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

Documento de Visão para o Sistema API Valve

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 Marco Rodrigo Costa e orientação acadêmica do professor José Laerte Xavier.

OBJETIVOS

Automatizar a construção de aplicações *Back-end for Front-end* (BFF) utilizando GraphQL para integrar diferentes aplicações de *back-end*. Baseada em uma interface gráfica de desenvolvimento ou diretamente em Notação de Dados Extensível (EDN, do inglês *Extensible Data Notation*), sem a necessidade de escrever código-fonte para definir as funcionalidades da aplicação. Sendo necessário apenas especificar as fontes de dados, estruturas e parâmetros para que uma aplicação seja gerada e possa ser implantada em ambientes com suporte à Máquina Virtual Java (JVM, do inglês *Java Virtual Machine*) para atender às necessidades de aplicações clientes que utilizarão a aplicação BFF como fonte de dados principal.

ESCOPO

O sistema será uma ferramenta de apoio para a construção de uma Interface de Programação de Aplicação (API, do inglês *Application Programming Interface*) que utiliza GraphQL por padrão, e realiza consultas e agregação de dados a partir de outras aplicações ou microsserviços. Por meio apenas de especificações será possível definir o seu funcionamento diretamente em notação EDN ou utilizando a interface de usuário do sistema. As especificações irão habilitar o recebimento de requisições via Protocolo de Transferência de Hipertexto (HTTP, do inglês *Hypertext Transfer Protocol*) já suportadas pelo GraphQL e irão mapeá-las para a realização de outras requisições HTTP baseadas no padrão de Transferência de Estado Representacional (REST, do inglês *Representational State Transfer*).

O arquivo EDN funcionará, basicamente, como um arquivo de configuração. As informações escritas diretamente nele ou por meio da interface gráfica (de forma simplificada) servirão apenas para contextualização. Todas as funcionalidades da aplicação já estarão implementadas de forma genérica, possibilitando que ela receba requisições com GraphQL e realize requisições REST, desde que sejam especificados os detalhes dessas requisições. Para isso, no arquivo serão definidas informações sobre a própria aplicação, os endereços dos serviços de *back-end* que serão consumidos como fontes de dados primárias e seus respectivos parâmetros para que os dados sejam consumidos. Além disso, para que o GraphQL funcione corretamente, será necessário definir as estruturas desses dados, a maneira como eles se relacionam, seus tipos, obrigatoriedade e descrições.

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 de composição de APIs que não oferece 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 e expor um único ponto de contato para o exterior permite focar mais na segurança e simplicidade da 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 adicional para essa tecnologia, como Spring e NestJS. No entanto, esses *frameworks* são totalmente focados na implementação diretamente em código-fonte, com suas respectivas linguagens de programação, assim como não possuem uma estrutura base para nenhum propósito específico. São estruturas genéricas que permitem a criação de qualquer aplicação, onde todas as lógicas e funcionalidades precisam ser construídas do zero, assim como qualquer eventual alteração.

Em suma, o propósito geral desse projeto é ter uma ferramenta geradora de serviços BFF, que pode ser considerada uma ferramenta *low code*, pois não propõe o desenvolvimento de código-fonte para implementar as funcionalidades suportadas. Será possível que isso seja feito de forma visual por uma interface gráfica dentro do sistema ou escrevendo diretamente em um arquivo EDN dentro do diretório da aplicação. Apenas com essas especificações e com o código base já fornecido pela ferramenta, onde todas as funcionalidades já estarão implementadas de forma genérica, será possível gerar uma aplicação que receba e realize as requisições especificadas de forma agregada. Por fim, essa aplicação poderá ser executada no próprio ambiente de desenvolvimento ou implantada em um servidor com sistema operacional Windows, Linux ou macOS, desde que haja suporte à JVM.

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 nenhum suporte para comunicação por mensageria ou *WebSockets* via Protocolo de Controle de Transmissão (TCP, do inglês *Transmission Control Protocol*).

A ideia de componentização e especificação por notação EDN permitiria a inclusão de outras funcionalidades futuramente ou, até mesmo, a implementação manual em meio ao código base fornecido pela ferramenta. No entanto, como foco deste projeto, optou-se por implementar apenas o suporte à realização de requisições HTTP que sigam estrita ou analogamente o padrão REST, pois esse é o padrão mais comum e expressivo em sistemas desse tipo.

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	Marco Rodrigo Costa
Qualificação	Orientador de Conteúdo
Responsabilidades	Auxiliar no acompanhamento do planejamento do projeto e fornecer orientações de conteúdo.

Nome	José Laerte Xavier
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 de usuário.

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 a 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 as especificações dos dados para criar o recurso.
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 especificações 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. Definir detalhes das Web APIs que poderão ser consumidas.	Crítico
2. Intermediar requisições HTTP de GraphQL para REST.	Crítico
3. Enviar e receber dados bilateralmente.	Importante
4. Abstrair o versionamento de 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

Necessidade: Automação para criação de Web APIs	
Funcionalidade	**Categoria**
1. Fornecer uma implementação genérica para a realização de requisições HTTP no padrão REST.	Crítico
2. Fornecer uma implementação genérica para o recebimento de requisições HTTP com GraphQL.	Crítico
3. Habilitar o recebimento e realização de requisições a partir de especificações EDN.	Crítico
4. Fornecer uma interface gráfica para explorar a API gerada.	Útil

Necessidade: Facilitação para implementação de GraphQL	
Funcionalidade	**Categoria**
1. Abstrair a notação original de GraphQL para EDN.	Útil
2. Definir *schemas* por uma interface gráfica e notação EDN.	Crítico
3. Definir *queries* por uma interface gráfica e notação EDN.	Crítico
4. Definir *mutations* por uma interface gráfica e notação EDN.	Crítico

Necessidade: Facilitação para implementação de uma aplicação BFF	
Funcionalidade	**Categoria**
1. Fornecer uma interface gráfica para definir especificações da aplicação.	Importante
2. Carregar e exibir na interface gráfica as especificações definidas em um arquivo EDN.	Importante
3. Salvar as especificações definidas na interface gráfica em um arquivo EDN.	Crítico
4. Executar a aplicação a partir do código-fonte fornecido e de um arquivo EDN com as especificações definidas.	Crítico

INTERLIGAÇÃO COM OUTROS SISTEMAS

O sistema proposto não possui interligação com outros sistemas ao ser utilizado para a construção de uma aplicação BFF. No entanto, o funcionamento da aplicação BFF construída utilizando o sistema (e que também faz parte dele) baseia-se na comunicação com outras aplicações *back-end* e aplicações clientes. Essas outras aplicações são definidas pelo usuário nas especificações salvas no arquivo EDN e a comunicação ocorre por meio do protocolo HTTP, seguindo os formatos definidos pelo usuário e os padrões definidos pelo sistema.

RESTRIÇÕES

- Se houver a necessidade do desenvolvedor implementar algo diretamente em código-fonte para construir a aplicação BFF, o propósito da ferramenta se torna questionável, haja vista que o seu foco é abstrair essa construção utilizando apenas especificações por meio de uma interface gráfica ou notação EDN.

- 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 de código base fornecida pela ferramenta não for baseada em componentização não será viável gerar a aplicação a partir de especificações.
- Se a estrutura base e o formato das 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 separada para essa construção. A ideia é que a curva de aprendizado seja pequena para incentivar a utilização com padrões e convenções já consolidados.
- O uso da ferramenta proposta, assim como a execução da aplicação gerada por ela, necessitam de um ambiente com suporte à JVM, possibilitando que ambas tenham suporte à portabilidade entre diferentes sistemas operacionais (Windows, Linux e macOS).

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.