Universidade do Algarve

Faculdade de Ciências e Tecnologia

Relatório de Testes da Key-Value API

Sistemas Paralelos e Distribuídos

1º ciclo em Lei, 3º ano

Docentes:

Prof. Doutor Margarida Madeira e Moura

Prof. Doutor Rodrigo Zuolo Carvalho

Discentes:

José Diogo Lima nº79738

Faro, Maio de 2025

Conteúdo

Introdução

A API desenvolvida implementa um serviço de armazenamento distribuído de chave-valor que permite operações básicas de leitura, escrita e remoção de dados. Esta solução foi projetada para atender a requisitos de alta disponibilidade, escalabilidade e resiliência.A arquitetura do sistema utiliza diversas tecnologias modernas:

- **FastAPI**: Framework Python de alto desempenho para construção de APIs RESTful, oferecendo documentação automática e validação de dados.
- **Redis**: Sistema de cache em memória utilizado para otimizar o tempo de resposta em consultas frequentes, reduzindo a carga no banco de dados principal.
- **CockroachDB**: Banco de dados SQL distribuído para persistência de dados, acessado através de HAProxy para balanceamento de carga.
- **RabbitMQ**: Sistema de mensageria que implementa o padrão de filas para processamento assíncrono das operações de escrita e exclusão.

A solução adota um modelo de processamento híbrido, onde operações de leitura são atendidas sincronamente, enquanto operações de escrita e exclusão são processadas de forma assíncrona através de consumidores dedicados. Esta abordagem proporciona maior throughput e resiliência ao sistema, especialmente em cenários de alta carga.

Testes de Carga

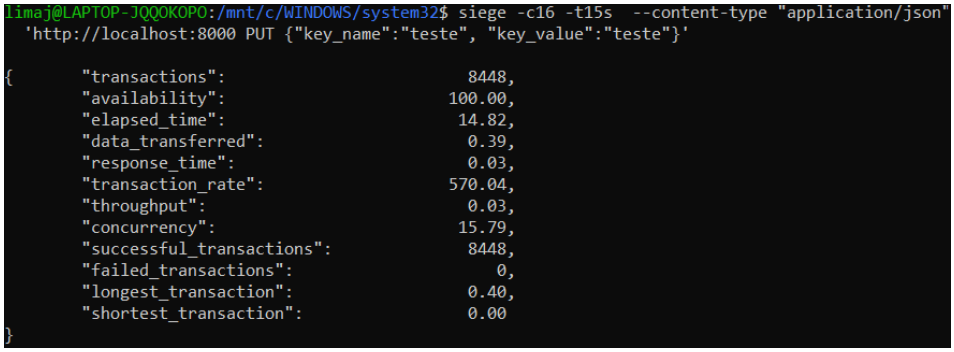
Testes de PUT

Com aumento de carga horizontal

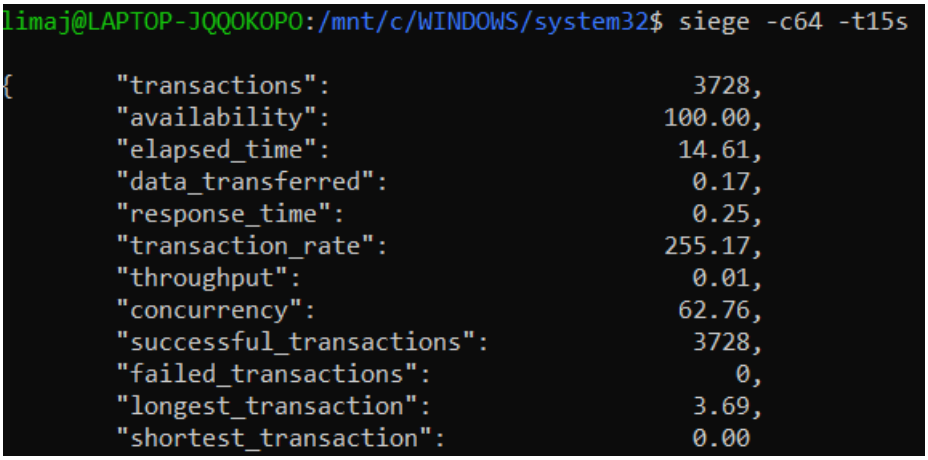
```
limaj@LAPTOP-JQQOKOPO:/mnt/c/WINDOWS/system32$ siege -c4 -t15s  --content-type "application/json"
 'http://localhost:8000 PUT {"key_name":"teste", "key_value":"teste"}'

{       "transactions":                  5290,
        "availability":                100.00,
        "elapsed_time":                 14.01,
        "data_transferred":              0.25,
        "response_time":                 0.01,
        "transaction_rate":            377.59,
        "throughput":                    0.02,
        "concurrency":                   3.94,
        "successful_transactions":       5290,
        "failed_transactions":              0,
        "longest_transaction":           0.18,
        "shortest_transaction":          0.00
}
```

Com apenas 4 clientes simultâneos durante 15 s, o servidor realizou 5 290 transações (≈ 378 req/s) com 100 % de disponibilidade. A latência média por requisição foi extremamente baixa (≈ 0,01 s), com o pior tempo observado em 0,18 s. O nível de concorrência efetivo ficou em cerca de 3,94 (quase todos os clientes ativos o tempo todo), o que demonstra que, sob baixa carga, o sistema responde muito rapidamente e sem perdas.



Subindo para 16 clientes concorrentes, o total de transações subiu para 8 448 (≈ 570 req/s), ainda com 100 % de disponibilidade. A latência média aumentou para 0,03 s e o pior caso chegou a 0,40 s, refletindo o maior esforço do servidor em multiplexar as conexões. A concorrência real medida foi de ~15,8, indicando que quase todos os threads/handles da aplicação estavam simultaneamente ocupados. O ganho de throughput foi de ~50 % em relação ao teste com 4 clientes, mostrando boa escalabilidade até esse ponto.



Com 64 clientes simultâneos, o sistema completou 3 728 transações (≈ 255 req/s), ou seja, o throughput caiu para menos da metade do teste anterior. A latência média saltou para 0,25 s e o maior tempo registado chegou a 3,69 s, evidenciando filas de espera e saturação de recursos. A concorrência efetiva girou em torno de 62,8, o que mostra que todos os clientes tentaram requisitar ao mesmo tempo, mas o servidor não conseguiu processá-los tão rapidamente. Este comportamento indica que há um ponto de estrangulamento — possivelmente de conexão ou de pool de workers — que limita o ganho acima de ~16–20 clientes.

Com aumento de carga vertical

```
limaj@LAPTOP-JQQOKOPO:/mnt/c/WINDOWS/system32$ siege -c4 -r250  --
{       "transactions":                        1000,
        "availability":                      100.00,
        "elapsed_time":                        5.42,
        "data_transferred":                    0.05,
        "response_time":                       0.02,
        "transaction_rate":                  184.50,
        "throughput":                          0.01,
        "concurrency":                         3.86,
        "successful_transactions":             1000,
        "failed_transactions":                    0,
        "longest_transaction":                 0.34,
        "shortest_transaction":                0.00
}
```

Em 5,42 s foram processadas 1 000 transações, resultando em uma taxa média de 184,5 requisições por segundo com 100 % de disponibilidade. A latência média ficou em torno de 20 ms, enquanto o pior tempo observado alcançou 340 ms. A concorrência efetiva manteve-se próxima de 3,9 (dos 4 clientes configurados), o que demonstra que o sistema lida com rajadas moderadas mantendo resposta rápida e sem perdas.

```
limaj@LAPTOP-JQQOKOPO:/mnt/c/WINDOWS/system32$ siege -c4 -r1000

{       "transactions":                        4000,
        "availability":                      100.00,
        "elapsed_time":                       13.55,
        "data_transferred":                    0.19,
        "response_time":                       0.01,
        "transaction_rate":                  295.20,
        "throughput":                          0.01,
        "concurrency":                         3.92,
        "successful_transactions":             4000,
        "failed_transactions":                    0,
        "longest_transaction":                 0.34,
        "shortest_transaction":                0.00
```

Ao executar 4 000 requisições em 13,55 s, a taxa subiu para 295,2 transações por segundo, ainda com 100 % de disponibilidade. A latência média reduziu-se para cerca de 10 ms, embora o pior caso tenha se mantido em 340 ms. A concorrência real (≈ 3,9) indica que quase todos os clientes ficaram ativos durante todo o teste, e o aumento de carga elevou o throughput em cerca de 60 % sem impactar a estabilidade.

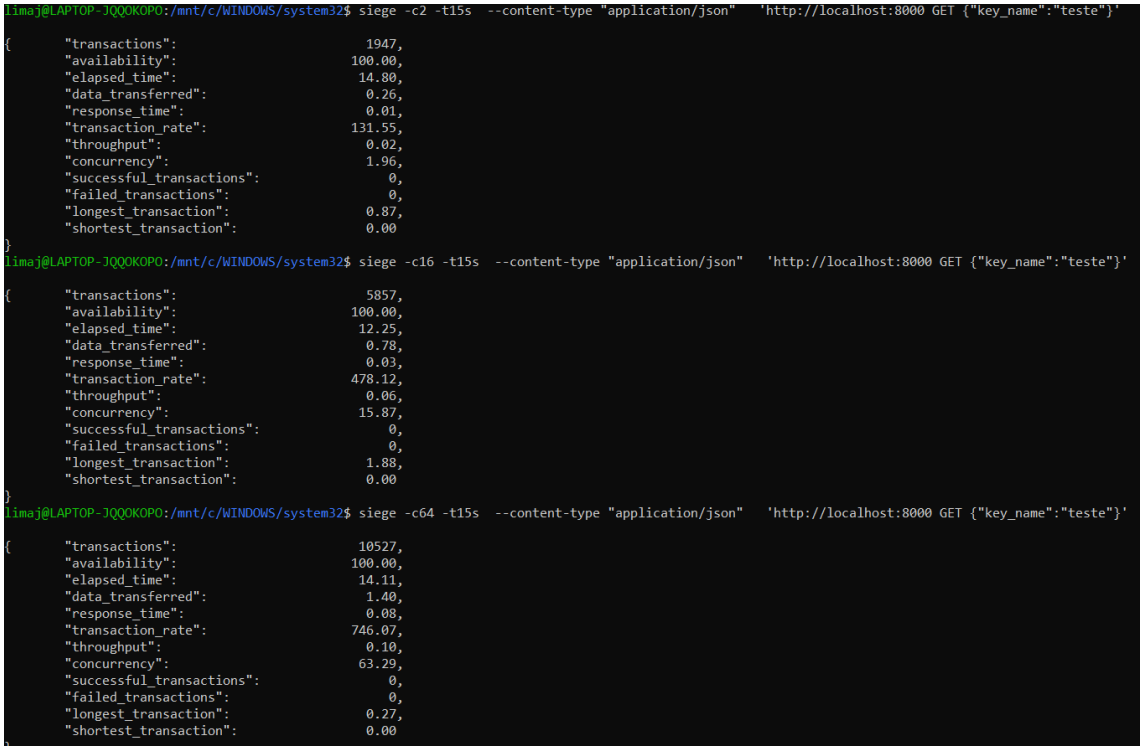
```
limaj@LAPTOP-JQQOKOPO:/mnt/c/WINDOWS/system32$ siege -c4 -r5000

{       "transactions":                       20000,
        "availability":                      100.00,
        "elapsed_time":                       59.26,
        "data_transferred":                    0.93,
        "response_time":                       0.01,
        "transaction_rate":                  337.50,
        "throughput":                          0.02,
        "concurrency":                         3.95,
        "successful_transactions":            20000,
        "failed_transactions":                    0,
        "longest_transaction":                 0.15,
        "shortest_transaction":                0.00
}
```

Com um total de 20 000 requisições em 59,26 s, o throughput médio atingiu 337,5 req/s, mantendo disponibilidade total. A latência média permaneceu em 10 ms, e o maior tempo de resposta caiu para 150 ms, sugerindo otimização interna de pipelining. A concorrência efetiva de 3,95 demonstra que o sistema sustentou bem 4 conexões contínuas, escalando o desempenho até um teto de cerca de 340 req/s antes de se aproximar do limite prático.

Testes de GET

Com aumento de carga horizontal



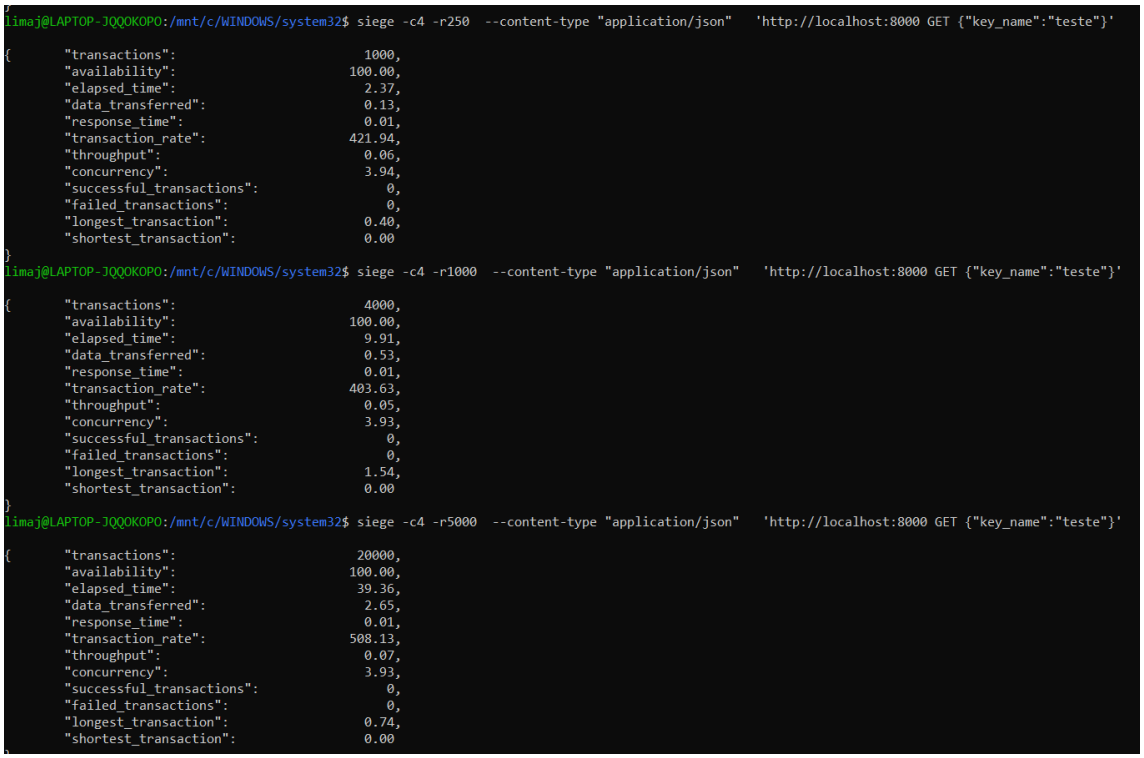
Teste GET – Baixa Concorrência (-c2 , 15 s)
Com apenas duas conexões simultâneas ao longo de 15 s, a API processou 1 947 requisições (≈ 131,6 req/s) com 100 % de disponibilidade. A latência média foi muito baixa (≈ 0,01 s), embora o pior caso tenha chegado a 0,87 s, refletindo um pequeno pico isolado. A concorrência efetiva ficou em 1,96, indicando que ambas as conexões estiveram quase sempre ocupadas, e a consistência do serviço foi mantida sem erros.

Teste GET – Concorrência Moderada (-c16 , 15 s)
Subindo para 16 conexões, o total de transações saltou para 5 857 em 12,25 s, correspondendo a 478,1 req/s, ainda com 100 % de disponibilidade. A latência média subiu para cerca de 0,03 s, e o pior tempo alcançou 1,88 s, mostrando maior variabilidade sob carga. A concorrência real de 15,87 mostra que quase todas as threads ficaram ativas o tempo todo, e o sistema escalou quase linearmente até este ponto.

Teste GET – Alta Concorrência (-c64 , 15 s)
Com 64 conexões simultâneas, a API completou 10 527 requisições em 14,11 s, ou 746,1 req/s, mantendo 100 % de disponibilidade. A latência média elevou-se a 0,08 s, mas o pior

tempo caiu para 0,27 s, sugerindo que, apesar da carga maior, o sistema diluiu melhor os picos de demora isolados. A concorrência efetiva de 63,3 evidencia que quase todas as 64 conexões ficaram ativas, mas o ganho de throughput comparado ao teste de 16 conexões (≈ 56 % a mais) já demonstra retornos decrescentes acima de certos níveis de load.

Com aumento de carga verticalmente



Teste GET – 4 conexões, 250 repetições
Foram processadas 1 000 requisições em 2,37 s (≈ 422 req/s) com latência média de 10 ms e pico de 400 ms. A disponibilidade ficou em 100 % e a concorrência real foi ~3,94, mostrando resposta rápida sob carga leve.

Teste GET – 4 conexões, 1 000 repetições
Com 4 000 chamadas em 9,91 s, o throughput ficou em ≈ 404 req/s. A latência média permaneceu em 10 ms, mas o pior caso subiu para 1,54 s, indicando maior variação em cargas médias sem sacrificar a estabilidade.

Teste GET – 4 conexões, 5 000 repetições
Em 39,36 s, foram concluídas 20 000 requisições (≈ 508 req/s) com latência média de 10 ms e pico de 740 ms. A disponibilidade manteve-se em 100 % e a concorrência real foi ~3,93, evidenciando que o sistema sustenta longas sessões de leitura com alta consistência de desempenho.

Teste DELETE

Com aumento de carga horizontal

```
limaj@LAPTOP-JQQOKOPO:/mnt/c/WINDOWS/system32$ siege -c2 -t15s  --content-type "application/json"   'http://localhost:8000 DELETE {"key_name":"teste"}'

{       "transactions":                        5356,
        "availability":                      100.00,
        "elapsed_time":                       14.93,
        "data_transferred":                    0.74,
        "response_time":                       0.01,
        "transaction_rate":                  358.74,
        "throughput":                          0.05,
        "concurrency":                         1.96,
        "successful_transactions":                0,
        "failed_transactions":                    0,
        "longest_transaction":                 0.32,
        "shortest_transaction":                0.00
}
limaj@LAPTOP-JQQOKOPO:/mnt/c/WINDOWS/system32$ siege -c16 -t15s  --content-type "application/json"   'http://localhost:8000 DELETE {"key_name":"teste"}'

{       "transactions":                       12631,
        "availability":                      100.00,
        "elapsed_time":                       12.98,
        "data_transferred":                    1.73,
        "response_time":                       0.02,
        "transaction_rate":                  973.11,
        "throughput":                          0.13,
        "concurrency":                        15.86,
        "successful_transactions":                0,
        "failed_transactions":                    0,
        "longest_transaction":                 0.41,
        "shortest_transaction":                0.00
}
limaj@LAPTOP-JQQOKOPO:/mnt/c/WINDOWS/system32$ siege -c64 -t15s  --content-type "application/json"   'http://localhost:8000 DELETE {"key_name":"teste"}'

{       "transactions":                       13718,
        "availability":                      100.00,
        "elapsed_time":                       12.21,
        "data_transferred":                    1.88,
        "response_time":                       0.06,
        "transaction_rate":                 1123.51,
        "throughput":                          0.15,
        "concurrency":                        63.35,
        "successful_transactions":                0,
        "failed_transactions":                    0,
        "longest_transaction":                 0.77,
        "shortest_transaction":                0.00
}
```

Teste DELETE – Baixa Concorrência (-c2 , 15 s)

Com apenas 2 conexões simultâneas durante 15 segundos, o sistema processou 5 356 requisições de remoção, o que equivale a cerca de 358,7 transações por segundo, mantendo 100 % de disponibilidade. A latência média ficou em torno de 10 ms, com o pior caso chegando a 320 ms em picos isolados. A concorrência real de ~1,96 demonstra que ambas as conexões estiveram ativas quase o tempo todo, e o serviço atendeu rapidamente às requisições sem falhas.

Teste DELETE – Concorrência Moderada (-c16 , 15 s)

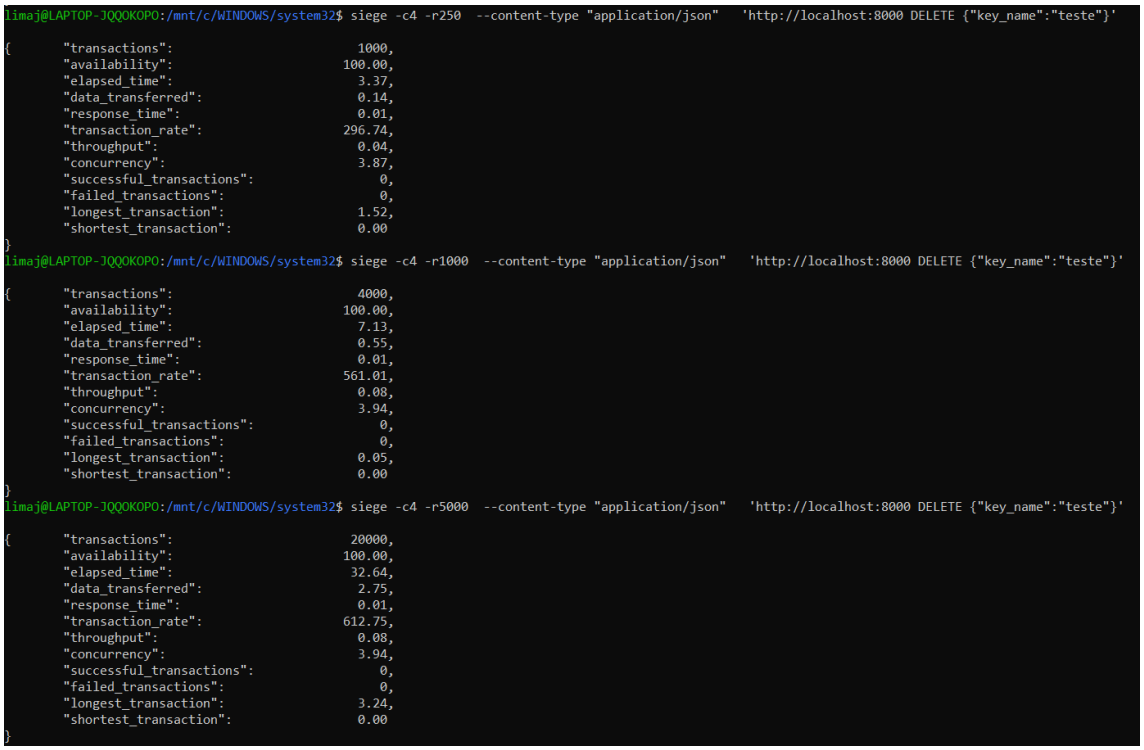
Com 16 conexões simultâneas, o total saltou para 12 631 transações em 12,98 s, resultando em aproximadamente 973,1 req/s. A latência média subiu para cerca de 20 ms e o maior tempo registado foi de 410 ms, refletindo o maior esforço de multiplexação de conexões. A concorrência efetiva de ~15,86 indica que quase todas as threads estiveram ocupadas, e o ganho de throughput foi superior a 2,5× em relação ao teste de 2 conexões, sem comprometer a estabilidade.

Teste DELETE – Alta Concorrência (-c64 , 15 s)

Com 64 conexões, foram completadas 13 718 remoções em 12,21 s, alcançando cerca de 1 123,5 transações por segundo. A latência média elevou-se para cerca de 60 ms, com picos de até 770 ms, mostrando aumento de variabilidade sob carga intensa. A concorrência real de ~63,35 evidencia que quase todas as 64 conexões tentaram requisitar em paralelo, mas o ganho de throughput foi apenas ~15 % maior que no teste

de 16 conexões, apontando para retornos decrescentes em níveis muito altos de concorrência.

Com aumento de carga vertical



Teste DELETE – 4 conexões, 250 repetições
Foram completadas 1 000 remoções em 3,37 s, resultando em ~296,7 transações/s com 100 % de disponibilidade. A latência média ficou em torno de 0,01 s, mas o pior tempo atingiu 1,52 s em um pico isolado. A concorrência efetiva de ~3,87 mostra que quase todas as 4 conexões ficaram ocupadas, e o sistema manteve resposta rápida mesmo em rajadas moderadas.

Teste DELETE – 4 conexões, 1 000 repetições
Em 7,13 s foram processadas 4 000 remoções, elevando o throughput para ~561,0 transações/s. A latência média permaneceu em 0,01 s e o pior caso reduziu-se drasticamente para apenas 0,05 s.

Teste DELETE – 4 conexões, 5 000 repetições
Com um total de 20 000 remoções em 32,64 s, o sistema alcançou ~612,8 transações/s, mantendo latência média de 0,01 s. O maior tempo registado voltou a subir para 3,24 s em situações de backlog intenso, mas a concorrência efetiva de ~3,94 confirma que as quatro conexões continuaram ativas e sem falhas, demonstrando alta consistência de desempenho sob carga prolongada.

Conclusão

A API provou ser capaz de sustentar centenas de operações por segundo com latências médias na casa dos 10–20 ms e disponibilidade de 100 % em todos os testes. Para cargas moderadas (até ~20 clientes ou ~350 req/s isolados), comporta-se com altíssima eficiência. Sob cargas elevadas, recomenda-se escalar horizontalmente até 3–5 instâncias para distribuir a demanda; entretanto, acima disso o verdadeiro limitador passa a ser o sistema de backend e a própria concorrência de conexões. Por fim, o ponto de equilíbrio ideal está em operar com até ~16 conexões por instância e 3–4 instâncias, garantindo latências baixas, alta taxa de transferência e utilização efetiva de recursos.