GraphQL vs REST: Um Experimento Controlado

Bruno Gomes Ferreira, João Pedro Mairinque de Azevedo, Matheus Vieira dos Santos, Marcio Lucas Machado Pereira

[1] Instituto de Ciências Exatas e Informática (ICEI)
Pontifícia Universidade Católica de Minas Gerais
Engenharia de Software
Belo Horizonte – MG – Brazil

1. INTRODUÇÃO

Quando se trata da comparação de abordagens arquitetônicas para APIs, destacam-se REST (Representational State Transfer) e GraphQL como duas das mais influentes. A tecnologia REST, devido à sua conformidade com o protocolo HTTP, tornou-se padrão por sua facilidade de uso e escalabilidade. Em contraste, o GraphQL, permite que os clientes especifiquem exatamente os dados necessários, reduzindo a quantidade de dados transferidos e otimizando a eficiência das consultas. Entender a eficiência das abordagens pode orientar desenvolvedores e empresas na escolha da tecnologia mais adequada para suas necessidades, resultando em melhorias significativas na performance e na utilização dos recursos dos serviços Web. [Brito and Valente 2020]

A arquitetura REST (Representational State Transfer), introduzido por [Fielding and Taylor 2002], é um estilo arquitetônico para sistemas distribuídos que visa melhorar o desempenho, disponibilidade e escalabilidade dos sistemas. Segundo [Brito and Valente 2020], as APIs RESTful são construídas em torno de recursos identificados por URLs e utilizam os métodos HTTP para realizar operações sobre esses recursos. A simplicidade e escalabilidade do REST, onde cada endpoint retorna uma representação do recurso em um formato padrão, tornaram-no uma escolha popular para o desenvolvimento de APIs na web.

Por outro lado, o GraphQL é uma linguagem de consulta para APIs desenvolvida pelo Facebook em 2015 e posteriormente transferida para a GraphQL Foundation. Descrito por [Hartig and Pérez 2017], o GraphQL oferece um sistema baseado em tipos e consultas que permite aos clientes especificarem exatamente os dados que necessitam de um serviço. [Lopes et al. 2020] Diferente do REST, onde os endpoints retornam um conjunto fixo de campos, o GraphQL permite que os clientes definam precisamente os campos desejados em uma única consulta. Essa flexibilidade e eficiência na transferência de dados são algumas das razões pelas quais o GraphQL tem ganhado popularidade, especialmente com o suporte de ferramentas avançadas como o GraphiQL.

Pensando nisso, as seguintes perguntas de pesquisa foram propostas para este trabalho:

- 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?

Baseado na literatura, tanto para [Lopes et al. 2020] e [Brito and Valente 2020] provou-se que GraphQL tem uma eficiência maior que REST, sendo assim as hipóteses

iniciais para esse trabalho é de que realmente o GraphQL será superior nas consultas tanto em tempo quanto tamanho. Fatores como qual das tecnologias possui menor tempo de resposta e qual delas tem um tamanho menor das consultas são preponderantes para determinar qual das tecnologias tem uma maior eficiência. Levando isso em conta, este artigo tem como objetivo realizar um experimento controlado em 1000 repositórios para avaliar quantitativamente os benefícios da adoção de uma API GraphQL.

Nas seções seguintes, será discutido inicialmente a Metodologia onde serão tratados detalhes necessários para possibilitar a reprodução e replicação do experimento. Após isso, o tópico de resultados tratá resultados numéricos e estatísticos com o resultado da execução do experimento e por fim o tópico de discussão será responsável pela interpretação do que foi encontrado como resultado.

2. METODOLOGIA

2.1. Desenho do Experimento

Nesta seção, será detalhada a estrutura do experimento projetado para investigar e comparar o desempenho de consultas GraphQL e REST em termos de tempo de resposta e tamanho das respostas. O experimento será conduzido utilizando a API do GitHub, que oferece suporte tanto para GraphQL quanto para REST, permitindo uma comparação direta entre os dois tipos de consultas.

Primeiramente, serão definidas as hipóteses nulas e alternativas em cada uma das questões de pesquisa que guiarão toda a investigação.

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

- Hipótese Nula (Ho) para RQ1: Não há diferença no tempo de resposta entre consultas GraphQL e consultas REST.
- Hipótese Alternativa (H1) para RQ1: Consultas GraphQL têm um tempo de resposta mais rápido do que consultas REST.

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

- Hipótese Nula (Ho) para RQ2: Não há diferença no tamanho das respostas entre consultas GraphQL e consultas REST.
- Hipótese Alternativa (H1) para RQ2: Consultas GraphQL resultam em respostas de tamanho menor do que consultas REST.

2.1.3. Variáveis Dependentes

As variáveis dependentes deste experimento são

- RQ1: Tempo de resposta da API (medido em segundos).
- RQ2: Tamanho da resposta da API (medido em bytes).

2.1.4. Variáveis Independentes

Quando se tratam de variáveis independentes, para uma melhor compreensão do experimento, optou-se pelos autores por separá-las em controladas e manipuladas. Variáveis controladas são aquelas que têm um valor fixo e são controladas pelos autores para assegurar um resultado replicável do experimento, enquanto as variáveis manipuladas são aquelas que são variadas durante o experimento. Em relação às variáveis independentes controladas, destacam-se:

- Quantidade de repositórios analisados (1000)
- Mesma query para REST e GraphQL
- Ambiente controlado: Todas as chamadas foram feitas no mesmo sistema operacional (Windows)
- Linguagem: Python
- API do Github
- Repositórios populares de acordo com a maior quantidade de estrelas

Para as variáveis independentes manipuladas, tem-se:

- Tipo de API (GraphQL vs. REST)

2.1.5. Objetos experimentais

- API de teste: API do GitHub que possui possibilidade de fazer consultas tanto em GraphQL quanto em REST.
- Consultas: Um conjunto de 1000 consultas que serão realizadas nos repositórios mais populares.

2.1.6. Tipo de Projeto Experimental

O projeto será do tipo Related Between Objects onde tanto o GraphQL quanto REST farão as mesmas consultas nos mesmos repositórios. Além disso, será utilizado o padrão Crossing, onde todos os fatores serão cruzados

2.1.7. Quantidade de Medições

Para esse experimento, a mesma query será feita para os 1000 repositórios mais populares do GitHub, sendo 1000 para GraphQL e 1000 para REST, totalizando 2000 medições. Para definir um repositório popular foi buscado informações como quando foi última atualização daquele repositório, número total de colaboradores, número de releases feitas e quantidade de Forks (quantas pessoas copiaram aquele repositório para seu GitHub).

2.1.8. Ameaças à Validade

- Interna: Variabilidade na carga do servidor durante os testes. Solução: Realizar testes em condições controladas e consistentes onde não haja sobrecarga da API

- Externa: Generalização dos resultados para outras APIs ou ambientes. Solução: Selecionar uma API representativa do uso comum de um desenvolvedor como o GitHub
- De Construção: Precisão na medição do tempo de resposta e tamanho das respostas. Solução: Usar um ambiente controlado, no caso, o experimento será realizado em um único computador

2.2. Ambiente de Trials

Para fazer o experimento, utilizou-se um computador que possui as seguintes especificações:

- Processador: Ryzen 5 7600 (12 CPUs) - DDR 5
- Memória RAM: 32 GB
- Sistema Operacional: Windows 10

2.3. Execução do experimento

Nesta seção, será detalhada a execução do experimento conforme o desenho previamente especificado, com o objetivo de poder comparar o desempenho de consultas GraphQL e REST em termos de tempo de resposta e tamanho das mesmas utilizando a API do GitHub para buscar os 1000 repositórios mais populares em cada uma dessas tecnologias.

Para a execução do experimento, foi inicialmente desenvolvido dois scripts em Python para coletar os dados necessários: um para realizar consultas REST e outro para consultas GraphQL. Ambos foram codificados para buscar informações sobre os 1000 repositórios mais populares no GitHub, ordenados pelo número de estrelas.

A execução dos scripts foi realizada em um ambiente controlado, assegurando que todas as chamadas fossem feitas sob as mesmas condições. Isso incluiu a utilização do mesmo sistema operacional e máquina, bem como a execução dos scripts em um período de tempo em que a variabilidade na carga do servidor do GitHub fosse mínima. Esse controle é essencial para garantir a validade interna do experimento, minimizando a influência de fatores externos nas métricas coletadas.

Após a coleta de dados, os repositórios são processados e salvos em um arquivo CSV, contendo informações como URL do repositório, nome, proprietário, número de estrelas, data de criação e data de atualização. Este processo assegura que todos os dados relevantes sejam capturados e organizados para análise posterior. Em suma, a execução do experimento seguiu rigorosamente o desenho proposto, utilizando scripts desenvolvidos especificamente para coletar e medir as variáveis dependentes em um ambiente controlado. A comparação das métricas obtidas permitirá a avaliação das hipóteses levantadas sobre o desempenho das consultas GraphQL e REST.

3. RESULTADOS

Neste tópico, os dados que foram obtidos durante o processo de coleta são apresentados e associados às suas questões de pesquisa correlatas.

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

A fim de investigar se as respostas às consultas GraphQL são mais rápidas que as consultas feitas em REST, foi realizado uma análise no tempo em que cada uma das tecnologias levou para trazer as respostas da `query` utilizada durante o experimento. A figura 1 abaixo apresenta o gráfico de histograma com os resultados da análise do tempo necessário para realizar 1000 chamadas de API em GraphQL e 1000 chamadas em REST.

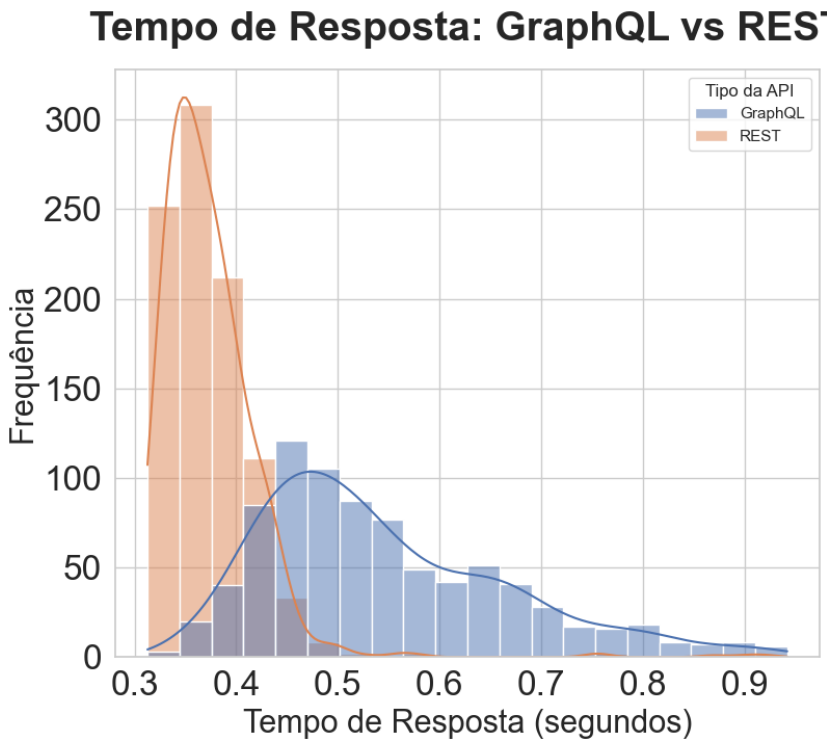

Figura 1. Histograma do tempo de resposta em segundos para GraphQL e REST em 1000 repositórios

A maioria das chamadas utilizando REST foram feitas em menos de 0,4 segundos, enquanto as chamadas feitas em GraphQL demonstraram ter uma maior variedade de tempo. No entanto, na maioria dos casos foi demonstrado que o tempo para obter a resposta superou os 0,4 segundos obtidos pela tecnologia REST demonstrando ter uma requisição mais lenta que a requisção feita em REST.

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

Para avaliar o tamanho das consultas de GraphQL e REST, traçou-se um gráfico de barras promovendo uma visualização comparativa através da média de tamanho de respostas. Os resultados obtidos podem ser vistos na figura 2.

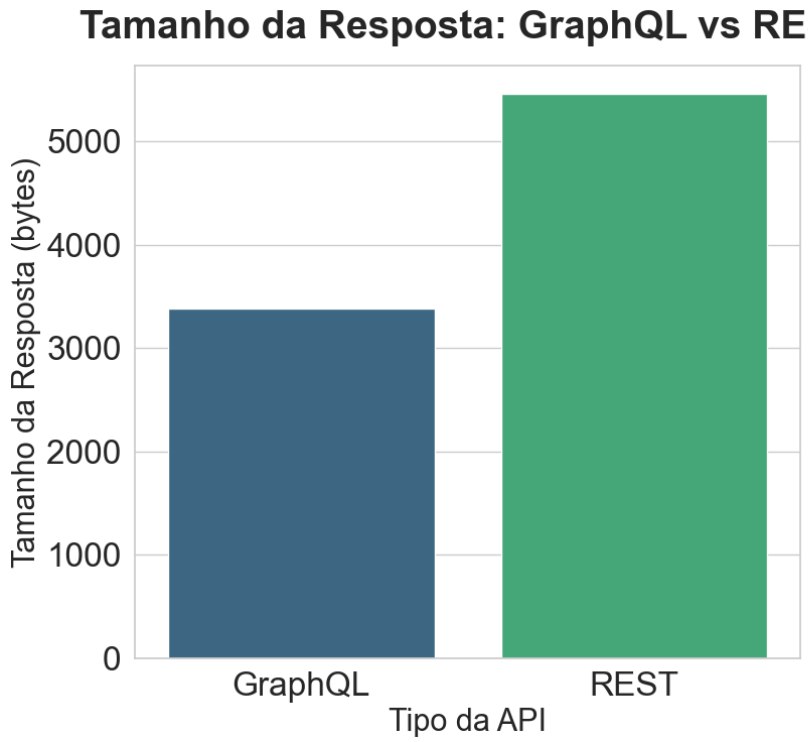

Figura 2. Média de tamanho das requisições em bytes feitas utilizando REST e GraphQL em 1000 repositórios

O tamanho da média de resposta do GraphQL é aproximadamente 3000 bytes, enquanto o REST ultrapassa o GraphQL, cerca de 5000 bytes. Isso sugere que o GraphQL pode ser mais eficiente em termos de uso de largura de banda, pois permite que os clientes solicitem exatamente os dados de que precisam, evitando o envio de dados desnecessários.

4. DISCUSSÃO

Como foi possível observar nos resultados, o tempo de resposta para requisições equivalentes, realizadas na API REST é predominantemente inferior quando comparado com a API GraphQL. Este resultado faz sentido quando olhamos para como a arquitetura destas diferentes APIs foram criadas e para quais propósitos elas são utilizadas, como definido por Muhammad Arya Dhika et al [Tsaqofi 2023] enquanto a API REST foi criada para se obter uma interface rápida e otimizada, a qual entrega todos os dados relacionados à uma entidade, mesmo sem estes serem requisitados, a API GraphQL foi criada para solicitar dados específicos, mesmo que de múltiplas entidades, entregues em uma resposta única.

Levando em consideração o resultado do tamanho da resposta, podemos aferir que as respostas da API GraphQL são sempre menores que as respostas da API REST, isso implica que, na resposta GraphQL, os dados recebidos do servidor correspondem apenas aos dados que foram requisitados, dessa forma sendo muito mais relevantes para o uso do solicitador, enquanto, na resposta da API REST, muitos dados residuais são enviados como resposta, causando, possívelmente, um maior trabalho de manipulação dos dados recebidos antes do seu uso apropriado.

Portanto, conclui-se que o uso de tais diferentes APIs, dá-se diretamente relacionado ao propósito do mesmo. Em situações nas quais a velocidade de requisições individuais é mais importante do que a capacidade de escolher exatamente como será construída a sua resposta, com dados de multiplas entidades, excluindo a necessidade de múltiplas requisições, a API REST será a melhor escolha. Caso contrário, a API GraphQL será mais apropriada.

Referências

Brito, G. and Valente, M. T. (2020). Rest vs graphql: A controlled experiment. In *IEEE International Conference on Software Architecture (ICSA)*, pages 81–90. IEEE.

Fielding, R. T. and Taylor, R. N. (2002). Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150.

Hartig, O. and Pérez, J. (2017). An initial analysis of facebook's graphql language. In *11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web (AMW)*, pages 1–10.

Lopes, I. S. M., de Faria Silva, L. H., and Ponciano, L. (2020). Análise do custo-benefício de graphql em comparação a rest e soap em aplicações para dispositivos móveis. In *Workshop de Engenharia de Software (WES)*, Belo Horizonte, Brasil. PUC Minas.

Tsaqofi, M. A. D. D. K. S. U. M. A. F. V. A. W. A. (2023). Comparing graphql and rest architecture in arabic learning games: A quality of service (qos) approach. *2023 11th International Conference on Cyber and IT Service Management*.