Pontifícia Universidade Católica de Minas Gerais

Laboratório de Experimentação de Software

GraphQL vs REST: Um Experimento Controlado

Ian Asenjo Dominguez Cunha

Jully Ketely Alves da Silva

Warley Leandro dos Anjos

Belo Horizonte
2023

INTRODUÇÃO

A linguagem de consulta GraphQL, proposta pelo Facebook como metodologia de implementação de APIs Web, representa uma alternativa aos populares APIs REST. Baseada em grafos, a linguagem permite que usuários consultem banco de dados na forma de schemas, de modo que se possa exportar a base e realizar consultas num formato definido pelo fornecedor da API. Por outro lado, APIs criados com base em abordagens REST baseiam-se em endpoints: operações pré-definidas que podem ser chamadas por clientes que desejam consultar, deletar, atualizar ou escrever um dado na base. Desde o seu surgimento, vários sistemas realizaram a migração entre ambas as soluções, mantendo soluções compatíveis REST, mas oferecendo os benefícios da nova linguagem de consulta proposta. Entretanto, não está claro quais os reais benefícios da adoção de uma API QraphQL em detrimento de uma API REST. Nesse contexto, o objetivo deste laboratório é realizar um experimento controlado para avaliar quantitativamente os benefícios da adoção de uma API GraphQL. Especificamente, as seguintes perguntas devem ser respondidas:

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?

1 Desenho do Experimento

A. Hipóteses Nula e Alternativa

Para a RQ1, “Respostas à consultas GraphQL são mais rápidas que respostas à consultas REST?”, aplica-se as seguintes hipóteses:

H0: A mediana do tempo de consulta da arquitetura REST é igual à mediana do tempo de consulta da arquitetura GraphQL.

H1: A mediana do tempo de consulta da arquitetura GraphQL é diferente da mediana do tempo de consulta da arquitetura REST.

Para a RQ2, “Respostas à consultas GraphQL tem tamanho menor que respostas à consultas REST?”, aplica-se as seguintes hipóteses:

H0: A mediana do tamanho das respostas às consultas REST é igual a mediana do tamanho das respostas às consultas GraphQL.

H1: A mediana do tamanho das respostas às consultas da arquitetura GraphQL é diferente da mediana do tamanho das respostas às consultas da arquitetura REST.

B. Variáveis Dependentes e Variáveis Independentes

- Variáveis Dependentes

No experimento temos duas variáveis dependentes: tempo e tamanho de resposta.

Para realizar a medição da velocidade de resposta das consultas foi utilizado um *script* em Python consultando a API https://api.github.com/ obtendo o tempo das respostas em segundos.

Para realizar a medição do tamanho das respostas retornadas pelas consultas foi utilizado um script em Python consultando a API https://api.github.com/ obtendo o tamanho das respostas em *bytes*.

- Variáveis Independentes

No experimento temos duas variáveis independentes, as manipuladas REST e GraphQL que são os tipos de arquiteturas API. E as controladas que são: velocidade da internet, máquina utilizada, tamanho da resposta da consulta e a API a ser consultada.

C. Tratamentos

Os tratamentos referem-se às diferentes condições ou grupos experimentais que são aplicados durante o experimento.

- Grupo de controle: utilizando uma API REST para realizar as consultas.
- Grupo experimental: utilizando uma API GraphQL para realizar as consultas.

D. Objetos Experimentais

Este experimento selecionou a API do Github disponibilizado através da *url* https://api.github.com para a realização da coleta de dados, tanto usando REST quanto GraphQL. Para a equalização de dados, retornadas pelas duas APIs, as requisições de informações são as mesmas, possibilitando uma confiabilidade nos resultados de tamanho e tempo de busca. Definindo o conjunto de consultas que foram executadas em ambas arquiteturas, as *queries* da **Figura 1** e **Figura 2** foram geradas para as respectivas APIs, Rest e GraphQL:

```
REST = {
  "REPOSITORY": "https://api.github.com/search/repositories?
q=stars:%3E100&per_page={per_page}"
}
```

Figura 1 - *Query* em Rest para coleta de repositórios mais populares

A coleta de dados de dados, usando Rest, representada na **Figura 1**, realiza uma busca de repositórios mais populares da plataforma Github, esperando retornar um JSON com todas as informações de cada repositório minerado.

```
search(query: "stars:>100", type: REPOSITORY, first: {per_page}) {
    repositoryCount
    nodes {
      ... on Repository {
        id
        name
        nameWithOwner
        isPrivate
        owner {
          login
          id
          avatarUrl
          resourcePath
          url
        }
        description
        isFork
        url
        homepageUrl
        diskUsage
        stargazerCount
        watchers { totalCount }
        primaryLanguage { name }
        hasIssuesEnabled
        hasProjectsEnabled
        hasWikiEnabled
        forkCount
        mirrorUrl
        isArchived
        isDisabled
        openIssues: issues(states: OPEN) { totalCount }
        licenseInfo {
          name
          key
          id
          spdxId
          url
        }
        forkingAllowed
        isTemplate
        repositoryTopics(first: 10) { nodes { topic { name } } }
        visibility
        forkCount
        defaultBranchRef { name }
        }}}}
```

Figura 2 - *Query* em GraphQL para coleta de dados dos repositórios mais populares.

A coleta de dados de dados, usando GraphQL, representado na **Figura 2**, realiza uma busca de repositórios mais populares da plataforma Github, esperando retornar uma árvore com as informações requisitadas na *query,* de cada repositório minerado.

E. Tipo de Projeto Experimental

Este experimento possui um fator (Arquitetura) e dois níveis (REST e GraphQL). Portanto, o desenho utiliza o controle local Blocking.

F. Quantidade de Medições

Neste experimento realizou-se uma coleta de dados de 500 repositórios do Github mais populares de consultas para cada API (GraphQL e Rest), totalizando 1000 dados coletados, para obter uma amostra significativa e reduzir a aleatoriedade nos resultados.

G. Ameaças à Validade

As ameaças à validade são possíveis fontes de erro, vieses ou influências que podem comprometer a validade e a confiabilidade dos resultados obtidos em um experimento.

As ameaças identificadas neste experimento são:

- Ameaças à validade interna: são fatores que podem afetar a capacidade de estabelecer uma relação causal entre as variáveis independentes e dependentes. Como exemplo temos: diferenças de hardware, otimização do script e configuração de rede.

- Ameaças à validade externa: referem-se à generalização dos resultados do experimento para além do contexto específico em que foi realizado. Como exemplo temos: cenários externos, consultas e cargas irreais e sobrecargas do lado do servidor.

- Ameaça de história: ocorrência de eventos externos ou mudanças no ambiente que possam influenciar os resultados do experimento de forma não relacionada às variáveis independentes. Temos como exemplo: atualizações e descontinuação das arquiteturas e do software utilizado para a consulta.

- Ameaça à validade de construção: refere-se a problemas relacionados à implementação ou configuração inadequada do experimento. Temos como exemplo: implementação incorreta dos scripts e configurações incorretas do ambiente.

- Ameaça à validade de conclusão: refere-se a problemas que podem levar a conclusões inválidas ou incorretas com base nos resultados obtidos. É importante evitar interpretações tendenciosas dos resultados que favoreçam uma API em detrimento da outra.

  Os resultados obtidos em um experimento específico podem não ser generalizáveis para todos os cenários ou sistemas.

2 Metodologia

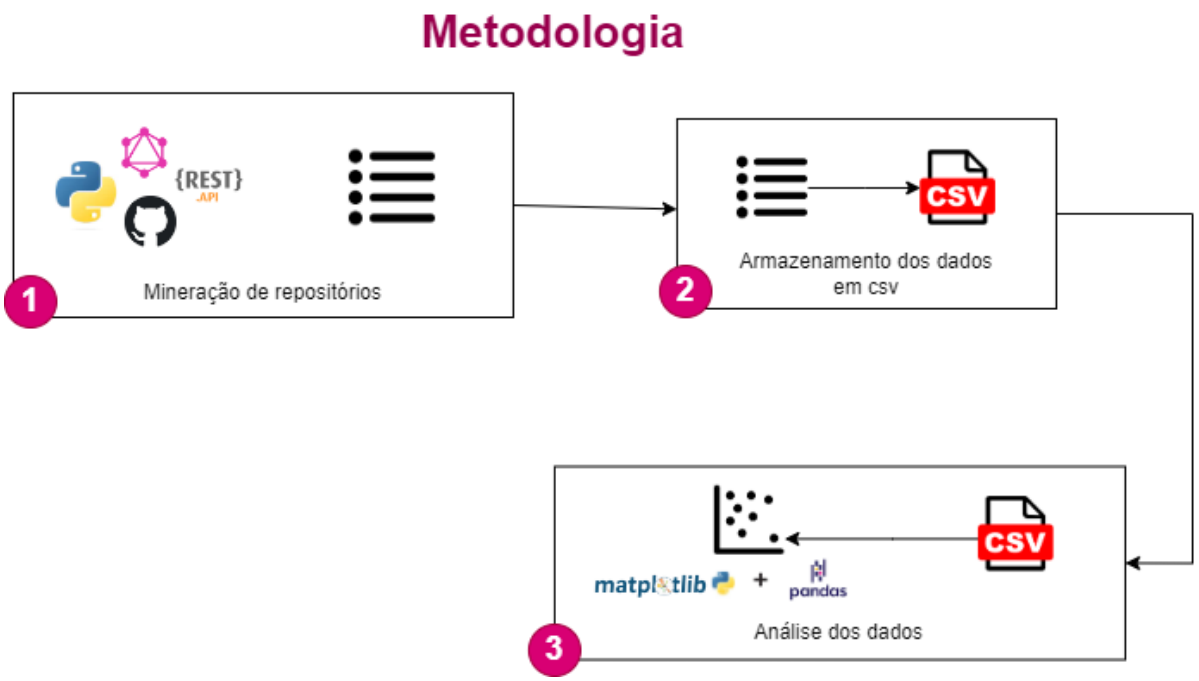

Figura 3 - Metodologia do processo de coleta e análise de dados

Na **Figura 3**, o processo da metodologia consiste em coleta dos 1000 repositórios mais populares do Github por meio de *script* python e pelo uso da API GraphQL e Rest do Github, ressaltando que, cada API realiza a coleta de 500 repositórios. Logo após, é realizada a análise dos dados, vulgo o tempo gasto para cada requisição e tamanho da resposta, por meio de gráficos plotados pelo *script* python com o uso das bibliotecas python matplotlib e pandas, comparando o resultados de ambos os tipos de requisições.

Preparação

Nome do dispositivo	DESKTOP-G1OQ0VH
Processador	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
RAM instalada	12,0 GB (utilizável: 11,8 GB)
Tipo de sistema	Sistema operacional de 64 bits, processador baseado em x64

Tabela 2 - Descrição técnica da máquina

A máquina *desktop* utilizada segue as especificações da **Tabela 2**, a qual o processador é um Intel Core i5 10º geração, 12GB de RAM e sistema operacional *Windows* x64. No entanto, demais especificações são irrelevantes, porém, pode afetar os resultados finais.

3 Análise de Resultados

Com base na RQ1, “Respostas à consultas GraphQL são mais rápidas que respostas à consultas REST?”, temos:

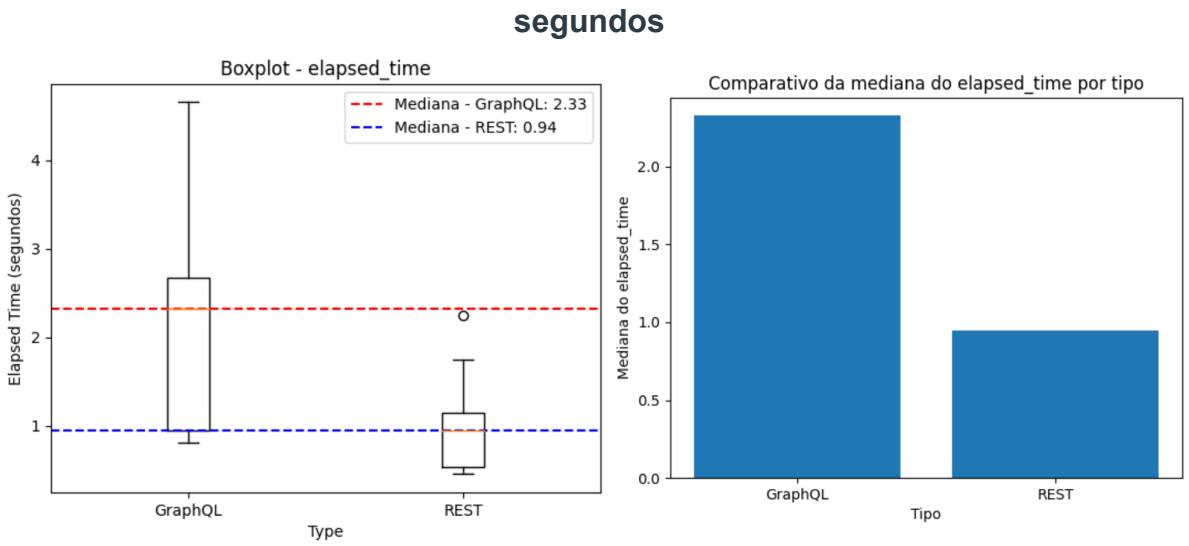

Figura 4 - Comparativo de mediana de tempo gasto na requisição das APIs demonstrado em *boxplot* (esquerda) e barra (direita)

A primeira hipótese nula (H0) afirma que a mediana do tempo de consulta da arquitetura REST é igual à mediana do tempo de consulta da arquitetura GraphQL. A segunda hipótese alternativa (H1) afirma que a mediana do tempo de consulta da arquitetura GraphQL é diferente da mediana do tempo de consulta da arquitetura REST.

Para realizar uma análise adequada, é recomendável utilizar testes estatísticos para comparar as medianas das amostras e determinar se há uma diferença significativa entre elas. Dito isso, um teste não paramétrico, como o teste de Mann-Whitney, foi

aplicado para comparar as medianas do tempo de requisição das duas arquiteturas.

Com base nesses resultados, **Figura 4**, podemos concluir que há evidências para rejeitar a hipótese nula (H0) de que as medianas do tempo de consulta das arquiteturas REST e GraphQL são iguais. Os dados sugerem que a arquitetura GraphQL apresenta uma mediana de tempo de consulta significativamente diferente da arquitetura REST.

Com base na RQ2, “Respostas à consultas GraphQL tem tamanho menor que respostas à consultas REST?”, temos:

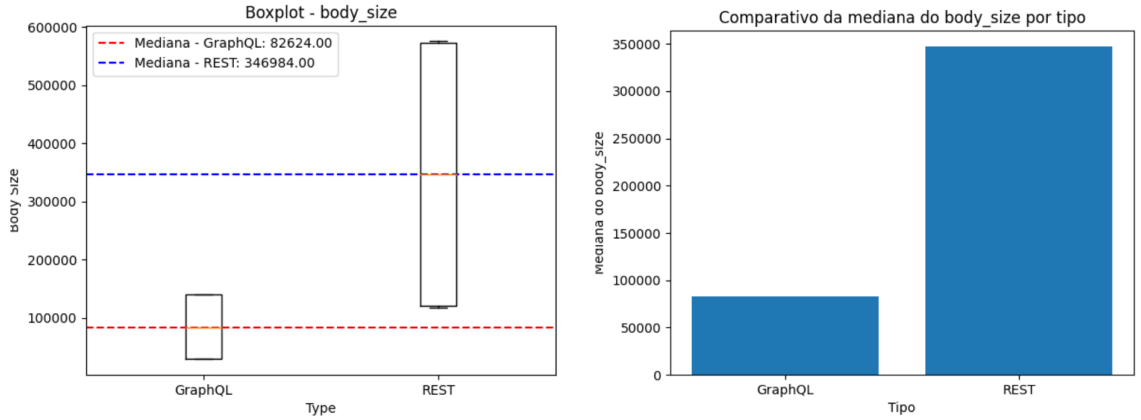

Figura 5 - Comparativo de mediana de tamanho de dados na requisição das APIs demonstrado em *boxplot* (esquerda) e barra (direita)

A primeira hipótese nula (H0) afirma que a mediana do tamanho das respostas às consultas REST é igual à mediana do tamanho das respostas às consultas GraphQL. A segunda hipótese alternativa (H1) afirma que a mediana do tamanho das respostas às consultas da arquitetura GraphQL é diferente da mediana do tamanho das respostas às consultas da arquitetura REST.

Para realizar uma análise adequada, é recomendável utilizar testes estatísticos para comparar as medianas das amostras e determinar se há uma diferença significativa entre elas. Dito isso, um teste não paramétrico, como o teste de Mann-Whitney, foi aplicado para comparar as medianas do tamanho de requisição das duas arquiteturas.

Com base nesses resultados, **Figura 5**, podemos concluir que há evidências para rejeitar a hipótese nula (H0) de que as medianas do tamanho das respostas às

consultas REST e GraphQL são iguais. Os dados sugerem que a arquitetura GraphQL apresenta uma mediana de tamanho de resposta significativamente diferente da arquitetura REST.

Medida	GraphQL	REST
Média	1.849779432	0.8526240279999999
Mediana	2.3262855	0.943318
Moda	0.804476	0.454347
Desvio Padrão	0.9073694441292411	0.32147190001286735
Variância	0.8233193081394081	0.10334418249788298
Valor Máximo	4.656771	2.249915
Valor Mínimo	0.804476	0.454347
Obliquidade	0.2223730323149515	0.222966856195095
Curtose	-1.3263615025260134	-1.0834665788869595

Tabela 3 - medidas de tendência central e medidas de variabilidade de tempo de requisição

Medida	GraphQL	REST
Média	83963,488	346803,6
Mediana	82624	346984
Moda	28708	573760
Desvio Padrão	55311,000957274744	227191,57232774614
Variância	3059306826,8956475	51616010536,75351
Valor Máximo	139236	575580
Valor Mínimo	28708	117012
Obliquidade	1,870826435134442e-05	-1,2471805506007967e-06
Curtose	-2,008023888713921	-2,008043318793945

Tabela 4 - medidas de tendência central e medidas de variabilidade de tamanho de dados

4 Conclusão

Enfim, conclui-se que as análises realizadas entre as arquiteturas REST e GraphQL mostraram que:

Tempo de consulta: Às respostas às consultas GraphQL tendem a ter um tempo de resposta maior em comparação com as consultas REST, indicando que as consultas GraphQL podem levar mais tempo para serem processadas.

Tamanho das respostas: As respostas GraphQL tendem a ter um tamanho menor em comparação com as respostas REST, sugerindo que as respostas GraphQL são mais compactas em relação às respostas REST.

No entanto, é importante ressaltar que essas conclusões são baseadas nos dados fornecidos nas análises específicas e nas hipóteses consideradas. Os resultados podem variar dependendo do contexto de uso, dos requisitos específicos do projeto e da implementação das arquiteturas REST e GraphQL.

Portanto, ao decidir entre REST e GraphQL, é essencial levar em consideração as necessidades do projeto e ponderar os benefícios e as limitações de cada arquitetura. A escolha dependerá dos requisitos de desempenho, tamanho das respostas, complexidade das consultas, flexibilidade de uso e outras considerações específicas do projeto. A experimentação e a análise cuidadosa são cruciais para avaliar o desempenho e a adequação das arquiteturas em um determinado contexto.