Marlon Henrique da Silva
marlon.silva.1137367@sga.pucminas.br

Documento de Visão para o Sistema BFF Squared

06 de março de 2022

Proposta do aluno Marlon Henrique da Silva ao curso de Engenharia de Software como projeto de Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo do professor José Laerte Xavier e orientação acadêmica do professor Marco Rodrigo Costa.

OBJETIVOS

Automatizar a construção de aplicações baseadas no padrão *Back-end for Front-end* (BFF) e que utilizem a linguagem GraphQL a partir de especificações em linguagem de notação, definindo as fontes de dados, estruturas e fluxos para que uma aplicação totalmente funcional seja gerada e possa ser implantada em qualquer ambiente para atender às necessidades específicas das aplicações clientes.

ESCOPO

O sistema será um framework para a construção de uma Interface de Programação de Aplicação (API, do inglês *Application Programming Interface*) que utiliza GraphQL como padrão e gera as funcionalidades de consulta e agregação de dados a partir de outras aplicações ou microsserviços apenas por um arquivo de definições com informações sobre a própria aplicação, os endereços e parâmetros para os dados serem consumidos, as estruturas desses dados, a maneira como eles se relacionam, além da sua tipagem, obrigatoriedade e descrição. Por meio apenas de notações será possível especificar todas essas características, além da possibilidade de sincronização ou paralelização das requisições e, até mesmo, da dependência entre elas. Os fluxos definidos por notação irão gerar operações de leitura e escrita já fornecidas pelo GraphQL, sendo mapeadas para operações equivalentes de requisições, normalmente, baseadas no padrão de Transferência de Estado Representacional (REST, do inglês *Representational State Transfer*).

Além do próprio diferencial do padrão BFF, a utilização do GraphQL potencializa, ainda mais, a flexibilidade e as vantagens da sua implantação, haja vista que a alternativa direta mais comum, conhecida como API *Gateway*, é uma derivação mais simples que faz apenas a composição de APIs sem oferecer a possibilidade de manipulação de dados de forma mais específica para as necessidades de cada aplicação cliente. Para ambos os casos, a abstração e ocultação de múltiplos microsserviços pode aumentar a segurança e diminuir a latência das aplicações, pois mantê-los em uma rede privada, expondo um único ponto de contato para o exterior, permite focar mais na segurança e simplicidade dessa exposição.

Para auxiliar na construção de APIs GraphQL, já existem *frameworks* totalmente dedicados, como o Apollo, além de outros *frameworks* mais genéricos que possuem suporte para essa tecnologia, como Spring e NestJS. No entanto, esses *frameworks* são totalmente focados para implementação diretamente em código fonte com suas respectivas linguagens de programação, assim como não possuem uma estrutura base para nenhum propósito específico, ou seja, são estruturas gerais que permitem a criação de qualquer aplicação, onde todas as lógicas e funcionalidades precisam serem construídas do zero, assim como qualquer eventual alteração.

Por fim, o propósito geral desse projeto é ter uma ferramenta geradora de serviços BFF, que pode ser parcialmente ou integralmente considerada como sendo um framework, mas que não propõe o desenvolvimento de código fonte para adicionar as funcionalidades e especificações de cada contexto, possibilitando que isso seja feito de forma transparente apenas pela definição de algumas informações em um formato de notação (como JSON, YAML ou EDN) e que, a partir disso, seja gerada uma aplicação pronta para ser implantada em qualquer ambiente.

FORA DO ESCOPO

Após a análise das funcionalidades do sistema, foi decidido que, inicialmente, não haverá suporte para a funcionalidade de *subscriptions* do GraphQL, assim como não haverá qualquer suporte para comunicação por mensageria ou *WebSockets* (TCP) na estrutura de definição, assim como na geração da aplicação final.

A ideia de componentização e definição por notação permitiria a inclusão dessas funcionalidades futuramente ou, até mesmo, a implementação manual a partir do código base fornecido pela ferramenta, mas, sendo a sua utilização mais expressiva, optou-se, neste momento, considerar apenas o suporte para comunicações HTTP que sigam estritamente ou analogamente o padrão REST.

GESTORES, USUÁRIOS E OUTROS INTERESSADOS

Nome	Marlon Henrique da Silva
Qualificação	Estudante/Desenvolvedor
Responsabilidades	Responsável pelo desenvolvimento do projeto ao longo do Trabalho de Conclusão de Curso.

Nome	José Laerte Xavier
Qualificação	Orientador de Conteúdo
Responsabilidades	Auxiliar no acompanhamento do planejamento do projeto e fornecer orientações de conteúdo.

Nome	Marco Rodrigo Costa
Qualificação	Orientador Acadêmico
Responsabilidades	Auxiliar no acompanhamento do planejamento do projeto e fornecer orientações acadêmicas.

Tipo de usuário	Pessoa desenvolvedora de *back-end*
Como poderá usar o sistema proposto	Criar um *middleware* de forma automatizada a partir das definições dos serviços de *back-end* e das necessidades das aplicações clientes para abstrair lógicas de carregamento e agregação de dados necessárias para a construção de interfaces.

Tipo de usuário	Pessoa desenvolvedora de *front-end/mobile*
Como poderá usar o sistema proposto	Modificar e utilizar o *middleware* gerado para abstrair as operações em diversas fontes de dados com a flexibilidade de usar apenas o necessário, especificando diretamente na aplicação cliente.

LEVANTAMENTO DE NECESSIDADES

1. Integração entre diversos microsserviços e diferentes aplicações clientes (web, *mobile*, *smart* TV etc), com características e demandas diferentes entre cada uma delas.
2. Controle e flexibilidade sobre os dados necessários para cada aplicação e/ou funcionalidade, sem, necessariamente, precisar de alguma alteração nos microsserviços. A API define apenas quais são os dados disponíveis e os clientes decidem quais utilizar e como carregá-los para simplificar a implementação e diminuir o consumo da franquia de internet dos usuários.
3. Automação para criação de Web APIs com um propósito muito bem definido, sem a necessidade de construir tudo do zero ou utilizar *frameworks* que fornecem diversos outros recursos que não serão utilizados, mas acabam tornando toda a estrutura do projeto mais complexa.
4. Facilitação para a implementação de GraphQL, sem a necessidade de saber fazê-la diretamente em código fonte. Sendo necessária apenas a compreensão do seu conceito e forma de utilização, usando somente notações para definir as estruturas de dados.
5. Facilitação para a implantação de uma aplicação BFF, sem a necessidade de saber fazê-la diretamente em código fonte. Sendo necessária apenas a compreensão do seu conceito e forma de utilização, usando somente notações para definir as fontes de dados, assim como os fluxos e relacionamentos entre elas.

FUNCIONALIDADES DO PRODUTO

Necessidade: Integração entre diversos microsserviços e diferentes aplicações clientes	
Funcionalidade	**Categoria**
1. Gerar uma composição de APIs apenas com os recursos necessários para cada aplicação cliente.	Importante
2. Intermediar requisições de aplicações *front-end* para aplicações *back-end*.	Crítico
3. Abstrair os versionamentos das APIs.	Útil

Necessidade: Controle e flexibilidade sobre os dados das requisições	
Funcionalidade	**Categoria**
1. Definir estruturas de dados diversas.	Crítico

2. Compor novas estruturas de dados a partir de outras.	Útil
3. Especificar os dados que serão retornados na requisição.	Crítico
4. Definir os parâmetros que podem ser utilizados para cada recurso independentemente.	Importante
5. Enviar e receber dados bilateralmente.	Crítico

Necessidade: Automação para criação de Web APIs	
Funcionalidade	**Categoria**
1. Definir configurações e recursos por meio de notação.	Crítico
2. Utilizar uma estrutura base construída por componentes.	Importante
3. Gerar funcionalidades a partir de notações.	Crítico
4. Construir uma aplicação completa apenas com especificações.	Crítico

Necessidade: Facilitação para implementação de GraphQL	
Funcionalidade	**Categoria**
1. Abstrair a notação específica de GraphQL.	Útil
2. Definir *schemas* sem código fonte.	Crítico
3. Definir *queries* sem código fonte.	Crítico
4. Definir *mutations* sem código fonte.	Crítico
5. Definir *resolvers* sem código fonte.	Crítico

Necessidade: Facilitação para implementação de uma aplicação BFF	
Funcionalidade	**Categoria**
1. Mapear uma requisição de API para outras várias.	Crítico
2. Definir um conjunto de requisições para compor um fluxo.	Importante
3. Definir sincronização ou paralelização para um conjunto de requisições.	Importante

4. Definir dependência de valores entre requisições.	Útil
5. Definir fluxo de dados entre requisições.	Útil

INTERLIGAÇÃO COM OUTROS SISTEMAS

Não há uma definição rígida de quais sistemas serão integrados ao usar essa abordagem, mas, naturalmente, a proposta deste *framework* pressupõe que a aplicação gerada irá interagir com diversas aplicações de *back-end* (microsserviços) e diferentes aplicações clientes (web, *mobile*, *smart* TV etc) dentro do cenário em que foi aplicada, a depender de cada contexto.

RESTRIÇÕES

- Se houver a necessidade do desenvolvedor implementar algo diretamente em código fonte, o propósito da ferramenta se torna questionável, haja vista que o seu foco é abstrair a construção utilizando notações de dados.
- Se a aplicação não funcionar corretamente com GraphQL, a sua utilização perde uma das suas principais vantagens frente a outras abordagens, como um API *Gateway* comum.
- Se a estrutura base fornecida não for baseada em componentização não será eficiente gerar a aplicação a partir das definições.
- Se a estrutura base e o padrão de notações não forem bem padronizados e flexíveis, não será viável estender a ferramenta futuramente com mais funcionalidades e suporte a novos recursos.
- O ideal é que não seja criada uma nova linguagem para as notações ou uma plataforma específica para essa construção. A ideia é que a curva de aprendizado seja pequena para incentivar a utilização baseada em padrões e convenções já consolidadas.
- A geração da aplicação depende exclusivamente das especificações feitas na linguagem de notação proposta e da estrutura base que será construída e fornecida, mas a sua aplicabilidade depende de cenários reais com outras aplicações em execução.
- Os ganhos de geração e utilização de uma aplicação BFF são potencializados pelo contexto de múltiplos microsserviços e diferentes aplicações clientes. Se o contexto for diferente desse, a sua aplicabilidade pode ser questionável.

DOCUMENTAÇÃO

Será elaborado um manual de uso indicando quais são as funcionalidades da ferramenta e como utilizá-las dentro do contexto proposto. Para detalhes mais aprofundados será elaborada uma documentação para descrever cada componente do sistema em um arquivo README no repositório do projeto. Como forma visual de esquematização, também serão elaborados modelos de caso de uso e outros diagramas UML.