UNIVERSIDADE FEEVALE

MATHEUS LESSA AZZI

COMUNICAÇÃO DE DADOS DECLARATIVA COMO AUXILIADORA NO DESENVOLVIMENTO DE APLICAÇÕES COM MÚLTIPLOS CLIENTES

(Título Provisório)

Anteprojeto de Trabalho de Conclusão

Novo Hamburgo
2017

MATHEUS LESSA AZZI

COMUNICAÇÃO DE DADOS DECLARATIVA COMO AUXILIADORA NO DESENVOLVIMENTO DE APLICAÇÕES COM MÚLTIPLOS CLIENTES

(Título Provisório)

Anteprojeto de Trabalho de Conclusão de Curso, apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação pela Universidade Feevale

Orientador: Guillermo Nudelman Hess

Novo Hamburgo
2017

RESUMO

Comunicação de dados é provavelmente a atividade mais importante em uma aplicação Web. Existem diversas técnicas e padrões para efetuar a comunicação entre cliente e servidor. Devido à evolução das capacidades dos Navegadores Web e a enorme quantidade de diferentes tipos de dispositivos conectados, o desenvolvimento de aplicações Web tornou-se mais complexo. O cenário não era semelhante ao atual quando REST passou a ser adotado pelos desenvolvedores como implementação padrão para APIs Web, o número clientes móveis ganhou mais tráfego online e requisições de dados não otimizadas tornaram-se mais visíveis. A necessidade de múltiplos *Round-trips* para compor um único componente de UI na aplicação é a uma limitação presente na arquitetura REST. Prejudicando dispositivos com recursos de rede limitados. Clientes oferecem experiências únicas, para tal, utilizam diferentes dados de uma mesma API. Observa-se cada vez mais a necessidade fornecer aos clientes o poder de solicitar exatamente os dados que eles precisam. Recentemente, ambos Facebook e Netflix cada uma, desenvolveram ferramentas para requisição de dados na Web. O objetivo deste trabalho é analisar as limitações da arquitetura REST que levaram a criação destas ferramentas e validá-las como alternativas para comunicação de dados entre cliente e servidor, com enfoque em casos onde há desenvolvimento de aplicações com múltiplos clientes.

Palavras-chave: GraphQL. Falcor. Comunicação de dados declarativa. Múltiplos clientes. JSON Graphs.

SUMÁRIO

MOTIVAÇÃO

Com a evolução constante da utilização da Web desde sua criação, passaram a ser desenvolvidos websites das mais variadas finalidades. Conforme a área do conhecimento relacionada ao Desenvolvimento Web foi se aprimorando, motivou-se a elaboração de diversas técnicas, padrões e arquiteturas com o intuito de solucionar problemas recorrentes.

Observa-se que a partir de 2005, passou-se a utilizar amplamente o protocolo HTTP *(HyperText Transfer Protocol)* em conjunto com o modelo cliente-servidor para a troca de informações. Um dos principais motivos que contribuiu na época para esta transição foi a facilidade de uma aplicação expor sua API *(Application Program Interface)* através do estilo de arquitetura REST *(REpresentational State Transfer)*, assim como os clientes em comunicarem-se com essas APIs (DUVANDER, 2013 apud MASO, 2016).

Verifica-se o uso majoritariamente de REST para efetuar a troca de mensagens entre cliente e servidor (CEDERLUND, 2016). Porém, com a evolução das capacidades dos navegadores web e a adoção dos usuários por serviços disponibilizados através deste meio, o desenvolvimento dessas aplicações tornou-se mais complexo. Em consequência, têm-se mais dados sendo trafegados entre cliente e servidor, interfaces mais ricas e dinâmicas, lidando com mais entidades ao mesmo tempo. O lado do cliente acaba absorvendo mais regras de negócios para prover melhores experiências ao usuário.

Como abordado em "Why Falcor?" (NETFLIX, 2015), o universo Web moveu-se de páginas de conteúdos para *Single-Page Applications* (SPAs), onde aplicações não são renderizadas pelo servidor. Hoje, existe uma enorme quantidade de dispositivos diferentes conectados, e essa demanda era inexistente quando REST passou a ser adotado pelos desenvolvedores como implementação padrão para APIs web. Empresas como Facebook e Netflix têm seus produtos disponíveis em quase todo tipo de dispositivo com GUI *(Graphical User Interface)* e conectividade, desde desktops, notebooks e tablets até smartphones, televisores, video games, *smartwatches* e óculos de realidade virtual.

Além de múltiplos dispositivos, há outra situação com a qual é preciso lidar. Conforme "The GitHub GraphQL API" (GITHUB ENGINEERING, 2016), grandes empresas de tecnologia expoem suas APIs para serem utilizadas por serviços de terceiros. Necessitando

uma grande quantidade de recursos expostos, pois cada serviço terceiro pode ter um objetivo diferente e utilizar dados específicos, mesmo assim, a API deve atender estas demandas. No caso do Github, estes serviços terceiros atuam como um complemento do produto da empresa, são chamados de integrações e fornecem funcionalidades extras para os usuários do Github. Portanto, é de suma importância para a empresa o sucesso da utilização dessa API por outras ferramentas ou empresas.

Comunicação de dados é provavelmente a atividade mais importante em uma aplicação. Ao consumir APIs RESTful, utilizam-se URLs para ler e escrever um único recurso, como um produto, uma pessoa, ou um comentário. Quando uma única página possui diversos componentes com informações complexas, por exemplo, exibir "amigos das pessoas que comentaram em uma foto", torna-se necessário múltiplas requisições (*requests*) e diferentes *endpoints* para preencher esses dados (BUNA, 2016).

Neste aspecto, Cederlund (2016) explica que como o uso de dispositivos móveis ganhou mais tráfego online, requisições de dados não otimizadas tornaram-se mais visíveis. Isto, devido a múltiplos *Round-trip Times* (RTT) e a significante sobrecarga em redes móveis comparando-as com redes com fio. Em razão do grande número de usuários com dispositivos móveis, otimizar a performance para estas plataformas torna-se importante ao fornecer serviços online. A necessidade de múltiplos *round-trips* para compor um único componente de UI *(User Interface)* é uma limitação da utilização de RESTful, prejudicando dispositivos com recursos de rede limitados.

Dispositivos oferecem diferentes experiências, portanto utilizam diferentes dados. Quando cada tipo de cliente requisita dados específicos aos *endpoints*, o tamanho da resposta da API *(payload)* que tenta atendê-los começa a aumentar. Consequentemente aumenta-se o tempo de resposta. Com o passar do tempo é necessário introduzir novas versões da API, enquanto versões anteriores ainda precisam ser mantidas. Isso deve-se ao fato de outros clientes dependerem da versão antiga, nas quais as alterações ainda não foram implementadas (GILLESPIE, 2015). Portanto, para o cliente não receber informações que não são necessárias naquele contexto é necessário manter versões de APIs, com o objetivo de diminuir o tempo e o tamanho da resposta, outra limitação da arquitetura REST.

Tentar solucionar estes problemas utilizando REST tornará a API muito complicada, diminuindo a facilidade de manutenção ao longo do tempo, além de acoplar cada vez mais o

cliente com o servidor (HELFER, 2016).

Em 2015 Netflix desenvolveu a Falcor (NETFLIX, 2016) e o Facebook tornou público a GraphQL (FACEBOOK, 2016). Ambas são ferramentas declarativas para requisição de dados, alternativas para as limitações expostas anteriormente. Empresas das mais diversas utilizam essas ferramentas. Segundo Helfer (2016), elas compartilham algumas características em comum: possuem mais de um cliente e ao menos um cliente móvel; preocupam-se com a latência e largura de banda; utilizam micro-serviços; desejam desacoplar o desenvolvimento *server-side* do *client-side*.

GraphQL é uma *query language* (linguagem de consultas) para APIs e um interpretador em tempo de execução que realiza as consultas em uma base de dados existente. Fornece uma documentação automática da API, e dá aos clientes o poder de solicitar exatamente os dados que eles precisam e nada mais. Tornando mais fácil evoluir APIs ao longo do tempo, além de proporcionar ferramentas que auxiliam o desenvolvimento (FACEBOOK, 2016).

Falcor é a uma plataforma de dados inovativa que impulsiona as UIs na Netflix. É uma biblioteca para requisição de dados eficiente. É um *Middleware*, ou seja, atua entre a aplicação e o banco de dados. Permite representar todos os dados do servidor como um único objeto, um grafo em JSON. Com isso, para a ferramenta não importa onde os dados estejam, seja em memória no cliente ou através da conexão no servidor. Os dados são a API, ao conhecer os dados, conhece-se a API (NETFLIX, 2016).

Ambas ferramentas trabalham para atingir os mesmos objetivos: solucionar a complexidade ao trabalhar-se com dados em aplicações modernas. Elas possuem similaridades como: ser apenas uma camada entre o *front-end* e o *back-end*; sintaxe diferentes, mas semântica similar para declarar exatamente qual dado é desejado; buscam os dados em apenas uma única requisição ao servidor; trabalham com grafos e possuem sistema de cache (HELFER, 2016). Entretanto, devido a suas naturezas com bases arquiteturais diferentes, cada uma atua de forma diferente, têm curvas de aprendizado distintas, flexibilidades específicas e esforços necessários para sua utilização.

Teses como *"Efficient data communication between a webclient and a cloud environment"* de Kit Gustavsson e Erik Stenlund (2016) e *"Performance of frameworks for*

declarative data fetching" de Mattias Cederlund (2016) comparam de forma quantitativa a performance destas ferramentas para requisição de dados. De outro modo, o trabalho "Um Modelo de Comunicação para Automação na Execução de Consultas de Dados sobre APIs Web" (MASO, 2016), realiza um estudo sobre o uso da linguagem GraphQL e formatos de descrição de API para propor um modelo de comunicação cliente-servidor.

Com base no contexto apresentado, o trabalho propõe-se à analisar e avaliar as ferramentas Falcor e GraphQL como soluções alternativas ao REST, com foco em aplicações desenvolvidas para múltiplos dispositivos.

OBJETIVOS

Objetivo geral

O objetivo deste trabalho é avaliar soluções alternativas à arquitetura REST para comunicação de dados entre cliente e servidor, com enfoque em casos onde há desenvolvimento de aplicações com múltiplos dispositivos e/ou serviços terceiros.

Objetivos específicos

- Analisar as limitações do uso do REST para aplicações com múltiplos clientes que levaram à necessidade de criação de ferramentas como Falcor e GraphQL;
- Apresentar as similaridades e diferenças entre Falcor e GraphQL;
- Abordar cenários em que é vantajoso o uso de cada uma das ferramentas acima, bem como quando não é vantajoso sua utilização;
- Construir um protótipo para validar se o uso das ferramentas Falcor e/ou GraphQL auxiliam no desenvolvimento de aplicações com múltiplos clientes.

METODOLOGIA

Conforme os conceitos metodológicos apresentados por Prodanov e Freitas (2013), pode-se classificar o trabalho, quanto a sua natureza, como **pesquisa aplicada**, pois tem como propósito gerar conhecimentos para aplicação prática à solução de problemas específicos. Utilizando o método científico **dedutivo**, investigando os problemas do âmbito geral e posteriormente direcionando aos contextos específicos.

Do ponto de vista de seus objetivos, o trabalho enquadra-se como **pesquisa exploratória**, com a finalidade de proporcionar mais informações sobre o assunto e explorar as soluções alternativas, bem como os cenários onde é benéfico o uso das mesmas.

Quanto aos procedimentos técnicos, caracteriza-se como pesquisa de cunho **bibliográfico**, a partir de o levantamento de referências teóricas analisadas, como teses, dissertações, livros e artigos publicados. Em conjunto com a pesquisa **experimental**, formulando hipóteses e submetendo-se à experimentação em condições de controle através do protótipo que será desenvolvido utilizando as ferramentas Falcor e GraphQL.

Relacionado à forma de abordagem do problema, utiliza-se **pesquisa qualitativa**, baseando-se em outros trabalhos relacionados, buscando validar se as alternativas propostas atendem aos problemas apresentados.

CRONOGRAMA

Trabalho de Conclusão I

Etapa	Meses			
	Mar	Abr	Mai	Jun
Entrega do Aceite de Orientação				
Elaboração do Anteprojeto				
Estudo sobre REST/RESTful				
Análise de necessidades atuais para desenvolvimento com múltiplos clientes.				
Análise de limitações do REST/RESTful para desenvolvimento com múltiplos clientes.				
Estudo sobre GraphQL				
Estudo sobre Falcor				
Elaboração do Trabalho de Conclusão I				

Trabalho de Conclusão II

Etapa	Meses			
	Ago	Set	Out	Nov
Comparar Falcor e GraphQL				
Implementar protótipos utilizando Falcor e GraphQL				
Validar ferramentas como auxiliadoras no desenvolvimento com múltiplos clientes.				
Elaboração do Trabalho de Conclusão II				
Apresentação à banca avaliadora				

BIBLIOGRAFIA

BUNA, Samer. **Learning GraphQL and Relay**. Birmingham, United Kingdom, Packt Publishing Ltd, 2016.

CEDERLUND, Mattias. **Performance of frameworks for declarative data fetching**. KTH Royal Institute of Technology, School of Information and Communication Technology, 2016.

FACEBOOK. **GraphQL**. Disponível em: <http://graphql.org/>. Acesso em: 07 mar. 2017.

GILLESPIE, Jacob. **From REST to GraphQL**. Disponível em: <http://0x2a.sh/from-rest-to-graphql-b4e95e94c26b#.otm0y7s96>. Acesso em: 06 mar. 2017.

GITHUB ENGINEERING. **The GitHub GraphQL API**. Disponível em: <http://githubengineering.com/the-github-graphql-api/>. Acesso em: 28 mar. 2017.

GUSTAVSSON, Kit; STENLUND, Erik. **Efficient data communication between a webclient and a cloud environment**. Lund University, LTH, Department of Electrical and Information Technology, Faculty of Engineering, 2016.

HELFER, Jonas. **Why GraphQL is the future**. Disponível em: <http://dev-blog.apollodata.com/why-graphql-is-the-future-3bec28193807#.879ph0i16>. Acesso em: 06 mar. 2017.

MASO, Matheus. **Um Modelo de Comunicação para Automação na Execução de Consultas de Dados sobre APIs Web**. Universidade Federal de Santa Catarina. Centro Tecnológico. Sistemas de Informação, 2016.

NETFLIX. **Falcor**. Disponível em: <http://netflix.github.io/falcor/>. Acesso em: 05 mar. 2017.

NETFLIX. **Why Falcor?**. Disponível em: <http://netflix.github.io/falcor/starter/why-falcor.html>. Acesso em: 05 mar. 2017.

PRODANOV, Cleber Cristiano; DE FREITAS, Ernani Cesar. **Metodologia do Trabalho Científico**: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico - 2ª Edição. Editora Feevale, 2013.