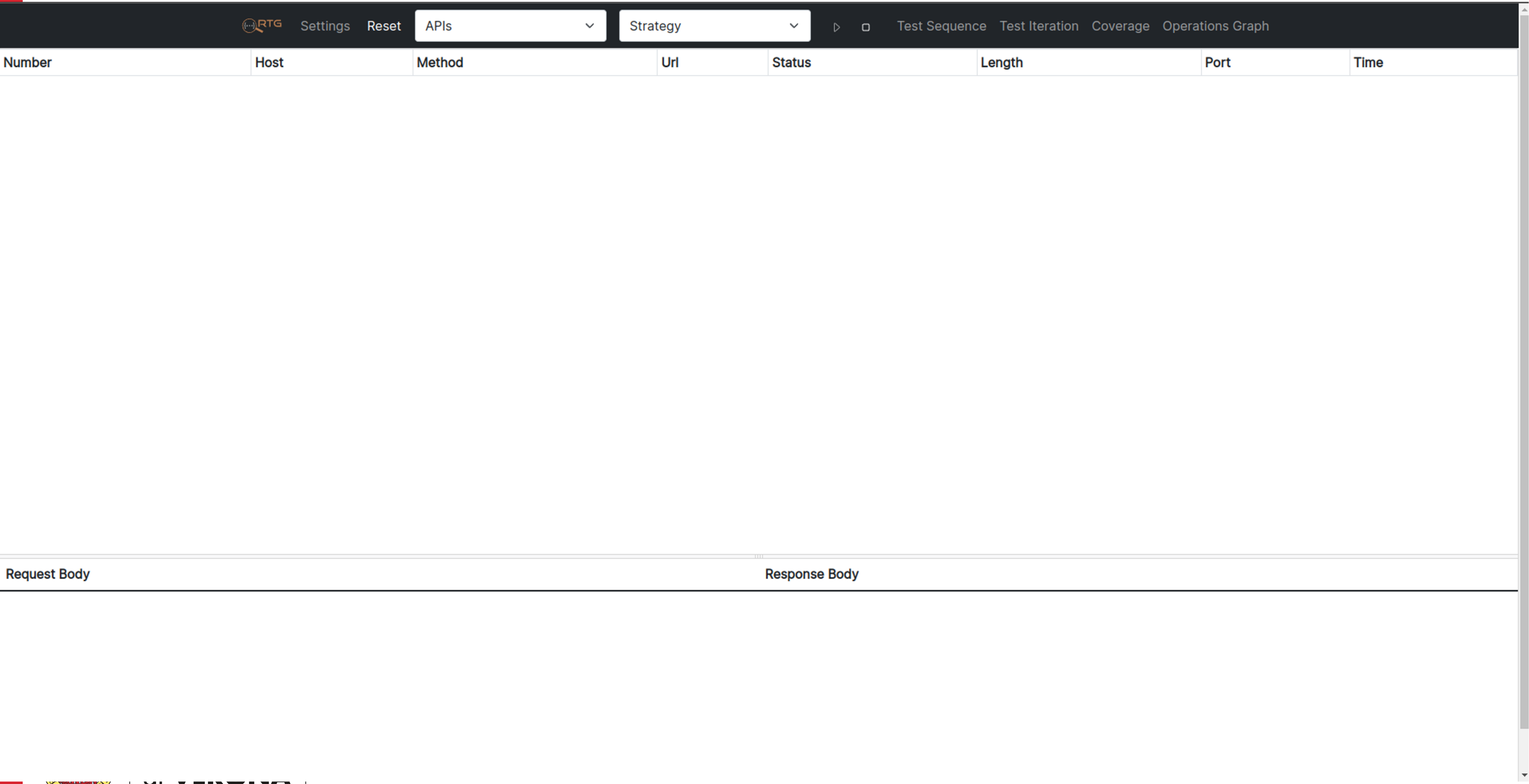
default

```
echo {"name": "apikey", "value": "davide", "in": "query", "duration": 6000}
```

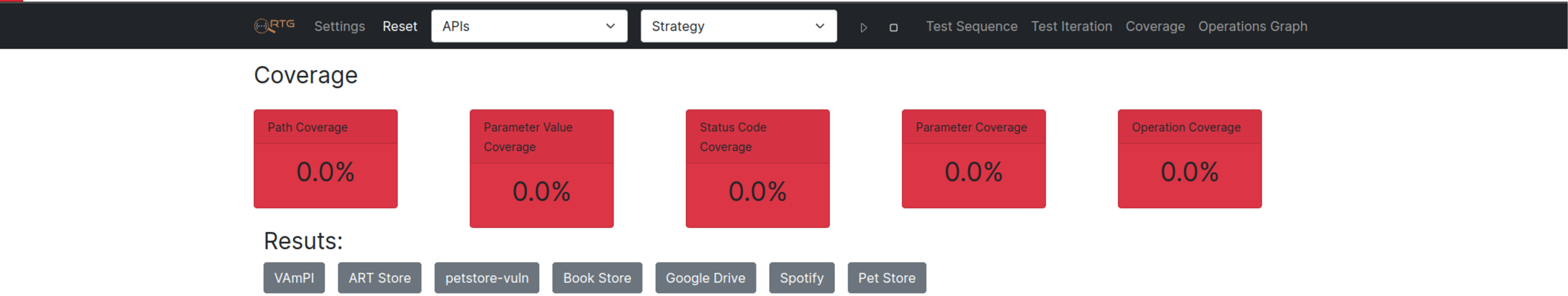
Submit



Test Sequence



Number	Host	Method	Url	Status	Length	Port	Time



Coverage

Path Coverage
0.0%

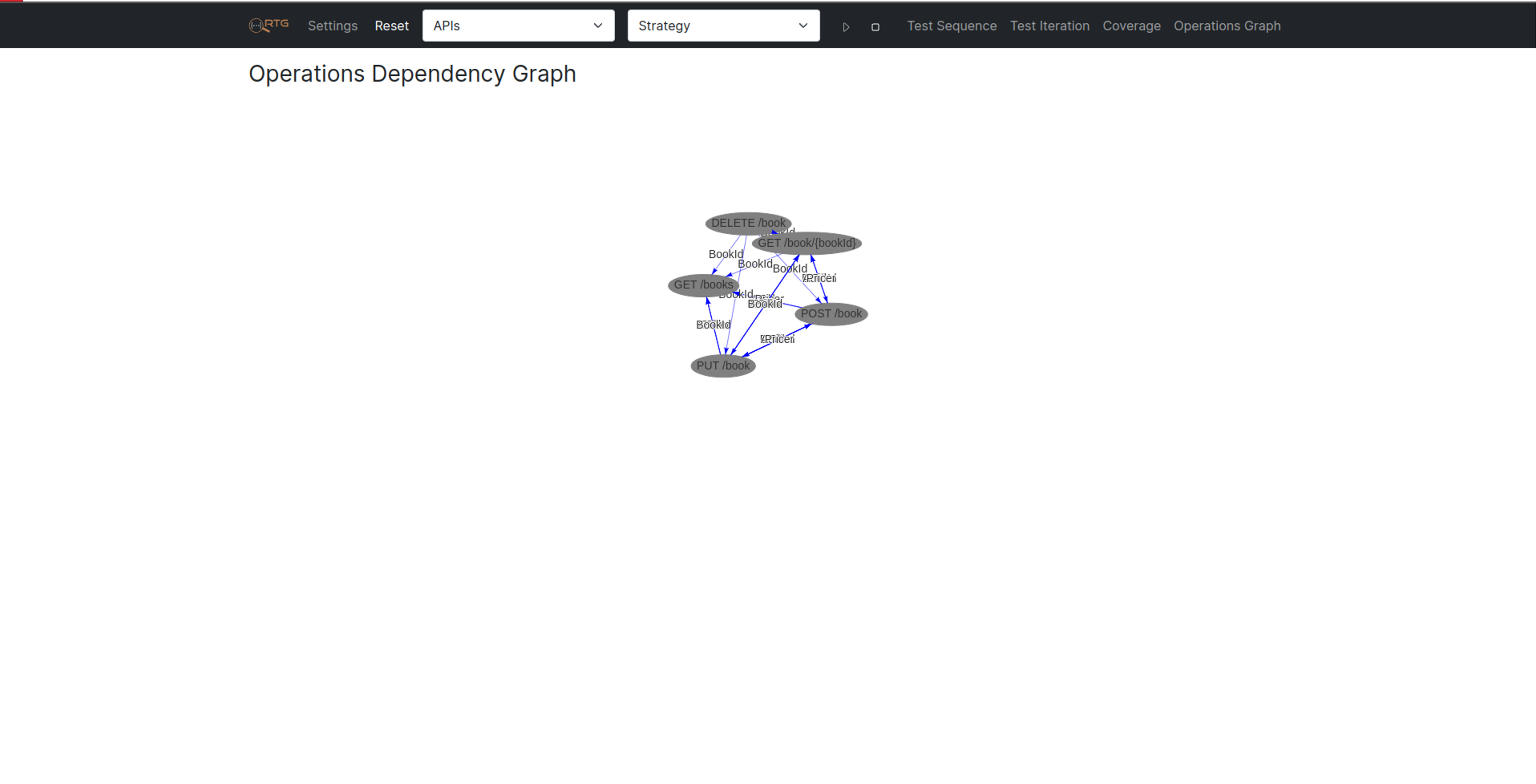
Parameter Value Coverage
0.0%

Status Code Coverage
0.0%

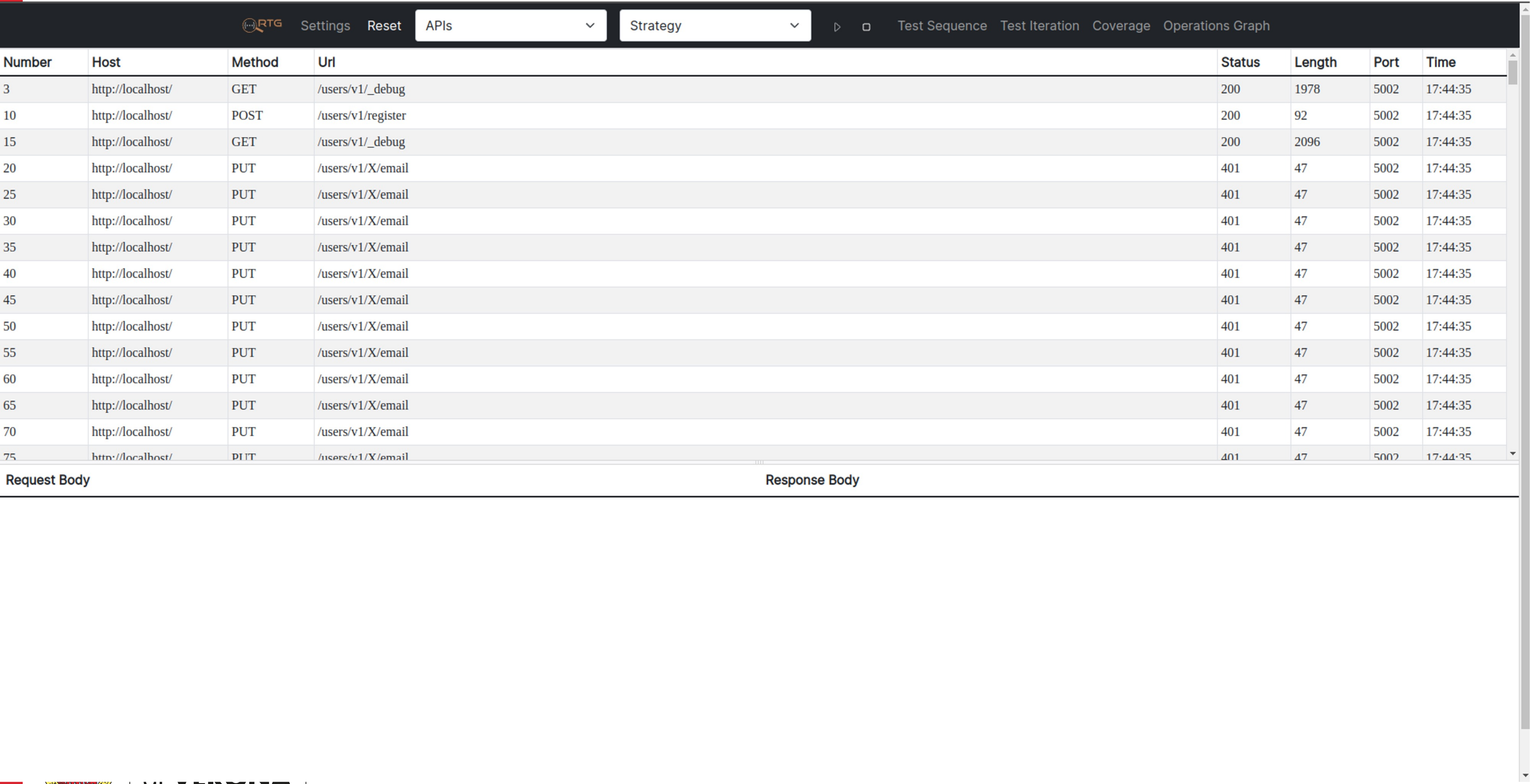
Parameter Coverage
0.0%

Operation Coverage
0.0%

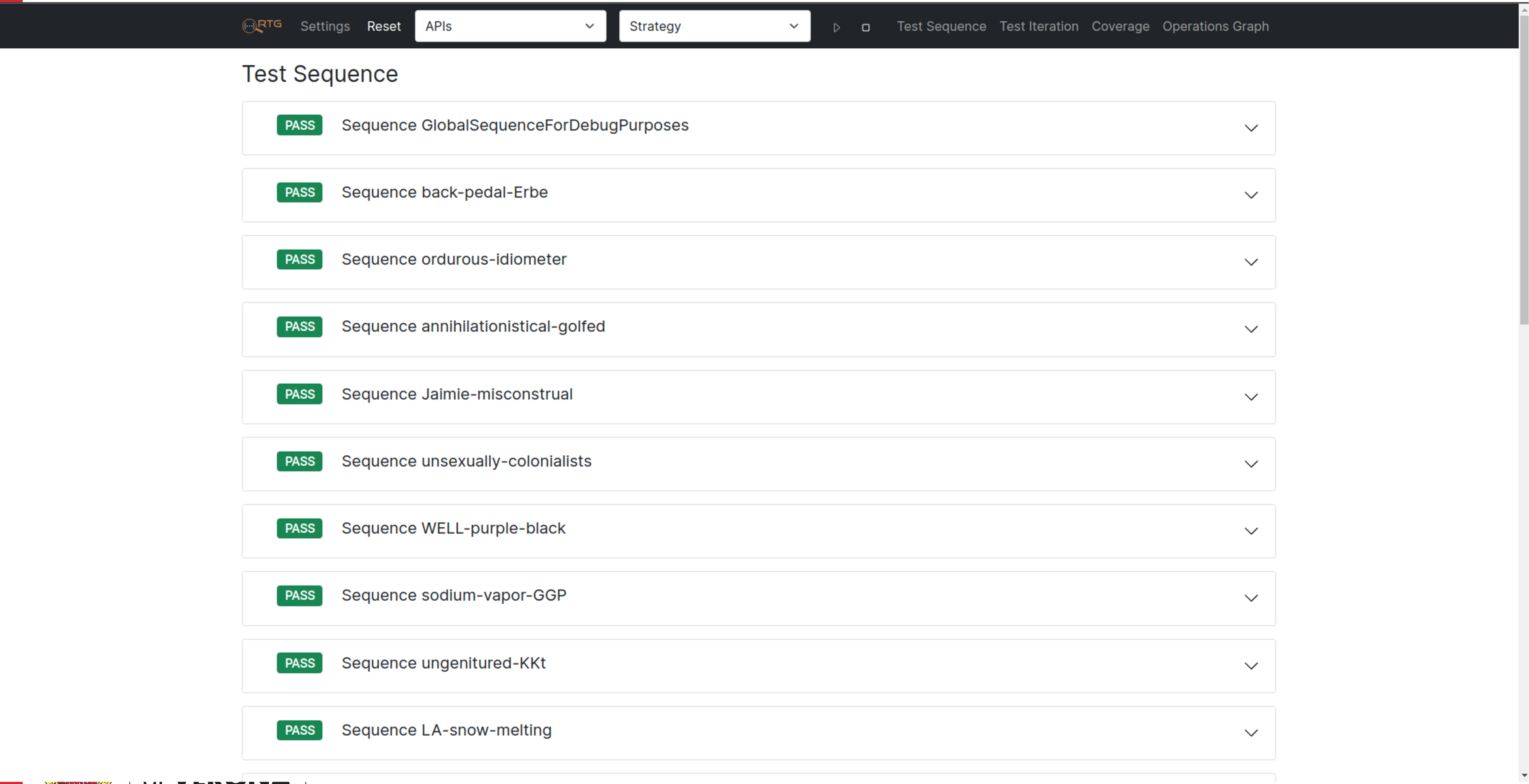
Resuts:



Operations Dependency Graph

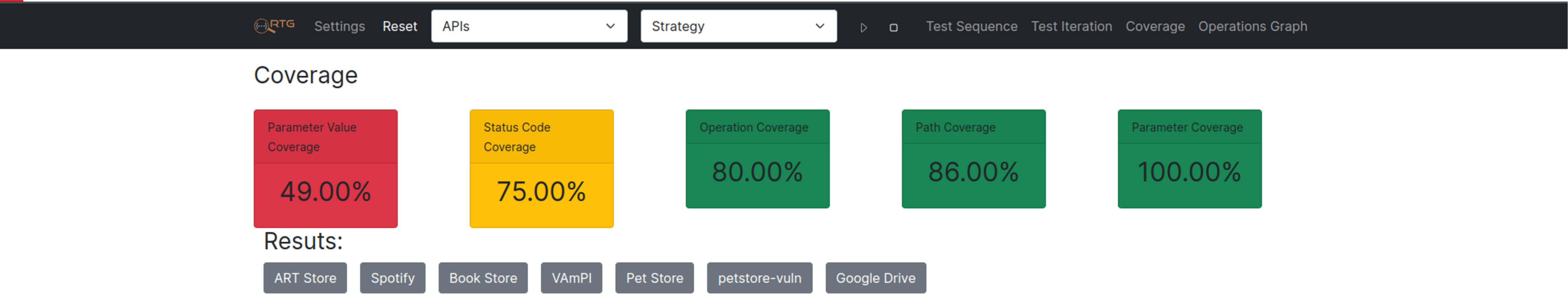


Number	Host	Method	Url	Status	Length	Port	Time
3	http://localhost/	GET	/users/v1/_debug	200	1978	5002	17:44:35
10	http://localhost/	POST	/users/v1/register	200	92	5002	17:44:35
15	http://localhost/	GET	/users/v1/_debug	200	2096	5002	17:44:35
20	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35
25	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35
30	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35
35	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35
40	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35
45	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35
50	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35
55	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35
60	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35
65	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35
70	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35
75	http://localhost/	PUT	/users/v1/X/email	401	47	5002	17:44:35



Test Sequence

- PASS Sequence GlobalSequenceForDebugPurposes
- PASS Sequence back-pedal-Erbe
- PASS Sequence ordurous-idiometer
- PASS Sequence annihilationistical-golfed
- PASS Sequence Jaimie-misconstrual
- PASS Sequence unsexually-colonialists
- PASS Sequence WELL-purple-black
- PASS Sequence sodium-vapor-GGP
- PASS Sequence ungenitured-KKt
- PASS Sequence LA-snow-melting



Coverage

Parameter Value Coverage
49.00%

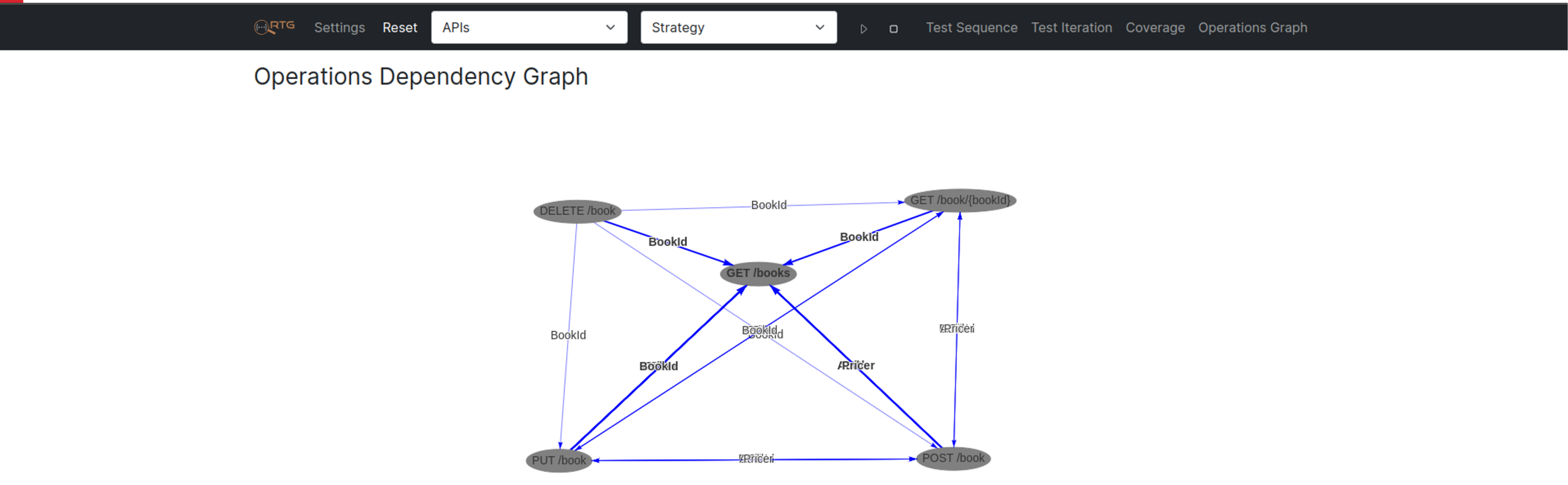
Status Code Coverage
75.00%

Operation Coverage
80.00%

Path Coverage
86.00%

Parameter Coverage
100.00%

Resuts:


RTG
Settings
Reset
APIs
Strategy
Test Sequence
Test Iteration
Coverage
Operations Graph
Operations Dependency Graph
DELETE /book
GET /book/{bookId}
GET /books
PUT /book
POST /book
BookId
BookId
BookId
BookId
BookId
BookId
BookId
Price
Price
Price