Pontifícia Universidade Católica de Minas Gerais

ICEI - Instituto de Ciências Exatas e Informática

Engenharia de Software

Júlia Vidal e Leandra Ramos

GraphQL vs REST - Um experimento controlado

Belo Horizonte

2025

SUMÁRIO

Análise de Fatores Determinantes da Felicidade Mundial: Um Estudo Exploratório com Power BI (2019)

1. Introdução

A arquitetura de APIs Web tem evoluído significativamente, com o estilo REST (Representational State Transfer) sendo o padrão dominante por anos, baseado em endpoints e verbos HTTP. No entanto, a complexidade de sistemas modernos motivou o surgimento do GraphQL, proposto pelo Facebook, que permite consultas baseadas em grafos e esquemas fortemente tipados, onde o cliente define a estrutura da resposta.

Apesar da crescente adoção do GraphQL e da promessa de otimização de dados, nem sempre são claros os benefícios quantitativos em comparação ao REST. O objetivo deste laboratório foi realizar um experimento controlado para avaliar quantitativamente o desempenho dessas duas abordagens. O estudo foi guiado pelas seguintes Perguntas de Pesquisa (Research Questions):

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

Hipóteses

Para cada pergunta de pesquisa, foram formuladas as seguintes hipóteses:

- H0 (Nula): Não há diferença significativa de tempo ou tamanho de resposta entre as tecnologias.
- H1 (Alternativa): Existe diferença significativa de desempenho entre REST e GraphQL

2. Metodologia

2.1. Objetos e Variáveis

Usamos a Rick and Morty API, uma API pública que fornece dados sobre personagens da série animada, suportando nativamente tanto endpoints REST quanto GraphQL.

Variável Independente: O tipo de arquitetura da API (Níveis: REST vs. GraphQL).

Variáveis Dependentes:

- Tempo de Resposta (ms): O tempo decorrido entre o envio da requisição e o recebimento completo da resposta.
- Tamanho da Resposta (Bytes): O tamanho total do payload (corpo da resposta) retornado pelo servidor.

2.2. Configuração do Experimento (Coleta de Dados)

Para a coleta de dados, foi desenvolvido um script automatizado na linguagem Python, utilizando as bibliotecas requests para chamadas HTTP, time para medição de latência e pandas para estruturação dos dados.

O experimento consistiu em 50 rodadas (iterações) sequenciais. Em cada rodada, o script executou as seguintes operações:

1. Requisição REST:
    - Foi realizada uma chamada GET ao endpoint /api/character.
    - Cenário: Esta chamada simula o comportamento padrão (e muitas vezes ineficiente) do REST, onde a API retorna todos os campos disponíveis do personagem (ex: id, nome, status, espécie, gênero, origem, localização, imagem, lista de episódios, url, data de criação), resultando em Overfetching.
2. Requisição GraphQL:
    - Foi realizada uma chamada POST ao endpoint /graphql.
    - Cenário: Foi enviada uma query específica solicitando apenas os campos name e status dos personagens. Isso simula o cenário ideal

do GraphQL, onde o cliente evita o Overfetching buscando apenas os dados necessários para a interface.

As medições de tempo foram calculadas subtraindo o timestamp de início do timestamp de fim da requisição (fim - inicio), convertendo o resultado para milissegundos. O tamanho foi extraído diretamente do comprimento do conteúdo da resposta (len(resp.content)). Os dados brutos foram salvos em um arquivo CSV (dados_experimento.csv).

2.3. Análise dos Dados

Após a coleta, os dados foram processados utilizando a biblioteca Pandas para gerar estatísticas descritivas e comparativas. A análise focou em:

- Estatística Descritiva: Cálculo de contagem, média, desvio padrão, mediana, mínimo e máximo para ambas as métricas (Tempo e Tamanho), agrupadas por tecnologia.
- Comparação Relativa:
    - Para o Tempo, calculou-se a diferença percentual entre as médias para determinar qual tecnologia foi mais rápida e em qual proporção.
    - Para o Tamanho, calculou-se a razão entre a média do REST e a média do GraphQL para quantificar quantas vezes o payload REST foi maior que o GraphQL.

3. Resultados e Discussão

Nesta seção, são apresentados os resultados das análises exploratórias realizadas sobre o dataset. O objetivo é responder às Questões de Pesquisa (RQs) formuladas para entender o que compõe a felicidade de uma nação.

3.1. Indicadores Consolidados de Desempenho

Figura 1 - Figura 1 (Cartões de KPI)

Fonte: Elaborado pelos autores (2025).

Os cartões apresentados consolidam os dados brutos das 50 rodadas do experimento através de médias aritméticas, permitindo uma comparação direta entre as abordagens. As métricas foram definidas da seguinte forma:

- % Redução Tamanho: Esta métrica relativa quantifica o ganho de eficiência no uso da rede. Foi calculada subtraindo a média do tamanho do payload GraphQL da média do REST, dividindo o resultado pela média do REST (fórmula: Média Rest - Média GraphQL / Média Rest). O resultado percentual indica quanto de largura de banda foi "economizada" ao optar pelao GraphQL.
- Média de Tempo (REST e GraphQL): Representam a latência média absoluta, calculada pela média aritmética do tempo de resposta (Tempo(ms)) de todas as iterações, com filtro de contexto aplicado especificamente para cada tecnologia (Tecnologia = "REST" e Tecnologia = "GraphQL").

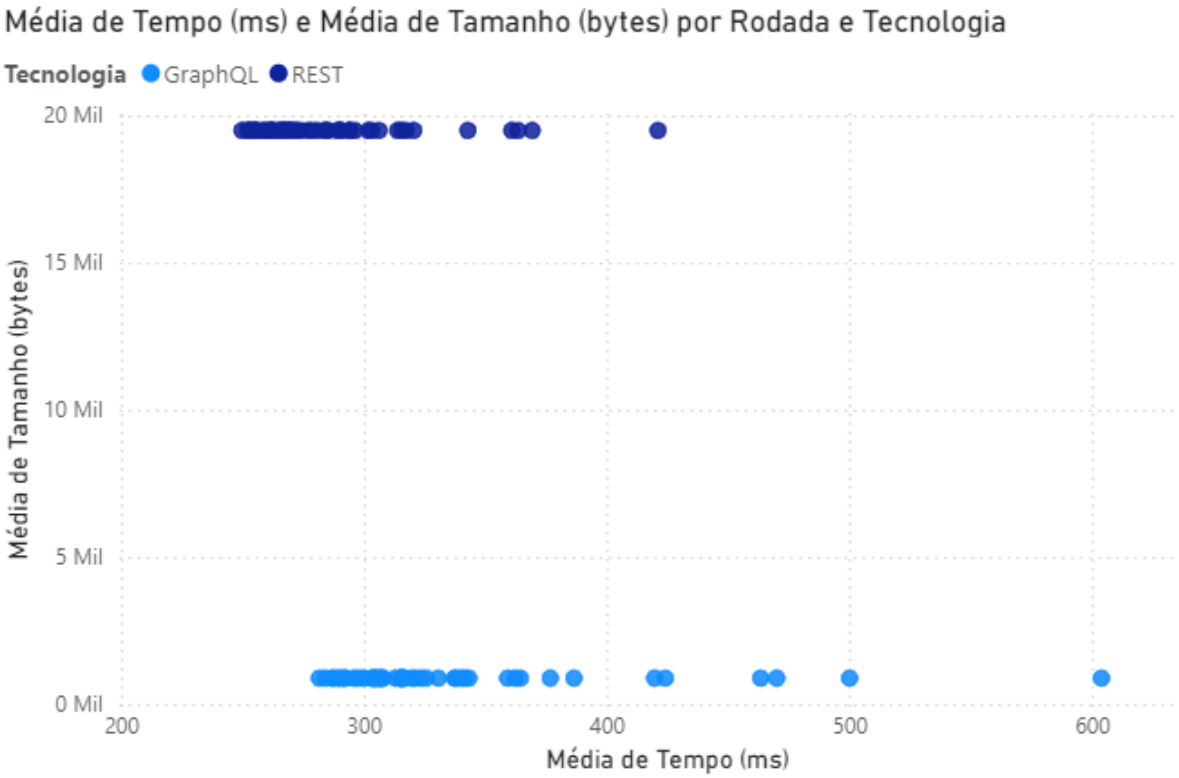
O resultado 95,47% demonstra que a consulta GraphQL foi capaz de satisfazer a necessidade de informação do cliente consumindo menos de 5% da largura de banda exigida pelo endpoint REST. Enquanto a abordagem REST obrigou o tráfego de um objeto JSON completo e redundante, a abordagem GraphQL permitiu a extração cirúrgica dos dados, validando sua superioridade em cenários de rede restrita.

Por outro lado, a comparação entre os indicadores de tempo revela uma inversão de expectativas em relação à primeira pergunta de pesquisa (RQ1). Ao contrastar a Média de Tempo REST de 286,01 ms com a Média de Tempo GraphQL de 336,15 ms, observa-se que a tecnologia REST foi aproximadamente 17% mais rápida. O que sugere que, para consultas de baixa complexidade, a economia de tempo no download obtida pelo GraphQL não foi suficiente para compensar o overhead computacional introduzido pelo servidor para processar a query dinâmica. Portanto, os indicadores mostram que o REST, apesar de trafegar um volume de dados significativamente maior, manteve-se mais performático devido à simplicidade de seu processamento no lado do servidor.

3.2. Correlação entre Tempo de Resposta (ms) e Tamanho do Payload (Bytes)

Enquanto a **Figura 1** apresentou os valores médios consolidados, a **Figura 2** aprofunda a investigação ao expor a distribuição individual das 50 rodadas de medição, permitindo visualizar a correlação direta entre o tempo de resposta (eixo X) e o volume de dados trafegados (eixo Y). A construção deste gráfico de dispersão utilizou a variável Rodada como elemento de detalhe, plotando cada iteração individualmente, enquanto as médias de Tempo(ms) e Tamanho(bytes) definiram o posicionamento cartesiano de cada ponto, categorizados pela legenda de Tecnologia.

Figura 2 - Tempo de Resposta (ms) e Tamanho do Payload (Bytes)



Fonte: Elaborado pelos autores (2025).

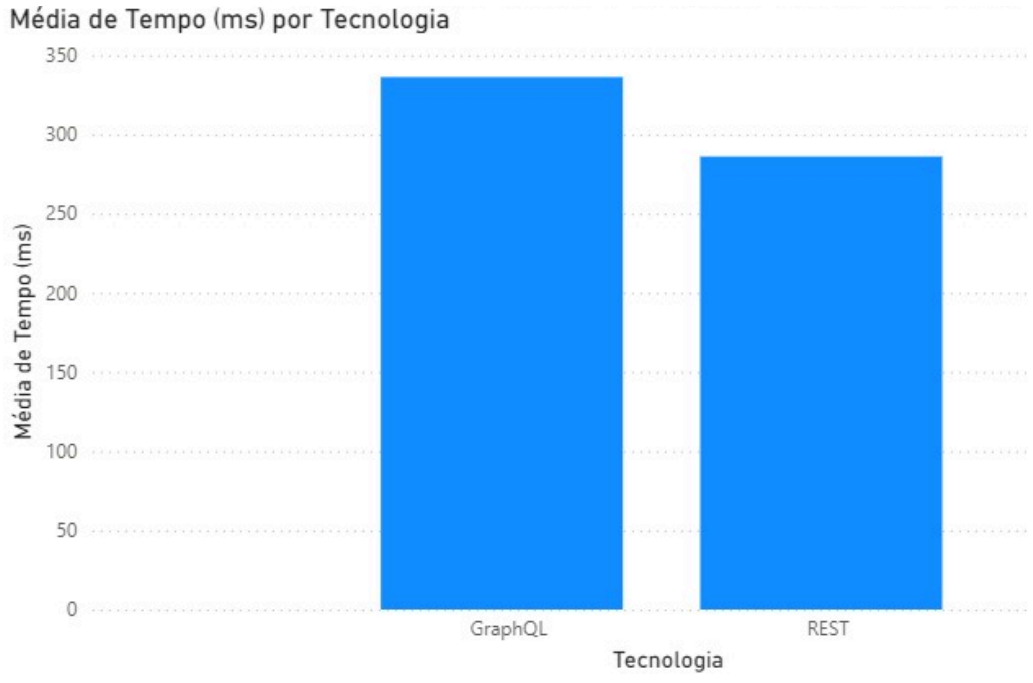
Na parte superior do gráfico, observa-se o agrupamento referente às requisições REST, formando uma linha horizontal quase perfeita em um patamar elevado do eixo Y. Isso confirma que o tamanho da resposta REST é fixo e invariável (~19 KB), independentemente de oscilações na rede ou no servidor; mas, sua posição elevada denuncia o custo de payload que penaliza a largura de banda.

O agrupamento referente ao GraphQL forma quase que uma linha horizontal na base do gráfico, visualmente colada ao eixo X, o que ilustra a eficiência extrema no tamanho da resposta (~883 bytes), que se mantém constante em todas as rodadas. A comparação entre as duas linhas horizontais (REST no topo e GraphQL na base) evidencia a magnitude da diferença de consumo de dados. Já a dispersão horizontal dos pontos ao longo dessas linhas permite analisar a RQ1: nota-se que, apesar da leveza do GraphQL, seus pontos estão distribuídos em uma faixa de tempo ligeiramente superior à do REST, reforçando que a redução drástica de tamanho não se traduziu proporcionalmente em redução de latência, devido ao custo de processamento da query.

3.3. Distribuição Estatística e Estabilidade do Tempo de Resposta

A **Figura 3** sintetiza o comportamento temporal das arquiteturas através de um gráfico de colunas clusterizado, projetado para facilitar a comparação direta da latência. Neste visual, a variável Tecnologia define o eixo horizontal, separando as amostras em duas barras, e a altura de cada coluna é determinada pela Média de Tempo(ms), oferecendo uma representação visual clara da "velocidade" de cada abordagem.

Figura 3 - Distribuição Estatística e Estabilidade do Tempo de Resposta



Fonte: Elaborado pelos autores (2025).

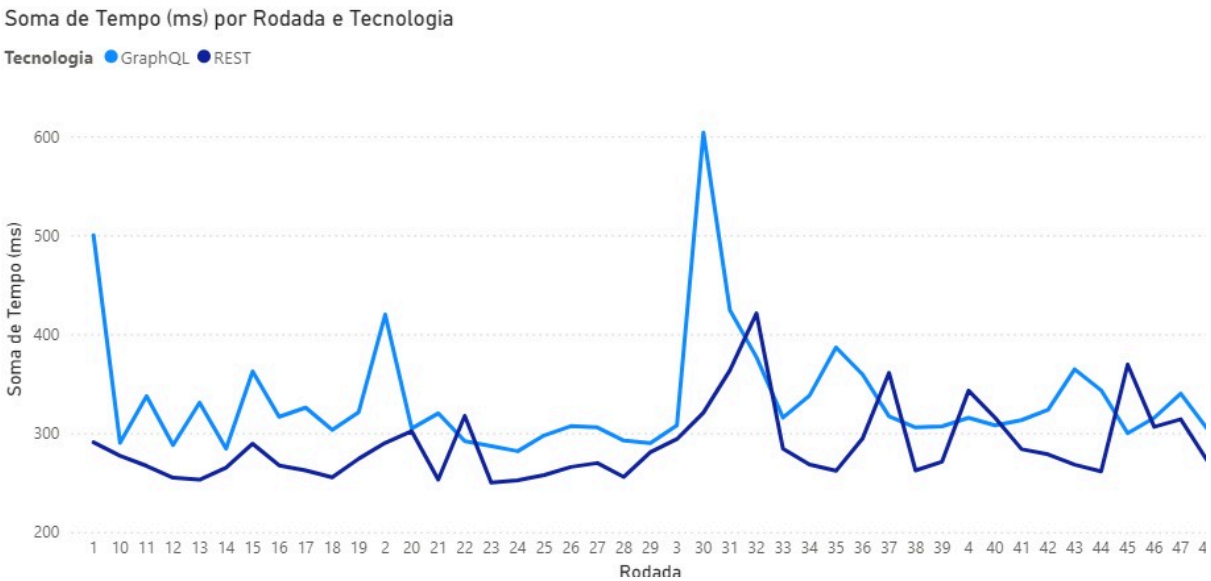
A leitura do gráfico revela uma diferença de altura perceptível entre as colunas, onde a barra referente ao GraphQL (à esquerda) atinge o patamar de 336 ms, superando visualmente a barra do REST (à direita), que atinge a marca de 286 ms. Essa representação gráfica confirma que o REST manteve um desempenho superior em termos de tempo de resposta. A diferença de altura entre as colunas ilustra o "custo" adicional do GraphQL: embora a barra do GraphQL não seja exorbitantemente maior, ela é consistentemente mais alta, quantificando o overhead de processamento que discutimos anteriormente.

Ao analisar este resultado em conjunto com as figuras precedentes, o gráfico de colunas serve como a "prova final" para a RQ1. Enquanto o gráfico de dispersão (**Figura 2**) mostrou que os pontos do GraphQL estavam "espalhados" mais à direita, a **Figura 3** consolida essa dispersão em um valor único e comparável. A menor estatura da coluna REST reforça a conclusão de que, para operações de leitura simples onde o volume de dados não é crítico, a arquitetura REST oferece uma vantagem competitiva de performance, entregando a resposta ao cliente com menor atraso global do que o motor de execução do GraphQL.

3.4. Estabilidade Temporal e Flutuação de Latência por Rodada

Para concluir a análise de desempenho, a **Figura 4** apresenta a evolução temporal das medições ao longo das rodadas do experimento, permitindo examinar a consistência das respostas de cada tecnologia em um cenário de execução contínua. Neste gráfico de linhas, o eixo horizontal representa a sequência cronológica das execuções (Rodada), enquanto o eixo vertical plota a Soma de Tempo (ms) registrada em cada iteração, com as séries diferenciadas pela legenda de Tecnologia. A visualização em linha foi usada para identificar padrões de comportamento intermitente e picos de latência que poderiam ser mascarados em médias simples.

Figura 4 - Gráfico de Linha / Evolução Temporal



Fonte: Elaborado pelos autores (2025).

A análise das oscilações revela uma disparidade significativa na estabilidade das duas arquiteturas. A linha representativa do REST, embora apresente variações naturais de rede, mantém-se confinada em um intervalo relativamente estreito, oscilando entre um mínimo de 264,34 ms e um máximo de 369,35 ms. Essa amplitude reduzida demonstra que o endpoint REST oferece um comportamento robusto e previsível, onde o desvio entre a melhor e a pior execução é contido, garantindo uma experiência de uso homogênea.

Em contraste, a linha referente ao GraphQL exibe um comportamento muito mais volátil e "nervoso", caracterizado por picos e quedas acentuados que rompem a constância da latência. A amplitude de variação do GraphQL é drasticamente maior, estendendo-se de um mínimo de 291,84 ms até atingir picos de 603,95 ms. Esse valor máximo, que é quase o dobro da média da própria tecnologia, atua como um outlier severo, indicando que o motor de execução do GraphQL está sujeito a momentos de instabilidade ou "engasgos" no processamento da query. Portanto, esta figura consolida a conclusão de que o REST não apenas foi mais rápido na média (como visto na **Figura 3**), mas também significativamente mais confiável e estável ao longo do tempo, enquanto o GraphQL apresentou uma imprevisibilidade que pode ser crítica em sistemas sensíveis a latência.

4. Conclusão

O presente experimento controlado atingiu seu objetivo de avaliar quantitativamente as diferenças de desempenho entre as arquiteturas REST e GraphQL, fornecendo evidências empíricas para orientar a escolha tecnológica em projetos de software. A análise dos dados coletados permite responder de forma conclusiva às perguntas de pesquisa formuladas no desenho do estudo.

Quanto à RQ1 ("Respostas às consultas GraphQL são mais rápidas que respostas às consultas REST?"), os resultados obtidos refutam a hipótese de que a menor transferência de dados implicaria necessariamente em maior velocidade. Pelo contrário, o experimento demonstrou que o REST foi consistentemente mais rápido, apresentando uma latência média inferior (~286 ms) em comparação ao GraphQL (~336 ms). Conclui-se que, para consultas simples onde o volume de dados não é proibitivo, o custo computacional (overhead) exigido pelo motor do GraphQL para interpretar a query e montar a resposta supera o ganho de tempo obtido na rede. O REST, por sua natureza estática e direta, mostrou-se mais eficiente em termos de tempo de processamento.

Quanto à RQ2 ("Respostas às consultas GraphQL têm tamanho menor que respostas às consultas REST?"), a resposta é afirmativa e os resultados são contundentes. O GraphQL proporcionou uma redução de 95,47% no tamanho do payload, trafegando apenas ~883 bytes contra os ~19 KB do REST. Isso comprova a eficácia superior do GraphQL na gestão de dados, eliminando completamente o problema de Overfetching ao permitir que o cliente solicite apenas os campos estritamente necessários (name e status), enquanto o REST obrigou o tráfego de objetos completos e redundantes.

Em suma, o experimento revela um claro trade-off arquitetural: o GraphQL deve ser a escolha preferencial em cenários onde a largura de banda é o recurso escasso (redes móveis, IoT) ou onde a flexibilidade do cliente é prioritária. Já o REST permanece como a opção mais robusta para cenários que exigem latência mínima, previsibilidade de desempenho e cacheamento simplificado, onde o volume de dados extra não impacta a experiência do usuário tanto quanto o tempo de resposta.

5. Referências

Qual é a diferença entre GraphQL e REST? Disponível em: https://aws.amazon.com/pt/compare/the-difference-between-graphql-and-rest/ . Acesso em: dezembro 2025.

KAGGLE. **World Happiness Report Dataset**. Disponível em: https://www.kaggle.com/. Acesso em: dezembro 2025.

CHA, Alan et al. **A principled approach to GraphQL query cost analysis**. In: ACM JOINT MEETING ON EUROPEAN SOFTWARE ENGINEERING CONFERENCE AND SYMPOSIUM ON THE FOUNDATIONS OF SOFTWARE ENGINEERING, 28., 2020, Virtual Event. Proceedings [...]. New York: ACM, 2020. p. 257-268. Disponível em: https://dl.acm.org/doi/abs/10.1145/3368089.3409670 . Acesso em: dezembro 2025.

MAVROUDEAS, Georgios et al. **Learning GraphQL Query Cost**. In: IEEE/ACM INTERNATIONAL CONFERENCE ON AUTOMATED SOFTWARE ENGINEERING (ASE), 36., 2021, Evento Virtual. Proceedings [...]. Piscataway: IEEE, 2021. p. 1146-1157. DOI: 10.1109/ASE51524.2021.9678513. Disponível em: https://ieeexplore.ieee.org/document/9678513. Acesso em: dezembro 2025.

BRITO, Gleison; VALENTE, Marco Tulio. REST vs GraphQL: A Controlled Experiment. In: IEEE INTERNATIONAL CONFERENCE ON SOFTWARE ARCHITECTURE (ICSA), 2020, Salvador. Proceedings [...]. Piscataway: IEEE, 2020. p. 81-91. DOI: 10.1109/ICSA47634.2020.00016. Disponível em: https://ieeexplore.ieee.org/document/9101226. Acesso em: dezembro 2025