# Tukano in Kubernetes

Second Project

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Pedro Cavaleiro 57974

José Romano 59241

Cloud Computing Systems

2024/2025

# 1. Introduction

## What we need to do

In this second phase of the Cloud Computing project, the main objective is to adapt and implement the solution developed in the first phase, using the containerization technologies provided by Docker and Kubernetes orchestration. The proposal consists of migrating the services initially based on Azure PaaS to an approach based on Azure IaaS, ensuring that the functionalities implemented in the first phase are preserved.

More specifically, we need to carry out the following mandatory tasks:

- Deploy the application server in Azure Kubernetes Service (AKS);
- Implement a SQL database (PostgreSQL) in AKS, or opt for a NoSQL alternative such as MongoDB;
- Replace Azure Blob Storage with persistent volumes;
- Implement user session authentication for the Blob service;
- Test the deployment using artillery.

In addition, as part of the optional tasks, we are aiming to:

- Configure a caching service (Redis) in AKS, optimizing application performance;

## Our approach

To meet the proposed objectives, we decided to structure the solution in two main services:

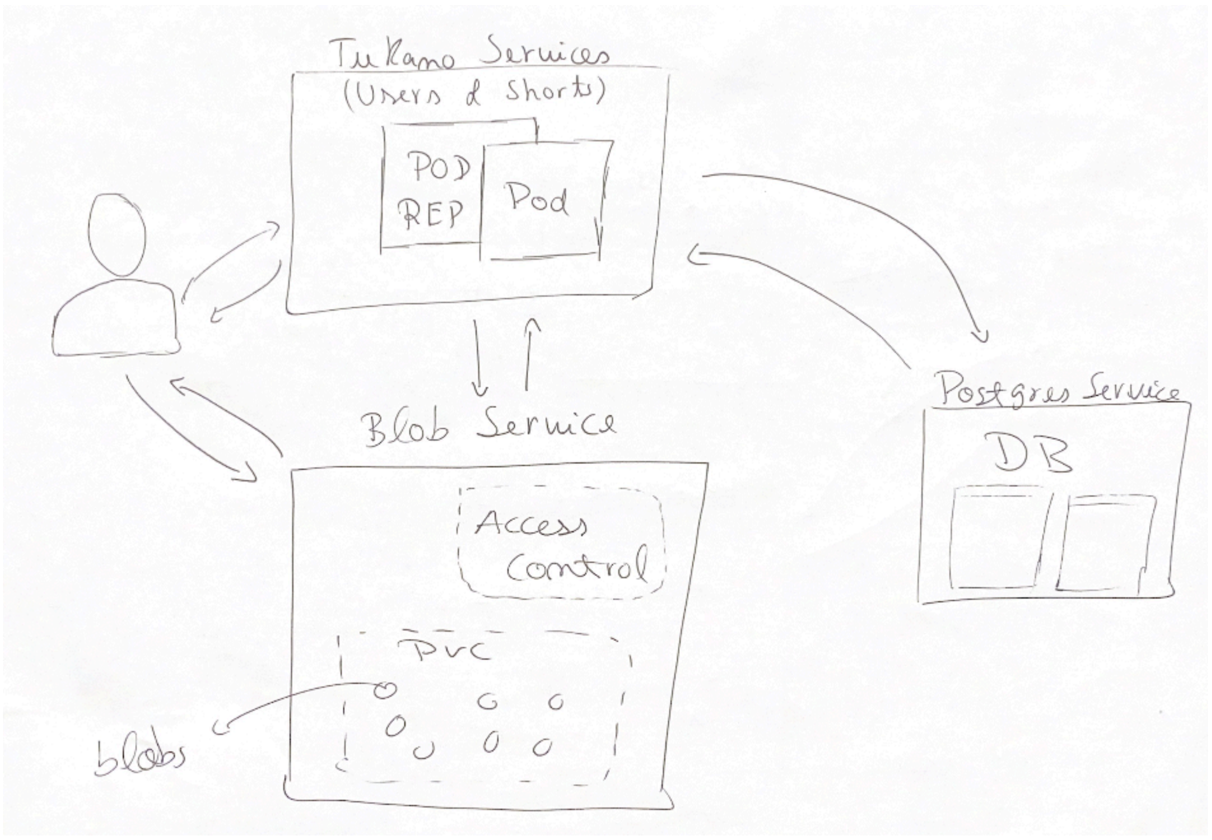
Service for Blobs: Focused on storing and managing multimedia files (such as videos), which represent the most load-intensive component in an application of this type.
Service for Users and Shorts: Responsible for managing users and the metadata associated with multimedia files.

This division was designed to optimize scalability and resource allocation. By separating the Blobs into an independent service, we are able to isolate the most demanding component, ensuring that the scalability of this service does not impact the others. Thus, in scenarios of increased load, we can scale only the resources needed for the Blobs service, avoiding wasting resources on less accessed components.

Another motivation for this approach is related to the nature of the data. While Users and Shorts deal with structured data and metadata, Blobs deal directly with multimedia files. This distinction allows us to apply specific strategies for each type of data, maximizing efficiency and ensuring a modular architecture.

With this strategy, we aim not only to meet the mandatory requirements but also to integrate the optional tasks that contribute most to the system's robustness and performance.

# 2. Architecture of our solution



## 2.1 Postgres runs on a different Service

In our project's architecture, the Postgres service was implemented as an isolated container within the Kubernetes cluster.

The Postgres service is exclusively responsible for managing relational data, supporting the operations carried out by the users and shorts components (from the Tukano service). These interact directly with Postgres to store and retrieve structured information, such as user and shorts data.

The limited interaction with Postgres reflects the division of responsibilities in the application: while the Postgres service handles structured data, unstructured data (blobs) is managed by an independent Persistent Volume Container. This approach allows the architecture to be flexible and easily scaled horizontally, ensuring that components can be updated or replaced without affecting the overall integrity of the system.

In addition, the isolation of the Postgres service in its own container promotes greater security. The physical and logical separation makes unauthorized access difficult, while computing resources can be allocated more efficiently, ensuring stable performance even under high load.

## 2.2 Tukano Services

Tukano Services is where we have assigned Users and Shorts. When necessary, these communicate with the Postgres Service to access the data in the table.

Within Tukano Services, we have two Pods, one of which acts as a replica in the event of a failure. Given that we haven't implemented a load balancer, this would be a solution that could mitigate some problems in the event of failures (but not as many as the load balancer).

## 2.3 Blobs Service

The Blob Service is the component dedicated to managing unstructured data, such as files, images and other binary content. It plays an essential role in the project architecture, providing efficient and secure storage for data that does not fit into relational models.

### 2.3.1 Access Control

To guarantee data security and integrity, Blob Service implements an Access Control system. This mechanism requires the user to be authenticated, logging in before performing any operations on the service. This prevents unauthorized access and ensures that only legitimate users can interact with the stored data.

Authentication reinforces the system's alignment with good security practices, protecting sensitive information and reducing the risk of attacks or misuse of the service.

### 2.3.2 Persistent Volume Container

Unstructured data managed by the Blob Service is stored using a Persistent Volume Claim (PVC), which is integrated into the service. The PVC provides an interface to physical storage and is mounted on the Pod running the Blob Service, allowing direct access to data while maintaining persistence regardless of the Pod's lifecycle.

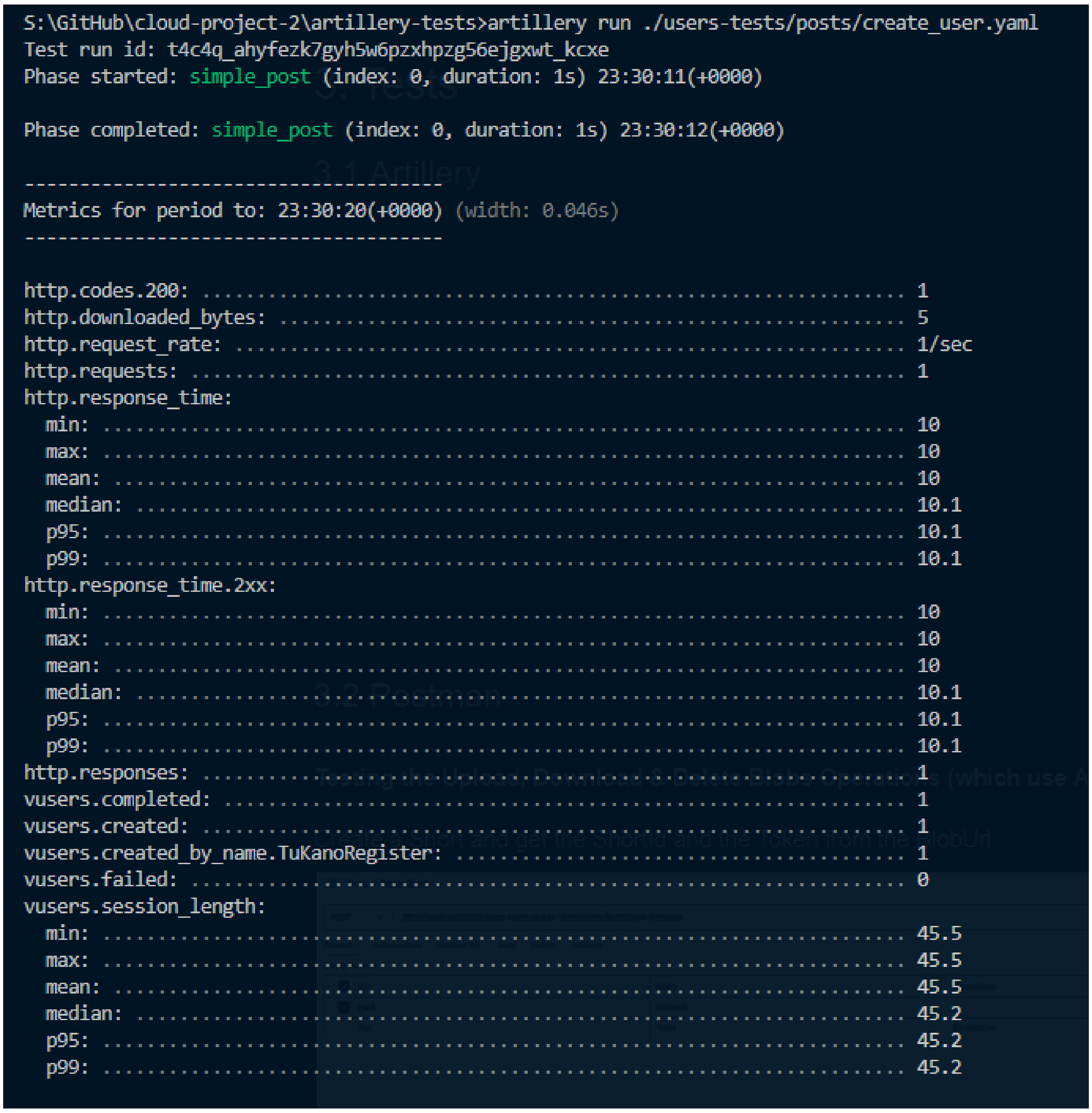
This approach allows the system to effectively manage a growing volume of unstructured data, ensuring that it remains available and intact even in the event of Kubernetes cluster failures. The use of PVC also facilitates service scalability, allowing storage expansion without interruptions to the system's operation.

# 3. Tests

## 3.1 Artillery

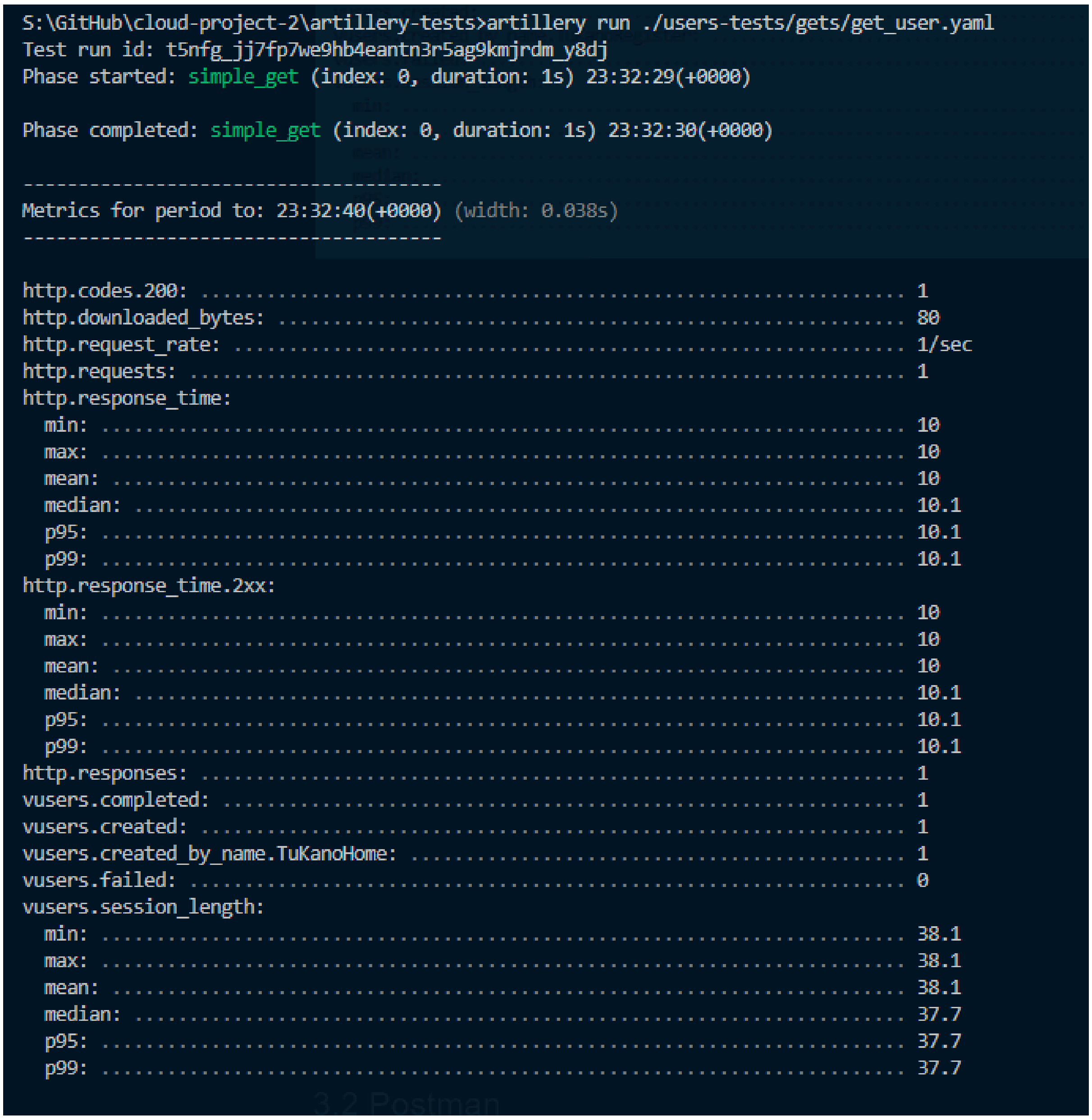
Although the tests are not finished, we have most of the operations for the users and shorts working correctly (the blobs operations were cast aside due to the cookie that we weren't able to capture in the .yaml artillery test files).
Although many of the Artillery tests that involved shortId's would work by acquiring the shortId through Postman but we decided not to include them in this list, other than that the remaining tests work.

Create User



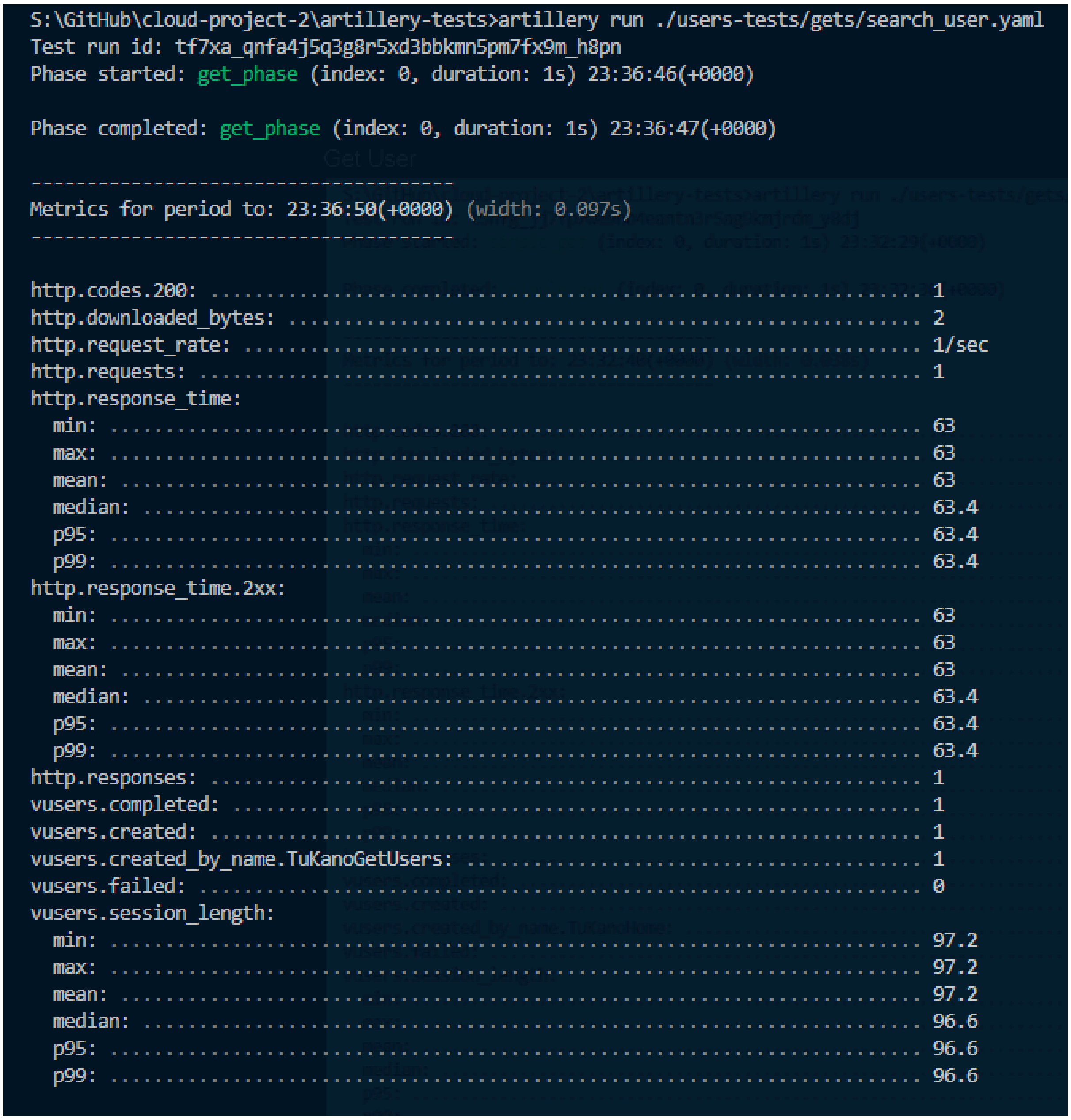
```
S:\GitHub\cloud-project-2\artillery-tests>artillery run ./users-tests/posts/create_user.yaml
Test run id: t4c4q_ahyfezk7gyh5w6pzxhpzg56ejgxwt_kcxe
Phase started: simple_post (index: 0, duration: 1s) 23:30:11(+0000)

Phase completed: simple_post (index: 0, duration: 1s) 23:30:12(+0000)

--------------------------------------
Metrics for period to: 23:30:20(+0000) (width: 0.046s)
--------------------------------------

http.codes.200: ................................................................ 1
http.downloaded_bytes: ......................................................... 5
http.request_rate: ............................................................. 1/sec
http.requests: ................................................................. 1
http.response_time:
  min: ......................................................................... 10
  max: ......................................................................... 10
  mean: ........................................................................ 10
  median: ...................................................................... 10.1
  p95: ......................................................................... 10.1
  p99: ......................................................................... 10.1
http.response_time.2xx:
  min: ......................................................................... 10
  max: ......................................................................... 10
  mean: ........................................................................ 10
  median: ...................................................................... 10.1
  p95: ......................................................................... 10.1
  p99: ......................................................................... 10.1
http.responses: ................................................................ 1
vusers.completed: .............................................................. 1
vusers.created: ................................................................ 1
vusers.created_by_name.TuKanoRegister: ......................................... 1
vusers.failed: ................................................................. 0
vusers.session_length:
  min: ......................................................................... 45.5
  max: ......................................................................... 45.5
  mean: ........................................................................ 45.5
  median: ...................................................................... 45.2
  p95: ......................................................................... 45.2
  p99: ......................................................................... 45.2
```

Get User



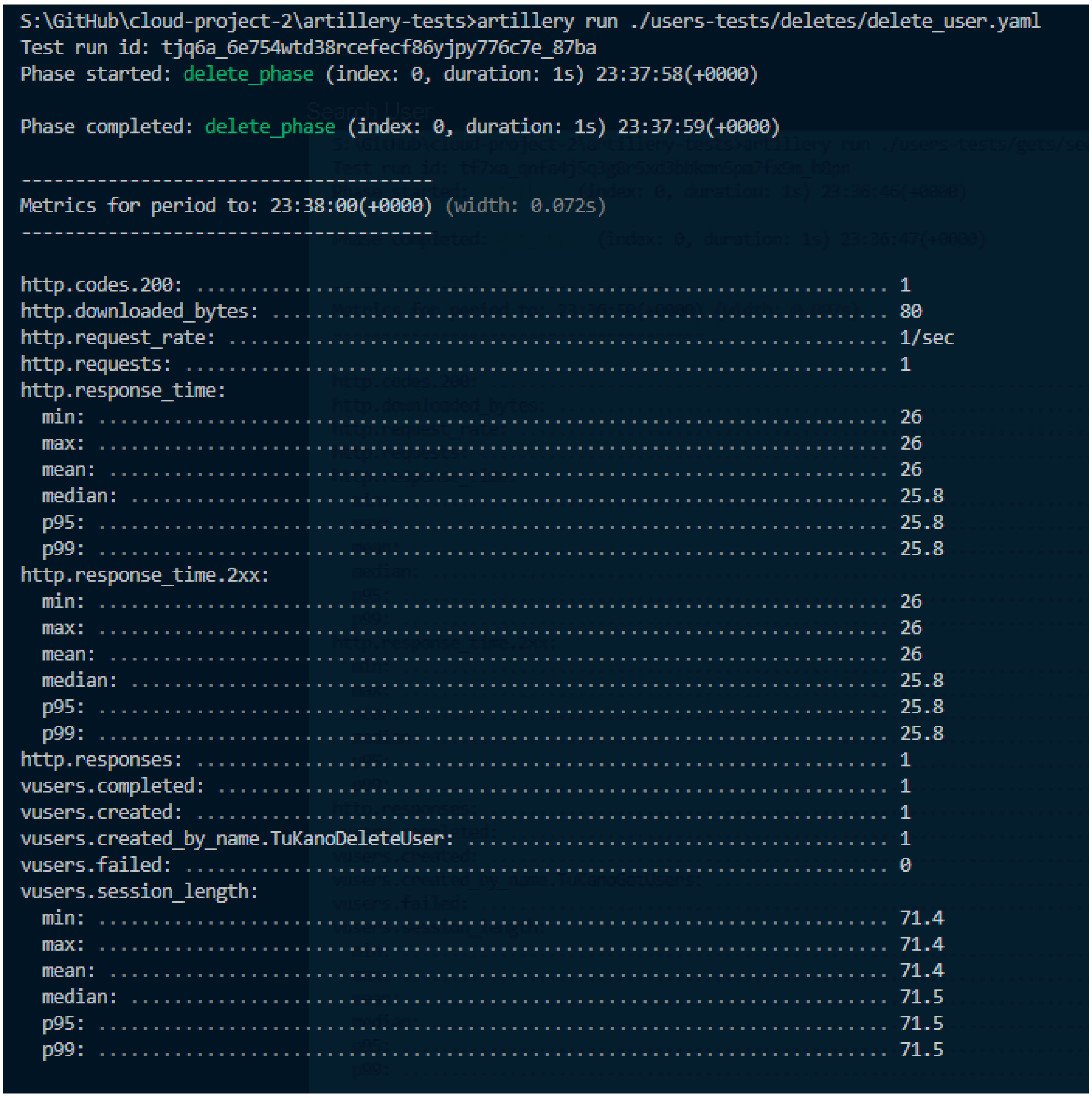
```
S:\GitHub\cloud-project-2\artillery-tests>artillery run ./users-tests/gets/get_user.yaml
Test run id: t5nfg_jj7fp7we9hb4eantn3r5ag9kmjrdm_y8dj
Phase started: simple_get (index: 0, duration: 1s) 23:32:29(+0000)

Phase completed: simple_get (index: 0, duration: 1s) 23:32:30(+0000)

--------------------------------------
Metrics for period to: 23:32:40(+0000) (width: 0.038s)
--------------------------------------

http.codes.200: ................................................................ 1
http.downloaded_bytes: ......................................................... 80
http.request_rate: ............................................................. 1/sec
http.requests: ................................................................. 1
http.response_time:
  min: ......................................................................... 10
  max: ......................................................................... 10
  mean: ........................................................................ 10
  median: ...................................................................... 10.1
  p95: ......................................................................... 10.1
  p99: ......................................................................... 10.1
http.response_time.2xx:
  min: ......................................................................... 10
  max: ......................................................................... 10
  mean: ........................................................................ 10
  median: ...................................................................... 10.1
  p95: ......................................................................... 10.1
  p99: ......................................................................... 10.1
http.responses: ................................................................ 1
vusers.completed: .............................................................. 1
vusers.created: ................................................................ 1
vusers.created_by_name.TuKanoHome: ............................................. 1
vusers.failed: ................................................................. 0
vusers.session_length:
  min: ......................................................................... 38.1
  max: ......................................................................... 38.1
  mean: ........................................................................ 38.1
  median: ...................................................................... 37.7
  p95: ......................................................................... 37.7
  p99: ......................................................................... 37.7
```

Search User



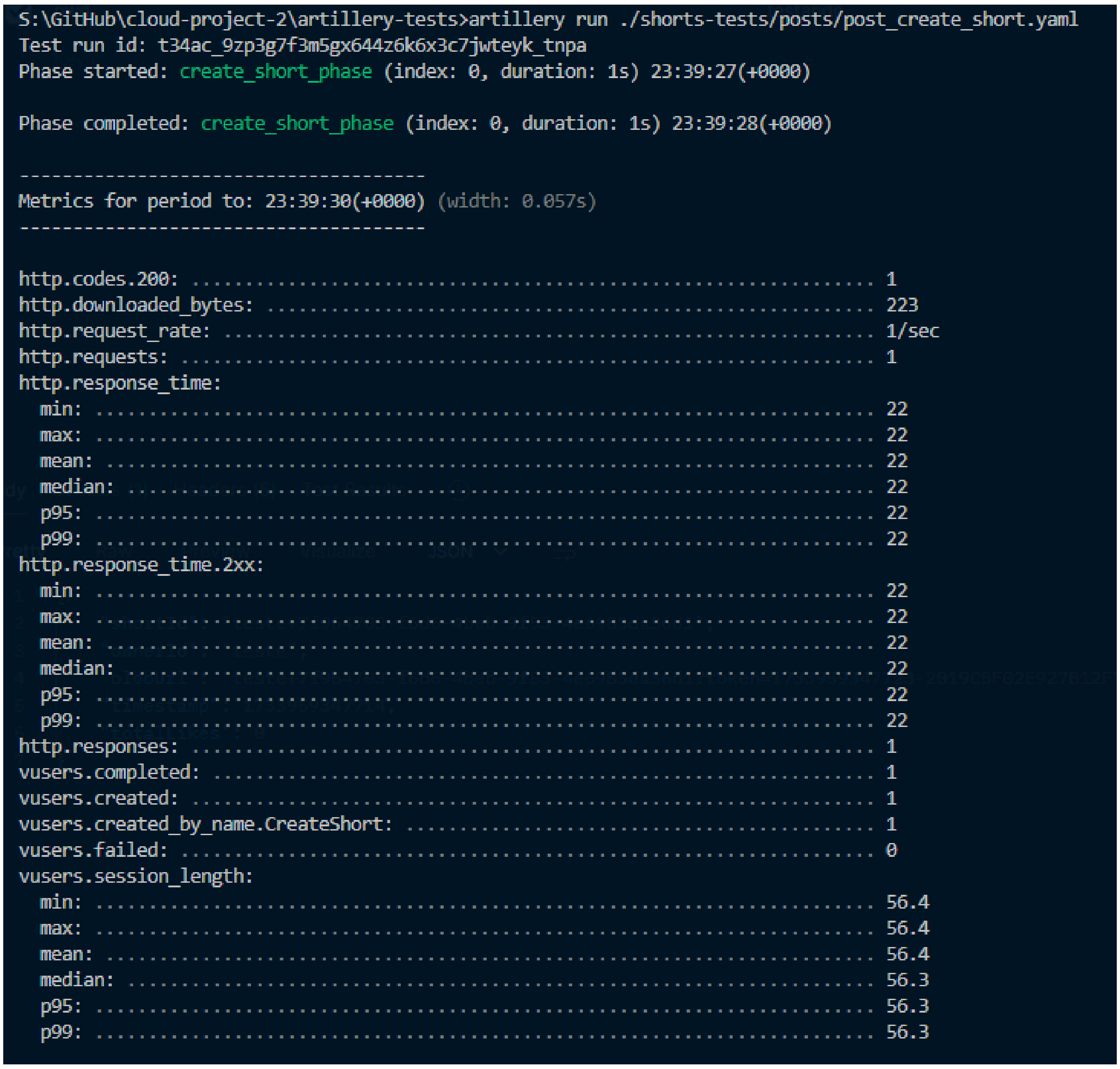
```
S:\GitHub\cloud-project-2\artillery-tests>artillery run ./users-tests/gets/search_user.yaml
Test run id: tf7xa_qnfa4j5q3g8r5xd3bbkmn5pm7fx9m_h8pn
Phase started: get_phase (index: 0, duration: 1s) 23:36:46(+0000)

Phase completed: get_phase (index: 0, duration: 1s) 23:36:47(+0000)

--------------------------------------
Metrics for period to: 23:36:50(+0000) (width: 0.097s)
--------------------------------------

http.codes.200: ........................................................ 1
http.downloaded_bytes: ................................................. 2
http.request_rate: ..................................................... 1/sec
http.requests: ......................................................... 1
http.response_time:
  min: ................................................................. 63
  max: ................................................................. 63
  mean: ................................................................ 63
  median: .............................................................. 63.4
  p95: ................................................................. 63.4
  p99: ................................................................. 63.4
http.response_time.2xx:
  min: ................................................................. 63
  max: ................................................................. 63
  mean: ................................................................ 63
  median: .............................................................. 63.4
  p95: ................................................................. 63.4
  p99: ................................................................. 63.4
http.responses: ........................................................ 1
vusers.completed: ...................................................... 1
vusers.created: ........................................................ 1
vusers.created_by_name.TuKanoGetUsers: ................................. 1
vusers.failed: ......................................................... 0
vusers.session_length:
  min: ................................................................. 97.2
  max: ................................................................. 97.2
  mean: ................................................................ 97.2
  median: .............................................................. 96.6
  p95: ................................................................. 96.6
  p99: ................................................................. 96.6
```

Delete User



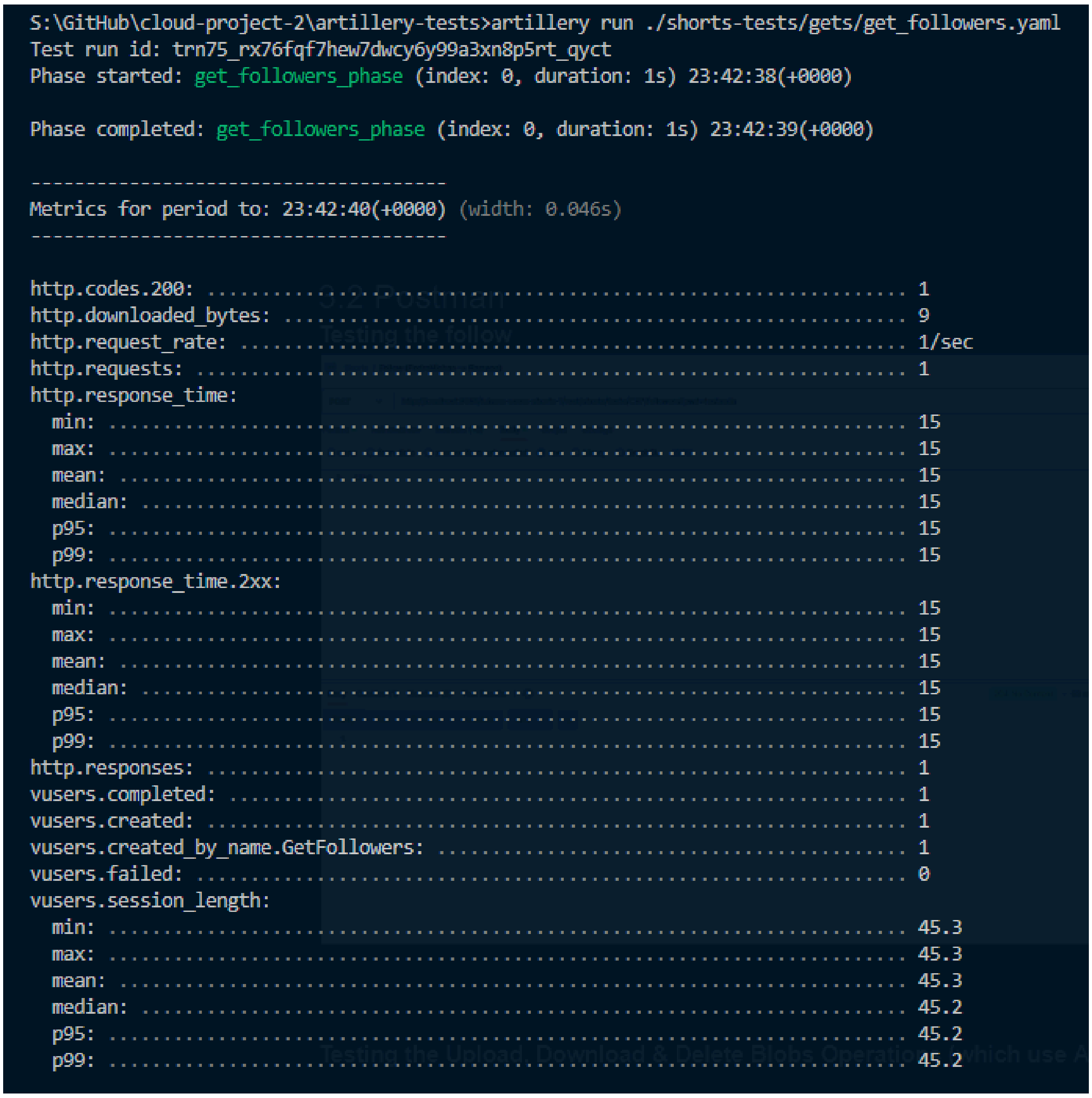
```
S:\GitHub\cloud-project-2\artillery-tests>artillery run ./users-tests/deletes/delete_user.yaml
Test run id: tjq6a_6e754wtd38rcefecf86yjpy776c7e_87ba
Phase started: delete_phase (index: 0, duration: 1s) 23:37:58(+0000)

Phase completed: delete_phase (index: 0, duration: 1s) 23:37:59(+0000)

--------------------------------------
Metrics for period to: 23:38:00(+0000) (width: 0.072s)
--------------------------------------

http.codes.200: ................................................................ 1
http.downloaded_bytes: ......................................................... 80
http.request_rate: ............................................................. 1/sec
http.requests: ................................................................. 1
http.response_time:
  min: ......................................................................... 26
  max: ......................................................................... 26
  mean: ........................................................................ 26
  median: ...................................................................... 25.8
  p95: ......................................................................... 25.8
  p99: ......................................................................... 25.8
http.response_time.2xx:
  min: ......................................................................... 26
  max: ......................................................................... 26
  mean: ........................................................................ 26
  median: ...................................................................... 25.8
  p95: ......................................................................... 25.8
  p99: ......................................................................... 25.8
http.responses: ................................................................ 1
vusers.completed: .............................................................. 1
vusers.created: ................................................................ 1
vusers.created_by_name.TuKanoDeleteUser: ....................................... 1
vusers.failed: ................................................................. 0
vusers.session_length:
  min: ......................................................................... 71.4
  max: ......................................................................... 71.4
  mean: ........................................................................ 71.4
  median: ...................................................................... 71.5
  p95: ......................................................................... 71.5
  p99: ......................................................................... 71.5
```

Create Short



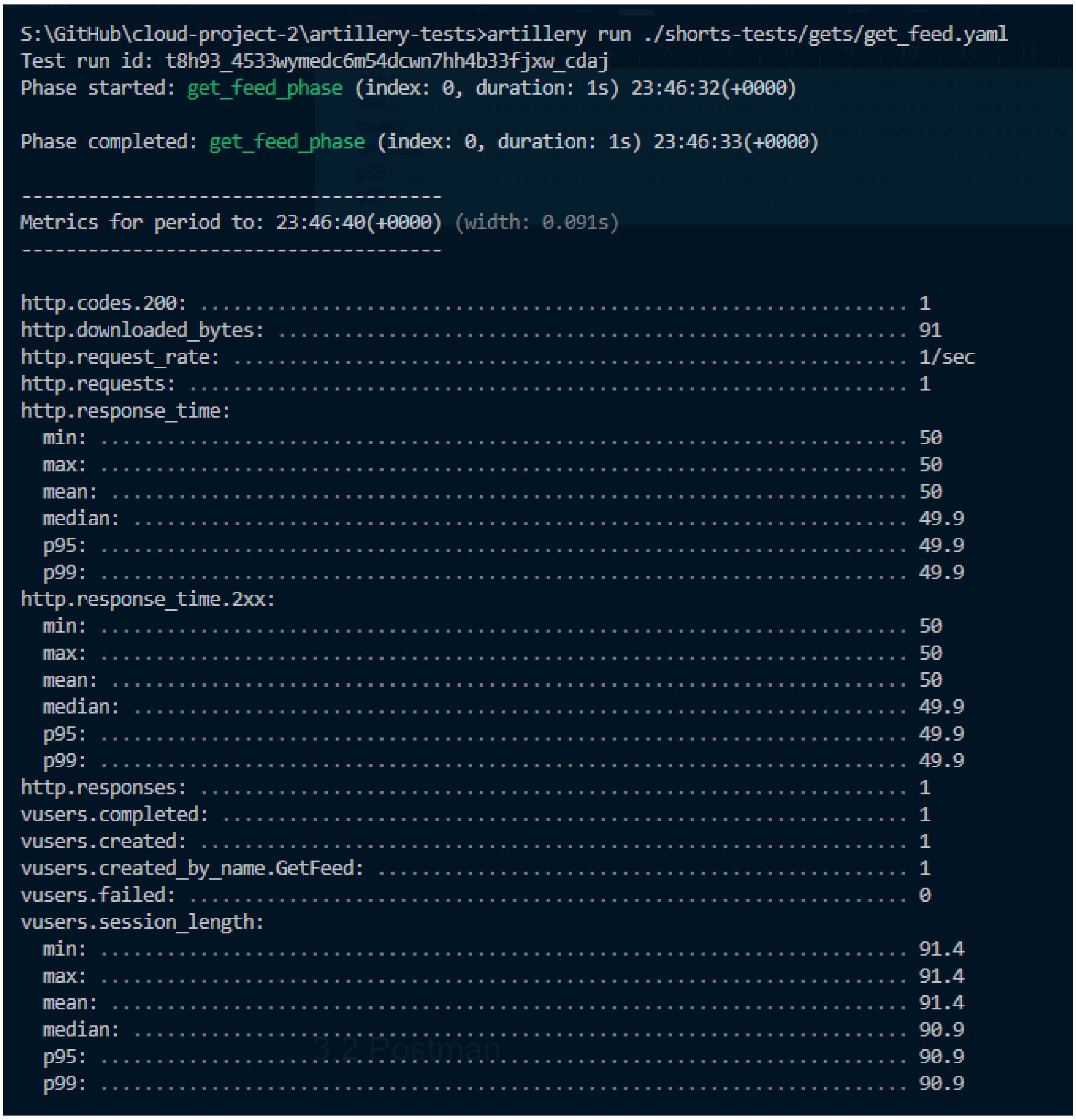
```
S:\GitHub\cloud-project-2\artillery-tests>artillery run ./shorts-tests/posts/post_create_short.yaml
Test run id: t34ac_9zp3g7f3m5gx644z6k6x3c7jwteyk_tnpa
Phase started: create_short_phase (index: 0, duration: 1s) 23:39:27(+0000)

Phase completed: create_short_phase (index: 0, duration: 1s) 23:39:28(+0000)

--------------------------------------
Metrics for period to: 23:39:30(+0000) (width: 0.057s)
--------------------------------------

http.codes.200: ................................................................ 1
http.downloaded_bytes: ......................................................... 223
http.request_rate: ............................................................. 1/sec
http.requests: ................................................................. 1
http.response_time:
  min: ......................................................................... 22
  max: ......................................................................... 22
  mean: ........................................................................ 22
  median: ...................................................................... 22
  p95: ......................................................................... 22
  p99: ......................................................................... 22
http.response_time.2xx:
  min: ......................................................................... 22
  max: ......................................................................... 22
  mean: ........................................................................ 22
  median: ...................................................................... 22
  p95: ......................................................................... 22
  p99: ......................................................................... 22
http.responses: ................................................................ 1
vusers.completed: .............................................................. 1
vusers.created: ................................................................ 1
vusers.created_by_name.CreateShort: ............................................ 1
vusers.failed: ................................................................. 0
vusers.session_length:
  min: ......................................................................... 56.4
  max: ......................................................................... 56.4
  mean: ........................................................................ 56.4
  median: ...................................................................... 56.3
  p95: ......................................................................... 56.3
  p99: ......................................................................... 56.3
```

Get Followers



```
S:\GitHub\cloud-project-2\artillery-tests>artillery run ./shorts-tests/gets/get_followers.yaml
Test run id: trn75_rx76fqf7hew7dwcy6y99a3xn8p5rt_qyct
Phase started: get_followers_phase (index: 0, duration: 1s) 23:42:38(+0000)

Phase completed: get_followers_phase (index: 0, duration: 1s) 23:42:39(+0000)

--------------------------------------
Metrics for period to: 23:42:40(+0000) (width: 0.046s)
--------------------------------------

http.codes.200: ................................................................ 1
http.downloaded_bytes: ......................................................... 9
http.request_rate: ............................................................. 1/sec
http.requests: ................................................................. 1
http.response_time:
  min: ......................................................................... 15
  max: ......................................................................... 15
  mean: ........................................................................ 15
  median: ...................................................................... 15
  p95: ......................................................................... 15
  p99: ......................................................................... 15
http.response_time.2xx:
  min: ......................................................................... 15
  max: ......................................................................... 15
  mean: ........................................................................ 15
  median: ...................................................................... 15
  p95: ......................................................................... 15
  p99: ......................................................................... 15
http.responses: ................................................................ 1
vusers.completed: .............................................................. 1
vusers.created: ................................................................ 1
vusers.created_by_name.GetFollowers: ........................................... 1
vusers.failed: ................................................................. 0
vusers.session_length:
  min: ......................................................................... 45.3
  max: ......................................................................... 45.3
  mean: ........................................................................ 45.3
  median: ...................................................................... 45.2
  p95: ......................................................................... 45.2
  p99: ......................................................................... 45.2
```

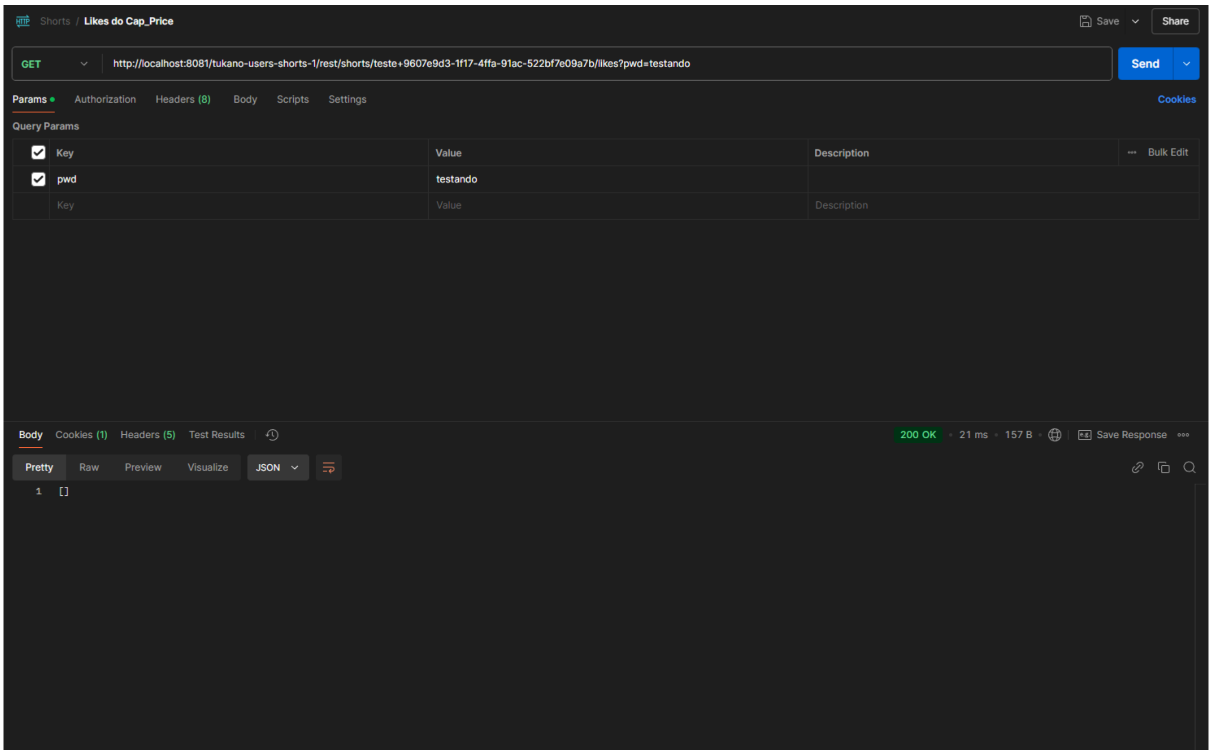
Get Feed



```
S:\GitHub\cloud-project-2\artillery-tests>artillery run ./shorts-tests/gets/get_feed.yaml
Test run id: t8h93_4533wymedc6m54dcwn7hh4b33fjxw_cdaj
Phase started: get_feed_phase (index: 0, duration: 1s) 23:46:32(+0000)

Phase completed: get_feed_phase (index: 0, duration: 1s) 23:46:33(+0000)

--------------------------------------
Metrics for period to: 23:46:40(+0000) (width: 0.091s)
--------------------------------------

http.codes.200: ................................................................ 1
http.downloaded_bytes: ......................................................... 91
http.request_rate: ............................................................. 1/sec
http.requests: ................................................................. 1
http.response_time:
  min: ......................................................................... 50
  max: ......................................................................... 50
  mean: ........................................................................ 50
  median: ...................................................................... 49.9
  p95: ......................................................................... 49.9
  p99: ......................................................................... 49.9
http.response_time.2xx:
  min: ......................................................................... 50
  max: ......................................................................... 50
  mean: ........................................................................ 50
  median: ...................................................................... 49.9
  p95: ......................................................................... 49.9
  p99: ......................................................................... 49.9
http.responses: ................................................................ 1
vusers.completed: .............................................................. 1
vusers.created: ................................................................ 1
vusers.created_by_name.GetFeed: ................................................ 1
vusers.failed: ................................................................. 0
vusers.session_length:
  min: ......................................................................... 91.4
  max: ......................................................................... 91.4
  mean: ........................................................................ 91.4
  median: ...................................................................... 90.9
  p95: ......................................................................... 90.9
  p99: ......................................................................... 90.9
```

## 3.2 Postman

### Testing the likes of a post owned by that same user



```
GET http://localhost:8081/tukano-users-shorts-1/rest/shorts/teste+9607e9d3-1f17-4ffa-91ac-522bf7e09a7b/likes?pwd=testando
```

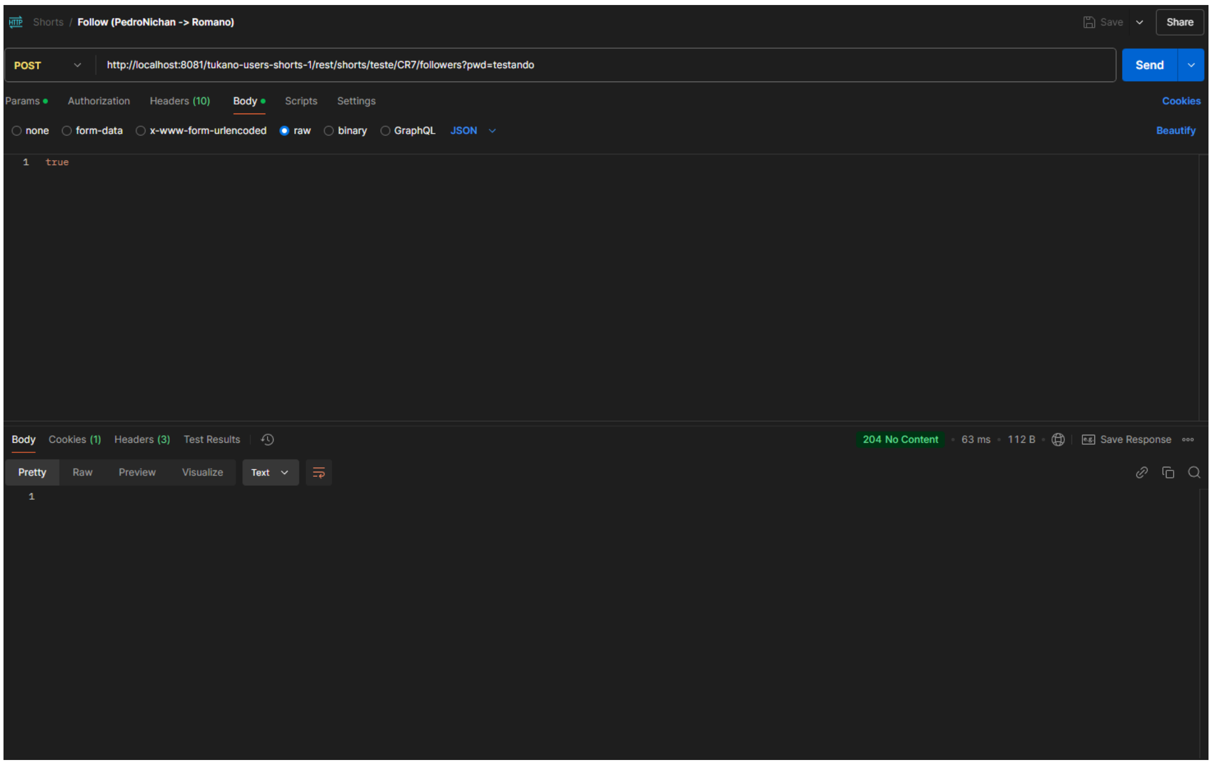
Query Params

| Key | Value | Description |
|---|---|---|
| ✓ pwd | testando | |

200 OK · 21 ms · 157 B

```
[]
```

## Testing the follow



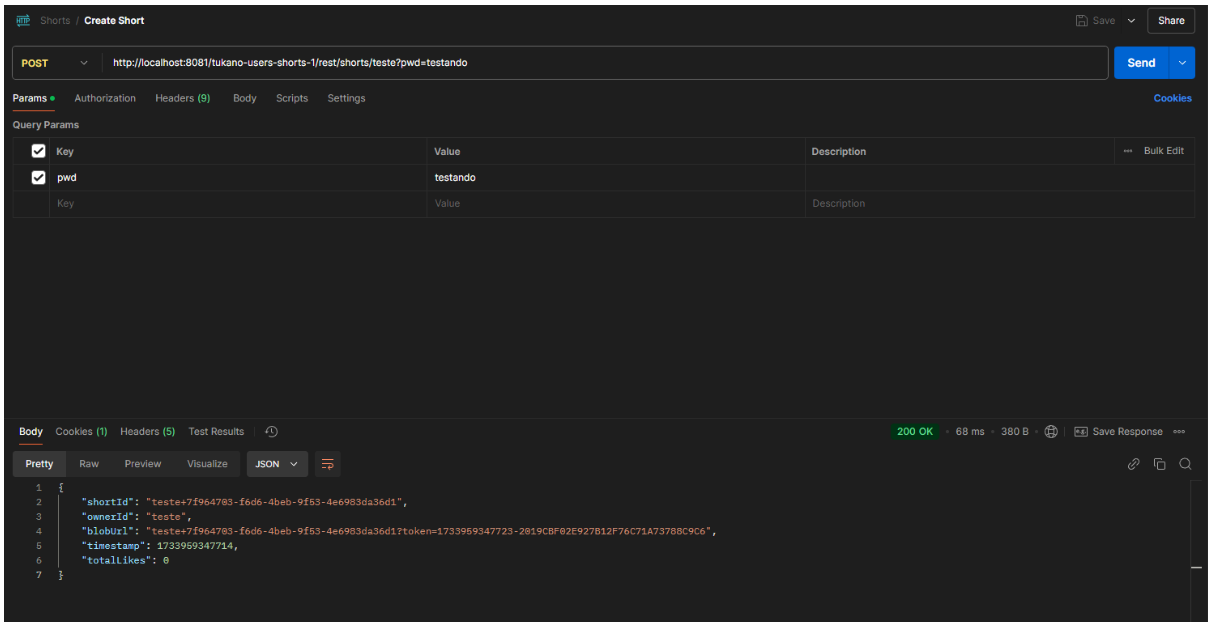
POST http://localhost:8081/tukano-users-shorts-1/rest/shorts/teste/CR7/followers?pwd=testando

```
true
```

204 No Content · 63 ms · 112 B

## Testing the Upload, Download & Delete Blobs Operations (which use Access Control):

Create a Short and get the ShortId and the Token from the BlobUrl



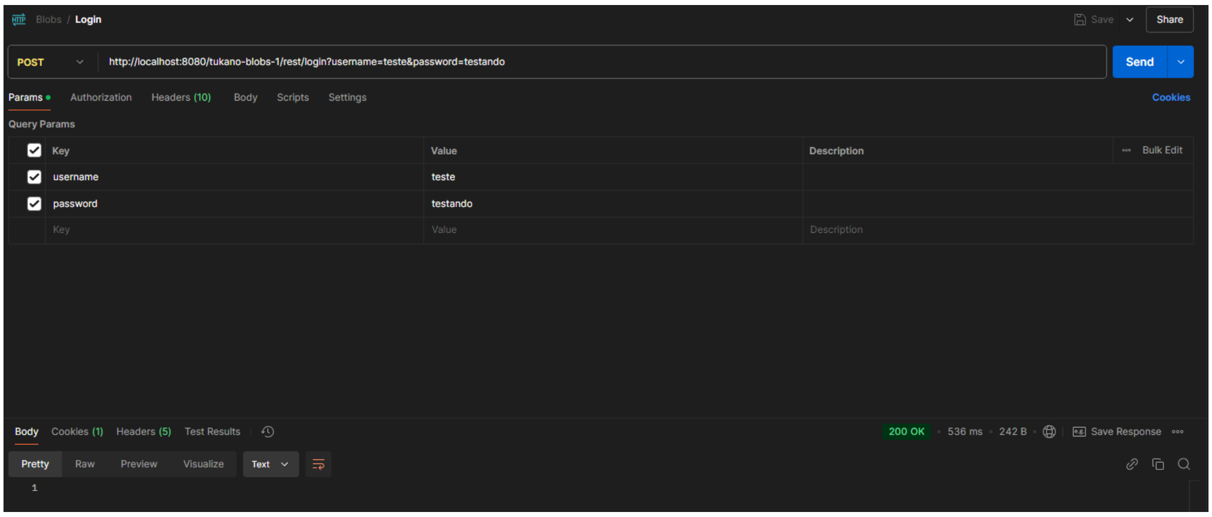
POST http://localhost:8081/tukano-users-shorts-1/rest/shorts/teste?pwd=testando

| | Key | Value | Description |
|---|---|---|---|
| ✓ | pwd | testando | |

200 OK · 68 ms · 380 B

```
{
    "shortId": "teste+7f964703-f6d6-4beb-9f53-4e6983da36d1",
    "ownerId": "teste",
    "blobUrl": "teste+7f964703-f6d6-4beb-9f53-4e6983da36d1?token=1733959347723-2019CBF02E927B12F76C71A73788C9C6",
    "timestamp": 1733959347714,
    "totalLikes": 0
}
```

Then we Login (because of the Access Control)


Blobs / Login

POST http://localhost:8080/tukano-blobs-1/rest/login?username=teste&password=testando

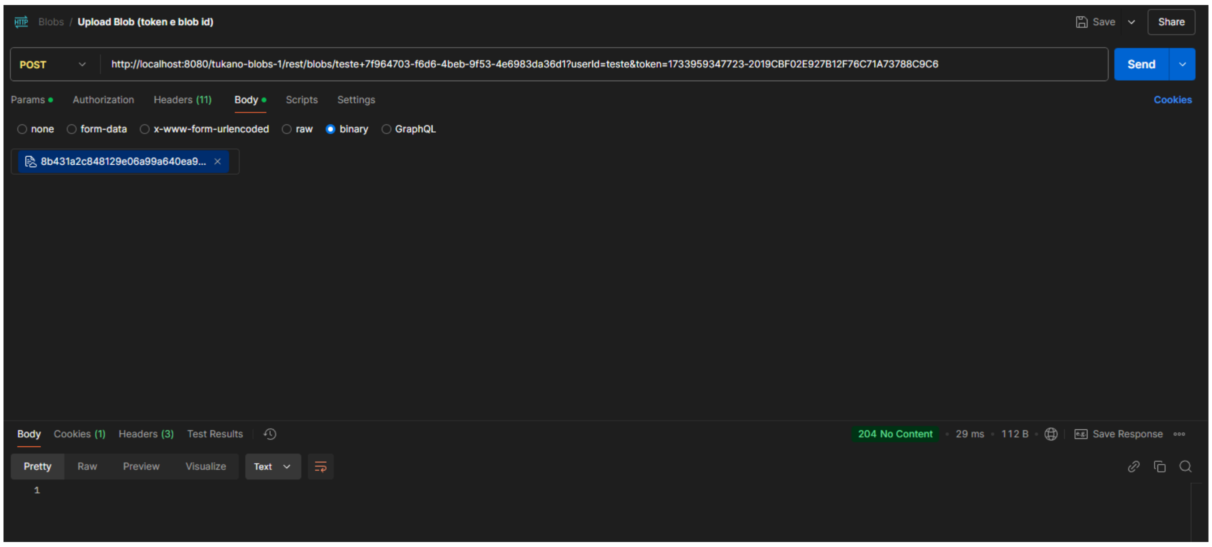
Query Params

| Key | Value | Description |
|---|---|---|
| ✓ username | teste | |
| ✓ password | testando | |

200 OK · 536 ms · 242 B

```
1
```

And now we can Upload it


Blobs / Upload Blob (token e blob id)

POST http://localhost:8080/tukano-blobs-1/rest/blobs/teste+7f964703-f6d6-4beb-9f53-4e6983da36d1?userId=teste&token=1733959347723-2019CBF02E927B12F76C71A73788C9C6

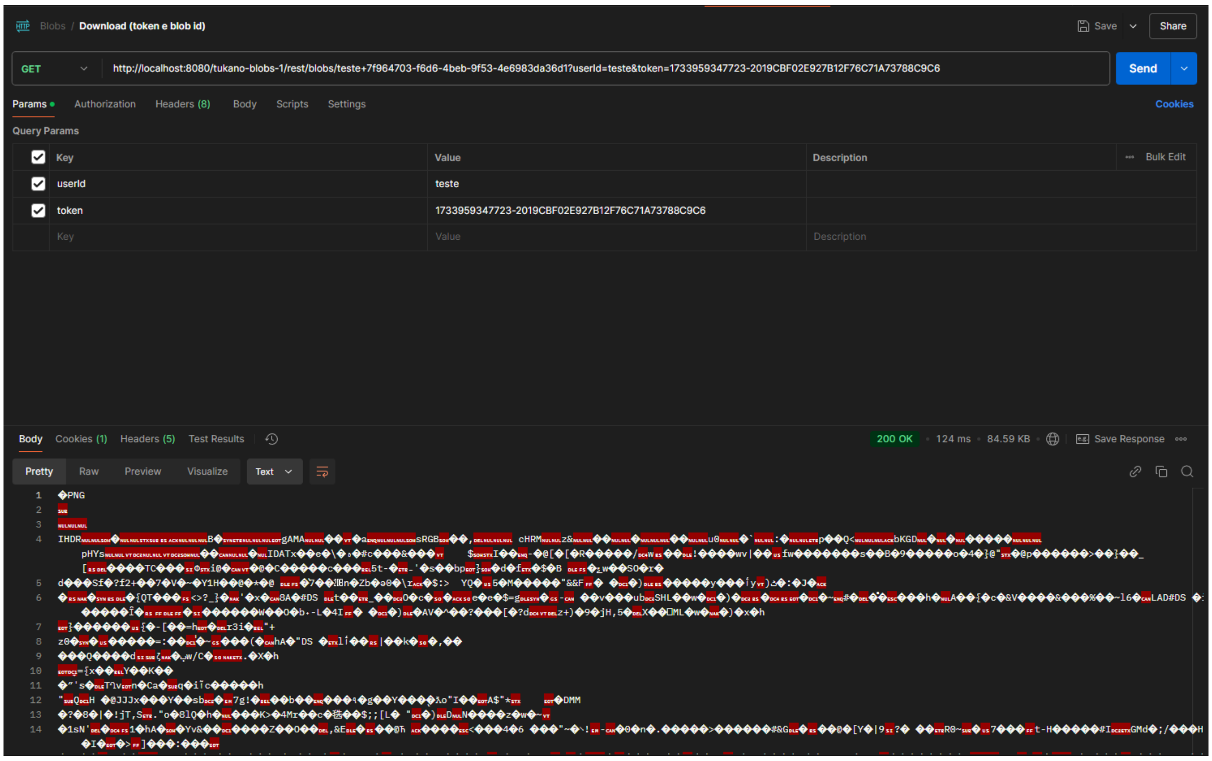
Body: ○ none ○ form-data ○ x-www-form-urlencoded ○ raw ● binary ○ GraphQL

8b431a2c848129e06a99a640ea9...

204 No Content · 29 ms · 112 B

```
1
```

Download also works



Blobs / Download (token e blob id)

```
GET http://localhost:8080/tukano-blobs-1/rest/blobs/teste+7f964703-f6d6-4beb-9f53-4e6983da36d1?userId=teste&token=1733959347723-2019CBF02E927B12F76C71A73788C9C6
```

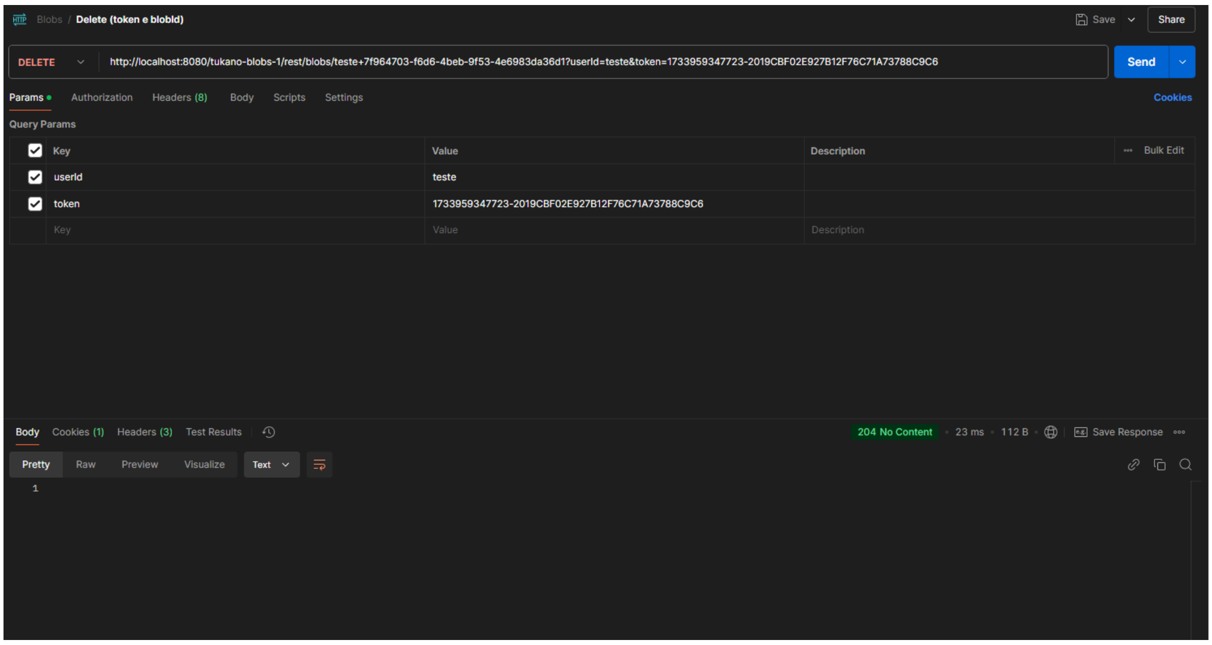
Query Params

| Key | Value | Description |
|---|---|---|
| userId | teste | |
| token | 1733959347723-2019CBF02E927B12F76C71A73788C9C6 | |

200 OK · 124 ms · 84.59 KB

```
�PNG
[illegible]
IHDR[illegible]sRGB[illegible]cHRM[illegible]
pHYs[illegible]IDATx[illegible]
[illegible]
```

Since we still have a valid token and access control cookie we can also perform a Delete Blob operation



Blobs / Delete (token e blobId)

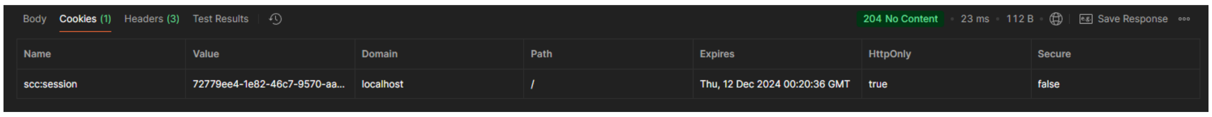
```
DELETE http://localhost:8080/tukano-blobs-1/rest/blobs/teste+7f964703-f6d6-4beb-9f53-4e6983da36d1?userId=teste&token=1733959347723-2019CBF02E927B12F76C71A73788C9C6
```

Query Params

| Key | Value | Description |
|---|---|---|
| userId | teste | |
| token | 1733959347723-2019CBF02E927B12F76C71A73788C9C6 | |

204 No Content · 23 ms · 112 B

Here’s a demonstration of the cookie working in the previous test



204 No Content · 23 ms · 112 B

| Name | Value | Domain | Path | Expires | HttpOnly | Secure |
|---|---|---|---|---|---|---|
| scc:session | 72779ee4-1e82-46c7-9570-aa... | localhost | / | Thu, 12 Dec 2024 00:20:36 GMT | true | false |

We have also thoroughly tested the other endpoints and they all worked, we just decided to not include more screenshots for the sake of making the report shorter.

# 4. Other notes

## 4.1 Implementation of Cache

We were going to implement it but we ended up not doing it.

## 4.2 Fixing deleteUser

We encountered some issues with the deleteUser command and we know for a fact that is something to do with the token, because if we comment the token argument we can run this command smoothly.

We could not solve it since we did not fully understand the issue in depth.