Customização de sistemas ERP baseado em microsserviços

Microservice-based customization of ERP systems

ABSTRACT

Customizing ERP systems in an organization can be a frequent activity in the face of business dynamics when changes need to be made either for continuous improvement of the business process or for the changes of the segment in which the company operates. These customizations may imply the creation of new features and how fast these customizations are made, and are deployed in production can be a factor of competitive advantage. To accomplish such customizations, this study proposes a roadmap, which uses the microservices architecture in a practical feature for a ERP in production and evaluates its application. The result is new complete feature, based on microservices best practices, as a software product which allows the user to access its data through an web interface.

Keywords**:** microservice, software customization, independence, ERP.

RESUMO

A customização de sistemas ERP em uma organização pode ser uma atividade frequente diante da dinâmica dos negócios quando mudanças necessitam ser realizadas, seja para melhoria contínua do processo de negócio ou pela própria natureza do segmento em que a empresa atua. Essas customizações implicam a criação de novas funcionalidades. A rapidez com que essas customizações são realizadas e disponibilizadas em produção pode ser um fator de vantagem competitiva. Para realizar tais customizações, este artigo propõe um roteiro de customização, que utiliza a arquitetura de microsserviços na implementação em um caso prático para um ERP implantado e avalia sua aplicação. O resultado obtido é uma nova funcionalidade completa, compatível com as melhores práticas de microsserviços em um produto de *software* que permite ao usuário acessar seus dados por meio de uma interface web.

Palavras chaves: microsserviços, customização de *software*, independência, ERP.

1. INTRODUÇÃO

A anatomia de um ERP tem como pilar tanto um banco de dados central que extrai ou alimenta dados em uma série de aplicativos que suportam diversas aplicações em uma organização, como o uso de um repositório ou banco de dados único que dinamiza consideravelmente o fluxo de informações (DAVENPORT, 2000). É natural que uma organização que utiliza um sistema ERP tenha que modificar o produto original para adicionar novas funcionalidades específicas ou estender funções já existentes. Essas customizações ou alterações de sistema, ocorrem devido a uma variedade de fatores, como uma funcionalidade do pacote que tem um comportamento que não atende suficientemente à necessidade da organização, tendência organizacional de decidir sobre alterações rápidas e ter poucas informações sólidas e incapacidade de levar aos tomadores de decisão uma visão de longo prazo sobre a evolução das soluções do ERP (SUDHAMAN e DANEVA, 2016).

Organizações, públicas ou privadas, que utilizem mais de um sistema ERP de diferentes fornecedores, podem necessitar de integrações entre funcionalidades de seus sistemas para obterem uma maior efetividade do fluxo do negócio. O envolvimento de dois ou mais fornecedores que utilizem linguagens de programação e repositórios de dados diferentes pode ser um obstáculo para que informações sejam compartilhadas. Essa heterogeneidade de ambientes podem ser limitadores e causarem efeitos colaterais que eventualmente comprometem o desempenho do sistema ERP e/ou a efetividade do fluxo do negócio.

Diante de um cenário por vezes desfavorável devido a barreiras financeiras ou de profissionais disponíveis na organização para suportar as demandas, soluções paliativas são adotadas. Dentre essas soluções, há o uso de planilhas e/ou arquivos texto exportados do ERP e por não estarem conectados diretamente ao banco de dados, podem demonstrar resultados defasados caso não haja uma frequência de atualização que garanta a fidedignidade das informações, o que pode não ser uma boa prática.

A contratação e treinamento de desenvolvedores para customizações de sistemas é um desafio, pois além de habilidades de programação que podem exigir uma linguagem proprietária da empresa que fornece o ERP, os desenvolvedores devem compreender a aplicação existente, a qual no decorrer do tempo já pode ter sido customizada. Essas customizações podem ser realizadas de três maneiras. Diretamente no código fonte, por meio de ferramentas de terceiros que se conectam diretamente ao banco de dados central para extrair informações, ou por uma interface limitada de conexão (DITTRICH e VAUCOULEUR, 2009).

Este artigo aborda o acesso ao banco de dados central para extração de dados que sirvam para a criação de uma nova funcionalidade e que funcione de forma independente com um repositório de dados também independente e exclusivo para esse objetivo.

Nos últimos anos surgiu o conceito da arquitetura de microsserviços que é uma abordagem para o desenvolvimento de uma única aplicação como um conjunto de pequenos serviços, cada um sendo executado em seu próprio processo e se comunicando usando mecanismos leves, os quais em sua maioria vêm de uma *Application Programming Interface* (API) com recursos *Hypertext Transfer Protocol* (HTTP) (FOWLER e LEWIS, 2014).

Microsserviços têm como características a divisão de sistemas em componentes por meio de serviços, organizados em torno de capacidades de negócio, produtos e não projetos,

governança descentralizada, gerenciamento de dados descentralizado, desenhados para tolerar falhas e preparados para evolução (FOWLER e LEWIS, 2014).

Dessa forma, a arquitetura de microsserviços pode ser uma alternativa para a criação de novas funcionalidades, já que permite a independência e flexibilidade de serem desenvolvidos sem que haja uma customização no pacote original do ERP.

2. METODOLOGIA

2.1 Conceitos

ERP:

Segundo Davenport (1998), um sistema ERP é um conjunto de aplicações ou módulos que ligam as informações das unidades de negócios de uma organização, tais como recursos financeiros, contábeis, manufatureiros e humanos, em um sistema único fortemente integrado, como uma plataforma comum para fluxo de informações em toda a empresa.

Para entender a necessidade do ERP nas empresas, e também seus riscos, é preciso definir qual problema ele se propõe a solucionar: a fragmentação e duplicidade de informações em grandes organizações.

Microsserviços:

Segundo Fowler e Lewis (2014), o estilo arquitetural de microsserviços é uma abordagem para o desenvolvimento de uma única aplicação como um conjunto de pequenos serviços, cada um sendo executado em seu próprio processo e se comunicando por meio mecanismos leves, os quais em sua maioria vêm com uma API com recursos HTTP. Esses serviços são construídos em torno de recursos de negócios, implementados de forma independente e disponibilizados automaticamente. Existe um mínimo de gerenciamento centralizado desses serviços, o qual pode ser escrito em diferentes linguagens de programação e usar diferentes tecnologias de armazenamento de dados.

Customização de sistemas:

Segundo Sudhaman e Daneva (2016), é natural que uma organização que utiliza um sistema ERP tenha que modificar o produto original para adicionar novas funcionalidades específicas ou estender funções já existentes. O tamanho e a complexidade das customizações podem variar desde a alteração de um simples relatório até o desenvolvimento completo de uma funcionalidade para atender um determinado processo de negócio.

Em um outro trabalho, Rothenberger e Srite (2009) também descrevem as customizações como a modificação de um pacote para satisfazer os processos de negócio existentes. Além disso, organizações que adotam sistemas ERP têm diversas opções para adequá-lo ao negócio que pode ser a modificação do próprio pacote ou a implementação de pacotes de terceiros desenvolvidos para trabalhar em conjunto com o ERP ou suplementar uma funcionalidade.

Sistemas monolíticos podem ser satisfatórios, porém podem trazer frustrações para os usuários, principalmente pela dificuldade de disponibilizá-las em ambiente web ou nuvem.

Ciclos de mudança estão interligados e uma mudança feita em uma pequena parte do sistema exige que o monolítico seja compilado e disponibilizado em sua totalidade. Ao longo do tempo, pode ficar difícil manter uma estrutura modular satisfatória e consequentemente torna-se mais difícil manter alterações. A escalabilidade requer que toda a aplicação seja escalada ao invés de partes dela, o que exige mais recursos (FOWLER e LEWIS, 2014).

Um sistema monolítico com uma base de código-fonte grande faz com que os desenvolvedores resistam a fazer alterações, especialmente aqueles que são novos na equipe.

Essa aplicação pode ser difícil de entender e modificar, com isso o desenvolvimento normalmente fica mais lento. Além disso, devido a essa dificuldade de compreensão ao implementar uma mudança corretamente, faz com que o processo seja mais demorado (RICHARDSON, 2017).

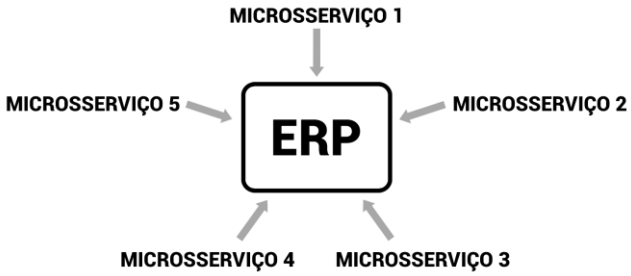
3. ROTEIRO PARA CUSTOMIZAÇÃO

3.1 Introdução

Nesta seção é apresentado o roteiro para customização de sistemas ERP com base na arquitetura de microssreviços. A idéia é prover novas funcionalidades por meio de microsserviços independentes do sistema ERP. Tipicamente, os microsserviços possuem bases de dados independentes. Assim, a integração se dá por acesso periódico à base de dados do ERP (figura 1).

Inicialmente são apresentadas e justificadas as atividades do roteiro. Em seguida, é apresentado um fluxo que verifica a adequação ou não da aplicação do roteiro em cada caso particular.

Figura 1. Proposta genérica para criação de novas funcionalidades.



3.2 Atividades do roteiro

A construção do roteiro é baseada em atividades sequenciais utilizando os conceitos e definições apresentados na literatura que relacionam a necessidade da customização de sistemas ERP e o uso de microsserviços. A figura 1 mostra o fluxo das atividades do roteiro em seguida, cada uma das atividades são apresentadas.

Figura 1. Fluxo para criação de nova funcionalidade por meio de microsserviço

3.2.1 Atividade 1 – Decomposição funcional

Adotando a arquitetura de microsserviços como uma alternativa para a criação de funcionalidades, toma-se como base o conceito apresentado por Fowler e Lewis (2014) que define a decomposição funcional dos processos de negócio em pequenos serviços. Da mesma forma, Richardson (2017) acrescenta que os serviços devem ser fracamente acoplados.

Nessa atividade, deve ser determinado qual é o objetivo da funcionalidade para que seja definida suficientemente de forma que possa ser representada como um único serviço ou uma classe, tendo exclusivamente uma única responsabilidade e que não dependa de outras funções.

3.2.2 Atividade 2 – Repositório de dados

O repositório de dados é um ponto importante a ser levado em consideração, já que há algum tipo de interação do serviço com o banco de dados para atender o objetivo definido para a funcionalidade.

Conforme Vienot, Lécuyer, Bell, Geambasu e Nieh (2015), funcionalidades que demandam integração e/ou compartilhamento de informações devem ter um repositório próprio de dados por poderem armazenar, em um único local, dados de diferentes fontes e não somente do sistema ERP principal. Portanto, o serviço cumpre sua finalidade utilizando-se do acesso a um único local e dessa forma pode aumentar a agilidade e melhorar a experiência do usuário.

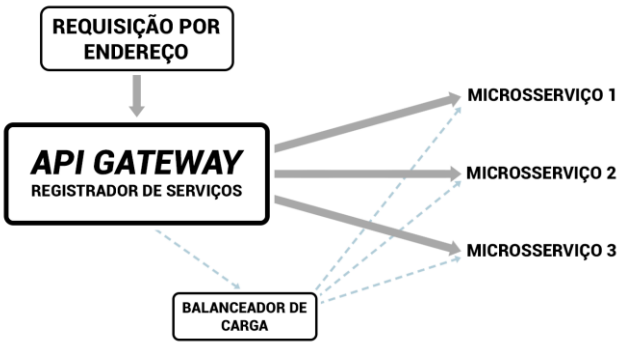
Não há uma obrigatoriedade de utilizar o mesmo repositório do sistema ERP que está sendo estendido, dessa forma dando maior liberdade de decisão para a equipe de desenvolvimento.

3.2.3 Atividade 3 – Servidor de aplicação (API Gateway)

Com a(s) funcionalidade(s) definidas, há a necessidade de um servidor de aplicação que seja responsável por responder as requisições de cada um dos serviços e que será representado por um endereço web.

Dessa forma, o API Gateway recebe todas as requisições, se houver um balanceador, há a distribuição para o servidor correspondente e partir desse servidor é direcionado ao serviço requisitado.

Figura 2. API Gateway



3.2.4 Atividade 4 – Definição dos endereços dos microsserviços

Para que o API Gateway possa fazer o roteamento correto para o serviço requisitado, será necessário definir um endereço para cada um dos microsserviços que será composto de [endereço_API_Gateway]/[endereço_microsserviço**N**].

Portanto, cada uma das funcionalidades têm um endereço correspondente e que está configurado no registrador de serviços para que o roteamento ocorra corretamente. Dessa forma, a responsabilidade do direcionamento está concentrada na infraestrutura que provê o acesso para todos os serviços ficando transparente para o usuário final o local físico que responde à requisição.

Figura 3. API Gateway e Registrador de Serviços

3.2.5 Atividade 5 – Criação de identificação de acesso (Access Token)

Os serviços podem apresentar informações do sistema ERP que estão sob uma política de acesso por serem confidenciais, estratégicas ou que só possam estar disponíveis a determinados usuários de acordo com seu perfil de acesso ou hierarquia dentro da organização.

Knoche (2016), Gadea, Trifan e Ionescu (2016) também recomendam a identificação por meio de credenciais e para serviços de busca de informações no banco de dados de um sistema ERP, pode ser gerada uma sequência de caracteres geradas randomicamente baseado em algoritmos conhecidos no meio computacional, como MD5 ou SHA-256.

No caso da busca de informações no banco de dados de um sistema ERP, podem haver webservices que por meio de requisições REST para atualizar os repositórios criados na atividade 2. Esses webservices, por acessarem diretamente a base de dados do ERP, é recomendado uma credencial de acesso que pode ser uma sequência de caracteres que é denominada token.

3.2.6 Atividade 6 – Disponibilizar em produção (Deployment)

Fowler e Lewis (2014) descrevem que os microsserviços criados são disponibilizados em produção pelos próprios desenvolvedores já que, por definição, são responsáveis pelo serviço como um todo, desde o código-fonte até a disponibilização em produção e o seu gerenciamento. Outro objetivo é que ocorram entregas contínuas que podem ser disponibilizadas em produção várias vezes ao dia e pode representar um ganho de produtividade na implantação de alterações ou novos serviços em produção e prontos para uso.

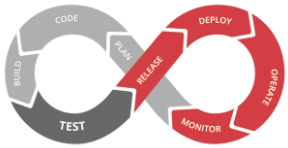
Da mesma forma que são disponibilizados, essas novas versões podem causar falhas e para isso medidas para garantir a resiliência do serviço devem ser adotadas.

Para atingir o objetivo dessa atividade que é a disponibilização dos serviços em produção, há premissas que devem ser aplicadas e para isso é utilizada a metodologia do DevOps apresentados por Stubbs, Moreira e Dooley (2015) e Guo, Wang, Zeng e Wei (2016).

Esse artigo não tem o objetivo de se aprofundar nas questões relacionadas ao DevOps, apenas utilizará o conceito principal denominado “esteira” para apresentar a infraestrutura

necessária para que os deployments ocorram de forma contínua e permitam ser escalados conforme a demanda.

Figura 4. Esteira DevOps



3.2.7 Atividade 7 – Observação do serviço

Na atividade anterior, há um monitoramento quanto à disponibilidade do serviço além da possibilidade de escalar os serviços mediante parâmetros de utilização e uso da infraestrutura com o objetivo de oferecer uma melhor experiência de uso do usuário em relação ao desempenho.

Além desse monitoramento, há outras possibilidades de análise do serviço por meio da criação de logs de acesso ou eventuais erros/mensagens que possam servir de alerta tanto para a equipe de gestão de uso do serviço como para os desenvolvedores que podem identificar padrões de comportamento e eventualmente oportunidades de melhorias visando atender as requisições dos usuários buscando a excelência do serviço.

3.2.8 Atividade 8 – Interface de visualização

A interface de visualização do microsserviço é uma atividade importante a ser levada em consideração, pois deve ser de fácil utilização e acesso para que seja o mais intuitivo possível para que evite uma demanda de interações entre o usuário e os departamentos responsáveis pelas informações para que não seja necessário um esforço de treinamento.

Gadea, Trifan, Inonescu (2016) e Richardson (2017) tratam essa atividade como interface do usuário (*user interface*) e há duas opções onde a primeira é gerada de acordo com um padrão retornado pelo servidor por meio de um fragmento em HTML e dessa forma todos os serviços podem manter um padrão de exibição como a marca da organização e denominada como *server-side page fragment composition*, a segunda opção o requisitante é encarregado de desenvolver seu próprio padrão de exibição para cada microsserviço, denominada por *client-side user interface composition*.

Este trabalho não tem o objetivo de comparar as opções para a exibição da interface, portanto essa atividade adota a opção onde cada microsserviço será responsável pela exibição dos seus dados (*client-side*).

3.2.9 Fluxo do Roteiro

A partir da descrição de cada uma das atividades, suas motivações e justificativas baseadas na literatura, há embasamento para criar uma sequência de atividades que compõe cada uma das atividades para definir um fluxo de passos sequenciais para a criação de novas

funcionalidades para estender um sistema ERP implantado utilizando-se da arquitetura de microsserviços para leitura e exibição de dados.

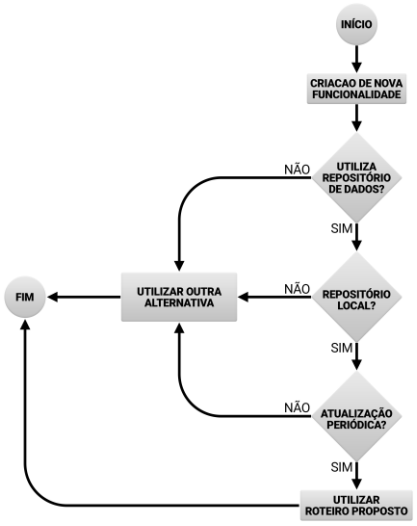
3.3 Requisitos e limitações da proposta

Esse roteiro deve ser aplicado avaliando o objetivo e as características do service a ser criado para que a aplicação desse roteiro obtenha os resultados esperados quanto ao atendimento das funcionalidades a serem criadas.

Para isso, é apresentado um fluxograma que tem o objetivo de avaliar se essa é a alternativa que converge com a proposta desse artigo.

Essa é a limitação do roteiro, sendo que, se o problema a ser resolvido não se enquadrar nesse fluxo, é possível que o roteiro não apresente os mesmos resultados.

Figura 6. Fluxograma de avaliação da funcionalidade



Nesse fluxograma quer se avaliar se a nova funcionalidade que necessita ser criada terá um repositório de dados e se é independente ao ponto de não haver integração em tempo real com qualquer outro repositório. Além disso, se a atualização desse repositório é periódica e que podem ser atualizados por processos agendados ou de forma espontânea pelo usuário.

Se essas premissas forem atendidas, o roteiro proposto torna-se uma alternativa para que o sistema ERP seja customizado para a criação das novas funcionalidades.

4. APLICAÇÃO DO ROTEIRO EM UM CASO PRÁTICO

4.1 Sobre o Sistema a ser customizado

O sistema ERP customizado é um sistema que está implantado em uma organização pública e composto de módulos de compras de materiais, contabilidade, patrimônio mobiliário, gestão de processos administrativos e recursos humanos.

A figura 7 apresenta como o ERP está organizado atualmente para atender as necessidades do negócio.

Figura 7 – ERP em funcionamento

O módulo de recursos humanos é o módulo que necessita de novas funcionalidades devido à descentralização dos funcionários que ficam alocados em diversos locais do município e estão geograficamente distantes do departamento responsável pela gestão do controle de folha de pagamento e recursos humanos.

Essa distância geográfica dificulta a obtenção de informações específicas por parte do funcionários já que quando necessitam buscar determinados dados referentes ao seu prontuário, devem entrar em contato telefônico, enviar um e-mail utilizando seu endereço corporativo ou deslocar-se até o departamento.

Essa interação torna-se um problema devido ao número de funcionários desse órgão que atualmente está em torno de 15.000 (quinze mil), em um município que tem aproximadamente 320 quilômetros quadrados com uma população aproximada de 1.300.00 (um milhão e trezentos mil) habitantes.

Diante desse cenário, o deslocamento é um problema dependendo do local onde o funcionário está lotado para exercer suas funções e ligações telefônicas causam um congestionamento nos ramais do departamento responsável.

Dessa forma, alternativas que facilitem o acesso de determinadas informações minimizam essas dificuldades e ao mesmo tempo diminui o número de interações entre os funcionários e o departamento que provê tais informações.

Identificadas essas dificuldades, uma alternativa é prover o acesso dessas informações por meio de acesso a um sistema informatizado utilizando uma identificação pessoal, por exemplo, usuário e senha, e que desta forma consiga obter tais informações de forma que não necessite uma interação direta com o departamento.

O acesso a esse sistema deve ser por meio de um endereço eletrônico utilizando um dispositivo que tenha acesso à internet, seja um computador pessoal, um telefone móvel ou um tablet.

Com isso, o funcionário tem a facilidade atender sua necessidade qualquer que seja sua localização e também fora do horário de expediente do departamento responsável.

Há uma preocupação da administração do órgão em oferecer alternativas para que o funcionário, independente de suas limitações, não seja cerceado de seus direitos. Funcionalidades disponíveis na internet podem facilitar que pessoas responsáveis por departamentos onde esses funcionários estão alocados, auxiliem seus pares que tenham dificuldades de acesso e obter suas informações evitando deslocamentos.

Diante dessa demanda apresentada, é possível identificar que há funcionalidades específicas, unitárias e independentes que buscam atender uma demanda objetiva, ou seja, um serviço de granularidade fina, que é implementado na forma de microsserviço.

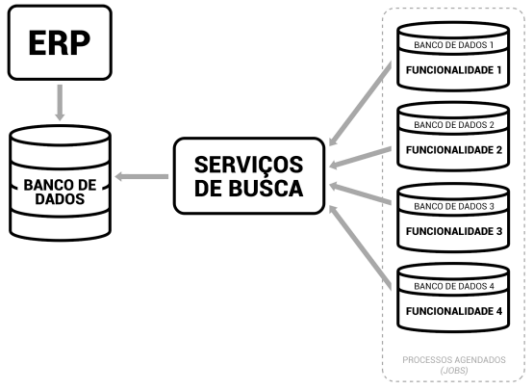
4.2 Funcionalidade a ser criada

A funcionalidade criada foi a consulta de dados de vale-transporte, tendo por objetivo apresentar os dados do benefício de solicitação voluntária do funcionário, onde são apresentadas as linhas de transporte público utilizadas, junto com seus valores unitários e totais, além do período referente aos créditos que serão disponibilizados ao funcionário. Por meio dessas informações, o funcionário pode conferir os valores creditados nos meios disponibilizados pelo órgão, como por exemplo, cartão de empresas públicas e/ou crédito em conta corrente. Por fim, com essas informações, o funcionário pode confrontar o desconto aplicado em folha de pagamento conforme a consolidação da leis do trabalho.

O fluxo do processo é dividido em 3 (três) blocos principais, sendo o primeiro deles referente ao sistema ERP que é estendido junto com seu banco de dados central que contém as informações de todos os módulos, o segundo bloco refere-se a serviços que proveem os dados obtidos do banco de dados central para os microsserviços e o terceiro bloco que são os serviços propriamente ditos que têm a interface, o repositório de dados independente e que atendem aos objetivos das funcionalidades.

A figura 8, apresenta um modelo genérico com os blocos e as interações que são necessárias entre si para atender a necessidade de criação de novas funcionalidades.

Figura 8– Fluxo do processo de criação dos microsserviços



Conforme a figura 8, o ERP é o sistema corporativo utilizado pelo órgão que contém seu repositório de dados que armazena e compartilha as informações entre os diversos departamentos que são responsáveis por criar e/ou consultar informações pertinentes a cada um dos seus papéis na estrutura do negócio. Segundo a definição da anatomia de um sistema ERP apresentado por Davenport (2000), essas características apresentadas vão ao encontro do modelo de um sistema ERP genérico independente de seus módulos.

O bloco denominado “Serviços de Busca”, são webservices do tipo REST, que utilizam o protocolo HTTP para obter os dados da base de dados central do sistema ERP e retorna os dados em formato XML, dependendo do tipo solicitado na requisição do serviço. Essa metodologia de busca segue a recomendação apresentada por Fowler e Lewis (2014) que define a obtenção de dados baseado em uma API por meio de um protocolo, nesse caso o HTTP.

Nesse bloco “Serviços de Busca” estão disponíveis serviços individualizados e responsáveis pela obtenção dos dados exclusivamente para atender um determinado conjunto de dados que se refere diretamente a uma única funcionalidade que é o microsserviço.

Seguindo a definição da arquitetura de microsserviços de Fowler e Lewis (2014), o terceiro bloco que apresenta as funcionalidades, é possível verificar que cada uma tem seu próprio repositório de dados para que dessa forma fiquem completamente independentes e permita a escalabilidade mediante parâmetros como número de requisições, uso de memória ou uso de unidades de processamento computacionais que possam eventualmente afetar o desempenho.

Como os repositórios são independentes e essas informações são atualizadas no sistema ERP em intervalos de datas específicos, há um processo agendado, também conhecido como job, que atualiza os repositórios periodicamente e essa a frequência pode ser definida pela área de recursos humanos juntamente com a equipe de desenvolvimento. Esses intervalos estão sendo adotados baseados na rotina de atualização desses dados que são definidos pela área e é uma rotina que obedece um cronograma de trabalho desse órgão.

Caso ocorra uma falha nessa atualização do processo agendado, os repositórios individuais de cada funcionalidade não são afetados e o serviço continuará funcionando, apenas estará defasado e essa falha pode ser um gatilho de aviso para as áreas envolvidas informando que há um problema a ser verificado e resolvido para que a sincronização seja realizada. Além disso, esse processo agendado pode ser executado manualmente a qualquer tempo, sem que comprometa o agendamento em vigência.

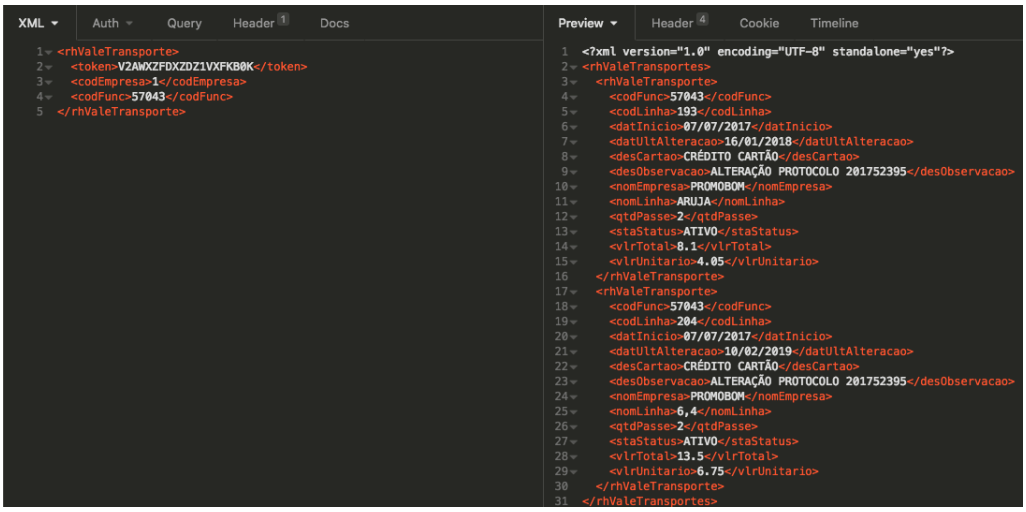
4.3 Resultados das atividades do roteiro aplicadas

Seguindo o fluxo apresentado do item 3.2 para a criação dos microsserviços, a primeira etapa refere-se a decomposição das funcionalidades, que contenham uma finalidade específica e granularidade fina (FOWLER e LEWIS, 2014).

A funcionalidade já foi definida acima como a consulta de dados de vale-transporte. Portanto, corresponde a atividade 1.

Na etapa2, para a criação das funcionalidades, é necessário obter as informações do banco de dados do sistema ERP e está no bloco de “Serviços de Busca”. Desta forma, foi criado um webservice, que retorna os dados desejados por meio de um serviço REST que disponibilizam os dados para que seja possível atualizar o repositório de dados independente para cada uma das funcionalidades.

Figura 9 – Webservice para os dados de vale-transporte



O serviço de busca obtém somente os dados correspondente a única funcionalidade, ou seja, cada busca é independente e atende apenas um objetivo que é atualizar o repositório de um serviço específico.

Baseado no webservice disponível e nos dados retornados por cada um desses serviços, criou-se o repositório para armazenar os resultados e dessa forma atender a atividade 2 que se refere a base de dados independente e disponível para a criação do serviço de consulta.

O repositório é atualizado com o conjunto de dados correspondente ao serviço de busca que atender a funcionalidade em questão e são atualizados por meio de processo agendado ou pode ser atualizado por meio de uma requisição espontânea.

A atividade 3 refere-se ao API *Gateway* que é o endereço principal do repositório de microsserviço. Neste artigo, é um endereço que direciona diretamente ao servidor de aplicações que provê os serviços. Em um caso real que esteja em produção, esse endereço pode ser substituído por um link mais amigável e que seja de fácil memorização.

O API Gateway para os serviços de busca que retornam os dados para serem atualizados nos repositórios está definido por um endereço genérico como:

http://255.255.255.255/RecursosHumanos/ws/servicos/[repositorio]

Para o microsserviço que retorna as informações as aplicações que apresentam os dados desejados, também é um endereço que direciona diretamente ao servidor de aplicações que provê os serviços e também pode ser modificado para um *link* mais intuitivo para os usuários finais.

Portanto, o API *Gateway* para as aplicações baseadas em microsserviços está definido por um endereço genérico como:

http://255.255.255.255:8080/ords/f?p=[microsservico]

Com o API *Gateway* definido, finaliza-se a atividade 3 e há um local central que é responsável por receber as requisições para o microsserviço. Apesar de não ser um requisito

obrigatório e nem apresentado anteriormente, esse endereço por ser parte de um balanceador de carga que pode apontar para mais de um servidor e distribuir as requisições realizadas.

Para a atividade 4, há a necessidade de definir os endereços individuais de cada serviço para o qual deseja-se acessar as aplicações (*endpoint*).

Portanto, os endereços de cada serviço estão definidos por um endereço genérico como:

http://255.255.255.255:8080/ords/f?p=VALETRANSPORTE

Dessa forma estão definidos os endereços individuais de cada serviço, e por se tratar de um *link*, também ser alterado para um endereço mais amigável e que facilite a memorização para acesso. Além disso, pode ser criado no site institucional do órgão para acesso dos usuários de cada um dos serviços. Com isso, finaliza-se a atividade 4 que se relaciona ao endereço direto do microsserviço que se deseja acessar.

A atividade 5 refere-se ao processo de disponibilização em produção denominado *deployment*, já que a gerência do serviço é de responsabilidade direta da equipe de desenvolvimento.

Como o servidor de aplicação foi definido na atividade 3 e tem seu endereço disponível, os microsserviços são disponibilizados em produção no endereço genérico *http://255.255.255.255:8080*.

A atividade 6 refere-se ao *access token*, que está relacionado aos acessos no serviços de busca que acessa a base de dados central do sistema ERP.

Para atualizar os repositórios, ao acessar os endereços disponíveis nos serviços de busca, é necessário o envio de uma requisição para obter a resposta. Nessa requisição, há uma *tag* que necessita ser preenchida com o conteúdo de um *token* válido para acesso aos dados.

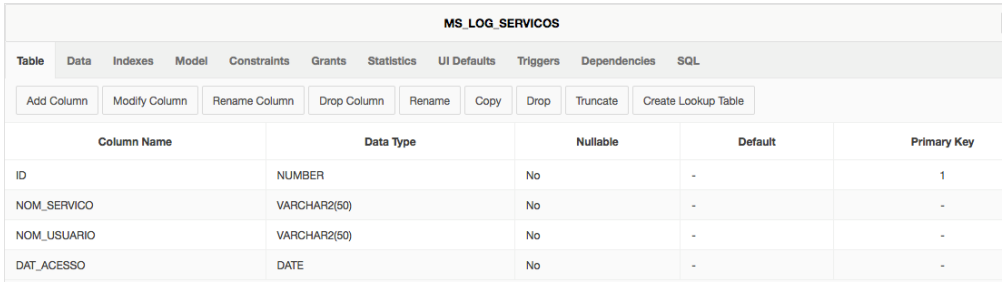
Segue um exemplo de um *token* gerado pelo sistema ERP utilizando um algoritmo a ser definido pelo responsável pela criação do serviço de deverá ser informado conforme segue:

<token>V2AWXZFDXZDZ1VXFKB0K</token>

Em relação aos serviços, quando o endereço individual é requisitado, imediatamente é requerido um usuário e senha ativos para obter o acesso e também é uma credencial de acesso para liberação da consulta dos dados.

Na atividade 7, é necessário realizar o registro de acesso aos serviços para que seja possível obter dados de número de consultas a cada um dos serviços por período e por usuário. Para isso foi criada uma estrutura que armazena esses acessos conforme figura 10:

Figura 10 – Registro de acesso aos serviços



Por fim, a atividade 8 refere-se a interface do usuário para acesso aos serviços e visualização das informações. Portanto, para o serviço criou-se um modo visual de acesso por meio de um endereço de acesso conforme definido na atividade 4.

A figura abaixo apresenta a tela de acesso ao serviço quando o endereço é requisitado em um navegador.

Figura 11 – Interface de acesso ao endereço do serviço

Com isso, a implementação da funcionalidade foi finalizada seguindo os conceitos da definição da arquitetura de microsserviços, o fluxo do processo criado para atender a demanda apresentada para a criação da funcionalidade com seus objetivos claros e definidos e aplicando as etapas do roteiro proposto no item 3.2 composto pela decomposição funcional que identificou a funcionalidade a ser criada, a criação de um banco de dados independente para cada funcionalidade, a definição de um endereço denominado *API Gateway* que é o servidor que provê acesso a funcionalidade, a definição dos endereços individuais de cada um dos microsserviços, a criação de token de acesso para a busca das informações do banco de dados central do sistema ERP, o acesso ao servidor de aplicação definido no API *Gateway* para a disponibilização do microsserviço em ambiente de produção (*deployment*), a criação de um repositório que registra o acesso ao serviço para a observação dos acessos efetuados e por fim a interface de acesso dos usuário por meio de um navegador com acesso à internet.

5. AVALIAÇÃO DO ROTEIRO NO CASO PRÁTICO

O roteiro proposto no item 3.2 está baseado nas melhores práticas dos microsserviços que foram citadas na revisão bibliográfica que apresentam arquiteturas, definições e conceitos de autores citados. Assim, a nova funcionalidade criada no sistema ERP com base neste roteiro é potencialmente aderente às melhores práticas de microsserviços encontradas na literatura.

Neste item, avaliaremos as questões de granularidade, isolamento e autonomia resultantes da aplicação do roteiro no caso prático.

A granularidade do microsserviço deve ser fina, ou seja, a funcionalidade que o microsserviço atende deve ser específica o suficiente para que seja responsável por apenas uma única finalidade, clara e direta, de modo que não tenha relação com outras funções que necessitem um controle de transação. Essa é uma premissa da etapa 1 do roteiro proposto que

se refere a decomposição funcional onde o serviço atende a um único objetivo específico que é a consulta dos dados de vale-transporte.

O isolamento tem o objetivo de garantir que o microsserviço é totalmente independente, não haja acoplamento e, no caso desse trabalho, não esteja ligado diretamente ao sistema ERP para que o serviço execute sua função. Para garantir essa validação, a execução da etapa 1 se refere a decomposição funcional para que o microsserviço tenha apenas uma funcionalidade específica e a execução da etapa 2 que cria um repositório de dados individual para o microsserviço fazem com que o serviço seja modularizado e não está diretamente ligado a nenhum outro serviço ou base de dados externa ao seu próprio repositório. Dessa forma, o microsserviço é executado individualmente e permite que seja parte de um contêiner para escalabilidade que é parte integrante da etapa 6.

A autonomia refere-se a independência do microsserviço e à elasticidade da aplicação. Essa validação está diretamente ligada ao isolamento, já que os microsserviços podem ser escalados de acordo com parâmetros definidos pela equipe de desenvolvimento quanto ao número de requisições, utilização de memória, capacidade de processamento de unidades computacionais ou quaisquer outros parâmetros que a equipe defina como relevante. Com o microsserviço isolado, pode fazer parte de uma esteira DevOps como apresentado na etapa 6 e também estar em um contêiner que podem ser criados, eliminados, duplicados ou movidos de acordo com a necessidade.

6. CONCLUSÕES

Este artigo partiu da necessidade de customização de sistemas ERP que é uma atividade frequente seja por mudanças no processo de negócio ou por alterações na legislação vigente. Essa demanda de mudanças faz com que as organizações necessitem adaptar-se à nova realidade, portanto ações para esse fim são tomadas podendo ser a customização do próprio ERP pela equipe interna ou por meio da contratação de recursos externos, ou ainda, por soluções paliativas como planilhas, bancos de dados paralelos e arquivos textos.

No caso de customizar diretamente o sistema ERP, pode haver barreiras por se tratar de um sistema composto de módulos que atendem os mais diversos departamentos de uma organização e acessam um único banco de dados centralizado. Essas alterações implicam que, quando finalizadas, deve-se atualizar o sistema ERP por completo em produção e esse procedimento pode causar comportamentos não esperados. Além disso, deve-se selecionar quais funcionalidades estarão em cada atualização em produção e somente quando todas estiverem prontas, será possível atualizar. Esse procedimento pode causar uma demora considerável para que o sistema ERP esteja adequado ao processo de negócio. Já as soluções paliativas fazem com que os dados fiquem em locais diferentes do banco de dados central e dificulta a consolidação das informações.

Nos últimos anos, surgiu a arquitetura de microsserviços como uma alternativa para o desenvolvimento de aplicações independentes, com uma função clara e objetiva e que pode ter seu próprio repositório de dados. São serviços que por natureza são desacoplados, porém podem ser integrados à fontes de dados para atender o seu propósito. Portanto, essa arquitetura permite que seja utilizada para estender um sistema ERP para a criação de novas funcionalidades que consultem ou alterem dados em um banco de dados central de forma

independente e pode utilizar tecnologias diferentes daquelas em que o sistema ERP está desenvolvido e da base de dados.

O artigo apresentou um roteiro que foi aplicado no desenvolvimento de uma funcionalidade real, porém não é possível afirmar que a conclusão após a implementação em cenários diferentes obtenham os mesmos resultados. Pode ser necessário que sejam feitos ajustes ou criação de outras etapas no roteiro e consequentemente modifiquem as atividades em uma nova implementação.

O roteiro proposto é composto por um fluxo de 8 atividades baseadas na literatura. Há ainda um fluxo de verificação de aplicabilidade do roteiro, onde requisitos associados ao repositório de dados necessário para a funcionalidade criada são avaliados.

O roteiro foi testado em uma implementação real e a nova funcionalidade do sistema ERP, além de atender aos óbvios requisitos funcionais, mostrou-se compatível com as melhores práticas de microsserviços.

5. REFERÊNCIAS BIBLIOGRÁFICAS

ALSHUQAYRAN, N.; ALI, N.; EVANS, R. A Systematic Mapping Study in Microservice Architecture. In: 9th International Conference on Service-Oriented Computing and Applications, IEEE, p. 44-51, 2016.

DAVENPORT, T. Mission critical: realizing the promise of enterprise systems. Harvard Business Press, 2000. 334p.

DAVENPORT, T. Putting the enterprise into the enterprise system. Harvard Business Review, v. 76, n. 4, 1998. 11p.

DITTRICH, Y.; VAUCOULEUR, S; GIFF, S. ERP Customization as Software Engineering: Knowledge Sharing and Cooperation. IEEE Software, v. 26, n. 6, p. 41-47, 2009.

FOWLER, M; LEWIS J. Microservices. 2014. Disponível em: <https://martinfowler.com/articles/microservices.html>. Acesso em: 16 maio 2017.

GADEA, C.; TRIFAN, M.; IONESCU, D.; IONESCU, B. A reference architecture for real-time microservice api consumption. Proceedings of the 3rd Workshop on CrossCloud Infrastructures & Platforms. ACM, p. 2, 2016.

GUO, D.; WANG, W.; ZENG, G.; WEI, Z. Microservices architecture based cloudware deployment platform for service computing. In: IEEE Symposium on Service-Oriented System Engineering (SOSE), IEEE, p. 358-363, 2016.

KITCHENHAM, B.; DYBÅ, T.; JØRGENSSEN, M. Evidence-Based Software Engineering. In: ICSE. IEEE, p. 357-363, 2004.

KNOCHE, H. Sustaining runtime performance while incrementally modernizing transactional monolithic software towards microservices. Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering. ACM, p. 121-124, 2016.

PRISMA STATEMENT. Consulta geral a homepage oficial. Disponível em: <http://www.prisma-statement.org>. Acesso em: 10 abr. 2020.

RICHARDSON, C. Microservice Architecture. 2017. Disponível em: <https://microservices.io>. Acesso em: 10 abr. 2020.

ROTHENBERGER, A.; SRITE, M. An investigation of customization in ERP system implementations. IEEE Transactions on Engineering Management, p. 663-676, 2009.

SJØBERG, D.; DYBÅ, T.; JØRGENSSEN, M. The future of empirical methods in software engineering research. In: FOSE. Washington, USA, p. 358-378, 2007.

STUBBS, J.; MOREIRA, W.; DOOLEY, R. Distributed systems of microservices using docker and serfnode. In: 7th International Workshop on Science Gateways (IWSG). IEEE, p. 34-39, 2015.

SUDHAMAN, P.; DANEVA, M. An approach to estimation of degree of customization for ERP projects using prioritized requirements, Journal of Systems and Software, v. 117, p. 471-487, 2016.

THÖNES, J; Microservices. IEEE Software, v. 32, p. 116, 2015.

VIENOT, N.; LÉCUYER, M.; BELL, J.; GEAMBASU, R.; NIEH, J. Synapse: A microservices architecture for heterogeneous-database web applications. Proceedings of the Tenth European Conference on Computer Systems, ACM, p. 21, 2015.

VILLAMIZAR, M.; GARCÉS, O.; CASTRO, H.; VERANO, M.; SALAMANCA, L.; CASALLAS, R.; GIL, S. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In: 10th Computing Colombian Conference, IEEE, p. 583-590, 2015.