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.**

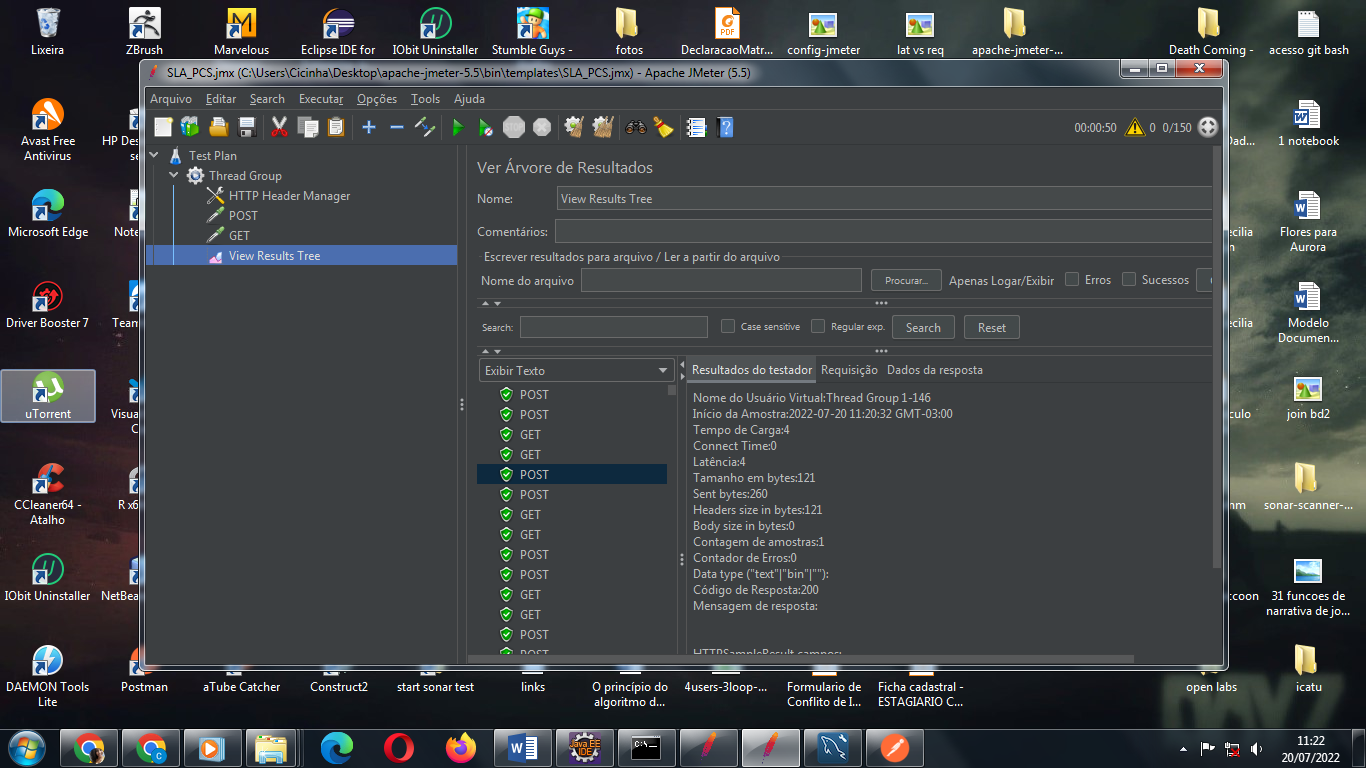
* **Seleção de produto disponível**
* Tipo de operação: leitura
* Arquivos envolvidos:
* ProductController - <https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/controllers/ProductController.java>
* ProductRepository - <https://github.com/LucasFPO/projeto-kipao/blob/main/src/main/java/br/unirio/kipao/repository/ProductRepository.ja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

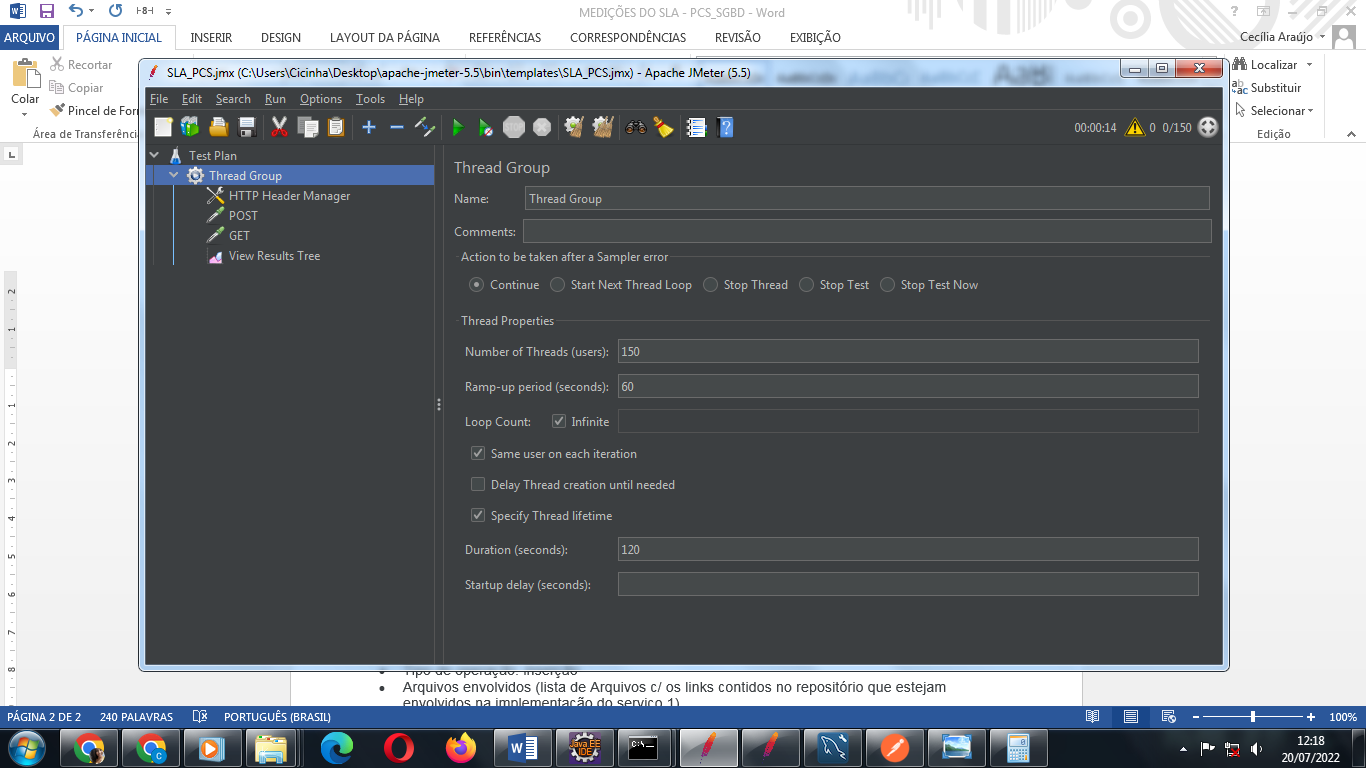
* Link da pasta dos resultados (arquivo index.html):

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

* Screenshot do programa:



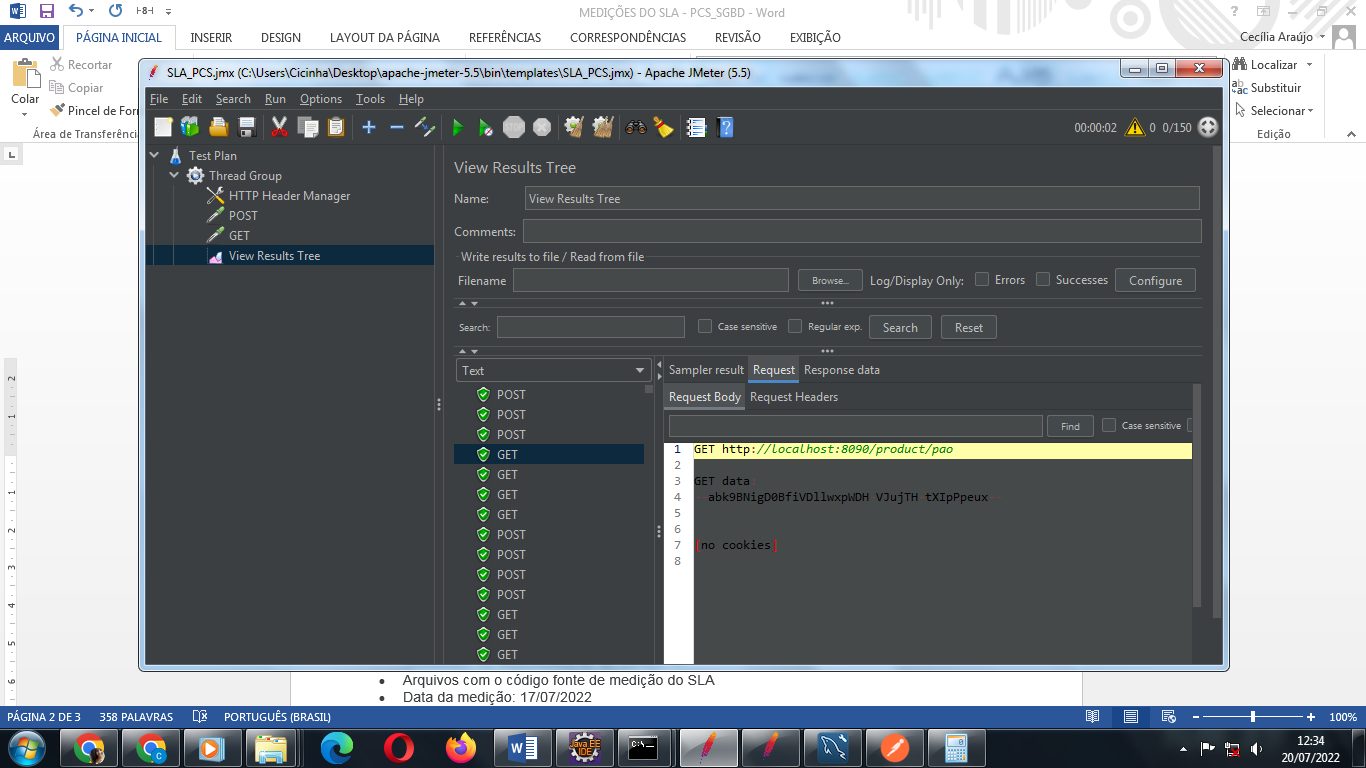
**Figura 1. Comportamento da execução do teste. Esquerda: plano de teste com duas requisições configuradas. Ao lado: requisições bem-sucedidas após chamada dos serviços. Direita: Resultado de uma das requisições com informações relevantes.**



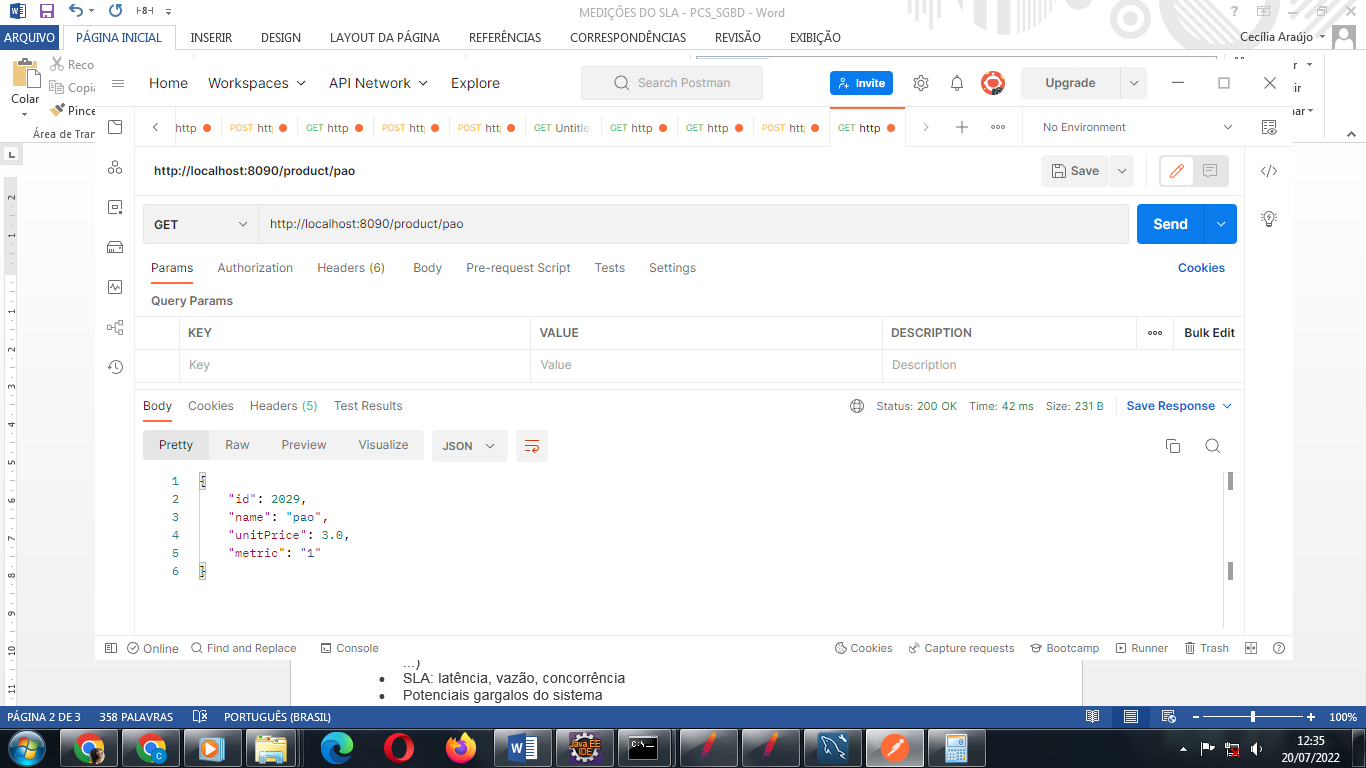
**Figura 2. Configuração de chamada de processo para ambos serviços.**

* 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.



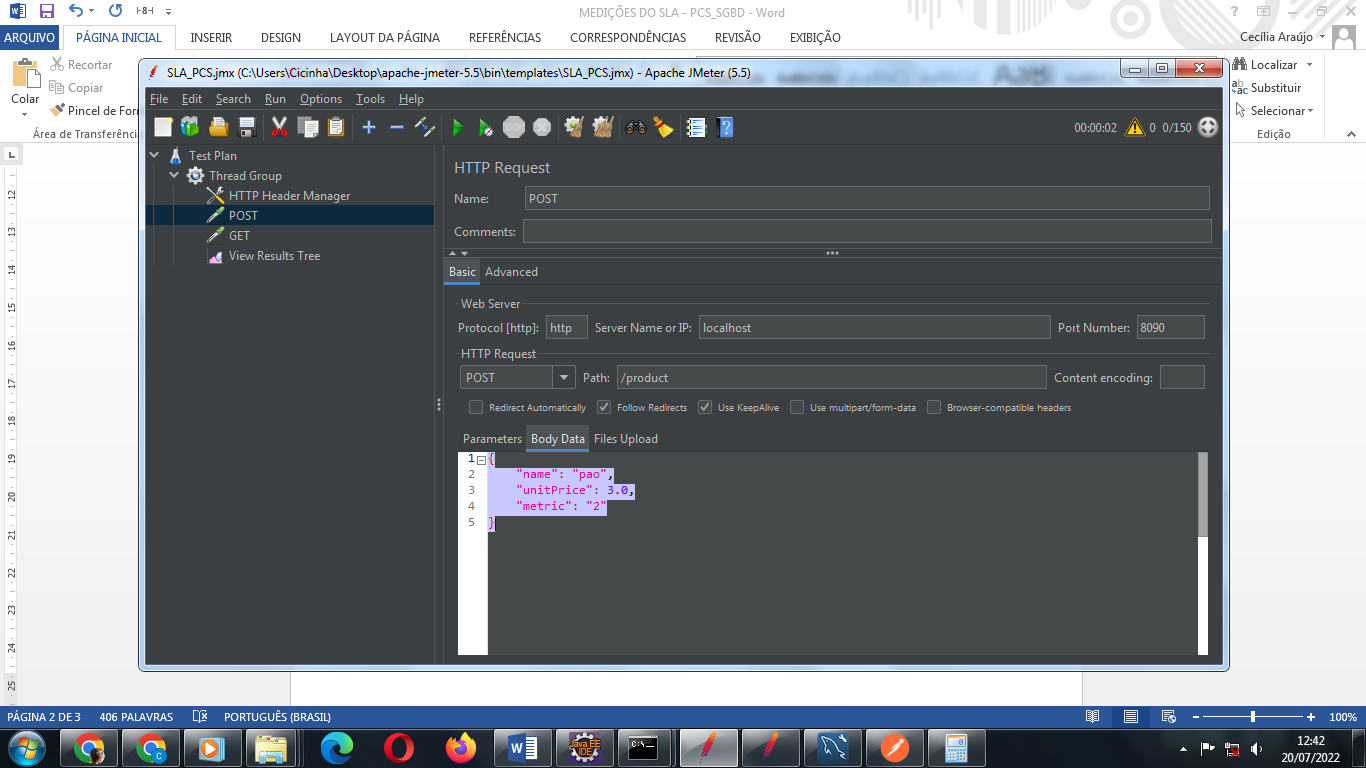
**Figura 3.**



**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:



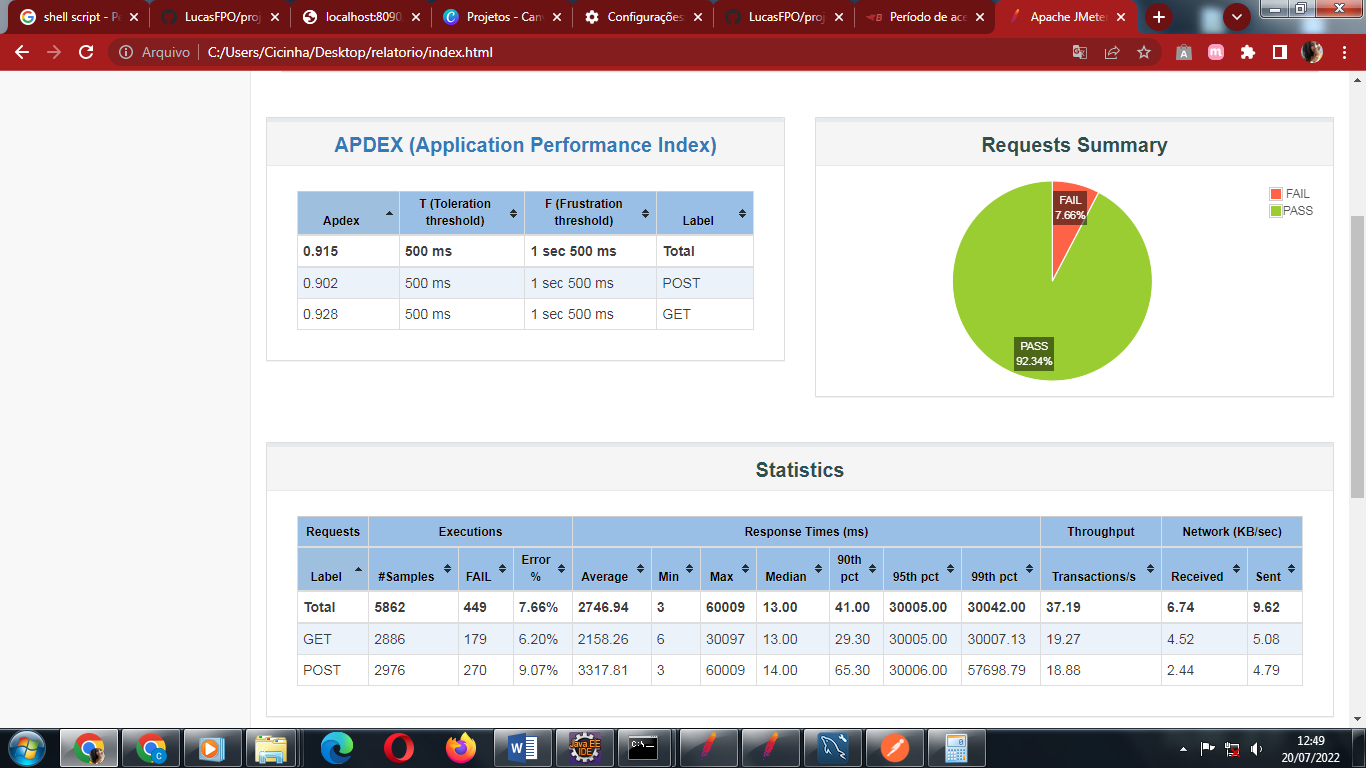
**Figura 5.**

**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.

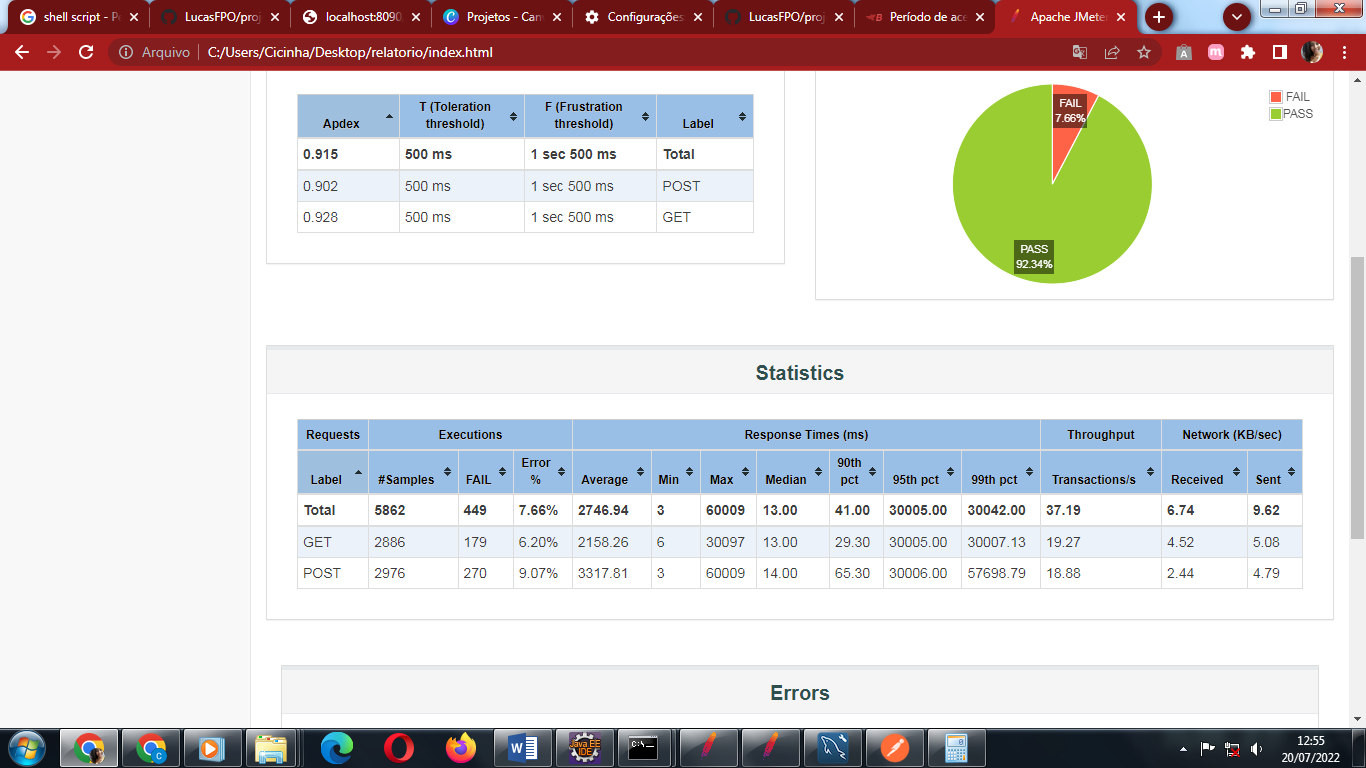
Nesse caso, temos os seguintes dados obtidos e que podem ser conferidos na pasta ‘Medicao\_SLA’:



**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:**



**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.

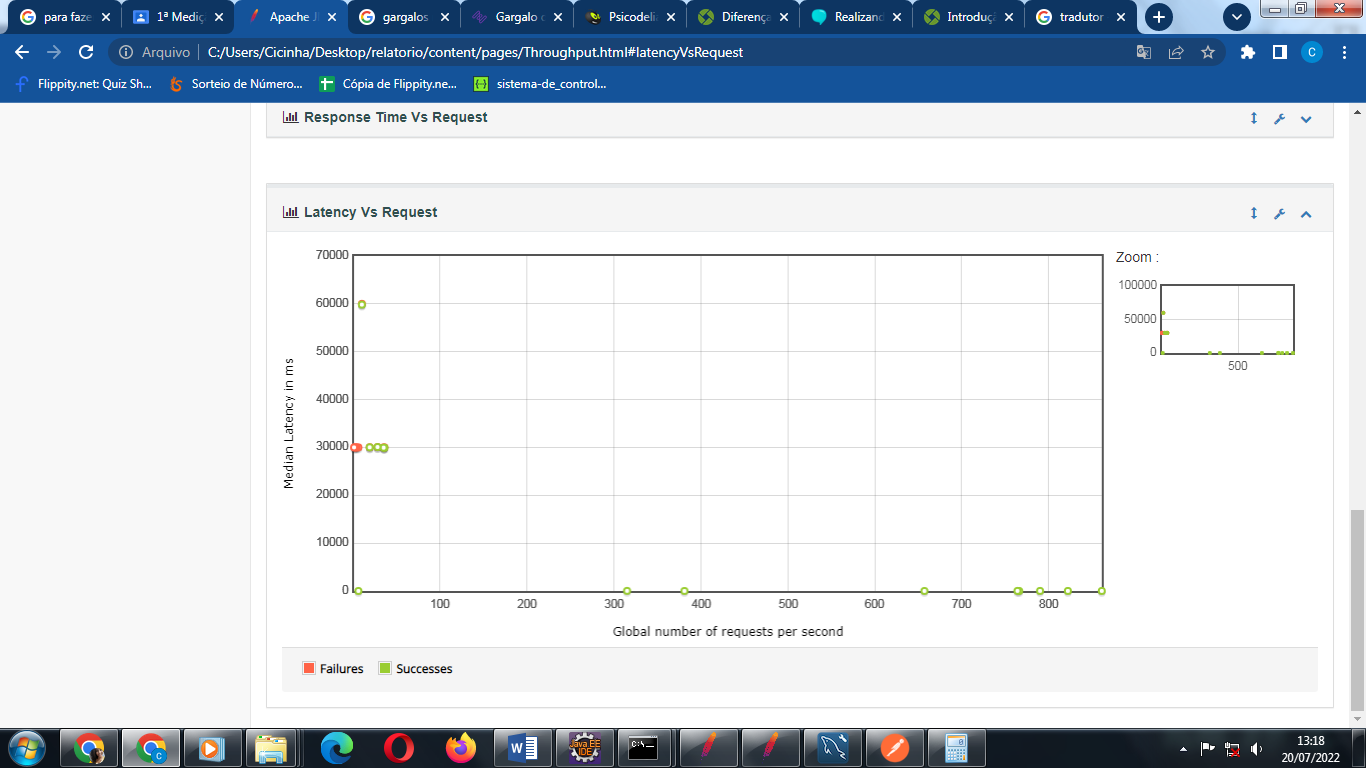
Para o 2° serviço usando POST e inserção no banco houve uma média de 3,3 segundos, mínima de 3ms e máxima de 1min. Para este serviço, o tempo de resposta mais recorrente foi de 14ms. É possível perceber já que há uma grande variação de tempo de resposta dentre milhares de requisições.

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.**

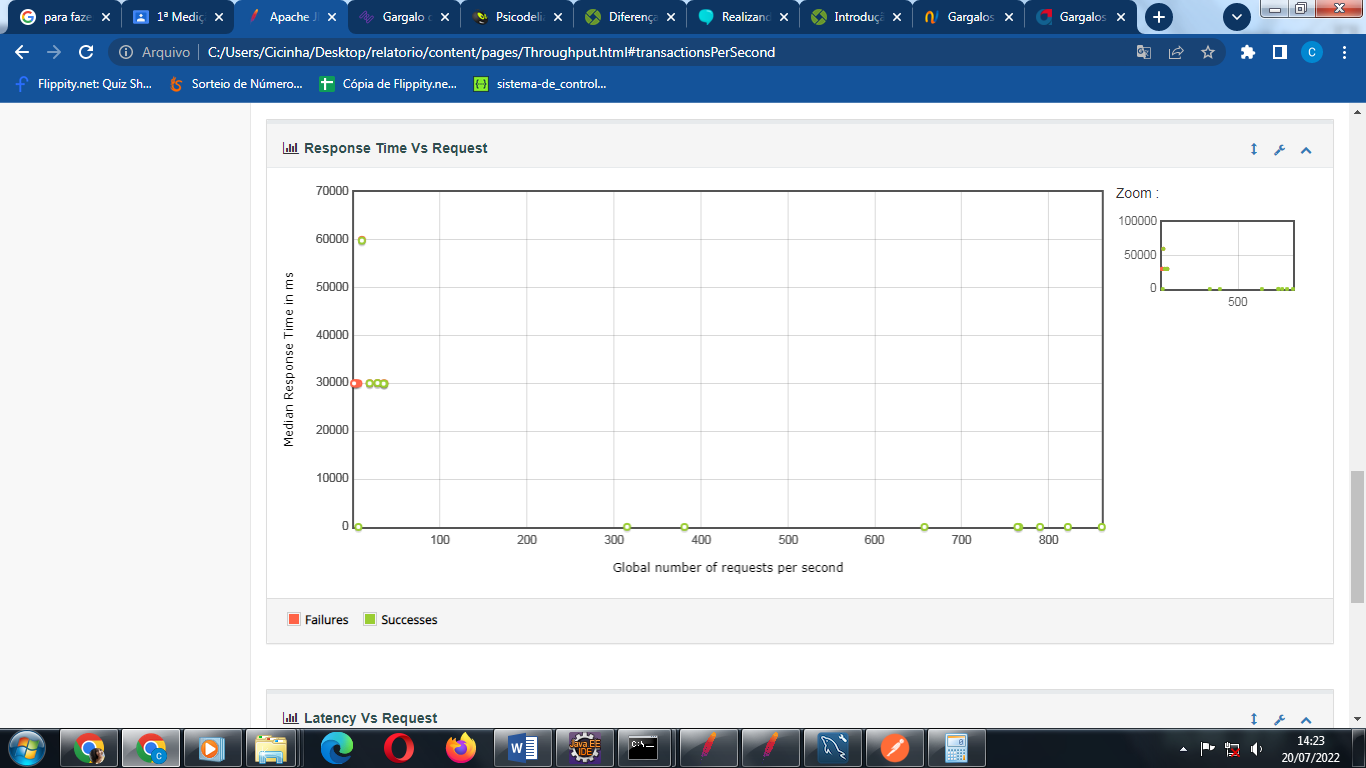
O gráfico mostra o número total de requisições por segundo em relação à latência em ms incluindo sucessos e falhas.



**Figura 9.**

**Vazão:**

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



**Figura 10.**

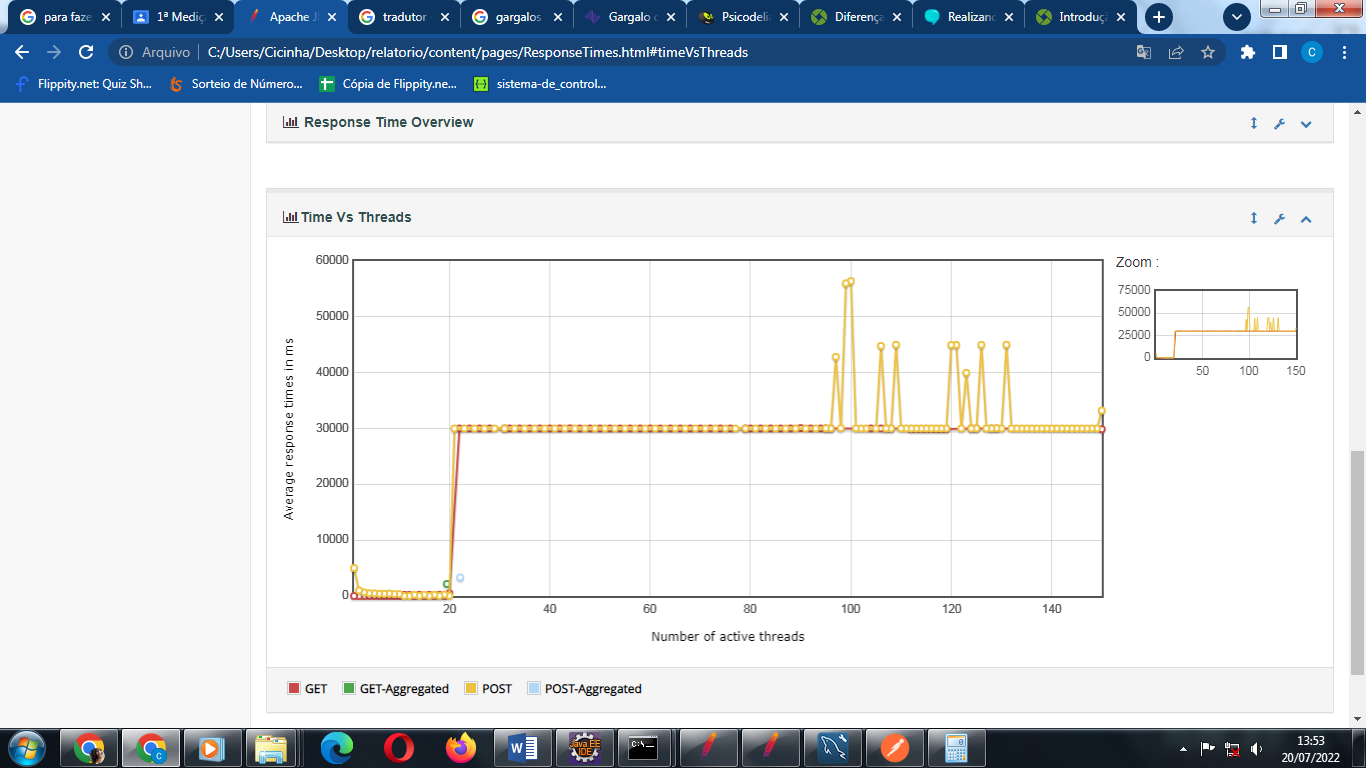
O gráfico a seguir apresenta a vazão das requisições dos dois serviços, mostrando o tempo de execução do teste pelo número de transações realizadas por segundo.



**Figura 11.**

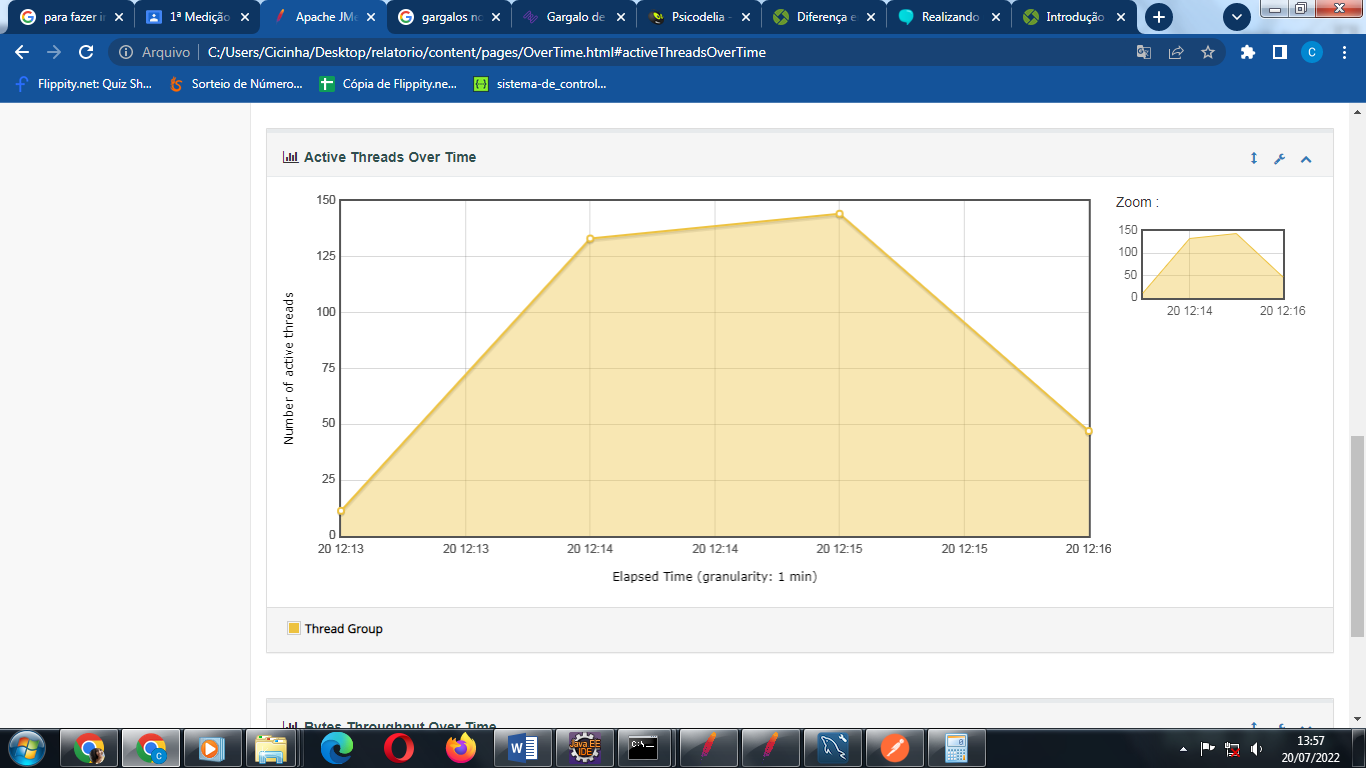
**Concorrência:**

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.

Para uma segunda medição após novas implementações é interessante aumentar o número de requisições por segundo em um teste de duração maior.