Relatório Final

Comparação de Desempenho entre APIs REST e GraphQL

Pedro Franco
Engenharia de Software
Laboratório de Experimentação de Software

Dezembro de 2024

1 Introdução

Este experimento investiga diferenças de desempenho entre APIs REST e GraphQL utilizando a Rick and Morty API. GraphQL permite que clientes especifiquem exatamente quais dados necessitam, enquanto REST retorna conjuntos pré-definidos, frequentemente causando *over-fetching* (dados desnecessários) ou *under-fetching* (múltiplas requisições).

1.1 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?

1.2 Hipóteses

Para RQ1 (Tempo de Resposta):

- H_0: $\mu_{GraphQL} \geq \mu_{REST}$
- H_1: $\mu_{GraphQL} < \mu_{REST}$ (GraphQL é mais rápido)

Para RQ2 (Tamanho da Resposta):

- H_0: $\mu_{GraphQL} \geq \mu_{REST}$
- H_1: $\mu_{GraphQL} < \mu_{REST}$ (GraphQL retorna menos dados)

2 Metodologia

2.1 Desenho Experimental

Tipo: Experimento controlado com desenho pareado (*paired design*)

Variável Independente: Tipo de API (REST ou GraphQL)

Variáveis Dependentes:

1. **Tempo de Resposta:** duração da requisição em milissegundos
2. **Tamanho da Resposta:** bytes transferidos no corpo da resposta

2.2 Objetos Experimentais

API: Rick and Morty API (`https://rickandmortyapi.com`)
Amostra: 50 personagens (IDs 1 a 50)
Justificativa: API pública com implementações equivalentes de REST e GraphQL, estrutura consistente e sem necessidade de autenticação.

2.3 Tratamentos

REST:

- Endpoint: `https://rickandmortyapi.com/api/character/{id}`
- Método: GET
- Retorna objeto completo (∼12 campos)

GraphQL:

- Endpoint: `https://rickandmortyapi.com/graphql`
- Método: POST
- Query: solicita apenas 3 campos (name, species, status)

2.4 Procedimento de Coleta

Estrutura Pareada: Para cada ID de 1 a 50:

1. Executar requisição REST
2. Registrar tempo e tamanho
3. Executar requisição GraphQL
4. Registrar tempo e tamanho

Total de medições: 100 requisições (50 pares)
Fase de warm-up: 10 requisições descartadas para estabilizar conexões

2.5 Ambiente de Execução

- **Linguagem:** Python 3.10
- **Bibliotecas:** requests 2.31.0, pandas 2.1.0
- **Localização:** Belo Horizonte, MG, Brasil
- **Conectividade:** Internet banda larga
- **Coleta:** Mesma máquina, mesma conexão, execução sequencial

2.6 Análise Estatística

1. Teste de Shapiro-Wilk para normalidade das diferenças pareadas
2. Se normalidade confirmada: Teste t pareado
3. Se normalidade violada: Teste de Wilcoxon pareado
4. Nível de significância: $\alpha = 0,05$
5. Cálculo de tamanho do efeito

3 Resultados

3.1 Estatísticas Descritivas

Tabela 1: Resumo dos dados coletados

Métrica	**REST**	**GraphQL**	**Diferença**
Tempo médio	898,59 ms	1076,24 ms	-177,66 ms
Tempo mediano	917,00 ms	1047,00 ms	-130,00 ms
Desvio padrão	142,35 ms	165,28 ms	152,13 ms
Tamanho médio	688 bytes	82 bytes	+607 bytes
Tamanho mediano	494 bytes	81 bytes	+413 bytes
Desvio padrão	426,84 bytes	8,12 bytes	424,95 bytes

3.2 Resposta à RQ1: Tempo de Resposta

Teste de Normalidade (Shapiro-Wilk):

- W = 0,9280, p-valor = 0,0046
- **Conclusão:** Normalidade rejeitada ($p < 0,05$)
- **Decisão:** Usar teste não-paramétrico (Wilcoxon)

Teste de Wilcoxon Pareado:

- W = 149,00, p-valor $< 0,0001$
- Tamanho do efeito: $r = -0,773$ (grande)

Decisão: NÃO REJEITAR H_0
Interpretação: A hipótese de que GraphQL é mais rápido não foi confirmada. Na verdade, REST foi significativamente mais rápido que GraphQL em média 177,66 ms (19,8%), com alta significância estatística (p ¡ 0,0001).

3.3 Resposta à RQ2: Tamanho da Resposta

Teste de Normalidade (Shapiro-Wilk):

- W = 0,4479, p-valor $< 0,0001$
- **Conclusão:** Normalidade fortemente rejeitada
- **Decisão:** Usar teste de Wilcoxon

Teste de Wilcoxon Pareado:

- W = 0,00, p-valor $< 0,0001$
- Tamanho do efeito: $r = -0,872$ (muito grande)

Decisão: REJEITAR H_0 com alta confiança
Interpretação: A hipótese de que GraphQL retorna menos dados foi fortemente confirmada. GraphQL apresentou redução de 88,1% no tamanho médio das respostas (607 bytes de economia), com diferença estatisticamente significativa (p ¡ 0,0001) e efeito muito grande.

3.4 Síntese

Tabela 2: Resumo dos testes de hipóteses

Métrica	Diferença	P-valor	Conclusão
Tempo	REST 19,8% mais rápido	$< 0,0001$	GraphQL mais lento
Tamanho	GraphQL 88,1% menor	$< 0,0001$	GraphQL muito menor

4 Discussão

4.1 Interpretação dos Resultados

Os resultados revelam um **trade-off fundamental** entre as tecnologias:
Tempo de Resposta (RQ1): Contrariando a expectativa inicial, REST foi significativamente mais rápido. Possíveis explicações:

- **Overhead de processamento:** GraphQL requer parsing de query, validação de schema e resolução de campos
- **Otimização da implementação:** Endpoints REST podem ter cache mais agressivo
- **Método HTTP:** GET (REST) é mais simples que POST com payload JSON (GraphQL)

Tamanho da Resposta (RQ2): A hipótese foi fortemente confirmada. GraphQL reduziu drasticamente o volume de dados:

- Economia de 88,1% em banda
- Consistência total: 100% dos casos GraphQL retornou menos dados
- Alta previsibilidade: desvio padrão de apenas 8,12 bytes
- Valida o conceito de eliminação de *over-fetching*

4.2 Implicações Práticas

Quando escolher REST:

- Latência é crítica (aplicações real-time, jogos)
- Consultas completas são comuns
- Simplicidade é prioridade

Quando escolher GraphQL:

- Economia de banda é crítica (mobile, IoT)
- Consultas parciais são comuns
- Múltiplos clientes com necessidades diferentes
- Overhead de 180ms é aceitável

4.3 Limitações

- **API específica:** Resultados dependem da implementação da Rick and Morty API
- **Cenário limitado:** Apenas consultas simples com 25% dos campos
- **Variabilidade de rede:** Medições em rede pública introduziram variação
- **Validade externa:** Não generalizável para queries complexas, mutations ou alta concorrência

4.4 Conclusão

Este experimento demonstra que não existe tecnologia universalmente superior. A escolha entre REST e GraphQL deve considerar:

- **REST:** Excelente para latência baixa (-19,8%)
- **GraphQL:** Excelente para eficiência de banda (-88,1%)

Ambas as hipóteses foram testadas com rigor estatístico. Os resultados fornecem evidências quantitativas de um trade-off real: GraphQL economiza banda mas adiciona latência de processamento. A decisão arquitetural deve ponderar qual métrica é mais crítica para o contexto específico.