RELATÓRIO

GraphQL vs REST: Um Experimento Controlado sobre Desempenho e Eficiência de Respostas

GraphQL vs REST: A Controlled Experiment on Performance and Response Efficiency

Lorrayne Oliveira [Pontifícia Universidade Católica de Minas Gerais | *lorrayne.marayze@gmail.com*]
Pedro Pires [Pontifícia Universidade Católica de Minas Gerais | *pedro.pires@gmail.com*]

Resumo. A linguagem de consulta GraphQL tem sido apresentada como alternativa às APIs REST tradicionais, prometendo maior flexibilidade e eficiência na comunicação cliente-servidor. Entretanto, evidências empíricas sobre os reais benefícios de desempenho e eficiência de GraphQL em comparação com REST permanecem limitadas. Este trabalho apresenta o desenho de um experimento controlado para avaliar quantitativamente duas dimensões principais: tempo de resposta e tamanho das respostas retornadas. O experimento utiliza um projeto crossover com medições repetidas, permitindo comparação direta entre as duas abordagens sob condições controladas. A execução sistemática deste experimento visa fornecer dados objetivos para subsidiar decisões arquiteturais relacionadas à escolha entre GraphQL e REST em projetos de desenvolvimento de software.

Abstract. The GraphQL query language has been presented as an alternative to traditional REST APIs, promising greater flexibility and efficiency in client-server communication. However, empirical evidence on the actual performance and efficiency benefits of GraphQL compared to REST remains limited. This paper presents the design of a controlled experiment to quantitatively evaluate two main dimensions: response time and the size of the returned responses. The experiment uses a crossover design with repeated measurements, allowing a direct comparison between the two approaches under controlled conditions. The systematic execution of this experiment aims to provide objective data to support architectural decisions related to the choice between GraphQL and REST in software development projects.

Palavras-chave: GraphQL; REST; API; Experimento Controlado; Desempenho de Software; Engenharia de Software Experimental

Keywords: GraphQL; REST; API; Controlled Experiment; Software Performance; Experimental Software Engineering

1 Introdução

A evolução das arquiteturas de software para sistemas distribuídos tem impulsionado o desenvolvimento de diferentes abordagens para implementação de APIs Web. As APIs REST (Representational State Transfer) consolidaram-se como padrão de mercado nas últimas décadas, oferecendo uma arquitetura baseada em recursos e operações HTTP bem definidas. Paralelamente, o Facebook introduziu GraphQL como linguagem de consulta alternativa, propondo um modelo baseado em grafos que permite aos clientes especificar precisamente quais dados desejam recuperar.

GraphQL fundamenta-se em schemas que descrevem a estrutura de dados disponível, permitindo consultas flexíveis e evitando problemas como over-fetching e under-fetching de dados. Em contraste, APIs REST dependem de endpoints fixos que retornam estruturas de dados predefinidas, frequentemente exigindo múltiplas requisições para obter informações relacionadas. Diversas organizações realizaram migrações parciais ou completas de REST para GraphQL, motivadas por promessas de melhor desempenho e eficiência.

Apesar do crescente interesse da indústria, a literatura científica carece de estudos experimentais rigorosos que quantifiquem objetivamente as diferenças entre estas abordagens. Avaliações anedóticas e estudos de caso existem, mas experimentos controlados que isolem variáveis e forneçam evidências estatísticas robustas são escassos. Esta lacuna dificulta decisões fundamentadas sobre qual tecnologia adotar em contextos específicos.

O presente trabalho propõe um experimento controlado para investigar duas questões centrais: (RQ1) Respostas às consultas GraphQL são mais rápidas que respostas às consultas REST? (RQ2) Respostas às consultas GraphQL têm tamanho menor que respostas às consultas REST? A investigação destas questões permitirá avaliar empiricamente se GraphQL oferece vantagens mensuráveis em termos de latência e eficiência de rede, aspectos críticos para aplicações modernas.

1.1 Objetivos

O objetivo principal deste trabalho é avaliar empiricamente as diferenças de desempenho e eficiência entre as tecnologias GraphQL e REST para implementação de APIs Web. A partir de medições quantitativas controladas, busca-se responder às seguintes questões de pesquisa:

- **RQ1:** Respostas às consultas GraphQL são mais rápidas que respostas às consultas REST?
- **RQ2:** Respostas às consultas GraphQL têm tamanho menor que respostas às consultas REST?

Como objetivo secundário, pretende-se fornecer evidências empíricas que permitam decisões fundamentadas sobre a escolha entre GraphQL e REST em projetos de desenvolvimento de software, contribuindo para o entendimento das vantagens e desvantagens práticas de cada abordagem em cenários controlados.

1.2 Hipóteses Nula e Alternativa

O experimento investiga duas dimensões principais, cada uma com seu par de hipóteses:

RQ1 (Tempo de Resposta):

- **Hipótese Nula ($H_{0,tempo}$):** Não existe diferença significativa entre o tempo médio de resposta de consultas GraphQL e o tempo médio de resposta de consultas REST equivalentes.
- **Hipótese Alternativa ($H_{1,tempo}$):** Existe diferença significativa entre o tempo médio de resposta de consultas GraphQL e o tempo médio de resposta de consultas REST equivalentes.

RQ2 (Tamanho da Resposta):

- **Hipótese Nula ($H_{0,tamanho}$):** Não existe diferença significativa entre o tamanho médio das respostas de consultas GraphQL e o tamanho médio das respostas de consultas REST equivalentes.
- **Hipótese Alternativa ($H_{1,tamanho}$):** Existe diferença significativa entre o tamanho médio das respostas de consultas GraphQL e o tamanho médio das respostas de consultas REST equivalentes.

1.3 Variáveis Dependentes

As variáveis dependentes representam as métricas que serão observadas e medidas durante o experimento:

- **VD1 - Tempo de Resposta:** Tempo decorrido entre o envio da requisição e o recebimento completo da resposta, **medido em milissegundos (ms)**. Esta métrica captura a latência total da comunicação.
- **VD2 - Tamanho da Resposta:** Quantidade de dados transferidos na resposta, **medida em bytes ou kilobytes (KB)**. Esta métrica quantifica a eficiência do uso da largura de banda.

1.4 Variáveis Independentes

A variável independente principal é o tipo de tecnologia de API utilizada:

- **VI1 - Tipo de API:** Variável categórica com dois níveis: GraphQL e REST. Esta variável representa o tratamento aplicado em cada medição.

Variáveis de Controle:

Para garantir validade interna, as seguintes variáveis serão mantidas constantes:

- Complexidade da consulta realizada
- Conjunto de dados retornado (equivalência semântica)
- Configuração do servidor (hardware, sistema operacional, recursos alocados)
- Condições de rede (mesma infraestrutura, mesmo horário)
- Volume de dados no banco de dados
- Tecnologia de implementação do servidor (linguagem, framework)

1.5 Tratamentos

Os tratamentos representam cada combinação das variáveis independentes (tipo de API e complexidade da consulta). Considerando os dois tipos de tecnologia e os quatro níveis de complexidade de consulta, tem-se:

Tratamentos REST:

- **T1a - REST Simples:** Consulta REST para recuperação de informações básicas de um único recurso (ex: dados de um usuário específico).
- **T1b - REST com Relacionamentos:** Consulta REST para recuperação de um recurso com dados relacionados, podendo exigir múltiplas requisições (ex: usuário e seus posts).
- **T1c - REST Complexa:** Consulta REST envolvendo múltiplos níveis de relacionamentos, como requirindo várias requisições sequenciais (ex: usuário, posts, comentários dos posts).
- **T1d - REST Lista:** Consulta REST para recuperação de múltiplos recursos com filtros aplicados (ex: lista de usuários com determinado critério).

Tratamentos GraphQL:

- **T2a - GraphQL Simples:** Consulta GraphQL equivalente a T1a, recuperando os mesmos dados através de uma única requisição com query específica.
- **T2b - GraphQL com Relacionamentos:** Consulta GraphQL equivalente a T1b, recuperando dados relacionados em uma única requisição através de campos aninhados.
- **T2c - GraphQL Complexa:** Consulta GraphQL equivalente a T1c, recuperando múltiplos níveis de relacionamentos em uma única requisição.
- **T2d - GraphQL Lista:** Consulta GraphQL equivalente a T1d, recuperando lista filtrada de recursos em uma única requisição.

Cada tratamento GraphQL será projetado para retornar exatamente os mesmos dados que seu tratamento REST correspondente, garantindo equivalência semântica para comparação válida das métricas de desempenho e tamanho de resposta.

Referências

Basios, M., Passos, L., Berger, T., and Khomh, F. (2017). Measuring api usability through automated analysis of their documentation. *IEEE Software*, 34(3):78–86.

GraphQL Foundation (2018). Graphql specification. Technical report, The Linux Foundation. Available at: https://spec.graphql.org/.

Indrasiri, K. and Siriwardena, P. (2021). Api design patterns and best practices. In *Microservices for the Enterprise*, pages 125–157. Apress.

Juristo, N. and Moreno, A. M. (2010). Basics of software engineering experimentation. *Springer Science & Business Media*.

Kampik, T., Amaral, M., Gonçalves, A., and Pereira, R. (2020). Network traffic analysis of graphql and rest apis. *Software: Practice and Experience*, 50(8):1458–1473.

Seabra, M., Nazareno, F., and Pinto, G. (2019). Performance comparison between graphql and rest. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pages 123–132. ACM.

Taelman, R., Verborgh, R., and Mannens, E. (2018). Comparative analysis of web apis and their machine-readable documentation. *Journal of Web Semantics*, 52:17–30.

Vázquez-Ingelmo, A., García-Peñalvo, F. J., and Therón, R. (2020). An analysis of the adoption of graphql in open source projects. In *Proceedings of the 8th International Conference on Technological Ecosystems for Enhancing Multiculturality*, pages 873–879. ACM.