Nomes: Cecília Araújo, Leone Daher e Lucas Oliveira

MEDIÇÕES DO SLA

Importante: para esta medição tratamos os resultados em gráficos e os comparamos entre os próprios serviços. Por este motivo, os resultados de latência, vazão e concorrência serão apresentados em gráficos juntos.

- **Seleção de produto disponível**
- Tipo de operação: leitura
- Arquivos envolvidos:
  - ProductController - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/controllers/ProductController.java
  - ProductRepository - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/repository/ProductRepository.java
  - ProductService - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/service/ProductService.java
  - Product - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/model/Product.java

- Arquivos com o código fonte de medição do SLA
  - Foi usado JMeter. O comando para a obtenção dos dados que se encontram na pasta Medicao_SLA é este:
    jmeter -n -t "path_plan_test.jmx" -l "path_resultado.jtl" -e -o "path_folder_results"

  - Link da pasta dos resultados (arquivo index.html):

    https://github.com/LucasFPO/projeto-kipao/tree/main/Medicao_SLA

  - Screenshot do programa:

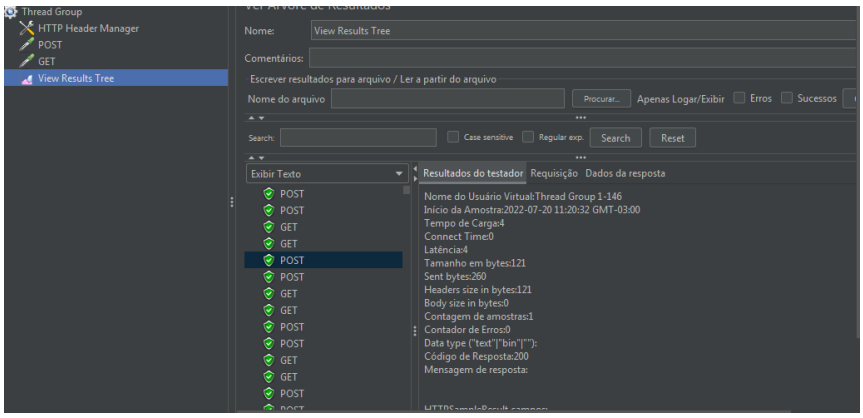


Figura 1. Comportamento da execução do teste. Esquerda: plano de teste com duas requisições configuradas. Ao lado: requisições bem-sucedidas após chamada dos serviços. Direita: Resultado de uma das requisições com informações relevantes.

Figura 2. Configuração de chamada de processo para ambos serviços.

- Data da medição: 20/07/2022
- Descrição das configurações:

  Para as chamadas dos serviços do sistema foi utilizado a linguagem de programação Java na IDE Eclipse, MySQL como banco de dados para armazenamento dos dados e configuração no JMeter para a realização das medições. Para a leitura adicionamos um produto a fim de ser possível obter retorno esperado.

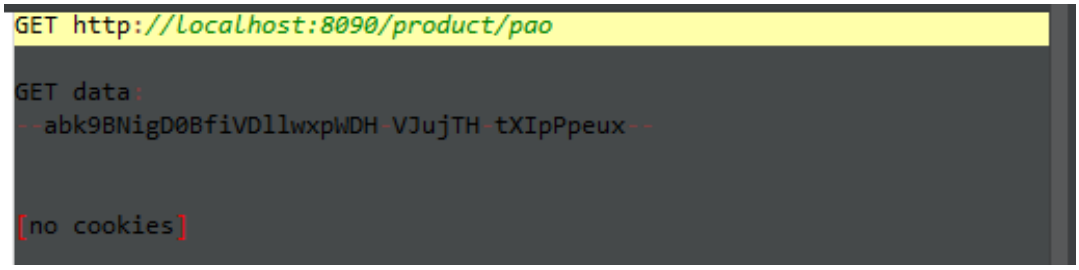


Figura 3.

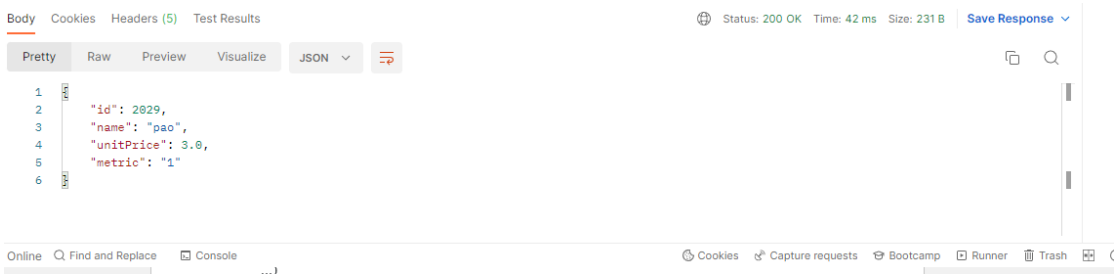


Figura 4.

- **Adição de produto na lista de produtos disponíveis**
- Tipo de operação: inserção
- Arquivos envolvidos:
  - ProductController - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/controllers/ProductController.java

    - ProductRepository - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/repository/ProductRepository.java
    - ProductService - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/service/ProductService.java
    - Product - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/model/Product.java
- Arquivos com o código fonte de medição do SLA
    - Tanto o serviço 1 como o serviço 2 encontram-se na mesma estrutura. Vide o item do serviço 1.
- Data da medição: última medição 20/07/2022
- Descrição das configurações:

Assim como no serviço 1 foi utilizado a linguagem de programação Java na IDE Eclipse, MySQL como banco de dados para armazenamento dos dados e configuração no JMeter para a realização das medições. Para a inserção do produto foi utilizada a seguinte estrutura do JSON:

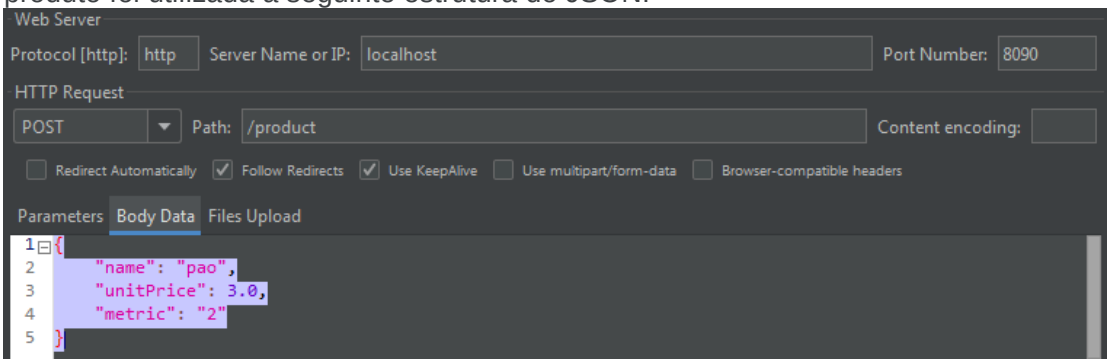


Figura 5.

Resultados dos serviços:

- SLA: latência, vazão, concorrência

Os microserviços que chamam os endpoints de busca dos produtos disponíveis e inserção de produto foi configurado para atender a 150 usuários (processos) por 1 minuto e que a cada segundo é adicionado mais 2 usuários. Esse aumento gradual tem o objetivo de fazer mais sentido ao resultado de teste visto que nem todos os usuários “requisitam” ao mesmo tempo. Foi, também, deixado 1 minuto de tempo em espera para verificar se o sistema pode lidar com a carga e se o desempenho permanece estável e não se deteriora.

Nesse caso, temos os seguintes dados obtidos e que podem ser conferidos na pasta ‘Medicao_SLA’:

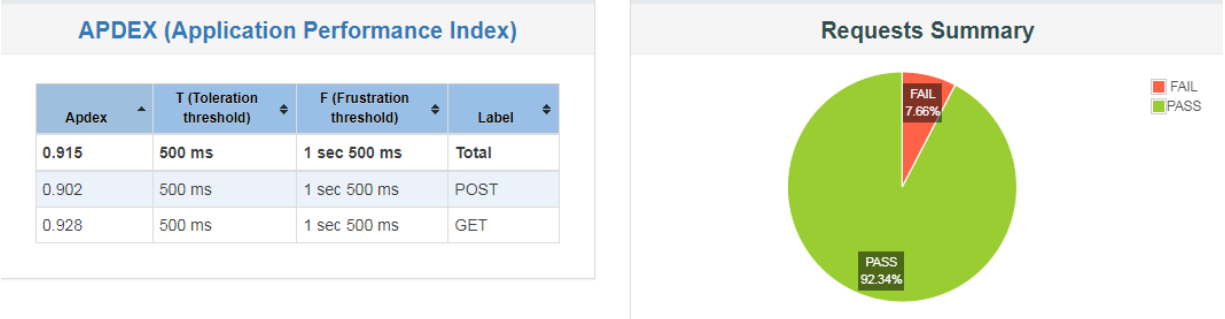


Figura 6.

No total foram bem-sucedidos mais de 92% das requisições demandadas, enquanto que menos de 8% falharam. Foram realizadas **5862 requisições**. O serviço 2 foi o que mais obteve chances de falha com cerca de 9%.

Latência:

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**5862**	**449**	**7.66%**	**2746.94**	**3**	**60009**	**13.00**	**41.00**	**30005.00**	**30042.00**	**37.19**	**6.74**	**9.62**
GET	2886	179	6.20%	2158.26	6	30097	13.00	29.30	30005.00	30007.13	19.27	4.52	5.08
POST	2976	270	9.07%	3317.81	3	60009	14.00	65.30	30006.00	57698.79	18.88	2.44	4.79

Figura 7.

A latência teve um tempo médio de 2,7 segundos no geral. O serviço 1 teve média de 2,1 segundos na qual a mínima foi de apenas 6ms e a máxima de quase 30 segundos. Entretanto, o tempo de resposta mais recorrente foi de 13ms.

Para o 2° serviço usando POST e inserção no banco houve uma média de 3,3 segundos, mínima de 3ms e máxima de 1min. Para este serviço, o tempo de resposta mais recorrente foi de 14ms. É possível perceber já que há uma grande variação de tempo de resposta dentre milhares de requisições.

O gráfico a seguir mostra o tempo de duração de teste em relação à média de tempo de resposta em ms dos dois serviços, lembrando que é esperado o aumento gradual devido ao ramp-up configurado.

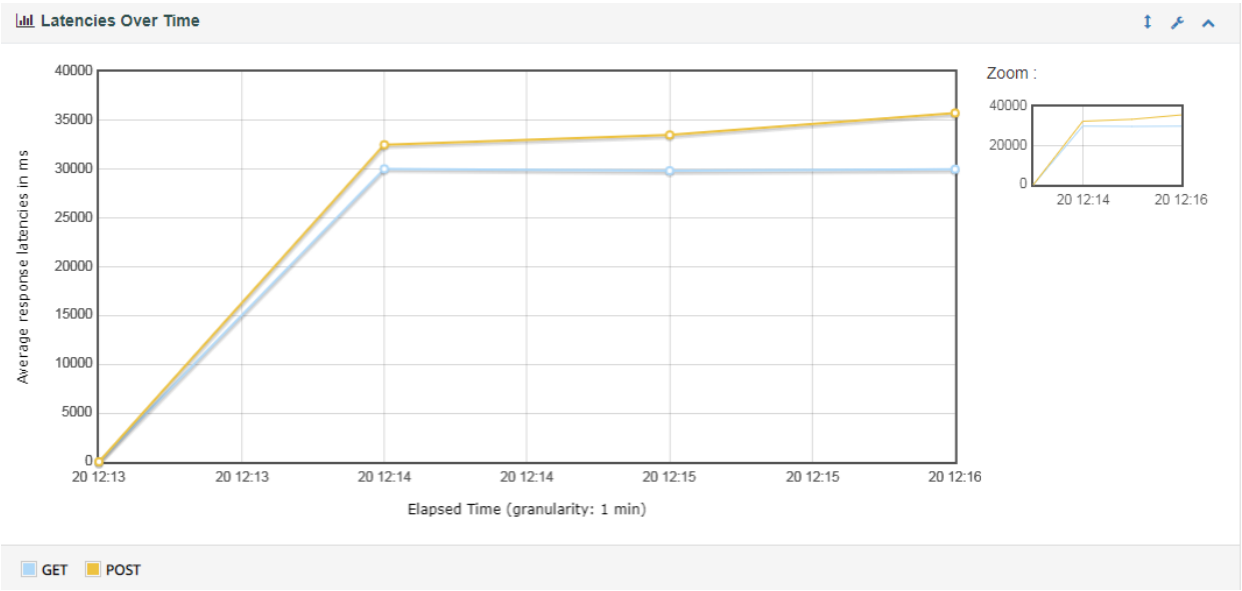


Figura 8.

O gráfico mostra o número total de requisições por segundo em relação à latência em ms incluindo sucessos e falhas.

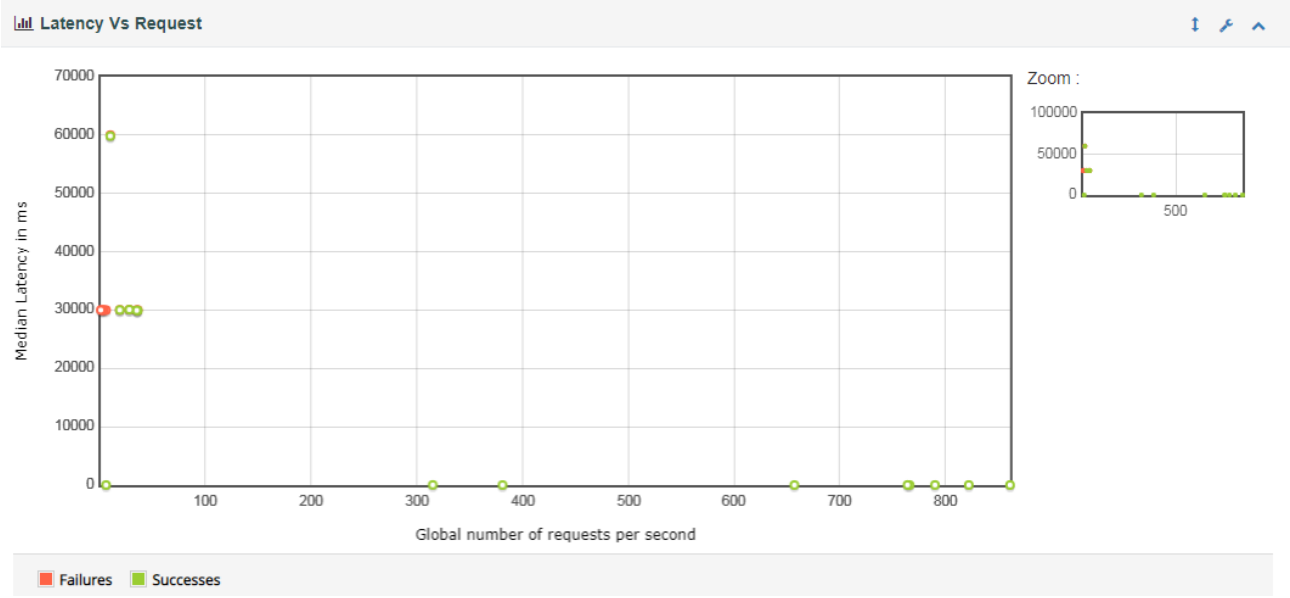


Figura 9.

Vazão:

O gráfico abaixo mostra o número de requisições por segundo pelo tempo médio de resposta incluindo falha e sucesso.

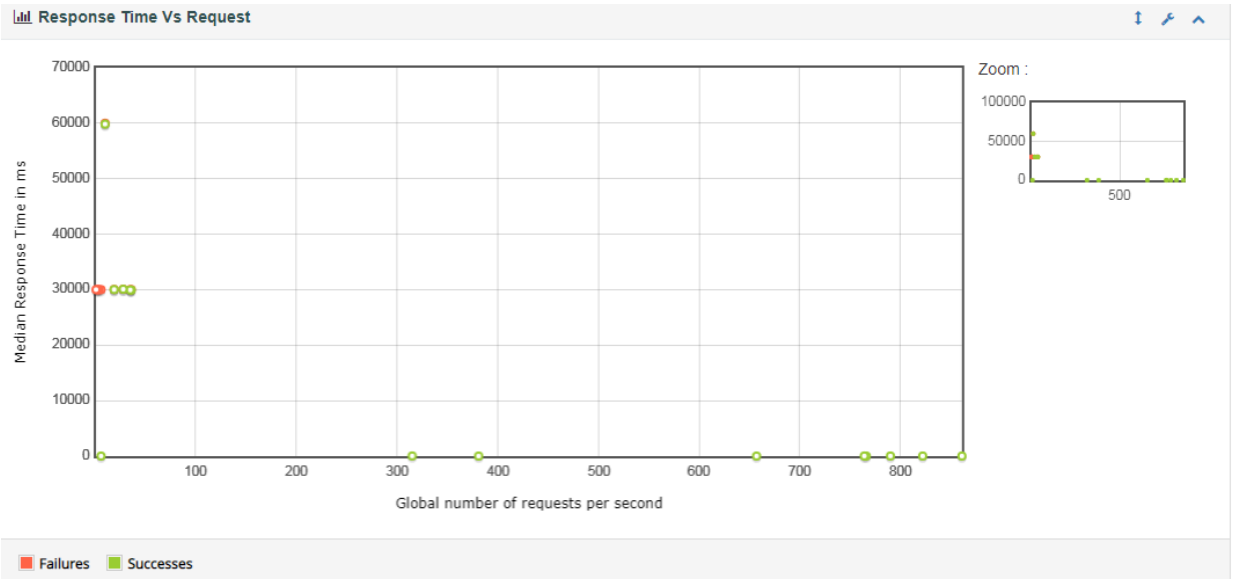


Figura 10.

O gráfico a seguir apresenta a vazão das requisições dos dois serviços, mostrando o tempo de execução do teste pelo número de transações realizadas por segundo.

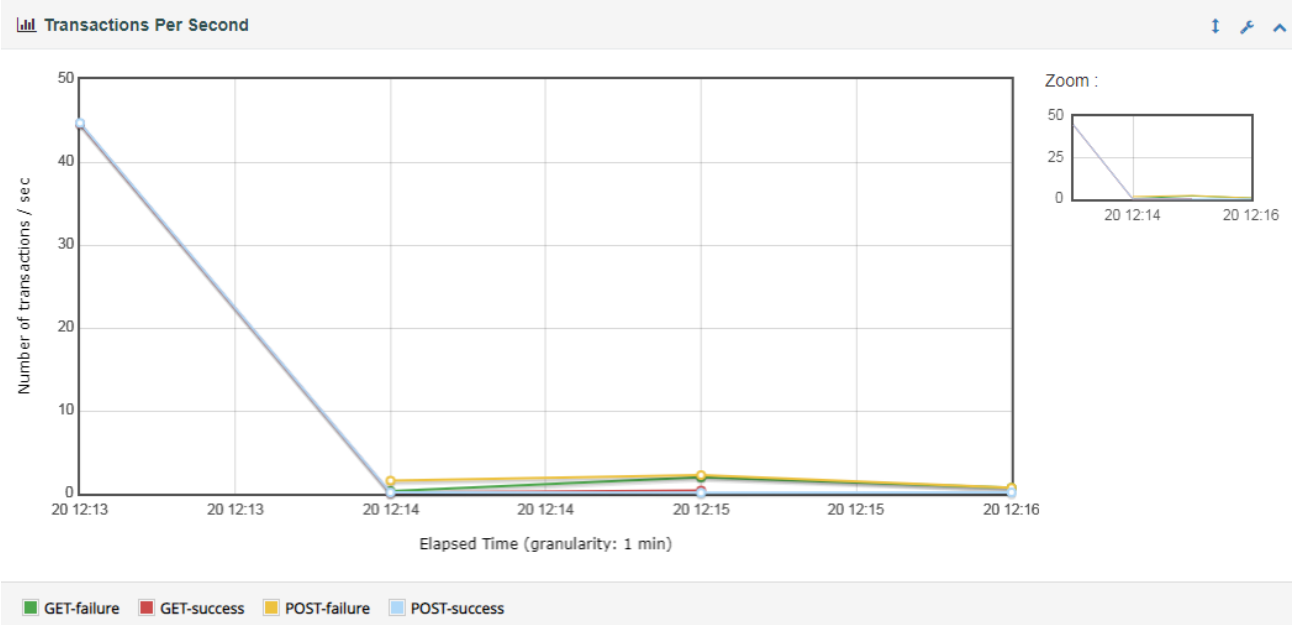


Figura 11.

Concorrência:

Não foi possível obter resultados para este item específico, no entanto, o gráfico a seguir mostra um resultado interessante em que mostra o número de requisições ativas em relação à média de tempo de resposta em ms.

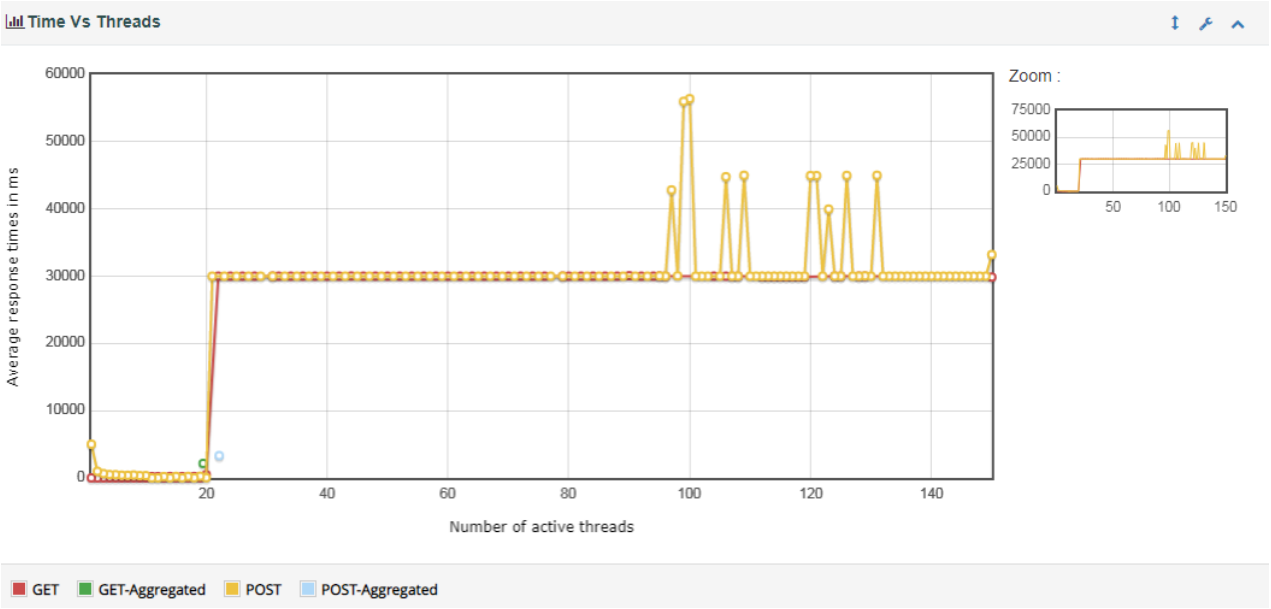


Figura 12.

O gráfico a seguir mostra a relação do tempo pelo número de requisições ativas.

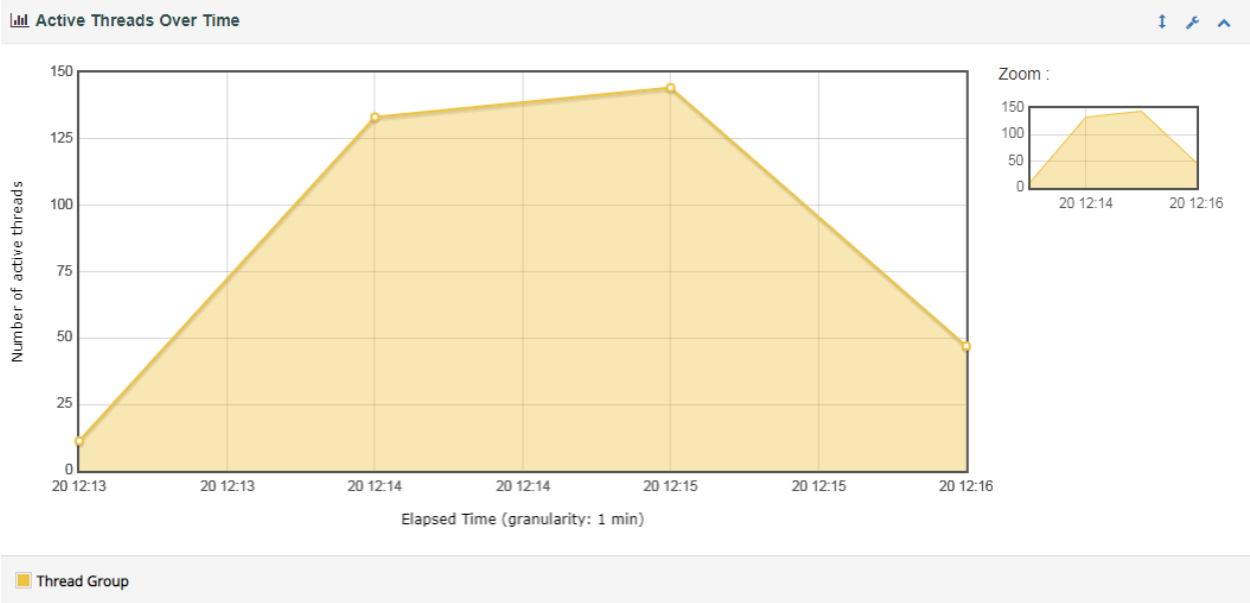


Figura 13.

- Potenciais gargalos do sistema

Como foi possível notar ao longo dos gráficos apresentados, o primeiro serviço que utiliza o endpoint GET teve um tempo de resposta médio menor que o segundo com POST, entretanto, conseguiu realizar menos requisições no tempo estipulado de 3 minutos. O acesso ao banco pode ser uma variável importante para este resultado. O segundo serviço teve mais falhas e levou mais tempo na latência, tal fato é como esperado dado que é necessário ler o dado, inserir na tabela, atualizar os índices, etc.

Há possibilidade de ter gargalo no sistema quando a requisição ao banco for necessária inserir dados mais robustos e também quando for necessário requisitar selects mais robustos. As falhas mostram que o sistema atende bem às demandas com muitas requisições, mas também apresenta instabilidade em alguns momentos.

2° Medição

- Melhorias/otimizações (listar arquivos modificados)
    - ProductController - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/controllers/ProductController.java
    - ProductRepository - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/repository/ProductRepository.java
    - ProductService - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/service/ProductService.java
    - Product - https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/model/Product.java
    - Application – https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/resources/application.yaml
    - KipaoApplication -

Serviços

- **Seleção de produto disponível**
- Tipo de operação: leitura
- Data de Medição: 31/07/2022

- **Adição de produto na lista de produtos disponíveis**
- Tipo de operação: inserção
- Data de Medição: 31/07/2022

- SLA:

Para esta segunda medição continuamos chamando os endpoints de busca dos produtos disponíveis e inserção de produto. A configuração também permaneceu para atender a 150 usuários (processos) por 1 minuto e que a cada segundo é adicionado mais 2 usuários. Esse aumento gradual tem o objetivo de fazer mais sentido ao resultado de teste visto que nem todos os usuários “requisitam” ao mesmo tempo. Como anteriormente foi, também, deixado 1 minuto de tempo em espera para verificar se o sistema pode lidar com a carga e se o desempenho permanece estável e não se deteriora.

- Vazão:

O gráfico abaixo mostra o número de requisições por segundo pelo tempo médio de resposta incluindo falha e sucesso.

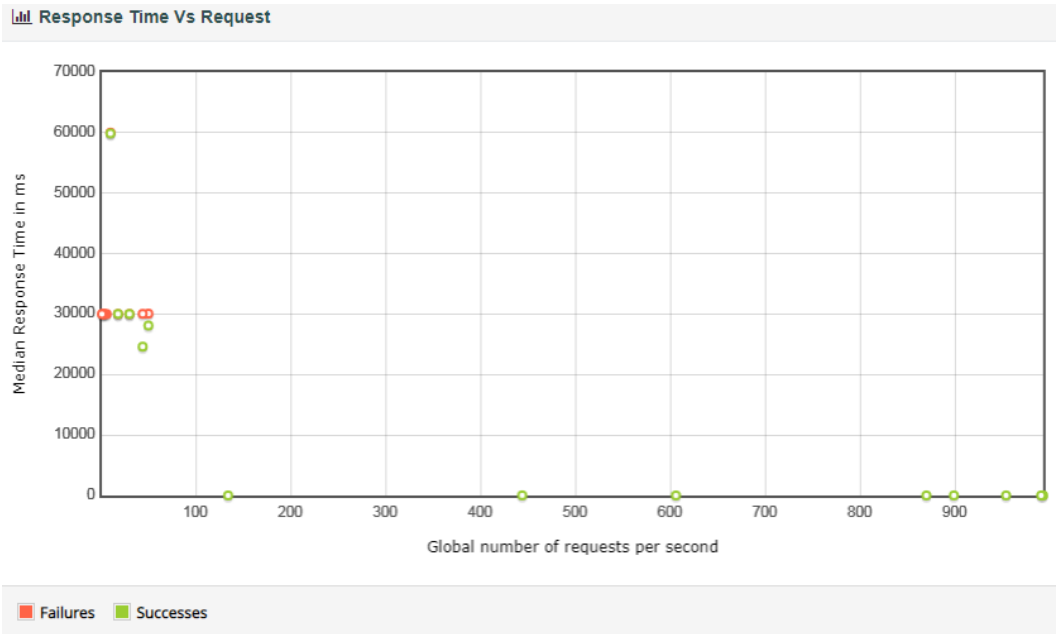


Figura 14.

O gráfico a seguir apresenta a vazão das requisições dos dois serviços, mostrando o tempo de execução do teste pelo número de transações realizadas por segundo.

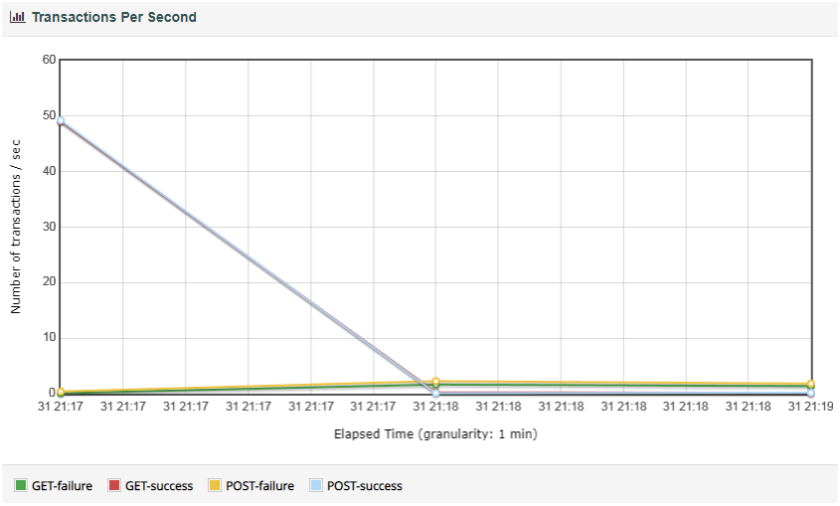


Figura 15.

- Latência

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**6390**	**453**	**7.09%**	**2504.16**	**3**	**60010**	**10.00**	**32.00**	**30005.00**	**30075.00**	**40.67**	**5.65**	**10.58**
GET	3150	184	5.84%	1973.28	4	30134	9.00	21.00	30005.00	30014.49	21.03	4.06	5.61
POST	3240	269	8.30%	3020.30	3	60010	12.00	42.00	30005.00	54551.02	20.62	1.78	5.24

Figura 16.

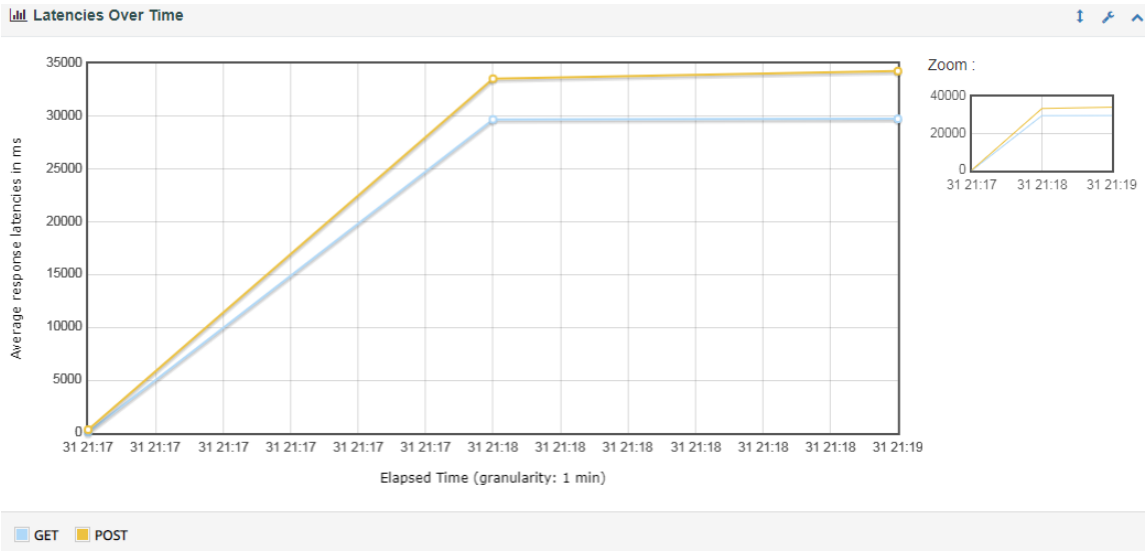


Figura 17.

No segundo teste de latência teve um tempo médio de 2,5 segundos no geral. O serviço 1 de leitura teve média de 1,9 segundos na qual a mínima foi de apenas 4 ms e a máxima, entretanto de 30 segundos. Entretanto, o tempo de resposta mais recorrente foi de 9ms.

Para o 2° serviço usando POST e inserção no banco houve uma média de 3 segundos, mínima de 3ms e máxima de 1min. Para este serviço, o tempo de resposta mais recorrente foi de 12ms. À medida que é demandada sempre novas requisições a média tende a crescer, porém chega num momento em que se estabiliza.

- Concorrência:

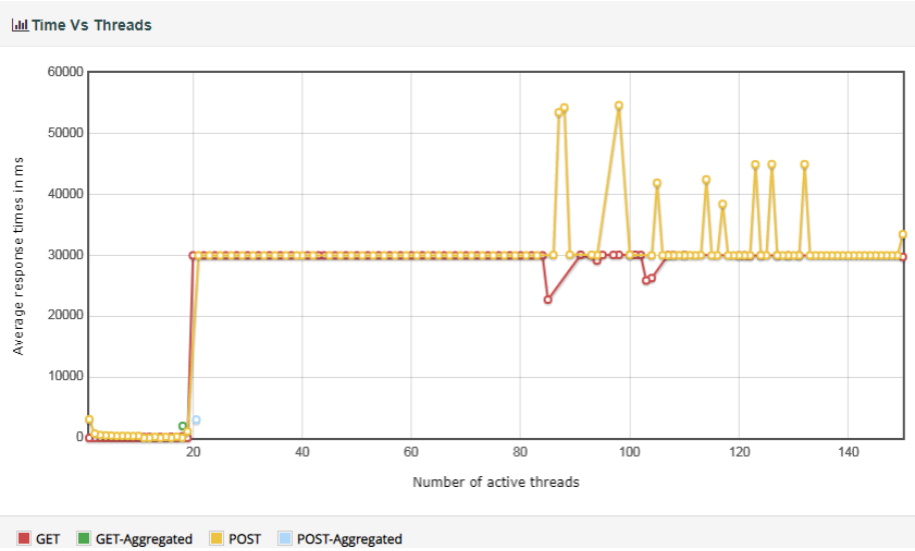


Figura 18.

A duração da medição que mostra o tempo de duração de teste em relação à média de tempo de resposta em ms dos dois serviços leva em conta que é esperado o aumento gradual de requisições devido ao ramp-up configurado.

Em vermelho mostra as requisições GET e em Amarelo as requisições POST. É possível observar uma estabilidade na maior parte do tempo entre ambos microsserviços e uma média de 3ms.

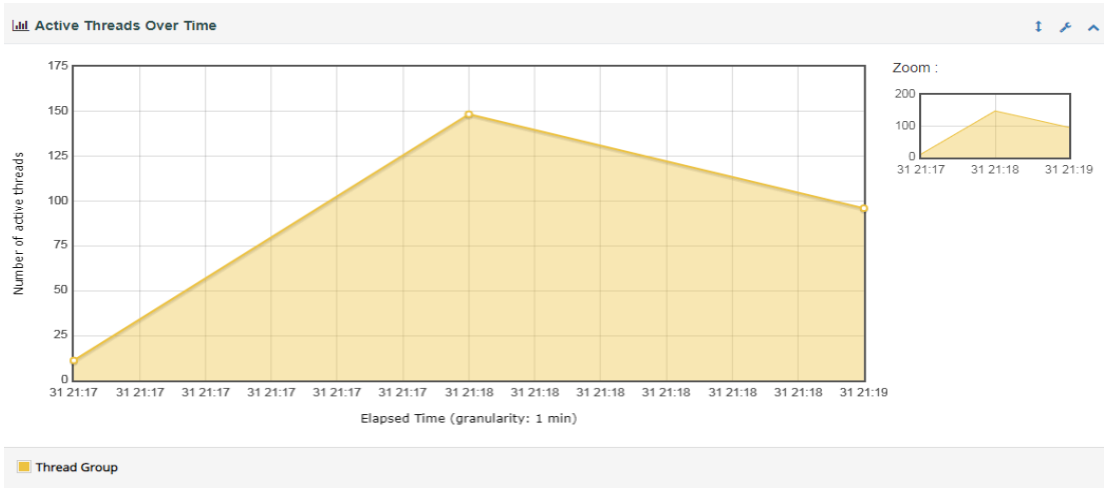
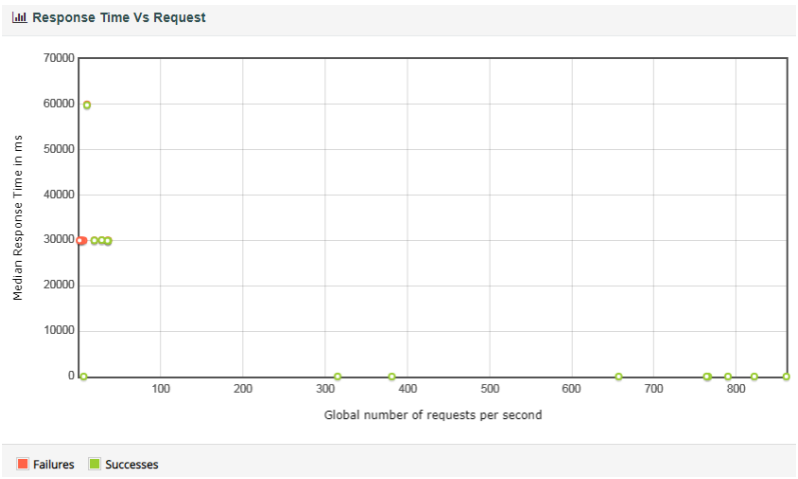
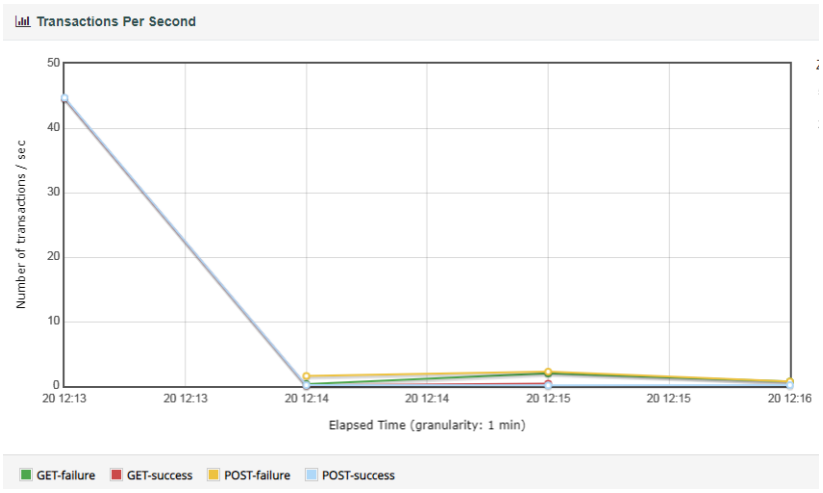


Figura 19.

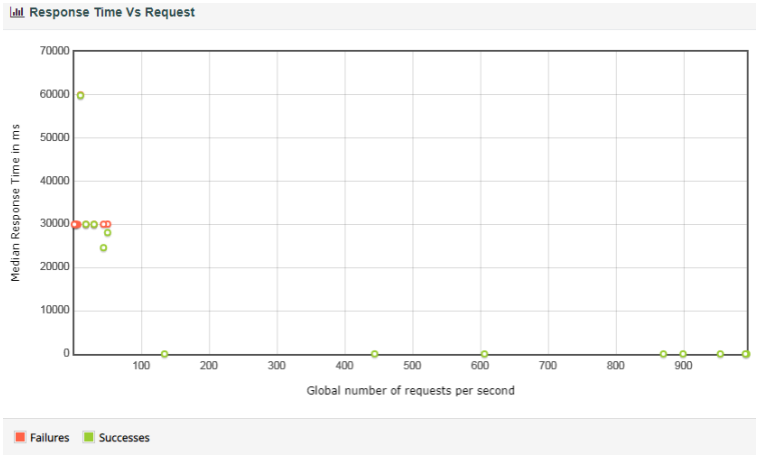
Número de threads ativos pelo tempo de execução da medição.

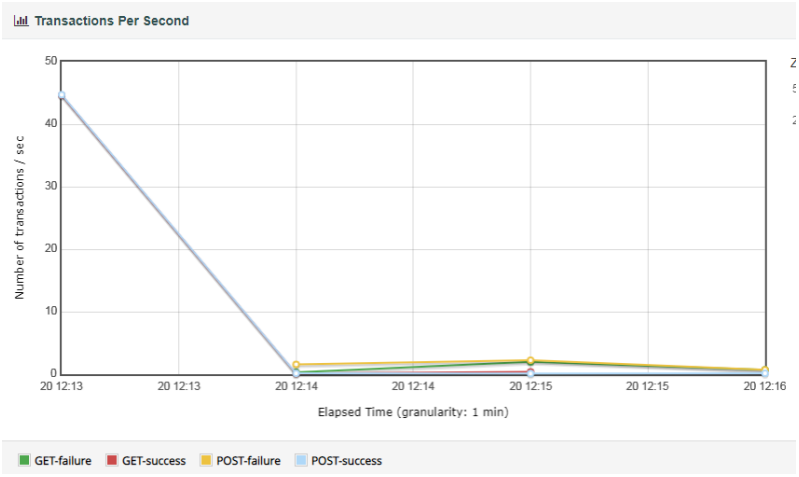
- GRÁFICOS comparativos das medições feitas
  - Vazão:





Vazão 1° medição.

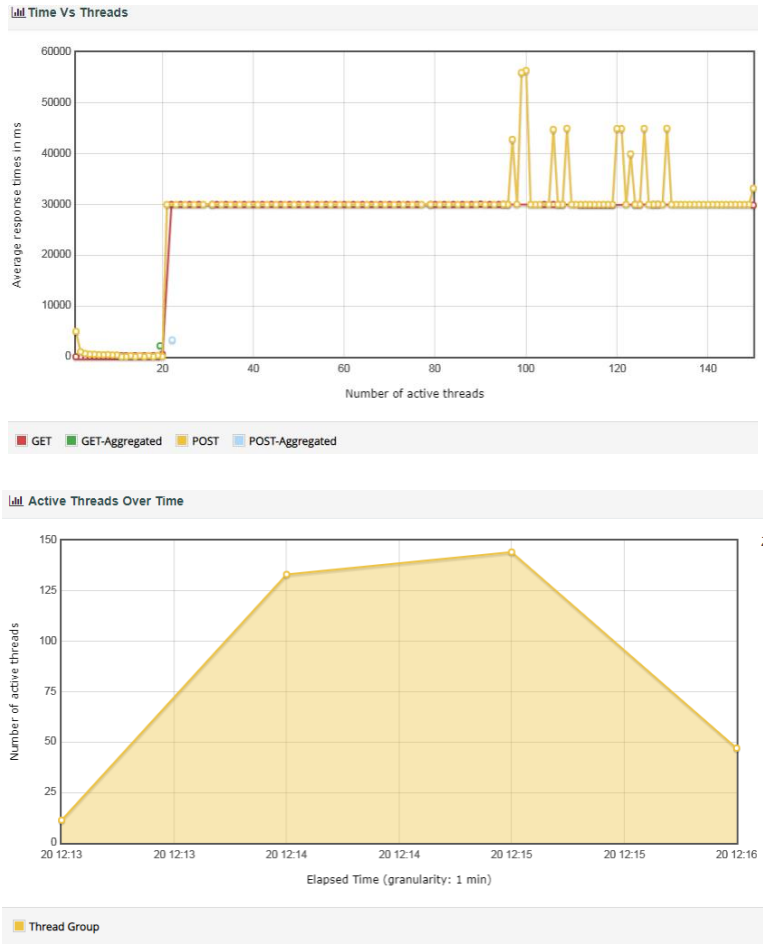




Vazão 2° medição.

A vazão entre as duas medições teve resultados bastante semelhantes, embora haja quase imperceptivelmente maior transação por parte da segunda medição. Neste gráfico, tanto o POST quanto o GET bem-sucedidos estiveram lado a lado, porém, a requisição GET foi executada mais rápido durante o teste e finalizada suas operações primeiro.

- Concorrência:



Concorrência 1° medição.

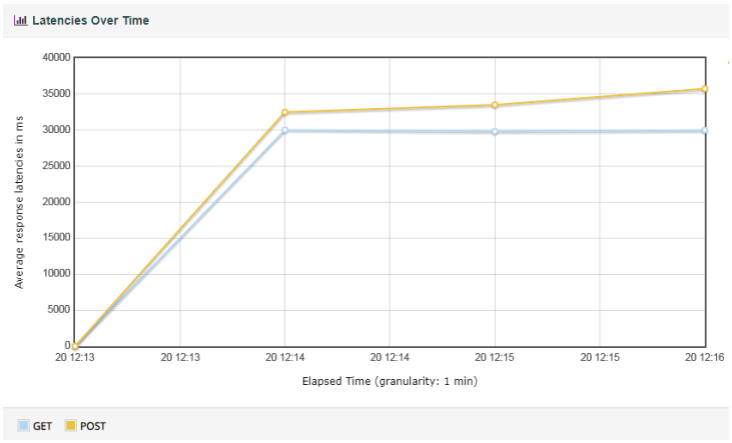
Concorrência 2° medição.

É normal haver picos e vales, o que deve ser levado em conta é se os resultados apresentados de fato se adequam ao comportamento esperado. O sistema conseguiu atender bem às demandas de mais de 6mil requisições em pouco tempo e sem desestabilizar o sistema, mesmo quando possuía mais de 140 threads ativos. Como nosso sistema é para uma padaria de poucas dimensões não haveria problema em manter este desempenho. No entanto, seria uma boa estratégia pensar na escalabilidade caso seja preciso futuramente.

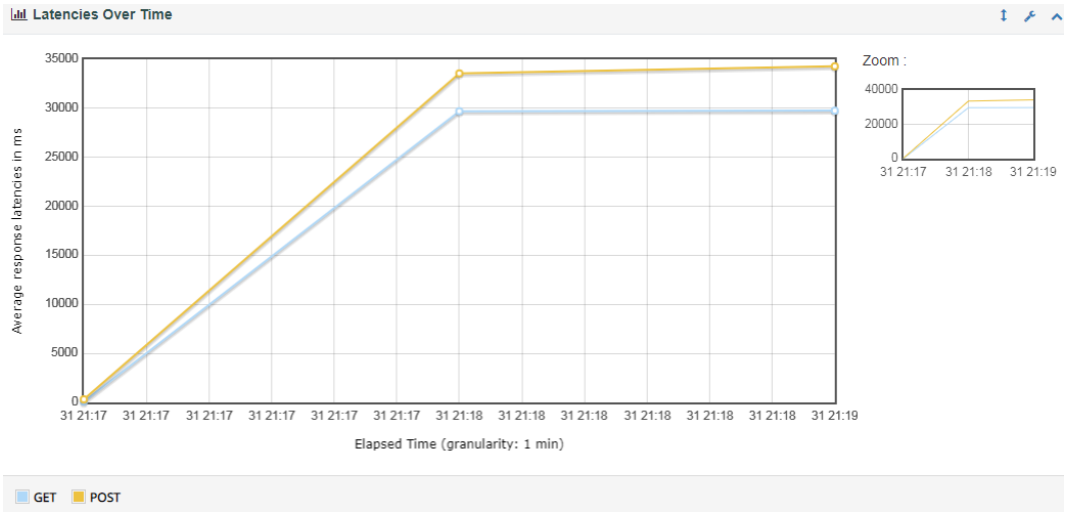
- Latência:

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**5862**	**449**	**7.66%**	**2746.94**	**3**	**60009**	**13.00**	**41.00**	**30005.00**	**30042.00**	**37.19**	**6.74**	**9.62**
GET	2886	179	6.20%	2158.26	6	30097	13.00	29.30	30005.00	30007.13	19.27	4.52	5.08
POST	2976	270	9.07%	3317.81	3	60009	14.00	65.30	30006.00	57698.79	18.88	2.44	4.79



Latência 1° medição.



Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**6390**	**453**	**7.09%**	**2504.16**	**3**	**60010**	**10.00**	**32.00**	**30005.00**	**30075.00**	**40.67**	**5.65**	**10.58**
GET	3150	184	5.84%	1973.28	4	30134	9.00	21.00	30005.00	30014.49	21.03	4.06	5.61
POST	3240	269	8.30%	3020.30	3	60010	12.00	42.00	30005.00	54551.02	20.62	1.78	5.24

Latência 2° medição.

O gráfico compara as duas medições de latência e pode-se observar que há uma melhora sutil na média do tempo de resposta. Pelas estatísticas das duas medições, houve maior quantidade de requisição pela mesma configuração que a anterior e menos erros de requisição, além de que, na segunda medição houve um melhor tempo (em ms) de resposta.

As mudanças implementadas partiram da estratégia de diminuir a necessidade de toda vez que houver uma requisição ele tenha que novamente abrir um acesso no banco. Para isso, a ideia de cache no framework Spring Boot auxilia na melhora final do status geral das requisições de maneira simples e sem necessidade de usar uma ferramenta mais robusta como um banco de dados em memória, como Redis, embora as boas práticas sugerem usar uma API Gateway para obter o máximo da performance e escalabilidade do

sistema. Para este simples projeto não é viável e também não há necessidade, porém é algo bastante interessante em projetos maiores.