INTEGRANDO MICRO SERVIÇOS EM UMA APLICAÇÃO WEB[1]

ERICK CARVALHO DE SÃO MIGUEL[2], RENATA MIRELLA FARINA[3]

[1]Projeto de pesquisa (Iniciação Científica) – CCA/Uniara - 2016

[2] Graduando em Sistemas de Informação, Uniara, Araraquara, erickcmiguel@gmail.com; [3]Mestre em Engenharia de Produção, USP, EESC, mirellafarina@yahoo.com.br

PALAVRAS-CHAVE: Micro serviços; WEB; PHP;

INTRODUÇÃO:

A constante expansão e inovação tecnológica no setor de desenvolvimento de aplicações vêm crescendo cada vez mais, o que provoca também a aparição de novos problemas ou desapoderamento de velhas tecnologias ou metodologias devido às novas tendências e demandas do mercado. Um velho problema do qual permanece até os dias atuais é a segunda lei de evolução de software: “a medida que um [sistema] evolui, sua complexidade aumenta, a menos que algum trabalho seja feito para mantê-la ou reduzi-la” (Lehman, 1980). Este mesmo problema foi mencionado por Robert Glass, que disse: “Para cada 25% de aumento em funções de negócio de um sistema, há um aumento de 100% na complexidade do serviço” (Robert Glass, 2002, p.50), ou seja, ao passo que uma aplicação cresce ela se torna mais complexa, o que necessita de um cuidado maior na hora de alterar, aprimorar ou integrar novas funções, cuidado que é mais custoso de ser aprendido e aplicado. O problema da complexidade é ainda mais aparente em uma estrutura monolítica, que graças a sua individualidade proporciona determinadas deficiências em alguns casos. Devido aos problemas que surgem com a complexidade, empresas como Netflix, Vtex e Spotify decidiram adotar outra estrutura para suas aplicações, conhecida por ser como um contraponto da estrutura monolítica: os micro serviços, introduzidos oficialmente em 2005 na "*CloudComputing Expo*" (Omed Habib, 2016), possui a premissa de: “desenvolver uma aplicação com um conjunto pequeno de serviços independentes” (Dmitry Namiot, 2014, p.1). Embora o termo micro serviço seja relativamente novo, o seu conceito básico já era utilizado há algum tempo, já

que fora extraído do SOA - Service Oriented Architecture. Proposto pela primeira vez em 1996 por W. Roy Schulte e Yefim Natis, o SOA é definido como: “Uma abordagem de design onde os múltiplos serviços colaboram para fornecer um conjunto final de recursos” (Sam Newman, 2015, p.22). Enquanto o SOA é uma orientação maior constituinte de compatibilidades baseadas em políticas, compartilhamento de contratos e afins, os micro serviços são uma especialização do SOA, focando mais na autonomia de serviços e sendo explicitamente delimitado. As propostas do micro serviço estabelecidas por Sam Newman (2015, p.12-22) dizem que, ao se fragmentar a aplicação em várias partes autônomas, obtêm-se várias vantagens tais como: autonomia, maior dinamismo, complexidade menor, possibilita heteronomia de tecnologias, facilidade de *deployment*, composabilidade, eleva a disponibilidade e escalabilidade de cada fragmento, o que consequentemente também eleva da aplicação como um todo.

METODOLOGIA

Foi criada uma aplicação monolítica em PHP, utilizando o *framework Laravel* versão 4.2, com uma banco de dados em MYSQL contendo três tabelas: *users*, *posts* e *comments*. As três tabelas demarcam um módulo da aplicação, cada um contendo um *CRUD.*

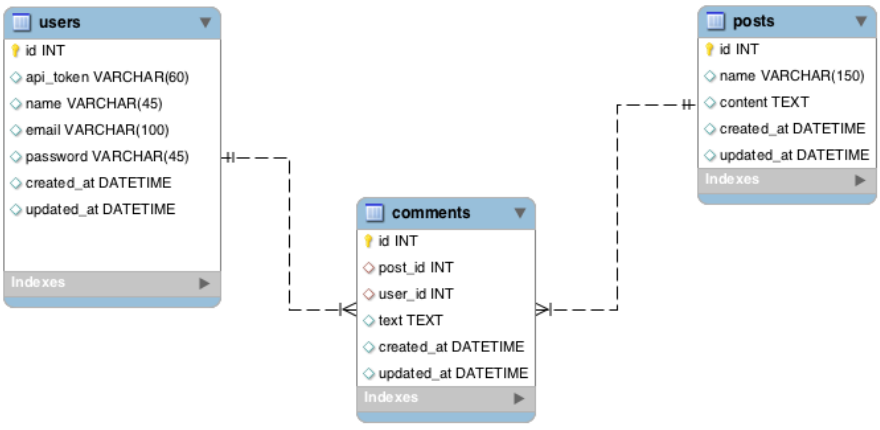

Figura 1 – Diagrama do banco de dados. FONTE: Autor

O módulo de comentários foi retirado da aplicação monolítica e integrado em uma aplicação separada, se tornando o micro serviço – apesar disto, utiliza o mesmo banco de dados. A aplicação monolítica contém uma *Helper Class*, que

com auxilio de um *third-party package* que utiliza *CURL* em *HTTP*, é capaz de realizar *requests* para o micro serviço. Foi criado uma *API* na aplicação do micro serviço, esta é usada para retornar dados e requisitar funções do módulo de comentários, como por exemplo: retornar uma lista de comentários de um determinado *post* e armazenar um novo comentário no banco de dados.

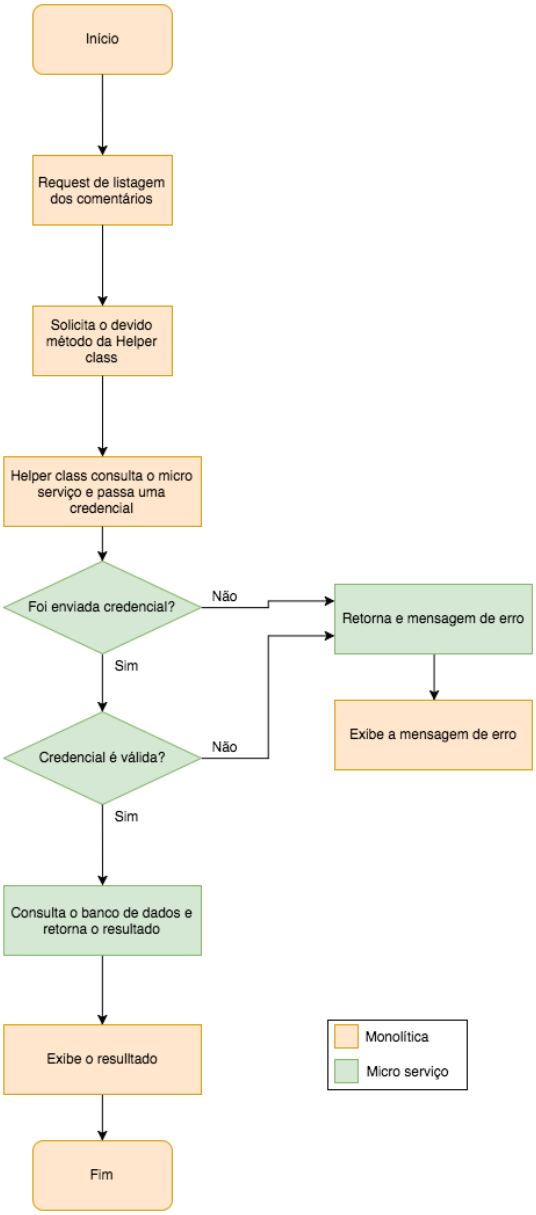

Figura 2 – Fluxograma de processos. FONTE: Autor

Após implementado, será apresentado os resultados obtidos em uma aplicação simples e comparados em relação alguns pontos da proposta de Sam Newman, sendo assim: checar se a autonomia foi obtida baseada na localização dos dados, se a Heteronomia de tecnologia foi alcançada através da atualização do framework *Laravel* do micro serviço da versão 4.2 para a 5,

se as aplicações podem se comunicar, se foi adquirido uma maior composabilidade e uma maior escalabilidade.

RESULTADOS E DISCUSSÃO

Devido a separação das aplicações, a autonomia de cada um foi obtida, já que a base de dados foi separada e isolada uma da outra. A Heteronomia de tecnologia também foi alcançada, a aplicação de micro serviço utiliza *Laravel 5.3* enquanto a monolítica utiliza a versão 4.2, ambas sendo estruturas muito diferentes uma da outra. Através da *Helper Class* tornou-se possível a aplicação monolítica se comunicar com o micro serviço, isso foi possível devido a compatibilidade de dados das aplicações. A composabilidade se tornou possível entre aplicações externas, contanto que usem as credências, é possível a utilização dos métodos do micro serviço em qualquer outra aplicação. Em contra ponto disto, se tornou mais custoso utilizar os métodos do micro serviço, já que para isto, é necessário realizar uma *REQUEST* em *HTTP*. A aplicação monolítica se tornou menor e devido a organização e isolamento dos dados, cada uma delas se tornou mais simples e portanto, menos complexa. No entanto, o módulo de usuários que é presente em ambas as aplicações, contradiz a ideia de diminuir a complexidade já que seus dados estão sendo duplicados, caso fosse necessário adicionar alguma regra ou lógica para o mesmo, teria que ser alterado na aplicação monolítica e na aplicação de micro serviço, portanto este é um fator que possa tornar a aplicação como um todo mais complicada e complexa de ser desenvolvida.

CONCLUSÃO

Foi possível alcançar as premissas estabelecidas, embora para isso foi-se obtido em troca de um custo, seja de performance ou de duplicação de informação. a aplicação se tornou menos complexa devido a agregação dos dados e mais fácil de ser desenvolvida, embora que para isto seja necessário uma adaptação de ambas as aplicações, para estabelecer uma forma de comunicação e disponibilidade de métodos. O módulo de usuário pode ser um empecilho na fluidez do desenvolvimento, portanto para que o fluxo atinja sua melhor eficiência seria necessário tomar alguma medida quanto a isto. Isto só é possível pois ambas as aplicações possuem similaridade de regras no módulo

de usuários, pois caso fosse necessário regras distintas nas aplicações, não seria viável integrar micro serviços visto que iria duplicar os dados. O desafio maior de integrar os micro serviços é saber o que pode ou deve ser agregado e o que deve ser separado, podendo ser demasiado trabalhoso integrar esta estrutura caso uma aplicação possua muitos módulos dos quais requerem interações entre si o tempo todo. No geral, micro serviços tornam a aplicação menos complexas como um todo, porém tornam-na mais complexa em alguns aspectos, o que não quer dizer necessariamente que suas desvantagens arruínam as premissas da estrutura.

REFERÊNCIAS BIBLIOGRÁFICAS

DMITRY NAMIOT. **On micro-services architecture.** International Journal of Open Information Tecnologies ISSN: 2307-8162 vol. 2, no. 9, 2014, p.1, Setembro. 2014.

LEHMAN, M. M. (1980). **Programs, life cycles and the laws of software evolution**. Proceedings of the IEEE, 68(9):1060–76.

W. Roy Schulte e Yefim Natis, **Service Oriented Architectures.** Gartner, 12 de abril de 1996.

OMED HABIB. **A quick Primer on Microservices.** SYS-COM Media, Inc. , 19 de fevereiro, 2016.

ROBERT GLASS. **Facts and Fallacies of Software Engineering.** Addison-Wesley Professional. v. 1, 7 de novembro de 2002.

SAM NEWMAN. **Building Microservices**. O'Reilly Media, Inc. , vol. 1, p.12-22