Um Modelo de Comunicação para Automação na Execução de Consultas de Dados sobre APIs Web

Mateus Maso[1], **Frank Siqueira**[2]

[1]Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brazil

mateus.maso@grad.ufsc.br, frank.siqueira@ufsc.br

Abstract. *In order to maintain client-server communication efficiency without the need of versioning Web APIs, services have come across problems while performing changes on their interface access specification due to the coupling caused by clients on the implementation of its fetching code. Seeking to develop a solution for the problem, this project conducts a study on the usage of GraphQL language and API description formats to propose a client-server communication model through automation in the execution of data queries. As a result, a tool foresaw by the proposed model is developed to validate its applicability and guide client-side developers to the implementation of data fetching code independent of API specification.*

Resumo. *A fim de manter a eficiência da comunicação cliente-servidor sem a necessidade do versionamento de APIs Web, serviços têm encontrado dificuldades em realizar mudanças na especificação de sua interface de acesso devido ao acoplamento causado por clientes na implementação em seu código de busca. No intuito de desenvolver uma solução para o problema encontrado, este trabalho realiza um estudo sobre o uso da linguagem GraphQL e formatos de descrição de API para propor um modelo de comunicação cliente-servidor através da automação na execução de consultas de dados. Como resultado, é desenvolvida uma ferramenta prevista pelo modelo proposto para validar sua aplicabilidade e direcionar desenvolvedores de clientes à implementação de um código de busca independente de especificação de API.*

1. Introdução

Em 2005, ocorreu uma grande transição no modelo de comunicação entre aplicações distribuídas, onde estas passaram a utilizar amplamente o protocolo HTTP e o modelo cliente-servidor para a troca de informações na World Wide Web. Hoje, empresas como Facebook e Netflix mostram que construir APIs Web é, não apenas essencial para entrar rápido no mercado de plataformas emergentes, como também um novo meio de agregar valor em seu próprio modelo de negócio e oferecer uma melhor experiência a seus usuários. [Duvander 2013, Art 2016]

Contudo, após anos de sua popularização, serviços têm mostrado dificuldades em realizar mudanças em suas APIs Web sem comprometer a comunicação de clientes. Isso porque clientes continuam implementando seu código de busca através de chamadas diretamente na API, ocasionando um acoplamento da especificação muitas vezes indesejado pelos serviços.

Com o objetivo de resolver o problema de acoplamento, este trabalho propõe um novo modelo de comunicação cliente-servidor através da automação na execução de consultas de dados sobre APIs Web. O modelo visa direcionar desenvolvedores de clientes à implementação de códigos de busca independente de especificação de API, resultando no desenvolvimento de clientes tolerantes às mudanças na especificação. Ainda, prevê o desenvolvimento de uma ferramenta que funciona como intermediadora na comunicação e automação de consultas, permitindo otimização de requisições e composição de serviços para consulta de dados.

2. Desenvolvimento

2.1. Planejamento de Projeto

Para que clientes possam trabalhar com dados remotos, é preciso que haja um meio de buscá-los antes de executar qualquer lógica que dependa deles. Para isso, a solução comumente adotada é a implementação de um código de busca para acesso remoto de dados em APIs de serviços[1]. Contudo, ao passar do tempo, fez-se necessário estabelecer um contrato de acesso entre o cliente e a interface do serviço. Isso porque a atual implementação do código de busca por clientes não prevê mudanças na especificação da API. Este contrato é, portanto, representado no modelo de comunicação da figura 1 através de chamadas entre o código de busca do cliente e a API do serviço.

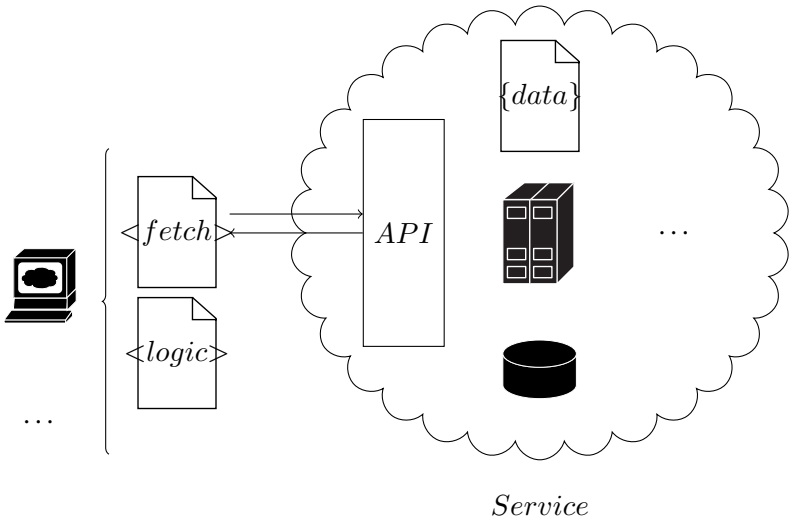

Figura 1. Modelo de comunicação entre cliente e serviço

O modelo de comunicação representado pode não revelar problemas para serviços com pouca demanda de acesso e diversidade de consultas. Contudo, ao longo do tempo e com o aumento na quantidade de contratos, alterações na especificação como, por exemplo, as de fluxo de dados[2] podem se tornar um desafio.

Mudanças no fluxo de dados pela API são inevitáveis em aplicações distribuídas. Elas ocorrem para abraçar a constante transformação por clientes na execução de consul-

[1]Infraestrutura distribuída (servidores, banco de dados, etc) que respondem a pedidos de operações oriundas de clientes em forma de requisições de API.

[2]Fluxo de acesso por clientes para consulta de dados sobre APIs de serviços.

tas de dados. Com isso, busca-se manter uma comunicação eficiente através da redução no número de chamadas executadas na API e do tamanho de dados transmitidos pela rede. Essas mudanças podem ser desde uma simples alteração no nome de um método ou número argumentos, até as mais complexas, como dados de retorno e estilo de arquitetura. A figura 2 ilustra o fluxo de dados entre um cliente e um serviço.

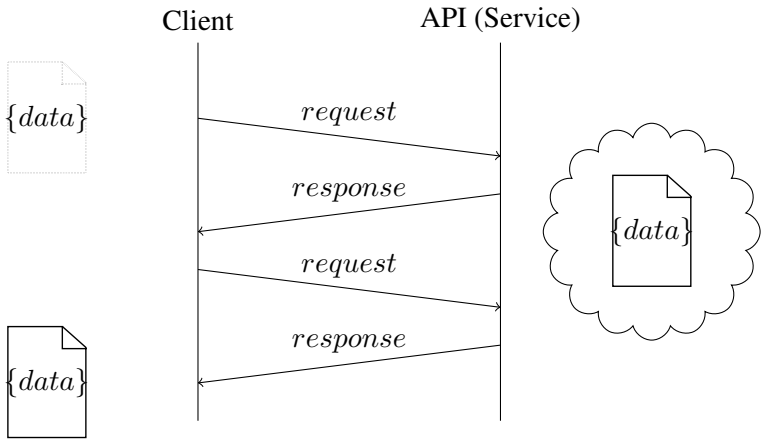

Figura 2. Fluxo de dados entre cliente e API

O impacto causado em clientes por mudanças no fluxo de dados de uma API felizmente é previsível, pois afeta diretamente, em sua maioria, o código de busca. Por outro lado, mudanças como a alteração de campos nas estruturas de dados, como renomear um campo ”nome”para ”nome_completo”, são de um grau de complexidade maior, pois pode não ter impacto no código de busca e sim na lógica da aplicação, onde é bem mais difícil de depurar erros.

2.2. Proposta de Modelo

Um novo modelo é proposto com o intuito de melhorar a comunicação cliente-servidor apresentada. Observado na figura 3, este prevê a criação de uma ferramenta no cliente para a intermediação da comunicação entre o código de busca e a API de serviços. Além disso, há a necessidade de reimplementação do código de busca para uma nova linguagem de consulta que seja interpretada pelo intermediador. Da mesma forma, soma-se a criação de um arquivo no serviço para descrição de metadados da API e outro no cliente para configuração, ambos essenciais para o funcionamento da ferramenta.

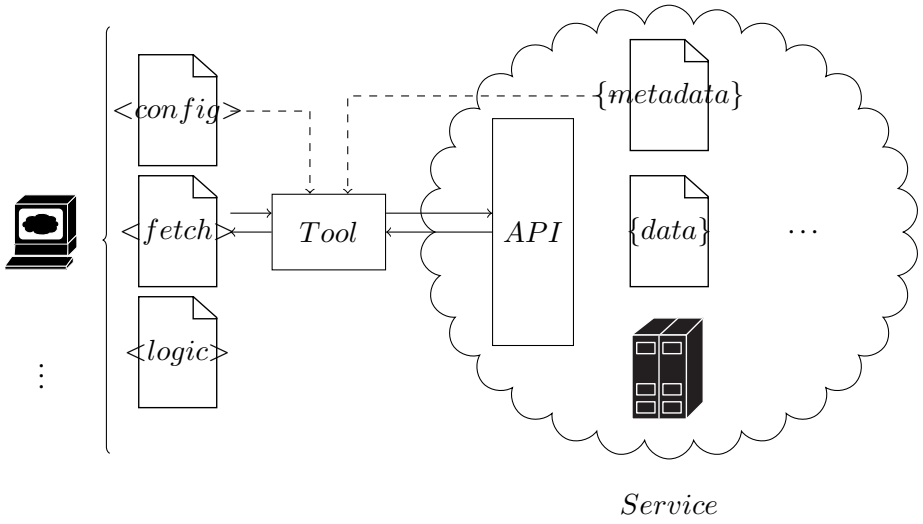

Figura 3. Modelo proposto para evitar contrato de comunicação

A proposta de eliminação de contrato visa também automatizar o acesso de clientes em APIs. Através da implementação de algoritmos na ferramenta, escolhe-se o caminho de acesso de melhor custo-benefício em relação aos serviços disponíveis para o cliente. O modelo proporciona, além disso, um ambiente escalável para consulta de dados através da composição de serviços, visto na figura 4.

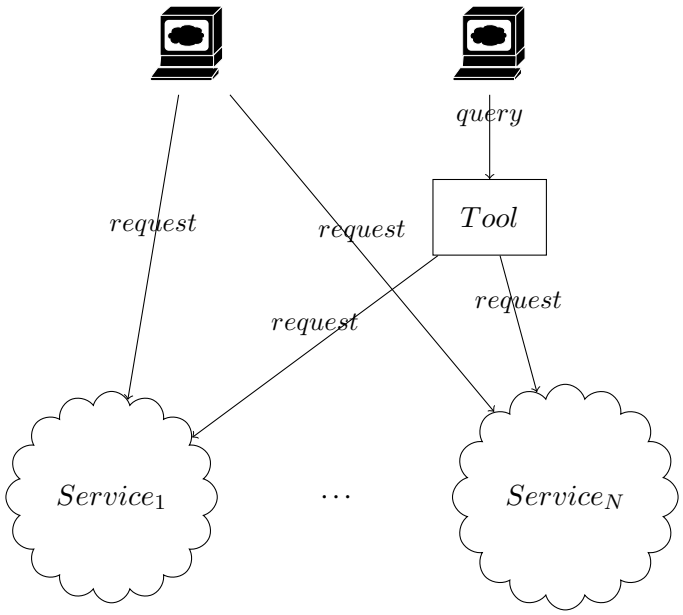

Figura 4. Composição de serviços através da ferramenta

2.3. Especificação da Ferramenta

Visando a aceitação da ferramenta prevista pelo modelo em ambientes de desenvolvimento de clientes, foi preciso pensar em uma especificação que apresentasse uma interface de baixa curva de aprendizagem, um fluxo de execução replicável e agnóstico à

plataforma, além do baixo impacto na base de código de clientes. Por conseguinte, a ferramenta proposta foi desenvolvida pensando na reutilização de tecnologias promissoras ou bem consolidadas no mercado de desenvolvimento.

Com a sucessão de estudos, a escolha foi em utilizar a tecnologia GraphQL como principal biblioteca para ajudar com intermediação da comunicação do modelo proposto. Ao invés da tecnologia Falcor (apontada durante o estudo), GraphQL foi o único que apresentou uma solução robusta que permitisse o acesso de APIs Web por clientes através de consultas em seu código de busca. Da mesma forma, buscou-se trabalhar com formatos abertos de descrição de metadados de APIs, como por exemplo o JSON Hyper-Schema. Através de adaptadores, a ferramenta permite acelerar a integração de serviços que já oferecem metadados em formatos suportados.

2.4. Implementação da Ferramenta

A fim de aplicar a especificação da ferramenta abordada no planejamento de projeto, foi proposta a implementação de onze funções, divididas em três categorias, para o uso de clientes na plataforma Web. Cada uma dessas funções levantadas resolve um problema com o intuito de chegar ao objetivo de representar os dois fluxos de execução. Dentro do conjunto de funções, quatro delas são responsáveis por criar o intermediador; três funções analisam as consultas através do formato AST; e as outras quatro transformam as consultas em requisições para API de serviços. Somente quatro são públicas e expostas para o cliente, sendo uma delas a principal para criação do esquema de consulta, e as outras três para serem sobrescritas por adaptadores.

3. Validação

Com o objetivo de validar o modelo proposto, é realizada uma pesquisa comparativa com foco no impacto da relação do uso da ferramenta implementada e mudanças no fluxo de dados sobre APIs Web. Para isso, é desenvolvido um ambiente de validação onde dois clientes, um com a ferramenta e o outro não, realizam três consultas de dados sobre uma API REST. Por fim, é aplicada uma série de quatro mudanças no fluxo de dados da API e coletados os dados para análise do impacto causado no código de busca desses clientes.

3.1. Resultados

Inicialmente, observa-se na figura 5 que o cliente 1 (sem o uso da ferramenta) não apresenta um bom índice de acerto das respostas. Dentre as quatro mudanças, obteve apenas 58% de acerto, sendo a C3 a única mudança em que conseguiu responder corretamente todas as perguntas, pois não houve mudança nas URIs que estava utilizando. Um resultado esperado, significando que houve quebra de contrato e impacto negativo no código de busca.

Em contrapartida, o cliente 2 (com o uso da ferramenta) apresenta um resultado no índice de acerto da figura 6 36.61% superior ao cliente 1. Com um total de 91,5% de acerto, apenas não completou com 100% de acerto pois a mudança C4 não permitiu que fosse possível acessar todos os dados necessários para a responder da pergunta Q2. Isso representa que a ferramenta foi capaz, através do intermediador, de evitar a criação de contrato e causar impacto negativo ao código de busca.

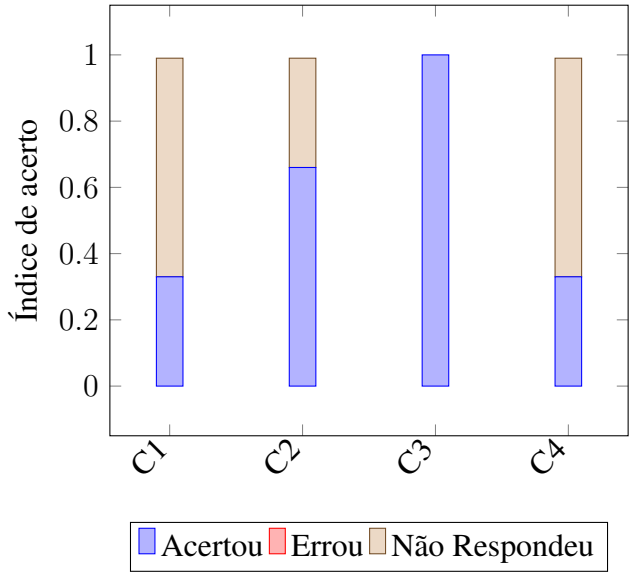

Figura 5. Índice de acerto sem o uso da ferramenta

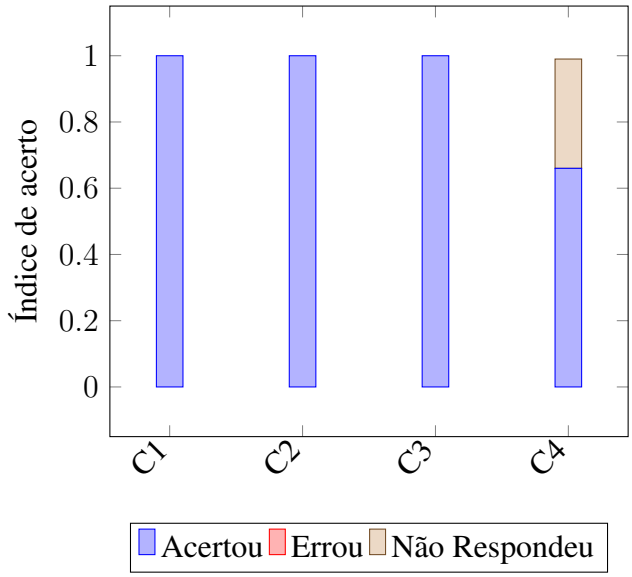

Figura 6. Índice de acerto com o uso da ferramenta

Ao analisar a mudança C3 isoladamente, onde ambos conseguem responder com 100% de acerto, percebe-se nas figuras 7 e 8 que o cliente 2 consegue realizar uma consulta com melhor desempenho (menor número de requisições e tamanho de dados) que o cliente 1. Isso porque, após a mudança e sem alterar o código de busca de dados, a ferramenta consegue remapear as requisições geradas pelas consultas GraphQL graças ao algoritmo que automatiza as chamadas à API.

Outro ponto importante é que, após a mudança C3 e a atualização dos metadados no cliente, o algoritmo da ferramenta percebe que há a possibilidade de realizar menos requisições em busca dos dados para responder as perguntas. Isso resulta em uma redução de aproximadamente 54% dos acessos entre o cliente 1 e cliente 2.

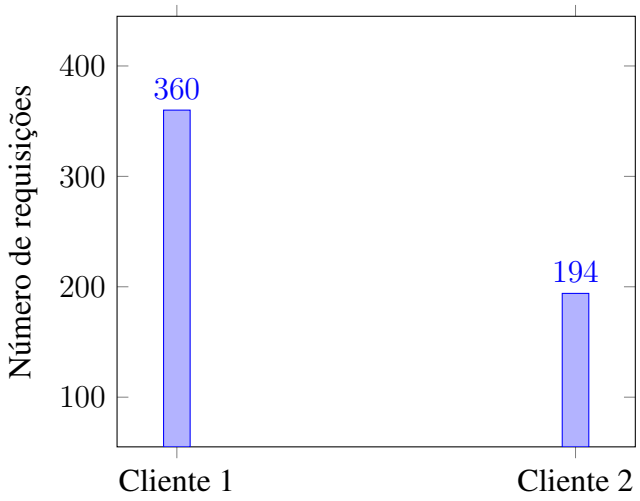

Figura 7. Comparação no número de requisições C3

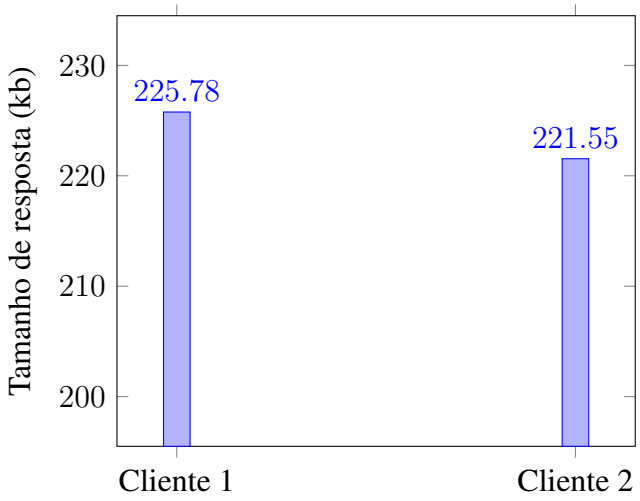

Figura 8. Comparação no tamanho de resposta C3

Apesar dos ganhos de performance e desenvolvimento através do uso da ferramenta vistos anteriormente, percebe-se na figura 9 um outro cenário que indica o lado negativo do seu uso causado pelo tempo de *overhead*[3]. Em média, a ferramenta atrasou em 1.7 segundos a execução na consulta de dados do cliente, sendo o tempo de busca de metadados responsável por somar mais da metade deste atraso inicial.

[3]Processamento em excesso

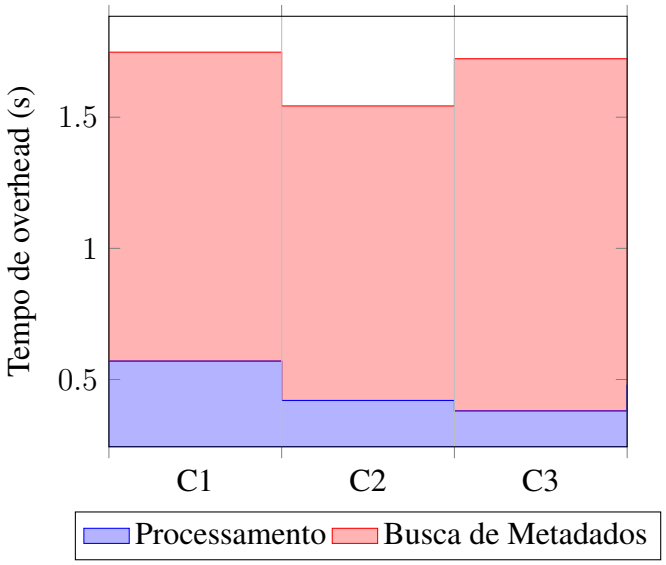

Figura 9. Overhead da ferramenta

4. Conclusão

Tornam-se evidentes, através deste trabalho, as dificuldades de serviços em realizar mudanças na especificação de APIs uma vez que estas já possuem clientes fazendo o seu acesso. Além disso, nota-se que é possível explorar novos modelos de comunicação a fim de manter a eficiência da comunicação cliente-servidor evitando versionamento.

O esforço dedicado neste trabalho em explorar esta área culminou no desenvolvimento de um novo modelo de comunicação. Buscou-se eliminar a preocupação de clientes em estar continuamente atualizando seu código de busca a cada mudança na API e direcionar desenvolvedores de clientes à implementação de códigos de busca independentes de especificação de API.

Apesar das vantagens do modelo, existe um investimento a mais com a integração da ferramenta para que clientes e serviços possam usufruir os benefícios demonstrados nos resultados dos testes de validação. Felizmente, serviços que disponibilizam descrições de metadados da sua API apresentam uma menor barreira de entrada e já podem ser consultados utilizando a ferramenta.

Por fim, uma das principais contribuições deste trabalho é estampar os benefícios da automação na execução de consultas de dados e composição de serviços que o modelo e a ferramenta podem proporcionar. Basta que os desenvolvedores de clientes se preocupem em escrever um código de busca através de linguagens de consulta como o GraphQL e dos serviços em disponibilizarem uma completa descrição dos metadados de APIs.

Referências

Art, A. (2016). Tracking the growth of the api economy — nordic apis —.

Duvander, A. (2013). Json's eight year convergence with xml.