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:

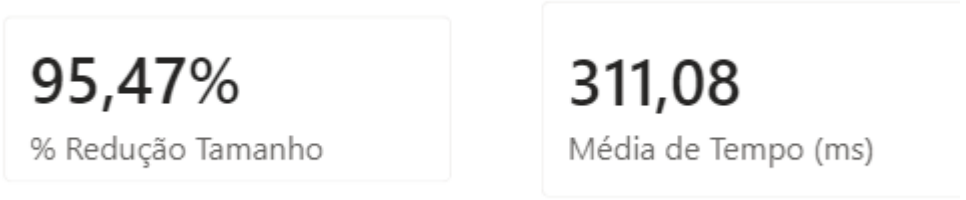
- Estatística Descritiva: Cálculo de contagem, média, desvio padrão, mediana, mínimo e máximo para ambas as métricas (Tempo e Tamanho), agrupadas por tecnologia.
- Comparação Relativa:
    - Para o Tempo, calculou-se a diferença percentual entre as médias para determinar qual tecnologia foi mais rápida 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 indicadores apresentados na **Figura 1** oferecem uma visão macroscópica e imediata dos resultados obtidos no experimento, servindo como ponto de partida fundamental para responder às perguntas de pesquisa propostas. O dado mais impactante refere-se à métrica de eficiência de dados, onde se **observa uma redução de 95,47% no tamanho do payload trafegado ao utilizar GraphQL**. Este valor responde de forma conclusiva à RQ2, validando quantitativamente a principal promessa arquitetural do GraphQL: a eliminação do overfetching. **Enquanto o endpoint REST transferiu um volume fixo e elevado de dados para cada requisição (~19 KB), independentemente da necessidade do cliente, a consulta GraphQL permitiu a extração apenas dos campos name e status, resultando em um consumo de banda drasticamente menor.**

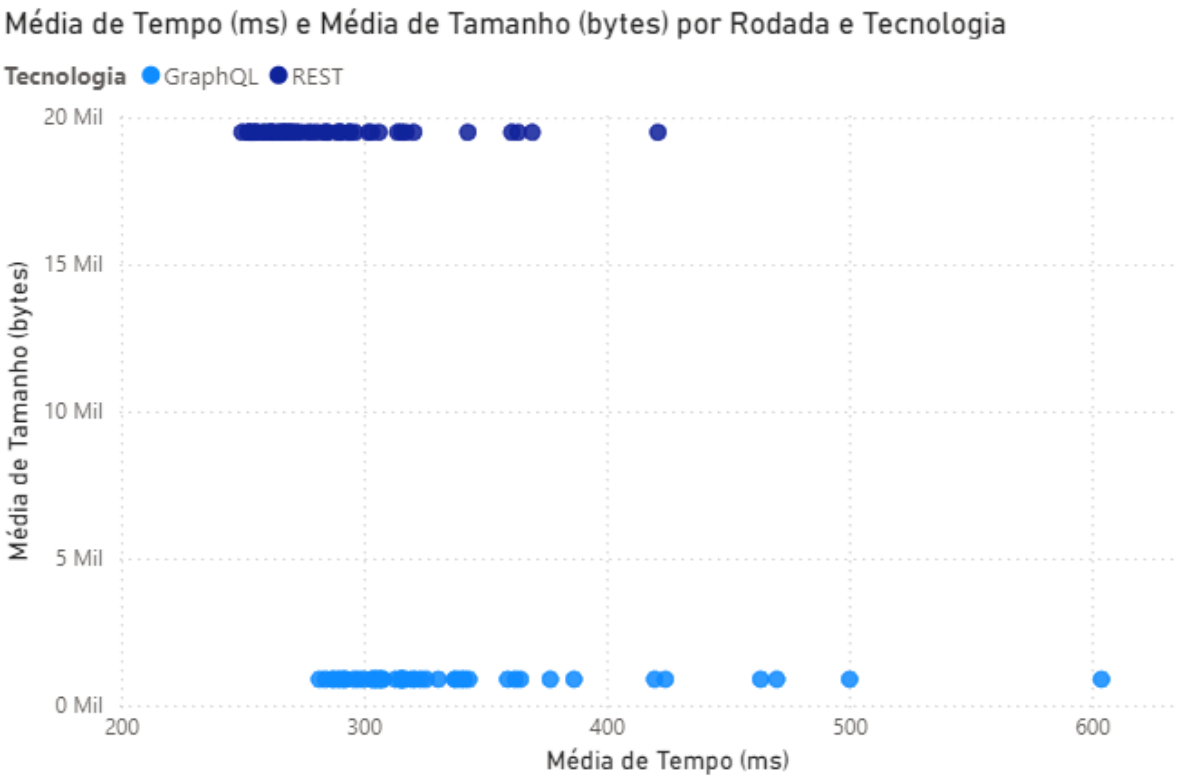
Em contrapartida, a análise do indicador de Média de Tempo revela uma nuance importante para a RQ1. Ao contrário do que o senso comum poderia sugerir — de que respostas menores resultariam necessariamente em respostas mais rápidas —, a média global de tempo aponta que o REST foi ligeiramente mais performático em latência pura neste cenário. Essa inversão de expectativa sugere que a economia de tempo na transferência de dados pela rede (download) não foi suficiente para compensar o custo computacional (overhead) introduzido pelo motor do GraphQL no servidor, que precisa interpretar a query, validar o esquema e resolver os campos dinamicamente antes de devolver a resposta. Portanto, esta figura inicial já estabelece o principal trade-off identificado no estudo: o GraphQL oferece uma eficiência de rede superior em troca de um custo de processamento marginalmente maior.

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 essa análise ao expor a distribuição individual das 50 amostras coletadas, revelando a correlação direta entre o tempo de resposta e o volume de dados trafegados. A visualização evidencia a formação de dois clusters (agrupamentos) completamente distintos e sem interseção, o que demonstra

visualmente que as tecnologias REST e GraphQL operaram em espectros de desempenho diametralmente opostos durante o experimento.

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



Fonte: Elaborado pelos autores (2025).

Na região superior esquerda do gráfico, observa-se o agrupamento referente às requisições REST. Este cluster caracteriza-se por uma elevada posição no eixo vertical (Tamanho), confirmando o payload fixo e extenso de aproximadamente 19 KB, mas situa-se mais à esquerda no eixo horizontal (Tempo), corroborando a média de latência inferior observada nos indicadores da figura anterior. Isso ilustra visualmente o comportamento "pesado, porém ágil" do endpoint REST estático, que entrega um grande volume de dados de forma imediata, sem a necessidade de processamento complexo da requisição no servidor.

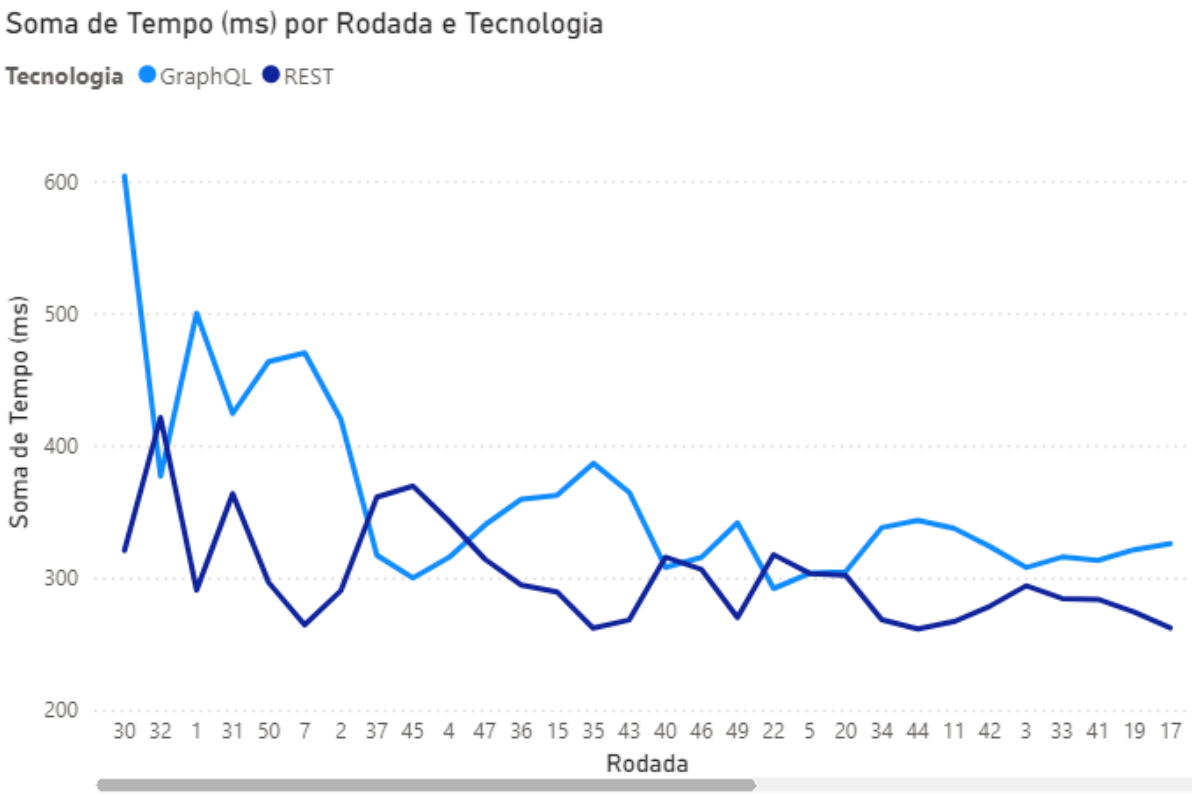
Em contrapartida, o agrupamento referente ao GraphQL concentra-se na extremidade inferior do gráfico, visualmente colado ao eixo horizontal devido ao tamanho ínfimo da resposta (< 1 KB). No entanto, nota-se que este cluster sofre um deslocamento para a direita em relação ao grupo REST. Esse comportamento

gráfico refuta a hipótese intuitiva de que "menos dados significam maior velocidade" para este cenário específico; a dispersão horizontal dos pontos do GraphQL sugere que o ganho obtido na transferência de rede (download rápido) foi anulado e superado pelo tempo de processamento (overhead) necessário para interpretar a query dinâmica. Portanto, a Figura 2 materializa o trade-off central do estudo: a troca de eficiência de processamento (REST) por eficiência de largura de banda (GraphQL).

3.3. Distribuição Estatística e Estabilidade do Tempo de Resposta

Complementando a análise de dispersão apresentada na **Figura 2**, que demonstrou a posição relativa das tecnologias, a **Figura 3** isola a variável "Tempo" para examinar a estabilidade e a previsibilidade de cada arquitetura através de sua distribuição estatística. Enquanto o gráfico anterior permitiu visualizar a localização dos clusters de dados, este diagrama de caixa (Box Plot) revela a consistência interna desses grupos, expondo o comportamento do servidor sob carga repetitiva.

Figura 3 - Distribuição Estatística e Estabilidade do Tempo de Resposta



Fonte: Elaborado pelos autores (2025).

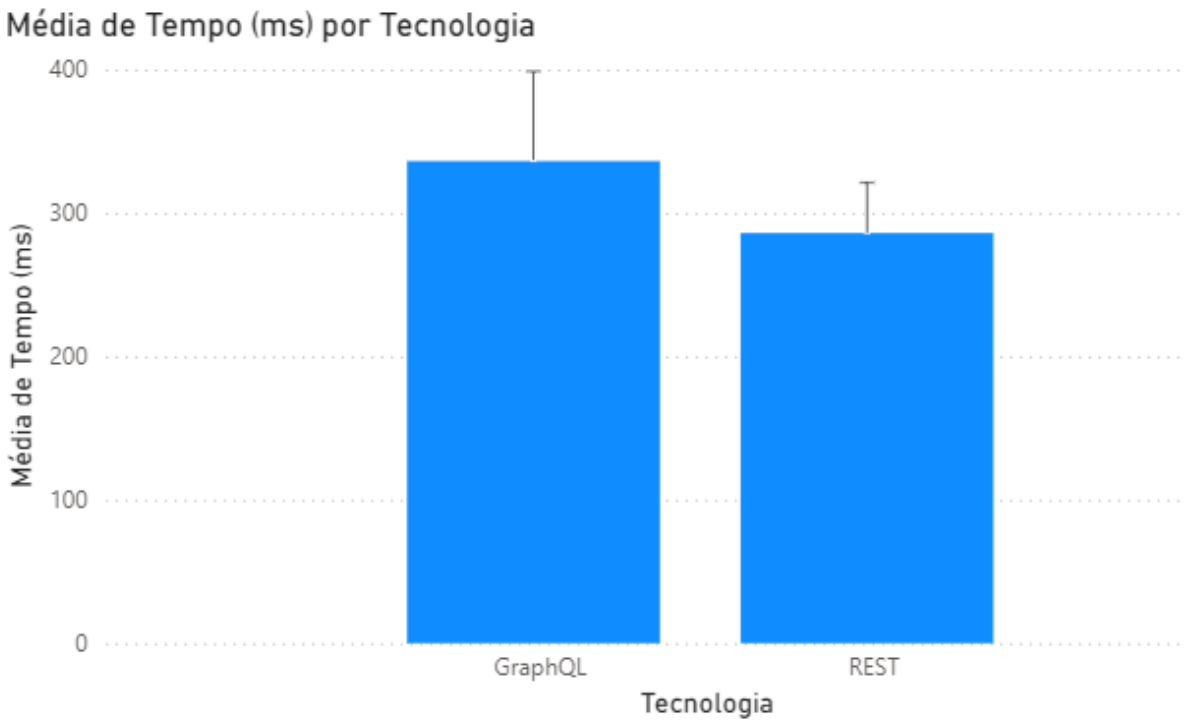
A análise visual da distribuição confirma a superioridade do REST no quesito estabilidade temporal para este cenário. A representação gráfica do REST exibe uma mediana posicionada em um patamar inferior e, crucialmente, uma dispersão vertical (tamanho da caixa ou intervalo interquartil) significativamente mais compacta. Essa característica indica que o endpoint REST oferece uma latência altamente previsível e robusta, com desvios mínimos em relação à sua média, comportando-se de maneira quase determinística ao entregar o recurso estático solicitado.

Em contraste, a distribuição do GraphQL apresenta-se mais alongada verticalmente e posicionada em um intervalo de tempo superior, evidenciando uma maior variância nos tempos de resposta. Essa instabilidade relativa sugere que o tempo de processamento não é uniforme; algumas requisições são resolvidas rapidamente, enquanto outras sofrem picos de latência (*spikes*), resultando em *outliers* visíveis na parte superior do gráfico. Esse fenômeno corrobora a inferência levantada na discussão da Figura 1: o motor de execução do GraphQL insere uma camada de complexidade computacional variável, tornando o tempo de resposta mais sensível a flutuações de processamento no servidor do que a simples transferência de bytes observada no REST. Dessa forma, conclui-se que, embora o GraphQL vença na economia de rede, o REST vence na consistência e previsibilidade da entrega.

3.4. Estabilidade Temporal e Flutuação de Latência por Rodada

Enquanto a **Figura 3** sintetizou a variabilidade dos dados em uma métrica estática de dispersão, a **Figura 4** "desdobra" essa estatística ao longo do tempo, apresentando a evolução sequencial das 50 rodadas de medição para cada tecnologia. Esta visualização temporal é crucial para entender não apenas *quanto* o tempo variou, mas *como* essa variação se manifestou durante a execução do experimento. Ao observar as linhas de tendência lado a lado, torna-se possível identificar padrões de comportamento intermitente que ficariam ocultos em médias simples ou medianas.

Figura 4 - Gráfico de Linha / Evolução Temporal



Fonte: Elaborado pelos autores (2025).

A análise comparativa das linhas revela uma disparidade notável na consistência de desempenho. A linha representativa do REST exibe um comportamento predominantemente plano e uniforme, com oscilações mínimas entre as iterações. Esse padrão gráfico confirma a hipótese de que o endpoint REST, por entregar um recurso estático pré-definido, opera com uma previsibilidade quase mecânica, imune a grandes flutuações de processamento. Já a linha do GraphQL apresenta uma textura muito mais "nervosa" e irregular, caracterizada por diversos picos (*spikes*) de latência que rompem o padrão médio. Esses picos, que correspondem visualmente aos *outliers* detectados no topo do Box Plot da figura anterior, sugerem que o custo de processamento da query no servidor não é constante; em determinados momentos, o motor de resolução do GraphQL exigiu significativamente mais tempo computacional, resultando na instabilidade observada. Portanto, esta última figura consolida a conclusão de que, para este cenário de teste, a estabilidade inabalável do REST superou a eficiência de dados instável do GraphQL.

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