

Raport i wnioski z testów wydajności

Streszczenie	2
Wyniki testów	3
1. Test baseline - 240 VU	3
2. Peak test - 500 VU	6
3. Stress test - 5000 VU	7
4. Spike test - 50/1200 VU	9
5. Soak test - nie wykonano	10
Wnioski	11

Streszczenie

W ramach zadania wykonano testy wydajności aplikacji Ladybug przy użyciu toola k6.io

Zaplanowano 5 następujących scenariuszy:

1. Test działania API w normalnych warunkach - 240 użytkowników w czasie 10 minut.
2. Test działania API w warunkach „godzin szczytu” - „peak test” - wzrost liczby użytkowników do 500 w 5 minut, następnie 50 minut z 500 użytkownikami, a ostatecznie spadek użytkowników do 0 w 5 minut.
3. Test działania API w warunkach ekstremalnych - „stress test”, czyli ilu użytkowników maksymalnie może korzystać z serwisu.
4. Soak test - czyli test długotrwałej pracy w warunkach normalnej eksploatacji.
5. Spike test - czyli test gwałtownych wahań liczby użytkowników.

Sprawdzeniu podlegało składanie zamówień przez wielu użytkowników, a następnie akceptacja tych zamówień przez pracownika. Akceptacja przez pracownika została zaprojektowana do wykonania przez jednego użytkownika, zalogowanego jako employee/ test, zwiększanie liczby użytkowników (vus) nie wpływa na wydajność.

Testy wykonano w środowisku MacOS oraz Windows.

Wartości graniczne, jakie ustalono są następujące:

Całkowity czas: $p(95) < 750 \text{ ms}$

Całkowity czas: $\text{avg} < 400\text{ms}$

Check failure rate < 0.05

Wartości powyższe są dość wysokie, zważywszy, że aplikacja działa w sieci lokalnej, ewentualne straty na przesył danych mogą wydłużyć czas reakcji. Graniczny czas w warstwie UI powinien wynosić mniej niż 2 sekundy, powyżej tej wartości istnieje zwiększone ryzyko, że użytkownik zrezygnuje z korzystania z aplikacji.

Wyniki testów

1. Test baseline - 240 VU

```
k6 run -o influxdb=http://localhost:8086/mydb script_post.js
```



```
execution: local
  output: influxdb=http://localhost:8086/mydb (http://localhost:8086)
  script: script_post.js

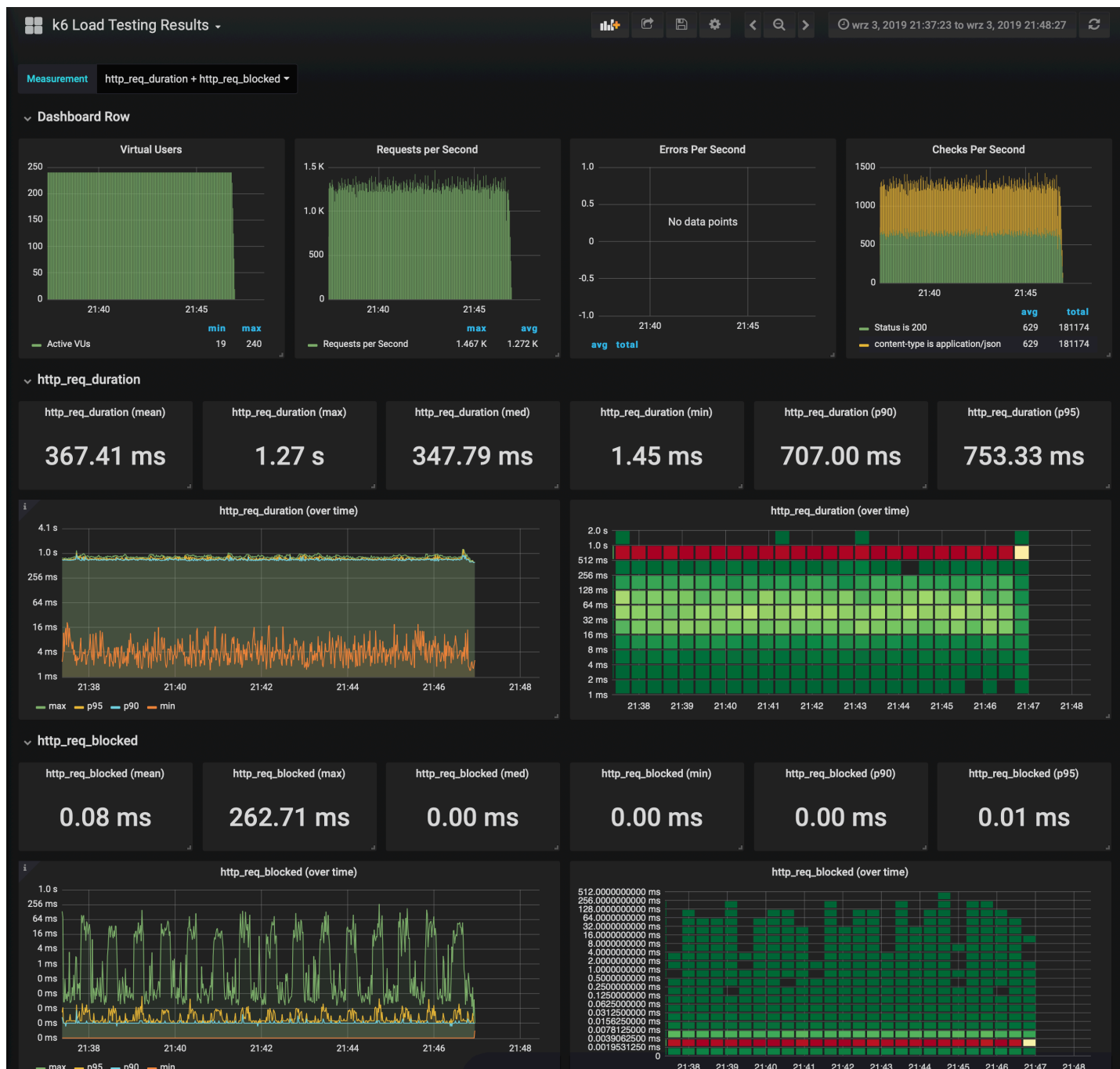
duration: -, iterations: -
  vus: 1, max: 240

done [=====] 10m20s / 10m20s

■ PART 1: Creating new orders.

  ✓ content-type is application/json
  ✓ Status is 200

checks.....: 100.00% ✓ 396412 x 0
data_received.....: 278 MB 448 kB/s
data_sent.....: 158 MB 254 kB/s
group_duration.....: avg=735.28ms min=603.14ms med=705.48ms max=2s p(90)=879.64ms p(95)=934.53ms
http_req_blocked.....: avg=78.41µs min=0s med=3µs max=262.7ms p(90)=4µs p(95)=5µs
http_req_connecting.....: avg=69.33µs min=0s med=0s max=244.89ms p(90)=0s p(95)=0s
✓ http_req_duration.....: avg=367.33ms min=1.45ms med=601.03ms max=1.27s p(90)=705.51ms p(95)=750.44ms
http_req_receiving.....: avg=373.96µs min=18µs med=174µs max=350.41ms p(90)=665µs p(95)=1ms
http_req_sending.....: avg=33.82µs min=5µs med=16µs max=93.55ms p(90)=33µs p(95)=46µs
http_req_tls_handshaking...: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_waiting.....: avg=366.92ms min=1.37ms med=600.84ms max=1.27s p(90)=705.15ms p(95)=750.05ms
http_reqs.....: 396421 639.388511/s
iteration_duration.....: avg=735.31ms min=603.15ms med=705.5ms max=2s p(90)=879.68ms p(95)=934.56ms
iterations.....: 198206 319.686997/s
vus.....: 1 min=1 max=240
vus_max.....: 240 min=240 max=240
```



W powyższym teście 240 VU wygenerowało ruch przez 10 minut i 20 sekund. Już dla takiego obciążenia wartości czasu AVG oraz p(95) są bliskie wartości granicznej.

Poniżej wykres oraz parametry dla funkcji akceptowania zamówień. W tym przypadku wszelkie wartości graniczne nie są przekroczone.

`k6 run script_put.js`



```
execution: local
output: influxdb=http://localhost:8086/mydb (http://localhost:8086)
script: script_put.js
```

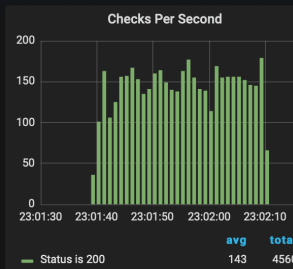
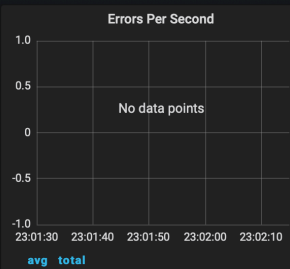
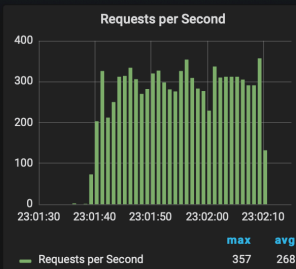
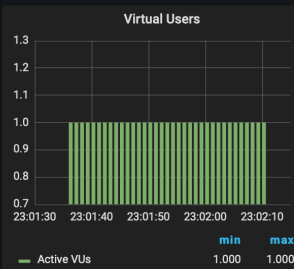
```
duration: -, iterations: 1
vus: 1, max: 1
```

```
done [=====] 1 / 1
```

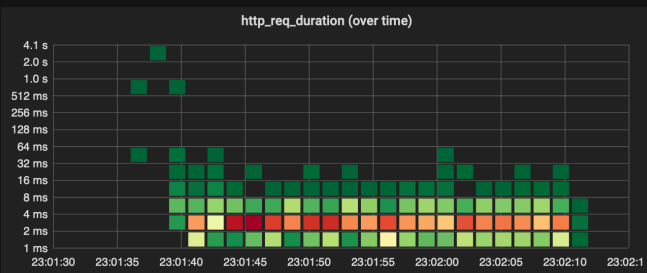
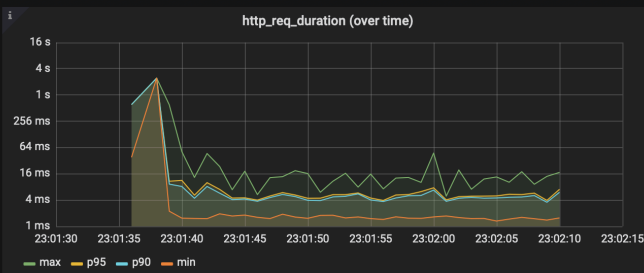
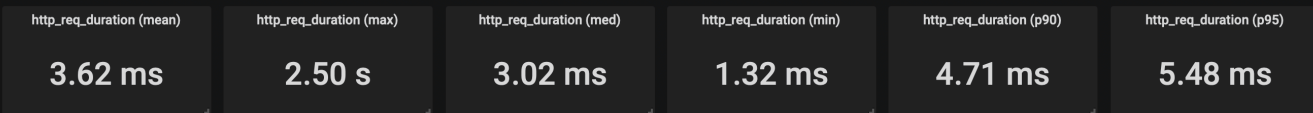
■ PART 2: Accepting orders.

✓ Status is 200

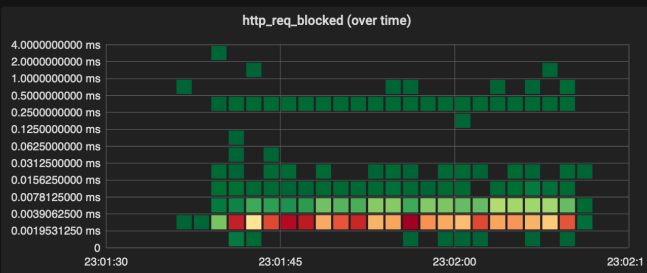
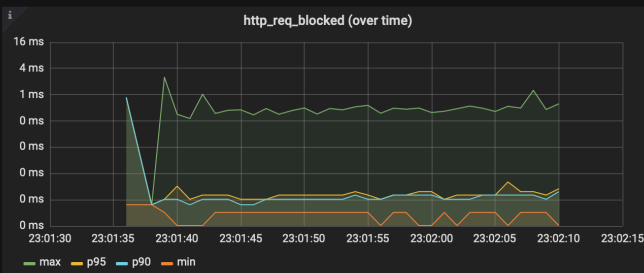
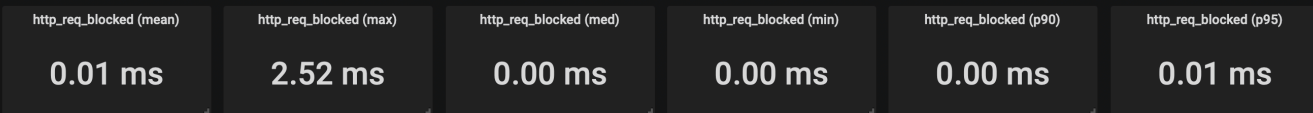
```
checks.....: 100.00% ✓ 4560 × 0
data_received.....: 10 MB 315 kB/s
data_sent.....: 4.3 MB 136 kB/s
group_duration.....: avg=31.62s min=31.62s med=31.62s max=31.62s p(90)=31.62s p(95)=31.62s
http_req_blocked.....: avg=7.81µs min=1µs med=3µs max=2.51ms p(90)=4µs p(95)=5µs
http_req_connecting.....: avg=3.63µs min=0s med=0s max=2.33ms p(90)=0s p(95)=0s
✓ http_req_duration.....: avg=3.62ms min=1.31ms med=3.02ms max=2.5s p(90)=4.71ms p(95)=5.47ms
http_req_receiving.....: avg=120.9µs min=50µs med=99µs max=50.68ms p(90)=162µs p(95)=198µs
http_req_sending.....: avg=20.45µs min=10µs med=18µs max=3.34ms p(90)=25µs p(95)=31µs
http_req_tls_handshaking.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_waiting.....: avg=3.48ms min=1.23ms med=2.89ms max=2.45s p(90)=4.54ms p(95)=5.29ms
http_reqs.....: 9124 288.504697/s
iteration_duration.....: avg=17.52s min=3.42s med=17.52s max=31.62s p(90)=28.8s p(95)=30.21s
iterations.....: 1 0.03162/s
vus.....: 1 min=1 max=1
vus_max.....: 1 min=1 max=1
```



✓ http_req_duration



✓ http_req_blocked




Dla pozostałych testów nie wygenerowano wykresów w Grafana ze względu na ograniczenia czasowe.

2. Peak test - 500 VU

```
k6 run script_post.js
```

Peak test przeprowadzony dla 500 VU również nie spełnił kryteriów czasu odpowiedzi.

```


execution: local-
output: -
script: script_post.js

duration: -, iterations: -
vus: 1, max: 500

done [=====] 1h0m0s / 1h0m0s

PART 1: Creating new orders.

Status is 200
content-type is application/json

checks.....: 100.00% 2238306 0
data_received.....: 1.6 GB 435 kB/s
data_sent.....: 890 MB 247 kB/s
group_duration.....: avg=1.46s min=602ms med=1.49s max=3.77s p(90)=1.8s p(95)=1.86s
http_req_blocked.....: avg=133.12µs min=0s med=0s max=640.46ms p(90)=0s p(95)=0s
http_req_connecting.....: avg=126.04µs min=0s med=0s max=640.46ms p(90)=0s p(95)=0s
[ ] http_req_duration.....: avg=731.65ms min=886.6µs med=676.99ms max=2.92s p(90)=1.17s p(95)=1.21s
http_req_receiving.....: avg=292.78µs min=0s med=0s max=305ms p(90)=920.1µs p(95)=1ms
http_req_sending.....: avg=71.5µs min=0s med=0s max=283ms p(90)=0s p(95)=0s
http_req_tls_handshaking....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_waiting.....: avg=731.28ms min=837.8µs med=676.25ms max=2.92s p(90)=1.17s p(95)=1.21s
http_reqs.....: 2238397 621.776843/s
iteration_duration.....: avg=1.46s min=602ms med=1.49s max=3.77s p(90)=1.8s p(95)=1.86s
iterations.....: 1119153 310.875783/s
vus.....: 1 min=1 max=500
vus_max.....: 500 min=500 max=500

ERRO[3613] some thresholds have failed

```

```

      .io
    }
  }
}

k6 run script_put.js

execution: local-
  output: -
    script: script_put.js

duration: -, iterations: 1
 vus: 1, max: 1

WARN[0061] Request Failed
ERRO[0061] Engine error
error="Get http://localhost:8080/api/orders/?size=1228716: net/http: request canceled (Client.Timeout exceeded while awaiting headers)"
error="setup: TypeError: Cannot read property 'content' of undefined at setup
t_airhelp/nowy2/airhelp-task-master/scri

ERRO[0061] Engine Error

```

```
k6 run script_post.js
```

```
k6 run script_post.js
```

Czasy $p(95) = 9.6s$ oraz $AVG = 4.95s$ przekroczyły znacznie założone limity.

Poniżej dodatkowe stress testy, w których czasy odpowiedzi są również znacznie przekroczone.

Dla 3500 VU

```

  A  NRG .io

execution: local-
output: -
script: script_post.js

duration: 1m0s, iterations: -
vus: 3500, max: 3500

done [=====] 1m0s / 1m0ss

PART 1: Creating new orders.

  Status is 200

checks.....: 100.00% 14662 0
data_received.....: 20 MB 342 kB/s
data_sent.....: 12 MB 194 kB/s
group_duration.....: avg=12.41s min=3.89s med=12.12s max=18.47s p(90)=14.89s p(95)=15.29s
http_req_blocked.....: avg=103.56ms min=0s med=0s max=1.85s p(90)=804ms p(95)=885ms
http_req_connecting.....: avg=102.75ms min=0s med=0s max=1.85s p(90)=798.99ms p(95)=884ms
  http_req_duration.....: avg=5.93s min=2.99ms med=5.71s max=13.83s p(90)=9.42s p(95)=9.95s
http_req_receiving.....: avg=560.14µs min=0s med=0s max=1.3s p(90)=1ms p(95)=1ms
http_req_sending.....: avg=9.01ms min=0s med=0s max=976ms p(90)=5.99ms p(95)=15.45ms
http_req_tls_handshaking...: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_waiting.....: avg=5.92s min=2.99ms med=5.7s max=13.82s p(90)=9.41s p(95)=9.91s
http_reqs.....: 31512 525.188515/s
iteration_duration.....: avg=12.45s min=654.41ms med=12.12s max=18.75s p(90)=15.01s p(95)=15.31s
iterations.....: 14662 244.361323/s
vus.....: 3500 min=3500 max=3500
vus_max.....: 3500 min=3500 max=3500

ERRO[0228] some thresholds have failed
```

Dla 2500 V

```

  A  NRG .io

execution: local-
output: -
script: script_post.js

duration: 5m0s, iterations: -
vus: 2500, max: 2500

done [=====] 5m0s / 5m0sm

PART 1: Creating new orders.

  Status is 200

checks.....: 100.00% 83984 0
data_received.....: 118 MB 392 kB/s
data_sent.....: 67 MB 223 kB/s
group_duration.....: avg=8.83s min=993.99ms med=8.45s max=17.42s p(90)=10.72s p(95)=11.36s
http_req_blocked.....: avg=6.47ms min=0s med=0s max=874.99ms p(90)=0s p(95)=0s
http_req_connecting.....: avg=6.43ms min=0s med=0s max=874.99ms p(90)=0s p(95)=0s
  http_req_duration.....: avg=4.4s min=1ms med=4.3s max=11.19s p(90)=6.45s p(95)=7.26s
http_req_receiving.....: avg=288.66µs min=0s med=0s max=991ms p(90)=999µs p(95)=1ms
http_req_sending.....: avg=232.48µs min=0s med=0s max=457ms p(90)=0s p(95)=0s
http_req_tls_handshaking...: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_waiting.....: avg=4.4s min=1ms med=4.3s max=11.19s p(90)=6.45s p(95)=7.26s
http_reqs.....: 168358 561.191243/s
iteration_duration.....: avg=8.83s min=618.25ms med=8.45s max=17.42s p(90)=10.72s p(95)=11.36s
iterations.....: 83984 279.945624/s
vus.....: 2500 min=2500 max=2500
vus_max.....: 2500 min=2500 max=2500

ERRO[0347] some thresholds have failed
```


4. Spike test - 50/1200 VU

Spike testy również pokazały że przy ww. obciążeniu nie spełnia zakładanych wymagań.



```
execution: local-  
output: -  
script: script_post.js
```

```
duration: -, iterations: -  
vus: 1, max: 1200
```

```
done [=====] 4m0s / 4m0sm
```

```
■ PART 1: Creating new orders.
```

```
■ Status is 200
```

```
■ content-type is application/json
```

```
checks.....: 100.00% ■ 93630 ■ 0
```

```
data_received.....: 66 MB 276 kB/s
```

```
data_sent.....: 39 MB 161 kB/s
```

```
group_duration.....: avg=2.71s min=601.98ms med=1.86s max=7.03s p(90)=4.81s p(95)=5.21s
```

```
http_req_blocked.....: avg=3.63ms min=0s med=0s max=305.99ms p(90)=0s p(95)=0s
```

```
http_req_connecting.....: avg=3.6ms min=0s med=0s max=305.99ms p(90)=0s p(95)=0s
```

```
■ http_req_duration.....: avg=1.36s min=998.8µs med=1.16s max=4.7s p(90)=2.66s p(95)=2.98s
```

```
http_req_receiving.....: avg=180.09µs min=0s med=0s max=187ms p(90)=991.1µs p(95)=1ms
```

```
http_req_sending.....: avg=287.32µs min=0s med=0s max=131.99ms p(90)=0s p(95)=0s
```

```
http_req_tls_handshaking...: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
```

```
http_req_waiting.....: avg=1.36s min=996.9µs med=1.16s max=4.7s p(90)=2.66s p(95)=2.98s
```

```
http_reqs.....: 95035 395.965939/s
```

```
iteration_duration.....: avg=2.71s min=601.98ms med=1.86s max=7.03s p(90)=4.81s p(95)=5.21s
```

```
iterations.....: 46815 195.055984/s
```

```
vus.....: 50 min=1 max=1200
```

```
vus_max.....: 1200 min=1200 max=1200
```

```
ERRO[0262] some thresholds have failed
```



```
execution: local-  
output: -  
script: script_put.js
```

```
duration: -, iterations: 1  
vus: 1, max: 1
```

```
done [=====] 1 / 1i
```

```
■ PART 2: Accepting orders.
```

```
■ Status is 200
```

```
checks.....: 100.00% ■ 48222 ■ 0  
data_received.....: 105 MB 412 kB/s  
data_sent.....: 46 MB 178 kB/s  
group_duration.....: avg=4m15s min=4m15s med=4m15s max=4m15s p(90)=4m15s p(95)=4m15s  
http_req_blocked.....: avg=5.99µs min=0s med=0s max=3.99ms p(90)=0s p(95)=0s  
http_req_connecting.....: avg=3.48µs min=0s med=0s max=3.99ms p(90)=0s p(95)=0s  
■ http_req_duration.....: avg=2.75ms min=940.8µs med=2.02ms max=24.41s p(90)=3ms p(95)=3.99ms  
http_req_receiving.....: avg=123.05µs min=0s med=0s max=692ms p(90)=995.5µs p(95)=1ms  
http_req_sending.....: avg=15.65µs min=0s med=0s max=2.99ms p(90)=0s p(95)=0s  
http_req_tls_handshaking...: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s  
http_req_waiting.....: avg=2.61ms min=935µs med=2ms max=23.72s p(90)=3ms p(95)=3.02ms  
http_reqs.....: 96448 376.850496/s  
iteration_duration.....: avg=2m21s min=27.95s med=2m21s max=4m15s p(90)=3m53s p(95)=4m4s  
iterations.....: 1 0.003907/s  
vus.....: 1 min=1 max=1  
vus_max.....: 1 min=1 max=1
```

5. Soak test - nie wykonano

Soak test nie został wykonany z racji długości testu.

Wnioski

Aplikacja w większości przeprowadzonych testów nie spełnia wyznaczonych przeze mnie wartości granicznych czasów odpowiedzi. Czas odpowiedzi w tym przypadku jest głównym czynnikiem według którego możemy uznać, że aplikacja nie działa poprawnie. Nawet przy obciążeniu rzędu 5000 VU nie udało mi się uzyskać **żadnych błędnych odpowiedzi ze strony serwera (Error rate = 0)**, każdy request był sprawdzany pod kątem statusu oraz zawartości. Ponadto, warto zauważyć, że **aplikacja jest skalowalna** - dla 1 VU średni czas odpowiedzi serwera wynosi 300ms (p(95)=606ms), dla 240 VU wynosi 360ms (p(95)=750ms), a dla 100 VU wynosi 306ms (p(95)=610ms). Co prawda nie pracuje szybko dla małej liczby użytkowników, natomiast wzrost liczby użytkowników, co do zasady nie powoduje dużego spadku wydajności.

Ilość VU, dla których aplikacja działa poprawnie (w granicach założonych wymagań) oscyluje w okolicy 240 VU. Scenariusze testowe przygotowane na githubie zostały zaktualizowane tak, aby ilości VU były bardziej adekwatne do aplikacji.

Zagrożeniem, które może spowodować długotrwałe działanie aplikacji pod dużym obciążeniem jest **zużycie całej dostępnej pamięci**. Soak test nie został przeprowadzony ze względu na ograniczenia czasowe oraz sprzętowe, natomiast wykonałem godzinny test z obciążeniem 750 VU, który spowodował **wzrost zużycia pamięci przez Java Platform SE z ok. 200 MB do prawie 2GB**.

Narzędzie k6 posiada pewne ograniczenia - przy bardzo dużej liczbie rekordów w bazie oraz ustawieniu parametru `setupTimeout` nawet powyżej 5 minut nie można było wykonać skryptu akceptującego zamówienia ze względu na engine error:

ERR0[0061] Engine Error