GraphQL versus REST - A Evolução dos Serviços Online

Átila Camurça Alves

16 de setembro de 2017

1 Introdução

A evolução dos computadores fez com que mais formas diferentes de acesso a informação sejam possíveis. Hoje em dia é possível, por exemplo, verificar sua conta de *e-mail* de seu computador pessoal, ou de seu *notebook*, de seu aparelho celular ou ainda de seu *smartwatch*. Cada um desses aparelhos com diferentes tamanhos físicos e diferentes poderes de processamento. Nesse cenário, voltamos um pouco da ideia de *Mainframes* para acesso remoto de arquivos ou serviços, mas agora com novas estruturas e arquiteturas, que ficou conhecido como Nuvem.

Para atender a estes diversos tipos de dispositivos citados acima que acessam os mesmos recursos surge o conceito de Serviços. O que se propõe é tornar o servidor e o cliente independentes, para que seja possível ter acesso aos mesmos recursos do servidor de vários clientes diferentes.

2 Definição

Em 2000, surge o conceito do REST (do inglês *Representational State Transfer*) criado por Roy Thomas Fielding em sua dissertação de Doutorado. Uma dos pontos-chave deste conceito é desacoplar a interface de usuário do armazenamento de dados, o que melhoraria a portabilidade da interface de usuário para múltiplas plataformas [**?**]. Esse padrão se popularizou graças a adoção por empresas como o *Twitter* em 2006, que fornecedia uma API (do inglês *Application Programming Interface*) para que terceiros pudessem fazer algum tipo de integração, como por exemplo *login* com a conta do *Twitter* em seu aplicativo ao invés de criar um sistema de *login* próprio.

Em 2015, o *Facebook* publica em formato *Open Source* o GraphQL, definido como uma *query language* (linguagem de consulta) para *API*'s. Permite que o cliente obtenha os dados de forma declarativa e hierarquica [**?**]. É um formato que adapta-se a qualquer banco de dados ou mecanismo de armazenamento, ou seja, é possível usá-lo para busca de arquivos em um sistema de arquivos remoto. Possui 3 tipos de operações básicas: *Query* para obtenção de dados; *Mutation* para alteração de dados; e *Subscriptions* para notificação em tempo real.

3 Estilo Arquitetônico e Arquitetura

3.1 REST

Em sistemas REST há série de restrições arquiteturais, tais como:

- **Cliente-Servidor**: separar a interface de usuário da lógica da aplicação.
- ***Stateless***: a requisição contém informações suficientes sobre o cliente para que o servidor entenda a requisição.
- ***Cacheable***: requisições devem ser passíveis de serem colocadas em *cache*.
- **Interface Uniforme**: identificação de recursos é feita através de URI e sua implementação é feita de forma abstrata da definição da interface.
- **Sistema em Camadas**: um cliente não deverá distinguir se está conectado ao servidor final ou a um intermediário.

Seu estilo arquitetônico se encaixa em uma arquitetura em camadas.

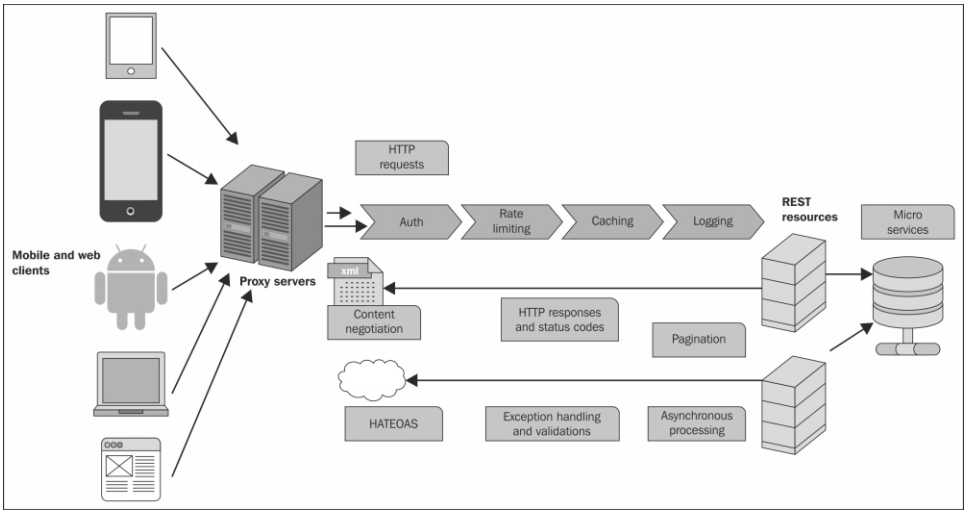

Figura 1: REST

3.2 GraphQL

Em sistemas GraphQL podem haver diversos tipos de arquiteturas diferentes, entre elas:

- Servidor GraphQL conectado a um banco de dados, como visto na Figura 2.
- Servidor GraphQL atuando como camada intermediária entre sistemas legados ou ainda de terceiros integrando-os através de uma única API GraphQL, como visto na Figura 3.
- Uma abordagem híbrida com a junção das 2 abordagem supracitadas, como visto na Figura 4.

Seu estilo arquitetônico é híbrido entre arquitetura em camadas e baseada em eventos, esta última utilizada para notificações em tempo real.

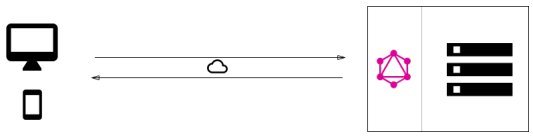

Figura 2: GraphQL com banco de dados

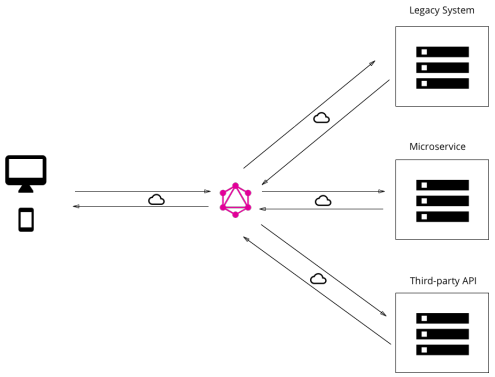

Figura 3: GraphQL com sistemas de terceiros ou legados

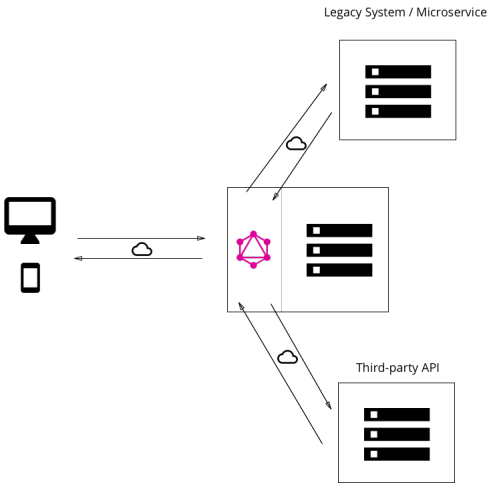

Figura 4: GraphQL híbrido

4 Comunicação

4.1 REST

A comunicação é feita através do protocolo HTTP, em que é recomendado o uso explícito dos métodos do HTTP para operações: criar, ler, atualizar e deletar, como é mostrado a seguir:

- Para criar um recurso no servidor, utilize o método `POST`;
- Para resgatar um recurso, use `GET`;
- Para modificar um recurso, use `PUT`;
- Para deletar um recurso, use `DELETE`.

4.2 GraphQL

A comunicação pode ser feita por qualquer protocolo, basta que para isso seja implementado no servidor. A única restrição é que haja somente um *endpoint* para a conversa entre cliente e servidor.

5 Nomeação

6 Dificuldades e Soluções

7 Considerações Finais