Flask application

Overload test report – Locust

Author: Piotr Piasta 75886

Test cases

Test No.	Users No.	Spawn ratio	Status
1	5	1	Passed
2	50	1	Passed
3	500	5	Passed
4	1000	10	Passed
5	5000	20	Warning

Wyniki testów obciążeniowych Locust pokazują, jak aplikacja docelowa działa pod różnymi obciążeniami. W tym przypadku wykonano 5 różnych scenariuszy, z różnymi liczbami użytkowników i szybkością generowania żądań.

W ogólnym ujęciu wyniki pokazują, że aplikacja może obsłużyć umiarkowane obciążenia bez problemów, z niskimi czasami odpowiedzi i bez błędów żądań. Jednak wraz ze wzrostem obciążenia czasy odpowiedzi również się wydłużają, a aplikacja zaczyna mieć trudności z obsługą żądań.

Case 1 – z 5 użytkownikami i szybkością generowania żądań wynoszącą 1, pokazuje, że aplikacja bez problemu może obsłużyć to obciążenie, z przeciętnym czasem odpowiedzi wynoszącym około 2 sekund i bez błędów żądań.

Case 2 - z 50 użytkownikami i szybkością generowania żądań wynoszącą 1, przeciętny czas odpowiedzi pozostaje w okolicach 2 sekund, ale liczba żądań na sekundę wzrasta do ponad 13, co jest ponad dziesięciokrotnie większe niż w przypadku 1.

Case 3 - z 500 użytkownikami i szybkością generowania żądań wynoszącą 5, aplikacja zaczyna mieć trudności przy tym obciążeniu, z przeciętnym czasem odpowiedzi wynoszącym ponad 2 sekundy i maksymalnym czasem odpowiedzi wynoszącym 3 sekundy dla niektórych żądań. Jednak liczba żądań na sekundę pozostaje wysoka - ponad 129.

Case 4 - z 1000 użytkownikami i szybkością generowania żądań wynoszącą 10, aplikacja jest wyraźnie przeciążona, z przeciętnym czasem odpowiedzi wynoszącym ponad 5 sekund i maksymalnym czasem odpowiedzi wynoszącym 22 sekundy dla niektórych żądań. Liczba żądań na sekundę również znacznie spada, do nieco ponad 100.

Case 5 – z 5000 użytkownikami i szybkością generowania żądań wynoszącą 20, aplikacja jest przeciążona i może nie być w stanie obsłużyć obciążenia, ponieważ połączenie zostaje odrzucone.

Te wyniki sugerują, że aplikacja może obsłużyć umiarkowane obciążenia bez problemów, ale zaczyna mieć trudności przy wzroście obciążenia. Aby poprawić wydajność pod dużymi obciążeniami, aplikacja może wymagać optymalizacji lub dodatkowych zasobów, takich jak więcej serwerów lub lepszy sprzęt.

Case 1 – 5 users / spawn 1

During: 9.04.2023, 11:48:00 - 9.04.2023, 11:48:18

Target Host: http://localhost:5000

Script: locustfile.py

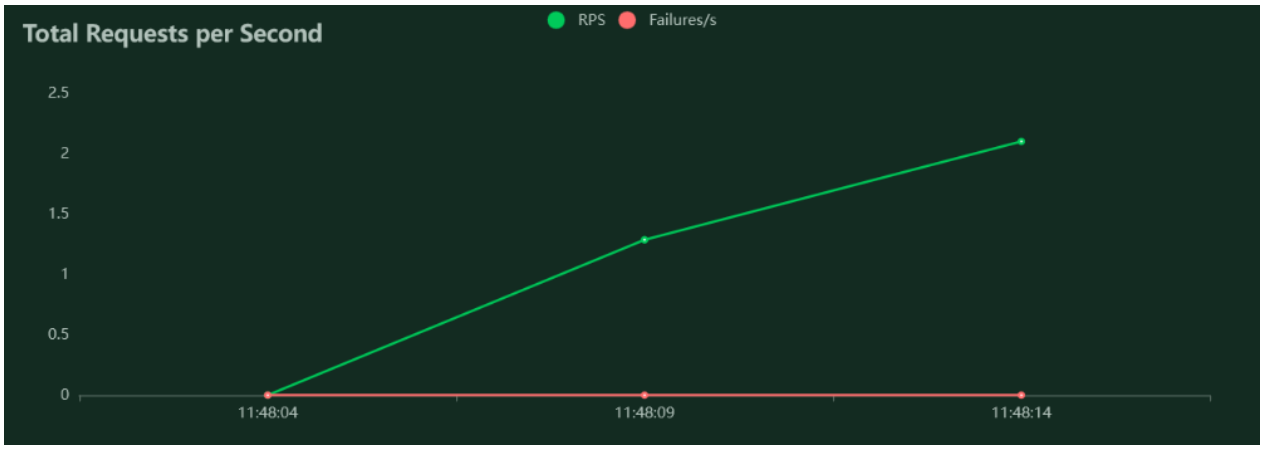
Request Statistics

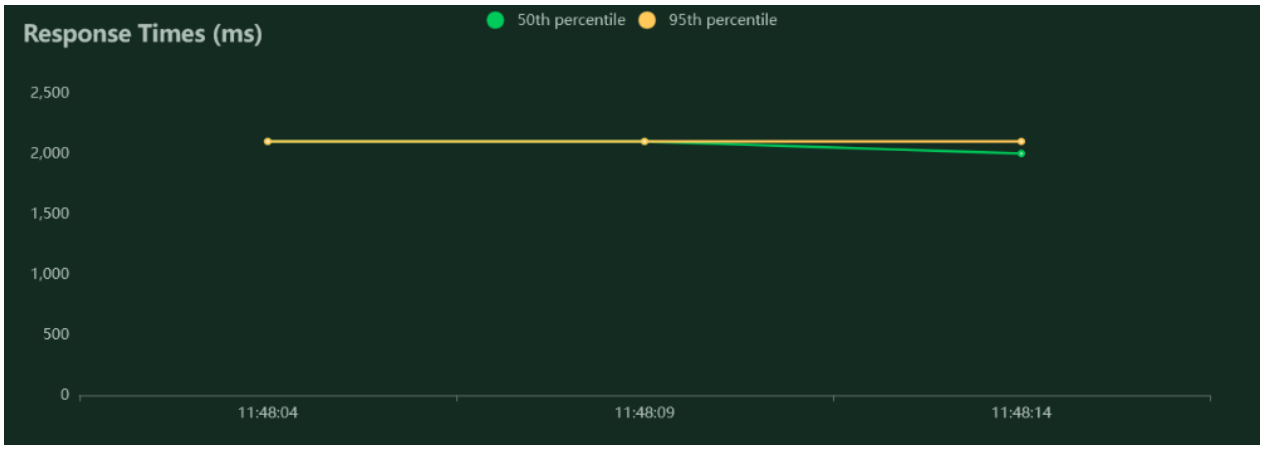
Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/	23	0	2050	2047	2059	12	1.2	0.0
POST	/hello	16	0	2051	2041	2072	319	0.9	0.0
	Aggregated	39	0	2051	2041	2072	137	2.1	0.0

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/	2000	2000	2100	2100	2100	2100	2100	2100
POST	/hello	2100	2100	2100	2100	2100	2100	2100	2100
	Aggregated	2000	2100	2100	2100	2100	2100	2100	2100

Charts





Final ratio

Ratio per User class

- 100.0% HelloWorldUser
  - 50.0% hello
  - 50.0% hello_name

Total ratio

- 100.0% HelloWorldUser
  - 50.0% hello
  - 50.0% hello_name

Case 2 – 50 users / spawn 1

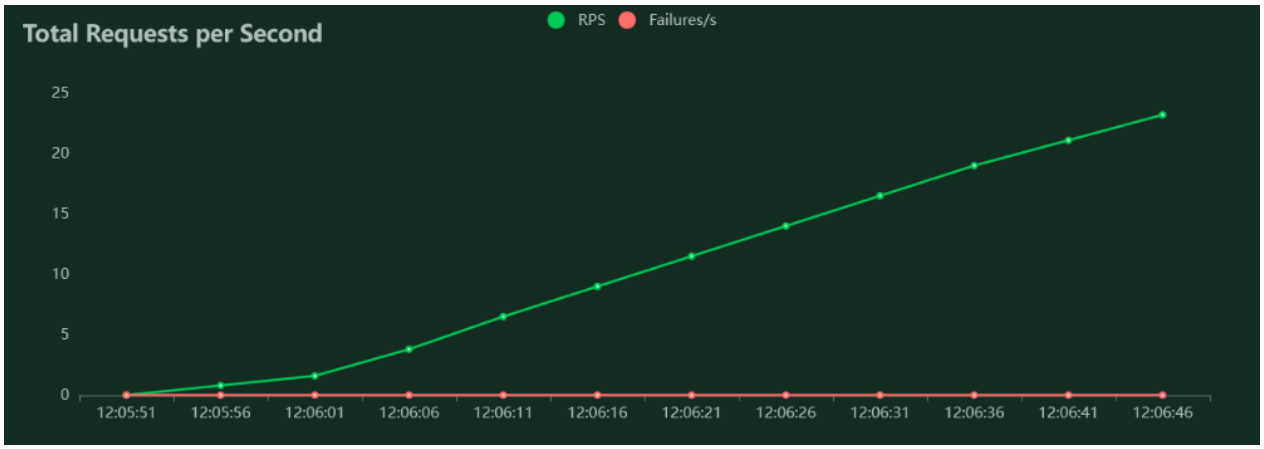
Request Statistics

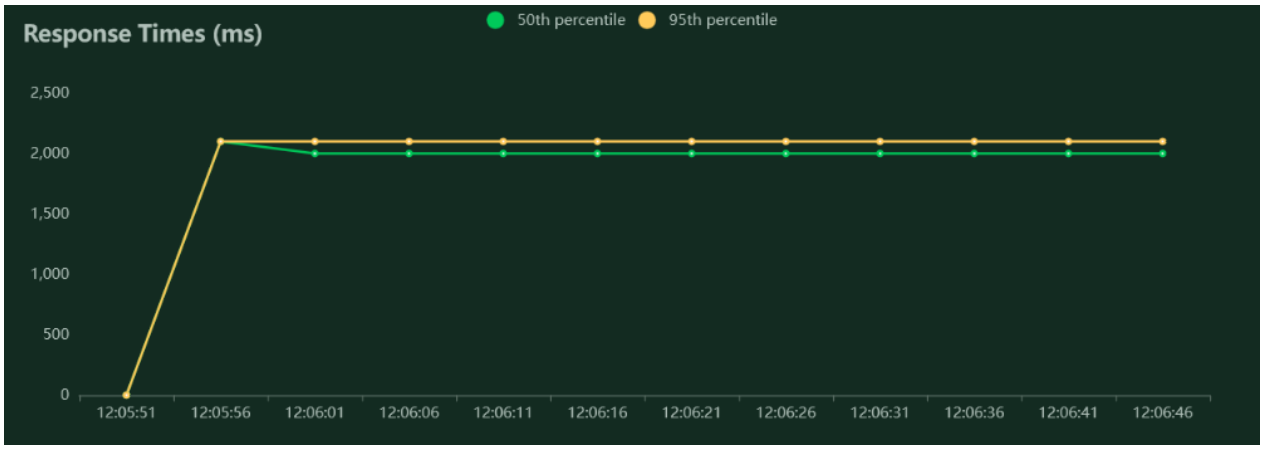
Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/	395	0	2047	2019	2069	12	6.8	0.0
POST	/hello	387	0	2048	2022	2080	319	6.7	0.0
	Aggregated	782	0	2048	2019	2080	163	13.5	0.0

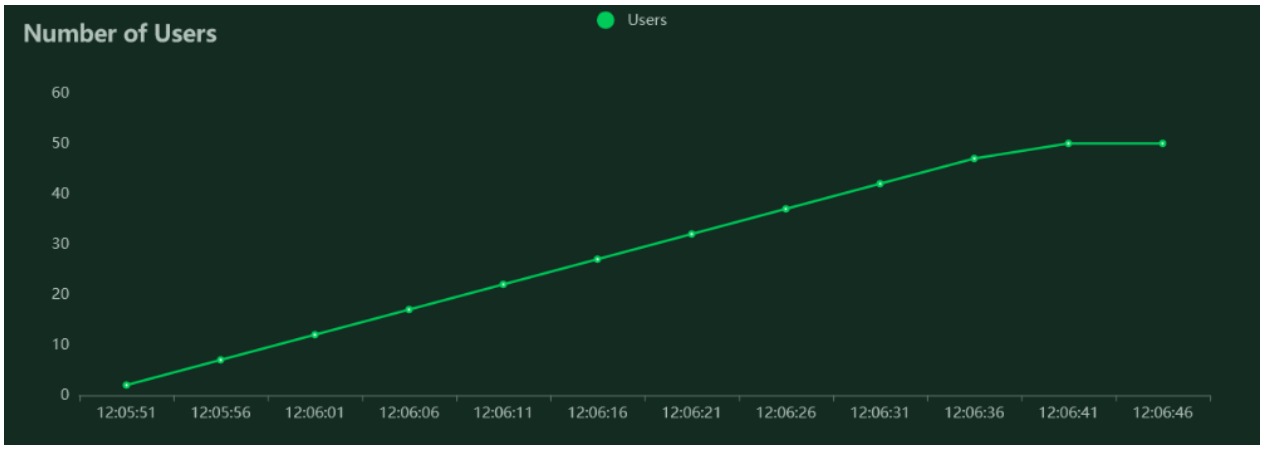
Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/	2000	2000	2100	2100	2100	2100	2100	2100
POST	/hello	2000	2100	2100	2100	2100	2100	2100	2100
	Aggregated	2000	2100	2100	2100	2100	2100	2100	2100

Charts







Final ratio

Ratio per User class

- 100.0% HelloWorldUser
  - 50.0% hello
  - 50.0% hello_name

Total ratio

- 100.0% HelloWorldUser
  - 50.0% hello
  - 50.0% hello_name

Case 3 – 500 users / spawn 5

During: 9.04.2023, 12:10:32 - 9.04.2023, 12:12:22

Target Host: http://localhost:5000

Script: locustfile.py

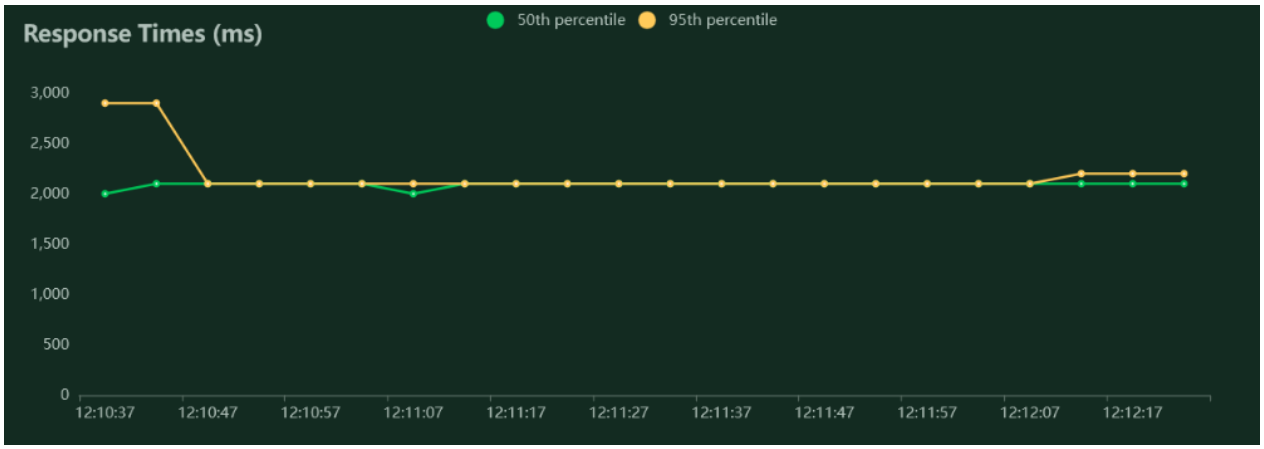
Request Statistics

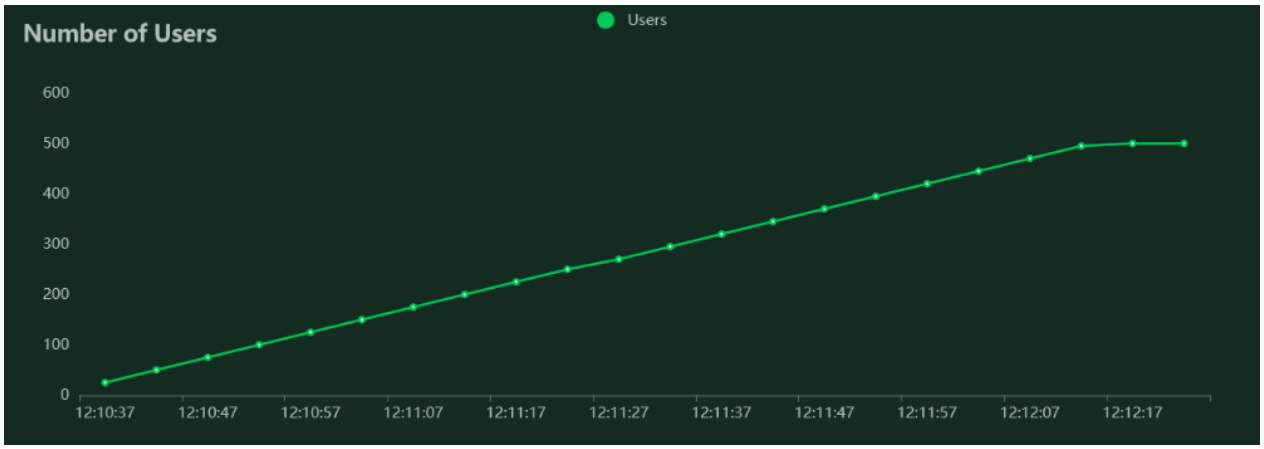
Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/	7134	0	2066	2021	2962	12	65.1	0.0
POST	/hello	7001	0	2068	2020	2921	319	63.9	0.0
	Aggregated	14135	0	2067	2020	2962	164	129.1	0.0

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/	2100	2100	2100	2100	2100	2100	2200	3000
POST	/hello	2100	2100	2100	2100	2100	2100	2200	2900
	Aggregated	2100	2100	2100	2100	2100	2100	2200	3000

Charts





Final ratio

Ratio per User class

- 100.0% HelloWorldUser
  - 50.0% hello
  - 50.0% hello_name

Total ratio

- 100.0% HelloWorldUser
  - 50.0% hello
  - 50.0% hello_name

Case 4 – 1000 users / spawn 10

During: 9.04.2023, 12:15:21 - 9.04.2023, 12:17:17

Target Host: http://localhost:5000

Script: locustfile.py

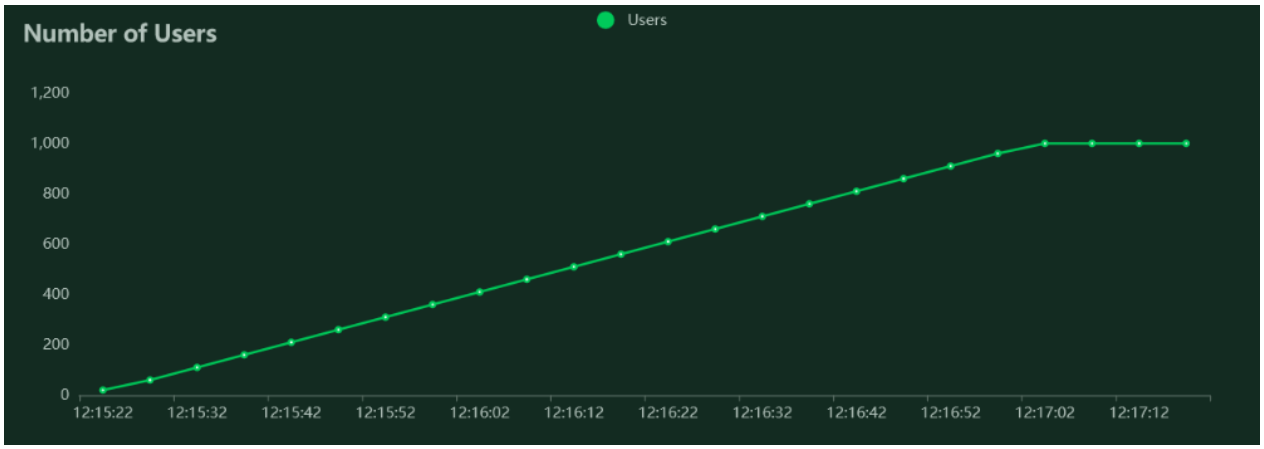
Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/	14966	0	2174	2011	3558	12	128.7	0.0
POST	/hello	14765	0	2180	2010	3622	319	126.9	0.0
	Aggregated	29731	0	2177	2010	3622	164	255.6	0.0

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/	2100	2100	2100	2200	2400	2800	3100	3600
POST	/hello	2100	2100	2100	2200	2500	2800	3100	3600
	Aggregated	2100	2100	2100	2200	2400	2800	3100	3600

Charts



Final ratio

Ratio per User class

- 100.0% HelloWorldUser
  - 50.0% hello
  - 50.0% hello_name

Total ratio

- 100.0% HelloWorldUser
  - 50.0% hello
  - 50.0% hello_name

Case 5 – 5000 users / spawn 20

During: 9.04.2023, 11:34:45 - 9.04.2023, 11:39:25

Target Host: http://localhost:5000

Script: locustfile.py

Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/	60931	335	6007	2022	20921	11	216.9	1.2
POST	/hello	60514	320	6133	2022	21480	317	215.5	1.1
	Aggregated	121445	655	6070	2022	21480	164	432.4	2.3

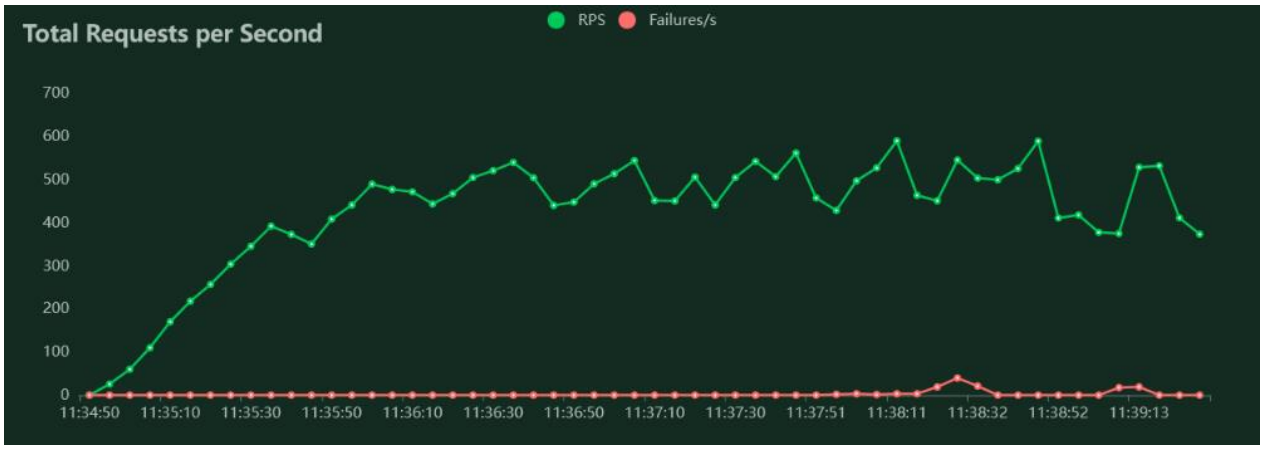
Response Time Statistics

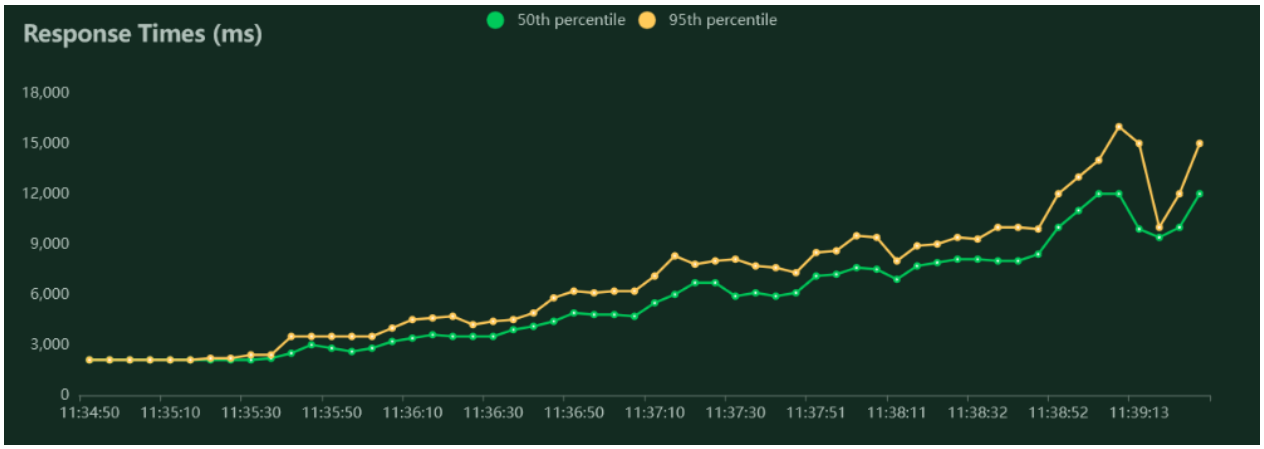
Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/	5800	6700	7500	8200	10000	12000	14000	21000
POST	/hello	5900	6900	7600	8400	10000	12000	14000	21000
	Aggregated	5800	6800	7600	8300	10000	12000	14000	21000

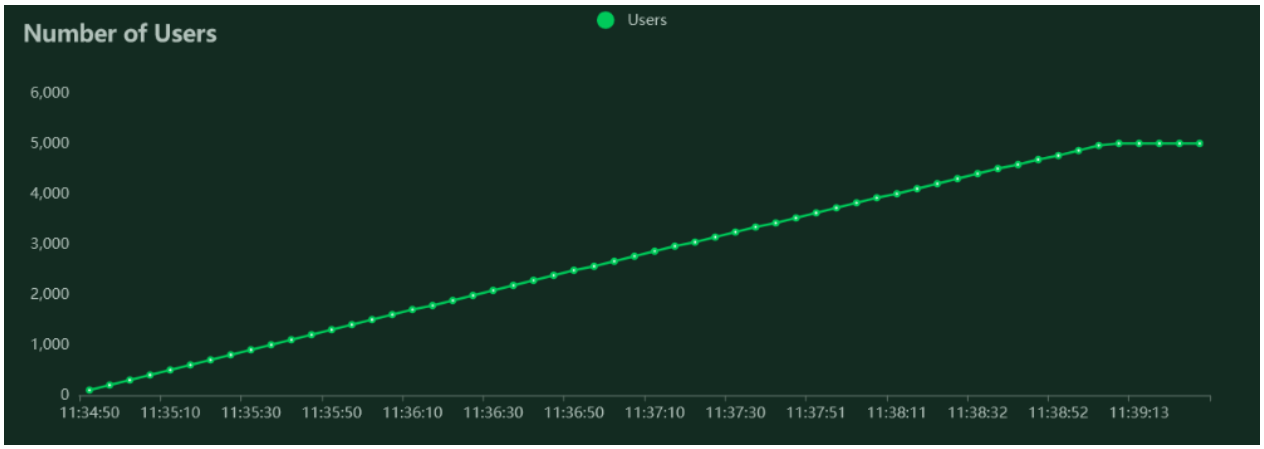
Failures Statistics

Method	Name	Error	Occurrences
GET	/	[Errno 10061] [WinError 10061] Nie można nawiązać połączenia, ponieważ komputer docelowy aktywnie go odmawia.	335
POST	/hello	[Errno 10061] [WinError 10061] Nie można nawiązać połączenia, ponieważ komputer docelowy aktywnie go odmawia.	320

Charts


Total Requests per Second
RPS
Failures/s
700
600
500
400
300
200
100
0
11:34:50
11:35:10
11:35:30
11:35:50
11:36:10
11:36:30
11:36:50
11:37:10
11:37:30
11:37:51
11:38:11
11:38:32
11:38:52
11:39:13


Response Times (ms)
50th percentile
95th percentile
18,000
15,000
12,000
9,000
6,000
3,000
0
11:34:50
11:35:10
11:35:30
11:35:50
11:36:10
11:36:30
11:36:50
11:37:10
11:37:30
11:37:51
11:38:11
11:38:32
11:38:52
11:39:13


Number of Users
Users
6,000
5,000
4,000
3,000
2,000
1,000
0
11:34:50
11:35:10
11:35:30
11:35:50
11:36:10
11:36:30
11:36:50
11:37:10
11:37:30
11:37:51
11:38:11
11:38:32
11:38:52
11:39:13

Final ratio

Ratio per User class

- 100.0% HelloWorldUser
  - 50.0% hello
  - 50.0% hello_name

Total ratio

- 100.0% HelloWorldUser
  - 50.0% hello
  - 50.0% hello_name