Laboratório 05

Eric Guimarães Caldas Jardim, Lucas Picinin Campos Lutti, Lucas Santos Rosa, Matheus Fontes Almeida Moreira Silva

[1] Instituto de Ciências Exatas e Informática
Pontifícia Universidade de Minas Gerais (PUC Minas)
Belo Horizonte – MG – Brasil

{1349194, 1254051, 1322257}@sga.pucminas.br

Resumo. *O trabalho desenvolvido consiste em analisar os benefícios e malefícios da adoção de um API GraphQL em detrimento da API REST. Nesse sentido, perguntas norteadoras serão levadas em conta para que a análise seja precisa, uma vez que questões como tempo de resposta, qualidade e esforço do desenvolvimento e necessidade de conhecimento técnico do desenvolvedor serão fatores considerados para relatório final dos ganhos e perdas da adoção do GraphQL.*

1. Introdução

Num contexto atual, onde o tempo para desenvolvimento tem se tornado cada vez mais valioso para grandes empresas e, principalmente, para desenvolvedores, as escolhas de tecnologias, frameworks, linguagens de programação, APIs, arquitetura, entre outros, são cada vez mais críticas e devem ser minuciosamente analisadas para que qualquer decisão seja tomada.

Nesse sentido, o artigo [C. Zhao et. al 2023] aborda a necessidade de estudar todos os efeitos da escolha de uma tecnologia para confecção de um sistema antes mesmo que o mesmo comece a ser implementado, visto que este fator é primordial para um software de qualidade.

Sob esse viés, surgiu a oportunidade de analisar as diferentes linguagens de consulta - GraphQL e REST. A primeira foi proposta pelo Facebook como alternativa para a outra, que é mais popular e utilizada no contexto de desenvolvimento. É fato que a combinação de ambas linguagens é utilizada em algumas empresas e essa junção pode ser totalmente benéfica. Contudo, ainda é preciso analisar quais são os verdadeiros ganhos com cada uma das APIs criadas baseadas em ambos modelos de requisição.

Deste modo, foi proposta uma atividade para que, baseado em algumas métricas escolhidas, o grupo pudesse comparar as linguagens de consulta GraphQL e REST a fim de entender quais são as vantagens e desvantagens da escolha de cada uma delas. Outrossim, perguntas norteadoras pré-definidas serão um alicerce para o desenvolvimento do projeto. São elas:

i) Respostas à consultas GraphQL são mais rápidas que respostas à consultas REST?

Métrica - tempo médio de um conjunto de requisições em cada uma das linguagens de consultas

Hipótese informal - Respostas a requisições GraphQL tendem a ter um tempo menor de espera em relação ao outro modelo de consulta (REST).

ii) Respostas à consultas GraphQL tem tamanho menor que respostas à consultas REST?

Métrica - LOC médio de requisições com o mesmo intuito em cada uma das linguagens de consulta

Hipótese informal - Respostas a requisições GraphQL tendem a ter um tamanho menor em relação ao outro modelo de consulta (REST).

Para responder a essas perguntas, este artigo será estruturado da seguinte forma: inicialmente, na Seção 2, será apresentada toda a metodologia utilizada durante o experimento, permitindo a visualização de todas as métricas e modelos de consulta utilizados para o entendimento e resposta das perguntas iniciais. Em seguida, na Seção 3, todos os resultados extraídos a partir do desenvolvimento da metodologia serão apresentados, assim como todas as hipóteses informais apresentadas na introdução serão discutidas, permitindo pontuar se foram acertadas ou não. E, por fim, a Seção 4 é onde haverá a dissertação e conclusão de todo o projeto, permitindo, também, pontuar trabalhos futuros que possam corroborar para a pesquisa desenvolvida no artigo.

2. Metodologia

A metodologia utilizada neste estudo tem como abordagem o desenvolvimento de consultas utilizando REST e GraphQL para a mesma API, que será do GitHub, sendo que haverão diferentes níveis de complexidade dessas consultas, onde existirão 500 diferentes tipos de requisições para que possa haver uma robustez estatística. Posteriormente, haverá a coleta de dados e a discussão dos resultados obtidos.

Com o intuito de alcançar os objetivos definidos, este estudo avalia duas questões de pesquisa:

RQ1: Respostas à consultas GraphQL são mais rápidas que respostas à consultas REST?

RQ2: Respostas à consultas GraphQL tem tamanho menor que respostas à consultas REST?

A RQ1 visa entender a velocidade das duas linguagens de consulta. Enquanto a hipótese nula afirma que o modelo de requisição não afeta significativamente o tempo de resposta, a hipótese alternativa tem como premissa o fato de que há uma grande diferença de tempo de resposta entre os dois modelos.

Já a RQ2 tem como intuito analisar o tamanho das respostas provenientes de cada uma das linguagens de consulta. Enquanto a hipótese nula afirma que não há uma diferença exacerbada entre os dois modelos, a hipótese alternativa defende que o tipo de consulta influencia bastante no tamanho da resposta.

A complexidade das requisições será separada em três tipos: baixa, média e alta, conforme exemplificado abaixo:

- Baixa - GraphQL

```
'low': """
      {
        user(login: "octocat") {
          repositories(first: 10) {
            nodes {
              name
            }
          }
        }
      }
    """
```

- Média - GraphQL

```
'medium': """
      {
        user(login: "octocat") {
          repositories(first: 10) {
            nodes {
              name
              owner {
                login
              }
            }
          }
        }
      }
    """
```

- Alta - GraphQL

```
'high': """
      {
        user(login: "octocat") {
          repositories(first: 10) {
            nodes {
```

```
            name
            owner {
              login
            }
            description
            stargazerCount
            forkCount
          }
        }
      }
    }
  """
```

- Baixa - REST = https://api.github.com/users/octocat/repos?per_page=10
- Média - REST = https://api.github.com/users/octocat/repos?per_page=100
- Alta - REST = https://api.github.com/users/octocat/repos?per_page=250

Essas queries estão baseadas no repositório de um único usuário, cujo login é “octocat”, onde, de acordo com a complexidade, cada vez mais dados são coletados de um certo repositório.

2.1 Coleta de dados

A coleta de dados deste estudo baseia-se na confecção de um número pré-determinado de chamadas para a API do GitHub, sendo que haverá a mesma quantidade de requisições REST e GraphQL, além de mesclar o nível de complexidade e volume de resposta entre as linguagens de consulta. Foi extraído um conjunto de repositórios do GitHub, onde as requisições podem ser feitas para capturar diferentes tipos de dados em cada um deles, como os títulos dos projetos, autores, linguagens de programação, números de estrelas, entre outros.

2.2. Normalizar e tratar dados

Após a coleta dos dados, será realizada uma etapa de normalização e tratamento. Isso inclui a organização dos dados, a redução de redundâncias e inconsistências, a padronização dos formatos, a remoção de valores ausentes e a transformação dos dados em formatos compatíveis para análise posterior. Essa etapa é fundamental para garantir a qualidade e a consistência dos dados utilizados.

2.3. Calcular dados auxiliares a partir dos obtidos anteriormente

Com os dados coletados e tratados, serão calculados dados auxiliares para fornecer insights adicionais. Isso inclui métricas como tempo médio de resposta, tempo médio de espera, tamanho de cada requisição e respostas. Essas métricas foram calculadas com base nos critérios estabelecidos na metodologia.

2.4. Analisar dados obtidos

Para melhor visualização, os resultados de medições serão exportados para uma planilha, onde será possível o cálculo de médias e desvios padrões, assim como a realização de testes estatísticos simples, como o teste t.

3. Resultados

3.1 Caracterização do dataset

A visualização dos dados será feita através da ferramenta PowerBi. Após a execução das 1000 requisições, sendo 500 em GraphQL e 500 em REST, um "dataframe" será criado, utilizando a linguagem Python, a fim de facilitar a importação e visualização dos dados no PowerBi.

Nesse sentido, serão tratadas, através da biblioteca , todos os valores obtidos na primeira mineração, sendo que a mesma retorna a complexidade, o tempo de resposta e o tamanho de respostas de cada uma das requisições feitas. Um novo arquivo .csv contendo a média de cada um desses dados será gerada, e a visualização dos dados será baseada no mesmo.

3.2 GQM

3.2.1 i) Respostas à consultas GraphQL são mais rápidas que respostas à consultas REST?

Para analisar a velocidade das requisições e comparar os ganhos em relação a GraphQL e REST, foi adotada a métrica de tempo médio de resposta para cada uma das requisições, incluindo complexidade baixa, média e alta.

Conforme ilustrado na Figura 1, o tempo médio de resposta de todas as requisições ficou em 0,79 segundos para chamadas GraphQL, enquanto para REST o tempo médio foi de 0,83 segundos, englobando todas as complexidades. Nesse contexto, pode-se ainda afirmar que de todo o tempo gasto para realização das requisições, 48,56% foi destinado a chamadas GraphQL, enquanto 51,44% para REST.

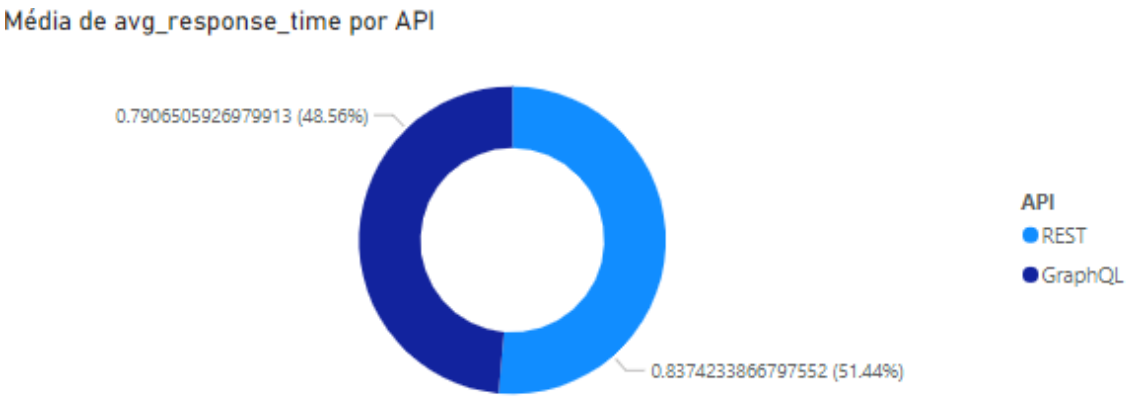
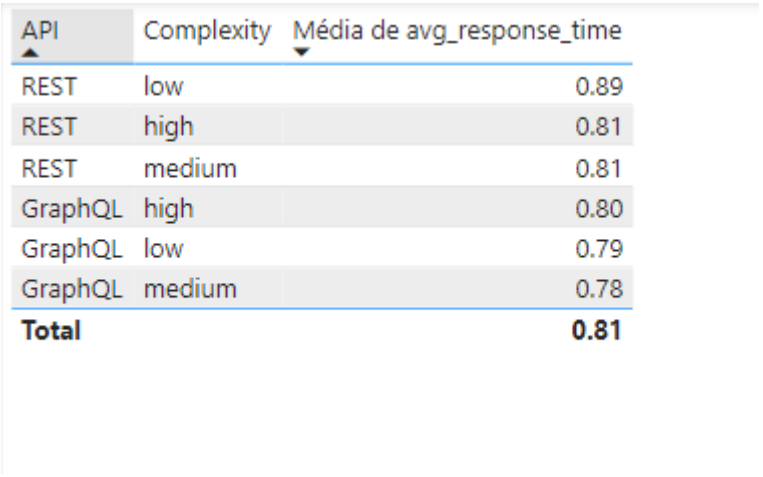


Figura 1. Tempo médio de resposta x Tipo de Requisição

Outrossim, outro fato perceptível na análise dessa questão de pesquisa foi o fato de que a tendência de similaridade se manteve de modo independente em relação ao tipo de complexidade, uma vez que não houve discrepância de tempo quando uma requisição fosse de complexidade simples ou alta, conforme demonstrado na Figura 2.



API	Complexity	Média de avg_response_time
REST	low	0.89
REST	high	0.81
REST	medium	0.81
GraphQL	high	0.80
GraphQL	low	0.79
GraphQL	medium	0.78
Total		**0.81**

Figura 2. API x Complexidade x Tempo médio de resposta

Em conclusão, é possível pontuar que as requisições em GraphQL foram levemente mais rápidas que as REST, mas esse tempo é quase imperceptível no estudo analisado, fazendo com que, nos cenários apresentados, não hajam grandes vantagens se for levado em conta apenas esse aspecto.

3.2.2 ii) Respostas à consultas GraphQL tem tamanho menor que respostas à consultas REST?

Para responder a essa pergunta, também decidiu-se por utilizar a métrica de média, nesse caso sendo o tamanho de resposta o aspecto a ser comparado. Sob esse viés, também foram coletados, junto ao tempo médio de resposta, esse novo dado, permitindo a visualização do mesmo através de sua normalização.

Nesse ponto, pode-se concluir que, num modo geral, as requisições em GraphQL tem um tamanho significativamente menor - em nossa base de testes - quando comparado a API REST. Esta discrepância pode ser vista na Figura 3, onde pode ser vista a média do tamanho de todas as respostas em ambas as linguagens de consulta, independente da complexidade.

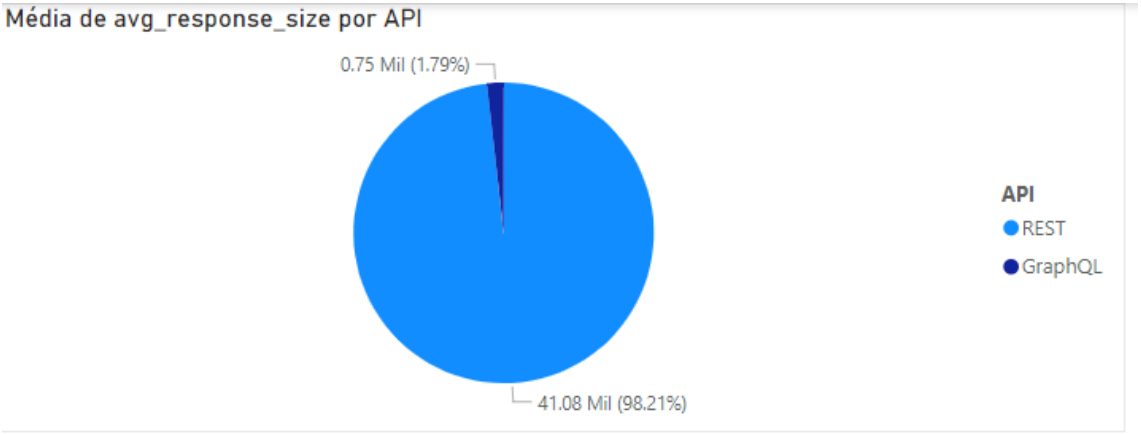


Figura 3. Tamanho médio de resposta x API

Ademais, outro ponto a ser analisado é a diferença crescente de acordo com a complexidade da chamada. Nas requisições feitas, as que utilizavam o modelo de consulta REST sempre tiveram o mesmo tamanho, sendo o mesmo muito maior quando comparado às consultas GraphQL. O tamanho de 41079 bytes se manteve inalterável durante todos os tipos de complexidade, enquanto o tamanho das respostas para GraphQL foi volátil.

Nesse âmbito, é importante ressaltar que, uma vez que de acordo com a complexidade o tamanho da resposta das consultas GraphQL varia, conforme visto na Figura 4, também é possível concluir que quanto menos complexa é a requisição, maior a diferença de tamanho de resposta entre ambos os tipos de linguagens de consulta.

API	Complexity	Soma de avg_response_size
GraphQL	high	1454
GraphQL	low	265
GraphQL	medium	529
REST	high	41079
REST	low	41079
REST	medium	41079
Total		**125485**

Figura 4. API x Complexidade x Tamanho médio de resposta

3.3 Comentários das hipóteses informais

3.3.1 Respostas a requisições GraphQL tendem a ter um tempo menor de espera em relação ao outro modelo de consulta (REST) - RQ 1.

Conforme analisado através da métrica de tempo médio de resposta para cada requisição, tanto em REST quanto em GraphQL, o tempo média de espera não foi um fator de grande discrepância entre os dois tipos de consulta.

Nesse sentido, pode-se concluir que a hipótese informal inicial não está completamente errada, uma vez que consultas da linguagem GraphQL tem um tempo menor de espera, mas não é um ponto, de acordo com nossa análise, que pode ser altamente considerado, levando em conta a proximidade dos dados.

3.3.2 Respostas a requisições GraphQL tendem a ter um tamanho menor em relação ao outro modelo de consulta (REST).

Essa hipótese informal levantada no início do estudo pode ser considerada como correta, uma vez que o tamanho médio das respostas de requisição REST foi consideravelmente maior quando comparada àquelas em GraphQL, independente da complexidade do caso.

4. Conclusões e trabalhos futuros

Para concluir, torna-se importante pontuar todos os passo a passo do estudo, visto que foram minuciosamente escolhidas para que houvesse uma quantidade de dados suficiente para haver uma base de comparação robusta, onde pudesse haver uma análise de qualidade envolvendo os tratamentos em GraphQL e REST.

Por um lado, percebeu-se que as requisições GraphQL, quando comparadas às de modelo REST, tem um tempo médio de respostas de consulta relativamente igual, não havendo grandes vantagens ou desvantagens quando este é o único ponto considerado. Por outro lado, quando o tamanho médio de resposta é discutido, a linguagem de consulta desenvolvida pelo Facebook tem uma grande vantagem, já que, independente da complexidade do caso, o tamanho da resposta, medido em bytes, é menor.

Nesse âmbito, as requisições em GraphQL são mais vantajosas por terem um tamanho menor em bytes pelos seguintes fatores: menor latência, menos consumo de banda, custo de armazenamento reduzido, carregamento de dados mais rápido, menor consumo de energia e aumento do número de usuários com acesso simultâneo.

Por fim, entende-se que, para o futuro, seria viável realizar uma pesquisa que abrangesse um número muito maior de requisições, onde apenas um computador com capacidades maiores pudesse executar os scripts necessários, a fim de haver uma quantidade mais robusta de dados. Este aspecto seria importante pois foi analisada uma tendência de crescimento no tamanho das consultas GraphQL de acordo com a complexidade e, então, seria interessante analisar se em algum caso esse tamanho poderia ultrapassar as respostas das consultas REST de acordo com o nível de complexidade.

Referências

C. Zhao, Y. Wang and J. Jiang, "Application of Software Engineering Technology in System Software Development," *2023 Asia-Europe Conference on Electronics, Data Processing and Informatics (ACEDPI)*, Prague, Czech Republic, 2023, pp. 502-506, doi: 10.1109/ACEDPI58926.2023.00101.