Migração de Sistemas Monolíticos para Microserviços

Guilherme L. D. Villaca[1]

[1]Universidade Estadual do Oeste do Paraná (Unioeste)
Cascavel – PR – Brazil

guidvillaca@gmail.com

Abstract. *Migrating monolithic systems to microservices has been an industry trend to address old software development issues such as scalability and maintainability. However, it is not simple to identify when these systems should migrate and what the consequences are. One technique for applying microservices is sensational in theory and when you plan to start and start zero development. In practice, issues such as splitting a database to meet the principles of independence, dynamism and deployment facilitated especially in an environment with many consolidated systems is no easy task.*

Resumo. *Migrar sistemas monolíticos para microserviços tem sido uma tendência da indústria para resolver velhos problemas enfrentados no desenvolvimento de software, como escalabilidade e manutenibilidade. Porém não é simples identificar quando estes sistemas devem migrar, e quais as consequências disso. A técnica de aplicação de microserviços é sensacional na teoria e quando se planeja previamente e inicia o desenvolvimento do zero. Na prática, questões como dividir uma base de dados para que atenda aos princípios de independência, dinamismo e deploys facilitados especialmente em um ambiente com muitos sistemas consolidados, não é tarefa fácil.*

Keywords: monolíticos, microserviços, reúso de software, reengenharia, migração, banco de dados.

1. Introdução

A técnica de desenvolvimento de software baseado em reúso é um processo de design que pode resultar em redução de tempo e custos de desenvolvimento além de aumentar a flexibilidade, manutenibilidade e confiabilidade do software. A principal razão para pensar em reúso é evitar trabalhos repetidos no desenvolvimento de software e usar do conhecimento e experiência adquiridos previamente e concentrar os esforços em partes mais críticas da aplicação. Seu foco é não necessitar mais desenvolver um software do zero [Yang et al.].

Reengenharia é uma forma para atingir o reuso de software e para entender os conceitos ocultos no domínio da aplicação. Seu objetivo é obter e manter o conhecimento incluído nos sistemas legados, usando-o como base para a continuidade e evolução estruturada do sistema. O código legado tem lógicas programadas, decisões de projeto, requisitos de usuários e regras de negócio que podem ser recuperadas e reconstruídas sem perder a semântica [Garcia et al. 2004]

Os sistemas legados são tradicionalmente construídos de forma monolítica, ou seja, um sistema composto por módulos que não são independentes da aplicação a que pertencem

[Dragoni et al.]. Para resolver as limitações dos sistemas monolíticos, os Microserviços estão sendo amplamente utilizados pela indústria por ser uma técnica que estimula uma melhor modularização e gestão através de serviços menores e autônomos [**?**]. Ter um sistema mais adaptável a mudanças é a melhor estratégia para modernizar sistemas legados [Kamimura et al. 2019].
O objetivo deste trabalho é analisar os sistemas desenvolvidos pelo Núcleo de Tecnologia da Informação (NTI) da Unioeste e verificar como pode ser resolvido os problemas enfrentados pelo setor. No NTI são cerca de 28 sistemas monolíticos e todos eles de alguma forma interligados. Em um cenário como esse com deve ocorrer a migração, de uma forma mais gradual e a longo prazo? Somente as pequenas partes interligadas poderiam se tornar microserviços? Apesar de existir no estado da arte guias e processos para migração, não há uma definição principalmente em um cenário como este.

2. Desenvolvimento

2.1. Revisão Bibliográfica

O desenvolvimento de sistemas pelas organizações segue um padrão onde o desenvolvimento começa pela solução especifica de um problema e a partir daí novas soluções são adicionadas, formando um ou mais sistemas maiores e complexos. Com o passar dos anos esses sistemas podem gerar problemas para a equipe de desenvolvedores. Se a implementação de uma nova funcionalidade ou correção de uma falha tiver qualquer barreira, seja por código mal escrito, complexidade estrutural, rotatividade na equipe onde perde-se conhecimento de domínio tem-se um dos seguintes problemas:(rever)

- Manutenibilidade: Que é a capacidade de um software ser modificado. Se um software não for construído com capacidade para lidar com a demanda de constantes mudanças, certamente irá consumir mais recursos e tornar a manutenção uma tarefa tediosa [Velmourougan et al. 2014].
- Escalabilidade: É a capacidade de um software para acomodar crescimento em termos de tamanho e complexidade do problema e em um segundo aspecto está relacionado a uma medida de eficácia quando usada em um contexto maior em escopo e complexidade do que o contexto para o qual foi originalmente projetado [Ibrahim et al. 2009].
- Confiabilidade: Eficácia do sistema quanto a estar livre de falhas por um período de tempo. É difícil atingir a confiabilidade de um software, pois quanto mais complexo, menos confiável ele se torna.[Pan 1999].
- Qualidade: (reescrever) Garantia de que o software atenda a todos os requisitos.

A primeira abordagem da utilização de microserviços surgiu na indústria, e o modo aplicado não é o mesmo hoje recomendado pelo estado da arte. Uma das principais diferenças está em como gerenciar a base de dados. Um sistema monolítico, geralmente, tem uma base de dados e o estado da arte sugere a separação dessa base onde cada microserviço tenha a sua própria base de dados. Portanto, ao migrar um monolítico para microserviços, deve-se também migrar a base de dados.

A divisão de um banco de dados é uma das principais questões quando se fala em migrar monolíticos para microserviços. Cada microserviço deve ser responsável pelos seus próprios dados, ou seja, independentes para que qualquer mudança não afete o sistema todo [Kholy and Fatatry 2019]. É importante entender como um sistema é dividido em componentes e como esses componentes interagem

2.2. Trabalhos Relacionados

2.3. Metodologia

3. Resultados Esperados

Resultados

4. Cronograma de Execução

Cronograma

Referências

Dragoni, N., Giallorenzo, S., Lafuente, A. L., and Mazzara, M. Microservices : yesterday , today , and tomorrow. pages 1–17.

Garcia, V. C., Lucrédio, D., Do Prado, A. F., De Almeida, E. S., and Alvaro, A. (2004). Using reengineering and aspect-based techniques to retrieve knowledge embedded in object-oriented legacy system. *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration, IRI-2004*, pages 30–35.

Ibrahim, H., Far, B. H., and Eberlein, A. (2009). Scalability Improvement in Software Evaluation Methodologies. *2009 IEEE International Conference on Information Reuse & Integration*, pages 236–241.

Kamimura, M., Yano, K., Hatano, T., and Matsuo, A. (2019). Extracting Candidates of Microservices from Monolithic Application Code. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, 2018-Decem:571–580.

Kholy, M. E. and Fatatry, A. E. (2019). Framework for Interaction between Databases and Microservice Architecture. *IT Professional*, 21(5):57–63.

Pan, J. (1999). Software Reliability. pages 1–9.

Velmourougan, S., Dhavachelvan, P., Baskaran, R., and Ravikumar, B. (2014). Software development Life cycle model to improve maintainability of software applications. *2014 Fourth International Conference on Advances in Computing and Communications*, pages 270–273.

Yang, X., Chen, L., Wang, X., and Cristoforo, J. A Dual-Spiral Reengineering Model for Legacy System.