Experimento Controlado: GraphQL vs REST

Relatório Final
Laboratório de Experimentação de Software

Gabriel Henrique Miranda Rodrigues

PUC Minas

1 Introdução

Este trabalho apresenta um experimento controlado para comparar o desempenho de duas tecnologias de APIs Web: GraphQL e REST. A API REST é a abordagem tradicional, baseada em endpoints que retornam dados completos de um recurso. Já o GraphQL, criado pelo Facebook em 2015, permite que o cliente especifique exatamente quais dados precisa receber.

O experimento foi realizado utilizando a API do GitHub, que oferece tanto uma versão REST quanto uma versão GraphQL para acessar os mesmos dados. Isso permitiu fazer uma comparação justa entre as duas tecnologias.

1.1 Questões de Pesquisa

O experimento buscou responder duas perguntas:

- **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?

1.2 Hipóteses

Para cada pergunta de pesquisa, foram definidas hipóteses a serem testadas estatisticamente:

RQ1 - Tempo de Resposta:

- H0: Não existe diferença significativa no tempo de resposta entre GraphQL e REST.
- H1: Existe diferença significativa no tempo de resposta entre GraphQL e REST.

RQ2 - Tamanho da Resposta:

- H0: Não existe diferença significativa no tamanho das respostas entre GraphQL e REST.

- H1: Existe diferença significativa no tamanho das respostas entre GraphQL e REST.

O nível de significância adotado foi de 5% (alfa = 0,05).

2 Metodologia

2.1 Desenho do Experimento

O experimento seguiu um projeto fatorial completo, combinando dois fatores: o tipo de API (REST ou GraphQL) e o nível de complexidade da consulta (simples, média ou complexa). Isso resultou em 6 tratamentos diferentes.

2.2 Variáveis

As variáveis dependentes, ou seja, aquelas que foram medidas, são:

- Tempo de resposta (em milissegundos)
- Tamanho da resposta (em bytes)

As variáveis independentes, controladas no experimento, são:

- Tipo de API: REST ou GraphQL
- Complexidade: simples, média ou complexa

2.3 Objetos Experimentais

Foram selecionados 10 repositórios populares do GitHub para realizar as consultas:

Repositório	**Descrição**
facebook/react	Biblioteca JavaScript
microsoft/vscode	Editor de código
tensorflow/tensorflow	Machine Learning
torvalds/linux	Kernel Linux
django/django	Framework Python
python/cpython	Linguagem Python
nodejs/node	Runtime JavaScript
kubernetes/kubernetes	Orquestração de containers
angular/angular	Framework TypeScript
vuejs/vue	Framework JavaScript

Tabela 1: Repositórios utilizados no experimento

2.4 Definição das Consultas

As consultas foram organizadas em três níveis de complexidade:

Consulta Simples: Busca apenas dados básicos do repositório como nome, descrição e número de estrelas. No REST, é feita uma única requisição. No GraphQL, uma única query solicitando apenas os campos necessários.

Consulta Média: Além dos dados do repositório, busca também os últimos 10 issues. No REST, são necessárias duas requisições separadas. No GraphQL, uma única query com os dados aninhados.

Consulta Complexa: Busca dados do repositório, issues, contribuidores e branches. No REST, são necessárias quatro requisições. No GraphQL, tudo é obtido em uma única query.

2.5 Quantidade de Medições

Foram realizadas 100 repetições para cada combinação de tratamento e repositório:

- 6 tratamentos × 10 repositórios × 100 repetições = 6.000 medições

2.6 Ambiente de Execução

O experimento foi desenvolvido em Python, utilizando a biblioteca requests para fazer as requisições HTTP. Para análise dos dados, foram utilizadas as bibliotecas pandas e scipy.

Antes de iniciar as medições, foram executadas 5 requisições de aquecimento que foram descartadas. A ordem dos tratamentos foi aleatorizada para evitar viés. Entre cada requisição, foi inserido um intervalo de 100ms para respeitar os limites da API do GitHub.

3 Resultados

3.1 Visão Geral

O experimento coletou 6.000 medições no total, sendo 3.000 para cada tipo de API. A tabela abaixo apresenta as estatísticas gerais:

API	**N**	**Média**	**DP**	**Mediana**	**Mín**	**Máx**
			Tempo de Resposta (ms)			
REST	3000	1071,73	612,65	918,63	303,06	4561,90
GraphQL	3000	548,47	246,66	459,09	276,51	3037,42
			Tamanho da Resposta (bytes)			
REST	3000	33839,08	26936,62	33205	-	-
GraphQL	3000	928,24	661,92	1137	-	-

Tabela 2: Estatísticas descritivas por tipo de API

De forma geral, o GraphQL apresentou tempo médio de resposta de 548,47ms, enquanto o REST teve média de 1071,73ms. Isso representa uma redução de 48,8% no tempo.

Em relação ao tamanho das respostas, o GraphQL retornou em média 928,24 bytes, contra 33839,08 bytes do REST. A redução foi de 97,3%.

3.2 RQ1: Tempo de Resposta

A primeira pergunta de pesquisa investigou se o GraphQL é mais rápido que o REST.

Métrica	REST	GraphQL
Média	1071,73 ms	548,47 ms
Desvio Padrão	612,65 ms	246,66 ms
Mediana	918,63 ms	459,09 ms
Mínimo	303,06 ms	276,51 ms
Máximo	4561,90 ms	3037,42 ms

Tabela 3: Comparação do tempo de resposta

O teste estatístico aplicado foi o teste t para amostras independentes. Os resultados foram:

- Estatística t = 42,33
- p-valor ¡ 0,001
- Cohen's d = 1,09

O p-valor menor que 0,001 indica que a diferença é estatisticamente significativa. O valor de Cohen's d igual a 1,09 indica um tamanho de efeito grande, ou seja, a diferença observada tem relevância prática.

Conclusão RQ1: A hipótese nula foi rejeitada. O GraphQL é significativamente mais rápido que o REST, com uma redução de 48,8% no tempo médio de resposta.

3.3 RQ2: Tamanho da Resposta

A segunda pergunta investigou se as respostas do GraphQL são menores que as do REST.

Métrica	REST	GraphQL
Média	33839,08 B	928,24 B
Desvio Padrão	26936,62 B	661,92 B
Mediana	33205 B	1137 B

Tabela 4: Comparação do tamanho da resposta

O teste estatístico apresentou os seguintes resultados:

- Estatística t = 69,34
- p-valor ¡ 0,001
- Cohen's d = 1,79

Novamente, o p-valor menor que 0,001 confirma diferença significativa. O Cohen's d de 1,79 indica um efeito muito grande.

Conclusão RQ2: A hipótese nula foi rejeitada. O GraphQL retorna respostas significativamente menores que o REST, com uma redução de 97,3% no tamanho médio.

3.4 Análise por Complexidade

Os resultados também foram analisados separadamente por nível de complexidade:

Complexidade	REST (ms)	GraphQL (ms)	Tempo	REST (B)	GraphQL (B)	Tam
Simples	459,21	393,98	-14,2%	6122	250	-9
Média	973,30	469,61	-51,8%	57398	1472	-9
Complexa	1782,68	781,81	-56,1%	37998	1063	-9

Tabela 5: Resultados por nível de complexidade

Os dados mostram que a vantagem do GraphQL aumenta conforme a complexidade da consulta cresce. Em consultas simples, a diferença de tempo é de 14,2%. Já em consultas complexas, essa diferença chega a 56,1%.

Isso acontece porque consultas REST complexas precisam de múltiplas requisições sequenciais, enquanto o GraphQL resolve tudo em uma única chamada.

Em relação ao tamanho, a redução se mantém acima de 95% em todos os níveis de complexidade. Isso ocorre porque o REST sempre retorna todos os campos do objeto, enquanto o GraphQL retorna apenas o que foi solicitado.

4 Discussão

Os resultados do experimento mostram vantagens claras do GraphQL sobre o REST nas duas métricas avaliadas.

4.1 Tempo de Resposta

A redução de 48,8% no tempo de resposta pode ser explicada por dois fatores principais. Primeiro, o GraphQL consolida múltiplas requisições em uma só, eliminando a latência de várias chamadas de rede. Segundo, ao retornar menos dados, o tempo de transferência também diminui.

Essa vantagem é especialmente relevante em aplicações móveis, onde a latência de rede costuma ser maior e cada milissegundo economizado impacta a experiência do usuário.

4.2 Tamanho da Resposta

A redução de 97,3% no tamanho das respostas é o resultado mais expressivo do experimento. Enquanto o REST retorna objetos completos com dezenas de campos, o GraphQL permite solicitar apenas os dados necessários.

Essa economia de bytes tem impacto direto no consumo de dados em redes móveis e no uso de memória do cliente. Em aplicações que fazem muitas requisições, a diferença acumulada pode ser substancial.

4.3 Limitações

O experimento foi realizado apenas com a API do GitHub. Outras APIs podem ter características diferentes que alterem os resultados. Além disso, as métricas avaliadas

não consideram aspectos como facilidade de desenvolvimento, curva de aprendizado ou complexidade de manutenção do código.

5 Ameaças à Validade

Alguns fatores podem ter influenciado os resultados:

- **Variação de rede:** A latência da internet pode variar durante o experimento. Para minimizar esse efeito, foram feitas 100 repetições de cada medição e a ordem foi aleatorizada.
- **Cache:** As APIs podem utilizar cache nas respostas. Foram utilizados headers para desativar o cache nas requisições.
- **Limitações do GitHub:** Os resultados são específicos para a API do GitHub e podem não se aplicar a outras APIs.
- **Definição de complexidade:** Os níveis de complexidade foram definidos de forma arbitrária e podem não representar todos os casos de uso reais.

6 Conclusão

Este experimento comparou o desempenho de APIs GraphQL e REST através de 6.000 medições realizadas na API do GitHub. Os resultados permitem responder às perguntas de pesquisa:

RQ1: Sim, as respostas GraphQL são mais rápidas. O tempo médio foi 48,8% menor, com significância estatística (p ¡ 0,001) e tamanho de efeito grande (d = 1,09).

RQ2: Sim, as respostas GraphQL são menores. O tamanho médio foi 97,3% menor, com significância estatística (p ¡ 0,001) e tamanho de efeito grande (d = 1,79).

Os resultados indicam que o GraphQL pode trazer ganhos de desempenho em cenários onde é necessário buscar dados de múltiplos recursos ou onde a economia de banda é importante. No entanto, a escolha entre as duas tecnologias deve considerar também outros fatores como a experiência da equipe e os requisitos específicos do projeto.

Referências

1. GitHub REST API Documentation. Disponível em: https://docs.github.com/en/rest
2. GitHub GraphQL API Documentation. Disponível em: https://docs.github.com/en/graphql
3. GraphQL Specification. Disponível em: https://graphql.org/