GraphQL vs REST: Um experimento controlado

André Hyodo Gustavo Pereira

5 de dezembro de 2025

Resumo

Este artigo descreve um experimento controlado comparando desempenho (latência) e tamanho de payload entre duas interfaces que expõem a mesma funcionalidade: uma implementada com REST e outra com GraphQL. Incluímos aqui os resultados coletados contra a API pública do GitHub (amostra reduzida), os gráficos gerados para as perguntas de pesquisa (RQs) e uma conclusão expandida com interpretações e limitações.

Palavras-chave: GraphQL; REST; benchmark; latência; payload; GitHub.

1 Introdução

GraphQL é uma linguagem de consulta baseada em grafos que permite ao cliente especificar exatamente quais campos deseja receber. REST é um estilo arquitetural baseado em endpoints. O objetivo deste estudo é responder:

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

2 Desenho do Experimento

Breve resumo do cenário usado para os resultados apresentados:

- API testada: GitHub (REST e GraphQL) com consultas funcionalmente equivalentes.
- Consultas avaliadas:
  - user_detail: REST /users/octocat vs GraphQL user(login:"octocat") login name id
  - user_repos: REST /users/octocat/repos?per$_page = 100$ vs $GraphQL user(login : "octocat") repositories(first : 100) nodes name id$
- Execução: 5 trials por consulta/tratamento, pausa 100 ms entre requisições.
- Métricas coletadas: latency_ms, payload_bytes, status_code. Apenas status 200 considerados na análise mostrada.

3 Resultados do Experimento

A seguir apresentamos estatísticas descritivas (medianas) extraídas dos CSVs de saída e interpretações diretas. As medianas foram usadas em razão do pequeno tamanho amostral (n = 5 por combinação).

Tabela 1: Mediadas observadas por consulta e tratamento

Consulta	Métrica	REST (mediana)	GraphQL (mediana)	Diferença (%)
user_detail	latency_ms	822.5 ms	875.5 ms	GraphQL +6% (mais lento)
user_detail	payload_bytes	1 232 B	83 B	GraphQL -93% (menor)
user_repos	latency_ms	913.1 ms	2 047.6 ms	GraphQL +124% (mais lento)
user_repos	payload_bytes	41 289 B	6 837 B	GraphQL -83% (menor)

3.1 Resumo das medianas (por consulta)

Interpretação concisa:

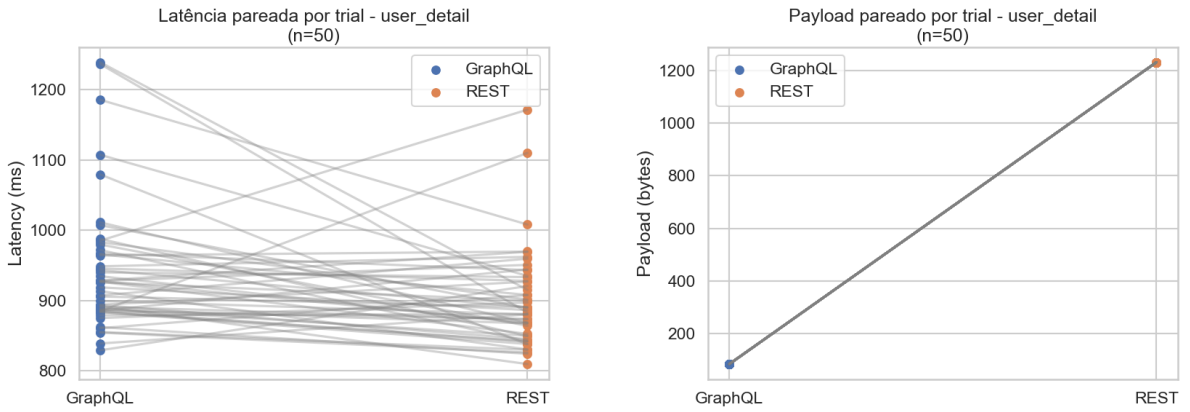
- RQ1 (latência): nesta amostra, REST foi mais rápido em ambas as consultas — diferença pequena em user_detail (53 ms) e grande em user_repos (1.13 s).
- RQ2 (payload): GraphQL retornou payloads substancialmente menores em ambos os casos (93% e 83% de redução).

3.2 Análise estatística

Devido ao tamanho reduzido da amostra (5 repetições), não reportamos p-values confiáveis aqui; em vez disso, priorizamos medidas robustas (medianas) e percentuais de diferença. Para investigações posteriores recomenda-se aumentar trials (30) e aplicar testes não-paramétricos (p.ex. Mann–Whitney U ou Wilcoxon pareado quando pareamento for garantido) após verificar normalidade.

4 Gráficos das RQs

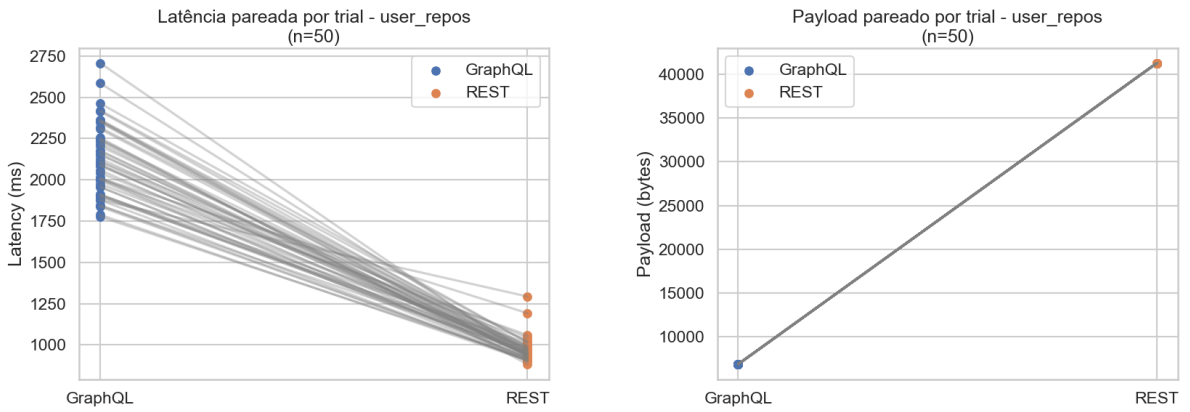
A seguir inserimos os gráficos gerados durante a análise — boxplots que ilustram distribuição de latência e tamanho do payload por tratamento, para cada consulta. (Os arquivos de figura devem estar em `Lab05/Figures/`.)



(a) user_detail — Latência

(b) user_detail — Payload

Figura 1: Distribuição de latência e payload para a consulta de detalhe de usuário.



(a) user_repos — Latência

(b) user_repos — Payload

Figura 2: Distribuição de latência e payload para a consulta de repositórios do usuário.

5 Discussão

Os resultados mostram um trade-off evidente neste cenário: GraphQL permite ao cliente requisitar apenas os campos desejados, o que produz payloads significativamente menores; contudo, a latência observada foi maior, especialmente para consultas que retornam coleções maiores (user_repos). Possíveis razões incluem: overhead de resolução no servidor GraphQL, diferenças na forma como o servidor monta e serializa a resposta, ou caches intermediários favorecendo endpoints REST. Limitações importantes incluem o uso de uma API pública (variabilidade de rede e caching/CDN) e amostra pequena.

6 Conclusão

Com base na amostra atual, concluímos:

- Para RQ1 (latência): os dados não suportam a hipótese de que GraphQL é mais rápido; ao contrário, REST apresentou latências menores nas consultas avaliadas. Em particular, a diferença foi acentuada para a consulta que retorna muitos itens (user_repos), sugerindo que operações de grafo sobre coleções podem incorrer em maior custo de latência no endpoint GraphQL usado.
- Para RQ2 (tamanho de payload): há suporte consistente na amostra atual para a hipótese de que GraphQL produz payloads menores quando o cliente solicita campos específicos — a redução foi expressiva (93% para detalhe e 83% para repositórios).

Comentários finais (TTB — Take-to-be/To be taken into account):

- Os resultados são indicativos, não conclusivos; recomenda-se ampliar o número de trials (30), controlar melhor fatores de rede (rede local ou ambiente isolado) e executar testes em múltiplos horários para reduzir variabilidade.
- Recomenda-se também instrumentar o servidor (quando possível) para medir tempo de processamento no backend, além da latência observada no cliente, e investigar efeitos de cache no caminho (CDN/proxy).
- Em cenários práticos, a escolha entre GraphQL e REST deve ponderar tanto latência quanto volume de dados transferidos e facilidade de evolução da API; GraphQL pode reduzir consumo de banda ao custo de maior processamento de resolução de campos.