UNIVERSIDADE FEDERAL DO PAMPA

Bolívar Teixeira de Teixeira

Avaliação Comparativa entre Implementação FaaS e Microserviço de um Software ERP

Alegrete
2021

Bolívar Teixeira de Teixeira

Avaliação Comparativa entre Implementação FaaS e Microserviço de um Software ERP

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Claudio Schepke

Alegrete
2021

Bolívar Teixeira de Teixeira

Avaliação Comparativa entre Implementação FaaS e Microserviço de um Software ERP

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em 09 de março de 2022.
Banca examinadora:

Prof. Dr. Claudio Schepke
Orientador
UNIPAMPA

Prof. Dr. Diego Luis Kreutz
UNIPAMPA

Profa. Dr. Aline Vieira de Mello
UNIPAMPA

Profa. Dra. Aline Vieira de Mello

UNIPAMPA

Assinado eletronicamente por **DIEGO LUIS KREUTZ**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 09/03/2022, às 16:59, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **ALINE VIEIRA DE MELLO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 09/03/2022, às 17:19, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **CLAUDIO SCHEPKE**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 09/03/2022, às 17:34, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0748938** e o código CRC **06FDFF32**.

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

T266a Teixeira, Bolívar Teixeira de Teixeira
Avaliação comparativa entre implementação FaaS e Microserviço de um Software ERP / Bolívar Teixeira de Teixeira Teixeira.
34 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade Federal do Pampa, CIÊNCIA DA COMPUTAÇÃO, 2021.
"Orientação: Claudio Schepke Schepke".

1. Computacao em nuvem. 2. Function as a Service. 3. Microservice. I. Título.

AGRADECIMENTOS

Em primeiro lugar, agradeço a minha família por terem me dado todo o apoio necessário durante a graduação. Em especial, agradeço aos meus avós Idalina e José Francisco, "In Memorian", por todos os ensinamentos e por acreditar na minha capacidade. Aos amigos conquistados na graduação, também aos amigos de longa data que sempre estiveram presentes em meus pensamentos. Aos meus colegas de curso, que de alguma forma ou de outra me auxiliaram nessa jornada. Também gostaria de deixar meu sincero agradecimento ao meu orientador, Claudio Schepke, que esteve presente nos momentos cruciais. Meu muito obrigado, espero tê-los em minhas lembranças para o resto da minha vida.

“Mesmo que a vida pareça difícil, há sempre algo
que você pode fazer para ter sucesso nela.“
(Stephen William Hawking)

RESUMO

Os serviços hospedados em nuvem cresceram e movimentaram bilhões de dólares nos últimos anos. Esse mercado atrativo é disputado por grandes provedores de hospedagem, que têm fornecido diferentes arquiteturas de software, as quais derivam principalmente do padrão *Service Oriented Architecture (SOA)*. Entre as arquiteturas mais adotadas estão *Microservice* e *Function as a Service* (FaaS). Embora FaaS seja uma evolução direta do padrão *Microservice*, os dois possuem diferenças que mudam completamente a forma de trabalhar com cada um. Como estudo de caso, são investigados os impactos de cada arquitetura em uma aplicação *Enterprise Resource Planning* (ERP). Os resultados obtidos mostram que a conversão da aplicação para as duas arquiteturas permitiu a reutilização de código, pois a aplicação é *stateless*. Em termos de performance, *Microservice* mostrou-se cerca de 40% melhor que FaaS para 10 diferentes tipos de requisições implementadas para a aplicação.

Palavras-chave: Arquitetura Orientada a Serviços. Função como Serviço. Microserviço. Software de Planejamento de Recursos Empresariais. Computação em Nuvem. Migração de Software. Avaliação de Desempenho.

ABSTRACT

Cloud-hosted services have grown and moved billions of dollars in recent years. Large hosting providers disputed this attractive market, which has provided different software architectures that derive mainly from the Service Oriented Architecture (SOA) standard. Microservice and Function as a Service (FaaS) are the most SOA adopted. Even though FaaS is a direct evolution of the Microservice standard, the two have differences that completely change the way they work with each one. In this case study, we investigate the impacts of each architecture on an Enterprise Resource Planning application. The results show that the conversion of the application to the two architectures allowed the reuse of code because the application is stateless. In terms of performance, Microservice proved to be around 40% better than FaaS for the 10 different types of requests implemented for the application.

Key-words: Service Oriented Architecture. Function as a Service. Microservice. Enterprise Resource Planning. Cloud computing. Software Migration. Performance Evaluation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API *Application Programming Interface*

AWS *Amazon Web Services*

ERP *Enterprise Resource Planning*

FaaS *Function as a service*

JSON *JavaScript Object Notation*

PaaS *Platform as a service*

SUMÁRIO

1 INTRODUÇÃO

A computação em nuvem tem sido amplamente utilizada nos últimos anos (BUYYA; VECCHIOLA; SELVI, 2013). Este modelo permite que tanto usuários empresariais como domésticos acessem recursos computacionais sob demanda, ou seja, podem fazer uso de acordo com a necessidade das aplicações. Para isso, a computação em nuvem possibilita dimensionar aplicações tanto verticalmente como horizontalmente, permitindo que usuários utilizem os recursos de acordo com as necessidades que possuem no momento.

Existem diversos padrões arquiteturais que podem ser utilizados para o desenvolvimento de *softwares* baseados em nuvem (Savchenko; Radchenko; Taipale, 2015). Dentre esses padrões, dois vem sendo amplamente utilizados nos provedores de nuvem: Microsserviços (Alshuqayran; Ali; Evans, 2016) e *Function as a service* (FaaS) (García López et al., 2018).

Microserviço é um padrão usado há um bom tempo na indústria de desenvolvimento de software, principalmente no que se refere à migração de aplicações monolíticas para essa arquitetura. A arquitetura consiste em diversos serviços desenvolvidos independentemente e que, após serem implementados, são acoplados e passam a funcionar conjuntamente com as outras partes da aplicação de microsserviços. Na Figura 1 ilustramos como são construídos os Microsserviços. O segundo padrão, FaaS, é baseado em computação sem servidor, onde o desenvolvedor cria a lógica de sua aplicação e executa através de *containers*, que são gerenciados pelo provedor de serviços de nuvem. Dessa forma, o desenvolvedor poupa tempo, pois não tem a necessidade de configurar o servidor para utilização e, desse modo, contribui para agilizar o desenvolvimento de sua aplicação.

Com o crescimento da computação em nuvem, as empresas que possuem seus

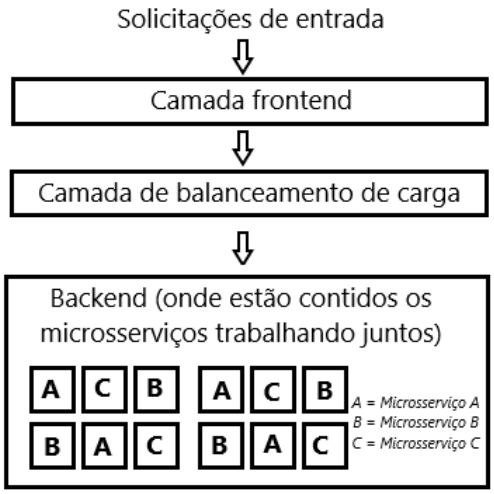

Figura 1 – Microsserviços.

Fonte: Adaptado de FOWLER, 2019

software em *datacenters* locais também precisam inovar e realizar a migração para não correrem riscos de perderem clientes. Usualmente estas empresas fazem o uso de *Enterprise Resource Planning*s (ERPs) que é um sistema de permite realizar a gerência dos dados da empresa de forma confiável e integrada (SAP, 2022). Ou seja, ERP engloba diversas funcionalidades ou áreas que uma empresa possui em apenas uma plataforma. Uma proposta para a modernização desse ERP pode ser a migração para a nuvem e fazer uso da arquitetura de microsserviços ou de FaaS para melhorar seu desempenho. Dessa forma, dispondo de internet o acesso poderá ser realizado em qualquer horário e local que seus clientes ou colaboradores precisarem. Além disso, outros fatores podem ser citados como: segurança, controle de processos na empresa, redução de custos com infraestrutura, mobilidade e escalabilidade.

1.1 Motivação e objetivos

Neste sentido, este trabalho tem como objetivo prover uma comparação direta entre os dois padrões de arquitetura: FaaS e Microsserviços em uma aplicação do tipo *Enterprise Resource Planning* (ERP). Para tanto, como estudo de caso foi escolhida a aplicação Escolar App[1], desenvolvida por Luis Guilherme Pires Moura. Para cumprir as demandas do projeto de atender um grande número de escolas, a aplicação foi migrada para as arquiteturas *Microservice* e FaaS. Para esta aplicação, foram abordados aspectos referentes ao desenvolvimento, tecnologias, performance e custos.

O trabalho contribui para a caracterização de critérios que auxiliam na escolha da melhor arquitetura para o desenvolvimento de aplicações do tipo ERP. Para tanto, os resultados estão baseados em métricas e metodologias validadas em trabalhos relacionados.

1.2 Organização do documento

O trabalho está organizado da seguinte maneira. No Capítulo 2 estão os artigos utilizados como referência para o desenvolvimento deste trabalho. No Capítulo 3 são apresentados os métodos, software e ambiente aplicados para a execução dos testes. O Capítulo 4 aborda a ferramenta utilizada para extração dos dados e discute os resultados obtidos nos testes realizados. Por fim, no Capítulo 5, são trazidas as considerações finais e as direções de trabalhos futuros.

[1] https://gitlab.com/hypnoinc/escolar-app

2 TRABALHOS RELACIONADOS

Neste Capítulo discorreremos sobre os trabalhos relacionados que tem pontos em comum com o trabalho proposto. Diante disso, realizamos uma revisão da literatura para compreendermos e verificarmos quais trabalhos utilizaram essas arquiteturas, como se deu a implementação, as métricas utilizadas e o modelo de arquitetura de nuvem adotada.

O trabalho (SHAHRAD; BALKIND; WENTZLAFF, 2019) ajuda a entender de forma mais aprofundada o funcionamento da virtualização presente na arquitetura *FaaS*, revelando que o problema de *Cold-start* característico da arquitetura se comporta de maneira diferente conforme o padrão de chamadas, partindo do cliente, separados em três padrões:

a) *Over-invoked*, quando o número de chamadas excede o número de *workers*, criando uma fila de execução;

b) *Under-invoked*, quando as chamadas são separadas com um tempo levemente superior a 55ms, que é o tempo que o *openwhisk* leva para por o *worker* em estado suspenso;

c) *Balanced*, onde as requisições são bem espaçadas para que o servidor tenha um bom espaço para resposta.

No experimento, o padrão de chamadas *over-invoked* apresentou maior tempo de *Cold-start*. Em nosso estudo de caso vamos medir o mesmo comportamento, porém como utilizamos provedores diferentes, esperamos um comportamento distinto e, como adicional, vamos comparar este tempo com a implementação em *Microservice*, para analisar o tamanho da diferença entre as duas arquiteturas.

Em (JR et al., 2017) é apresentado um estudo comparativo entre FaaS e *Platform as a service*s (PaaSs). A proposta é parecida com nosso trabalho, mas se diferencia no fato de analisar as arquiteturas em softwares já implementados e não no processo de desenvolvimento, uma vez que analisamos uma aplicação durante todo o seu desenvolvimento.

O artigo (IVANOV; SMOLANDER, 2018) contribui com a apresentação de métricas. Neste trabalho foi avaliado o impacto da arquitetura FaaS em práticas Devops. Dentre as 27 práticas Devops analisadas, 18 sofreram influência relacionadas ao padrão de arquitetura. Em nosso trabalho, nós também iremos considerar algumas métricas para avaliar de forma comparativa FaaS e *Microservice* usando uma aplicação real, além de considerar o impacto dessas arquiteturas no desenvolvimento do código.

No estudo de (BOGNER et al., 2019) foram realizadas entrevistas com equipes de desenvolvimento de dez empresas sobre a adoção e implementação de microsserviços e FaaS na indústria alemã. As entrevistas foram focadas nas tecnologias utilizadas, características das arquiteturas e como elas influenciaram na qualidade do *software*. Este estudo se assemelha a nossa avaliação pois faz a utilização de *software* monolíticos e necessitam

migrar para arquiteturas de microsserviços e FaaS. Porém, conforme o relato da maioria das equipes de desenvolvimento das empresas, não conseguiram implementar toda uma arquitetura de microsserviços ou de FaaS.

3 METODOLOGIA

Neste Capítulo é introduzida a metodologia proposta para a avaliação. O foco desse trabalho é realizar a avaliação comparativa entre as implementações de arquiteturas FaaS e Microsserviços executadas em um *software* de ERP e analisar os dados coletados. Na Seção 3.1 discorremos sobre o avanço da computação em nuvem, aplicação ERP e também sobre as arquiteturas utilizadas para o desenvolvimento do trabalho. Também foi necessário realizar uma migração do ambiente para um provedor de nuvem que atendesse a proposta e permitisse executar o necessário utilizar no projeto. Este conteúdo está disponível na Seção 3.2. Já na Seção 3.3 introduzimos a realização da migração do *software* legado monolítico para as arquiteturas propostas e também a utilização de um provedor de nuvem para a execução do mesmo. Por último, mencionamos o que iremos utilizar como métrica para avaliação do trabalho. Na Seção 3.4 falamos sobre o provedor de nuvem utilizado na orquestração deste trabalho, o motivo de sua escolha e também sobre algumas de suas funcionalidades. Na Seção 3.5, apresentamos o sistema ERP ao qual obtemos acesso para realizar essa avaliação. Neste caso, discorremos sobre algumas funcionalidades que a aplicação possuí e também acerca do desenvolvimento e sobre o desenvolvedor.

3.1 Proposta do Trabalho

Para realizar a comparação entre as duas arquiteturas optou-se por um estudo de caso envolvendo uma aplicação monolítica desenvolvida para ser utilizada na área da educação, essencialmente em escolas. A aplicação em questão possui algumas funções que são necessárias para se ter um melhor controle na área da educação escolar como: informações do aluno, notas, turmas entre outros. Porém, a mesma tem o seu formato monolítico, ou seja, é uma aplicação única que faz a utilização de um mesmo código.

A fim de cumprir com as demandas propostas para esta aplicação que visa atender diversas escolas, foi necessário realizar a migração da mesma para um provedor de nuvem, onde há mais liberdade para começar a estruturar uma aplicação ERPs, fazer ajustes e também trabalhar com as arquiteturas de *microservices* e FaaS escolhidas para essa avaliação.

3.2 Migração do projeto

Durante a migração do projeto foi necessário escolher um provedor de nuvem que atendesse a demanda da aplicação e também que os desenvolvedores conseguissem trabalhar com ambas as arquiteturas no mesmo ambiente. Dessa forma, optou-se pela utilização do *Amazon Elastic Compute Cloud (Amazon EC2)* (AMAZON, 2022c), que é um dos provedores que contempla diversos tipos de computação em apenas um ambiente. O serviço é conhecido como *Lambda*(AMAZON, 2022b) para FaaS e foi utilizado o serviço EC2 para a implementação em microsserviço.

Para evitar interferências no estudo, as arquiteturas são acessadas pelo mesmo cliente, recebendo requisições com dados no padrão *JavaScript Object Notation*s (JSONs) e acessadas no mesmo banco de dados em um servidor externo.

3.3 Aplicação Avaliada

A avaliação consiste na migração de um *software* legado monolítico, reutilizando sua estrutura de dados. O objetivo dessa migração era reescrever o *software* de acordo com os dois padrões de arquitetura já apresentados.

No desenvolvimento, foi utilizado uma aplicação monolítica, a qual foi migrada para uma arquitetura FaaS e posteriormente para o código escrito para a arquitetura de microsserviços. Através da Figura 2 é possível ter uma noção de como é o funcionamento de cada uma dessas arquiteturas.

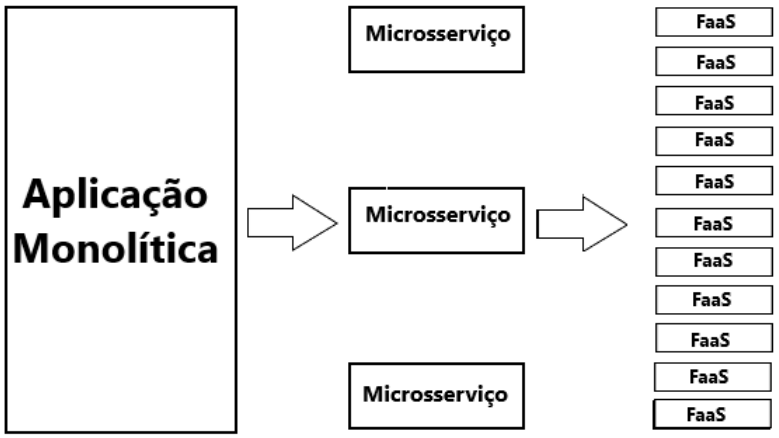

Figura 2 – Monolítica, Microsserviço e FaaS

Fonte: Do autor

Como métricas para avaliar os resultados obtidos foram considerados o tempo de resposta e valor do custo de processamento. Desta forma, serão levados em conta o tempo de processamento em milissegundos e custos relacionados a contratação do serviço em nuvem.

3.4 Ambiente de Testes

Para executar esse projeto foi escolhido o provedor de nuvem da Amazon, denominado *Amazon Web Services* (AWS). O motivo para esta escolha deste é que, além de ser o provedor de nuvem mais utilizado, também é o que mais possui funcionalidades. Sua documentação é constantemente atualizada. Dessa forma, quando surgiram dúvidas

na implementação ou algum erro foi possível recorrer a documentação para solucionar os mesmos.

O Amazon EC2 possui mais de 475 instâncias, diversas opções de processador, armazenamento, rede, sistema operacional e mais de uma opção de modelo de nuvem para a aplicação (AMAZON, 2022c). Tratando-se de infraestrutura, o EC2 possui um modelo seguro, confiável, econômico e de alta performance para atender as demandas do solicitado no projeto.

A utilização do EC2 foi através das instâncias do tipo T3 para a arquitetura de microsserviços. Nesse caso, a T3.micro que possui 1 GB de memória RAM e 2 vCPUs. Estas, são instâncias de uso geral, de baixo custo e alto desempenho que apresentam processadores Intel Xeon Platinum da série 8000 que, em questão de preço, são melhores do que as utilizadas em instâncias do tipo T2. Instâncias T3 são criadas no AWS *Nitro System*, que é composto por diversos componentes básicos que transferem várias funções tradicionais de virtualização para *hardware* dedicado, que permite alto desempenho, alta disponibilidade, segurança, além de diminuir a sobrecarga de virtualização (AMAZON, 2022a).

Já para colocar em prática a execução utilizando a arquitetura de FaaS foi utilizado o serviço de computação sem servidor AWS Lambda, que também é orientado a eventos e permite executar código para praticamente qualquer tipo de aplicação ou serviço de *backend*, sem a necessidade de provisionar ou gerenciar servidores. Alguns dos destaques de Lambda é a possibilidade de acioná-lo a partir de mais de 200 serviços da AWS, pagar conforme o uso da aplicação e também ter suporte nativo para várias linguagens de programação.

Durante a realização dos testes, analisamos a latência média (avg) do provedor AWS. Após realizada a coleta do resultado, tivemos uma média de 26,32 ms.

3.5 Estudo de Caso - Escolar App

O sistema Escolar App foi desenvolvido por Luis Guilherme Pires Moura para permitir o gerenciamento de escolas. Inicialmente o aplicativo possuía modelos para acesso de alunos, pais destes alunos e professores. Algumas das funcionalidades do sistema são: trocar mensagens com professores, alunos e também com os administradores da escola; conferir o horário de suas aulas e agenda; opção de visualização de eventos agendados na escola; uma agenda exclusiva para professores, onde é possível compartilhar com outros professores suas datas de trabalhos e provas; recebimento de avisos sobre datas importantes da escola; *feedback* e atualização de notas no momento em que os professores fizerem o fechamento e lançamento no sistema; e acompanhamento de frequência do aluno.

A cada nova versão, outras funcionalidades foram adicionadas e falhas de software corrigidas, como a aba de desempenho. Assim é possível que: os pais possam acompanhar o desempenho de seus filhos e a frequência do mesmo; há um sistema de consulta para a

secretaria, onde é possível verificar a avaliação de cada aluno e sua respectiva frequência e, por último, anotações de professores acerca de cada aluno.

A seguir, iremos discorrer sobre as funções testadas na aplicação e como elas se comportaram.

- login: a função login serve para o aluno, professor ou administrador do sistema possa ser autenticado ou não na plataforma. Caso o usuário consiga se autenticar, o sistema direcionará o usuário de acordo com a autorização que o mesmo possui dentro da plataforma.

- turma: esta função permite fazer a busca de turmas cadastradas na plataforma. Através dela é possível listar as turmas existentes conforme a *string* utilizada e também relacionar os alunos e professores que estão cadastrados nela.

- usuário: dentro dessa função estão englobados os cadastros validados de alunos, professores, diretores, secretários e pais de alunos.

- aluno: através dessa função é possível cadastrar determinado aluno, adicionar o email do mesmo, gerar uma senha e, após isso, salvar o cadastro. Também está disponível no nível de aluno a página de avaliação, que é constituída das notas desse aluno junto de sua matrícula e nome.

- disciplina: função que pode cadastrar ou excluir disciplinas que estejam dentro da grade curricular da escola. As disciplinas são determinadas de acordo com a sua sigla (ex: AL001), nome ou cor.

- nota: se tratando da função nota, ela é obtida de acordo com a disciplina ao qual determinado aluno está matriculado. A consulta das notas se dará através do período que está sendo pesquisado no sistema, podendo ser via bimestre, trimestre, semestre ou até mesmo anual. Também existe uma subcategoria e indica o conceito (CSA, CPA e CRA) atingido por esse aluno dentro dessas notas obtidas.

As implementações das aplicações desenvolvidas foram realizadas anteriormente a este trabalho por Luis Guilherme Pires Moura (Android APK) estão disponíveis para as versões FaaS [1] e Microsserviço[2].

[1] https://drive.google.com/file/d/1eIbkl-hxavTWE0KQ9mKZvDUghpLW1djy/view?usp=sharing
[2] https://drive.google.com/file/d/1mtXhPLI2b3M-C3SsBHYRyCf-1Mm5Nrj0/view?usp=sharing

4 RESULTADOS

Para extração dos dados quantitativos foi utilizado o *Postman API Platform* (POSTMAN, 2022). Através dele é possível registrar solicitações e executá-las em qualquer sequência ou latência, registrando o tempo de resposta retornado por cada *Application Programming Interface* (API). Na Tabela 1 são detalhadas as requisições *endpoints* bem como as duas arquiteturas, onde foram realizados os testes e seus respectivos tempos de execução.

Tabela 1 – Tabela com o tempo de resposta em ms de cada API por *endpoint*.

Requisições (endpoint)	**FaaS**	**Microservices**
login	106	89
turma	91	116
turma/salvar	280	38
usuário	102	38
aluno	78	39
aluno/salvar	333	40
aluno/senha	80	116
nota	102	51
disciplina	79	109
nota/salvar	294	49

Ao analisar a Tabela 1 é possível constatar que a arquitetura de Microsserviço sobressaiu-se em sete das dez requisições ao qual foram submetidas. Uma das causas desse evento é que a arquitetura de Microsserviço tem um tempo de resposta mais eficiente quando os dados tratados tem pouco tráfego de requisições. Por outro lado, a arquitetura FaaS utiliza *cold start*, o que causa atraso na execução da solicitação e, por esse motivo, não se sobressai em relação a arquitetura de Microsserviços. Nos demais casos, as APIs foram estressadas com menor tempo entre as requisições. Dessa forma, equilibrou-se a performance das duas nos diferentes *endpoints*.

Vale ressaltar que ambas arquiteturas tem custos de utilização diferentes no ambiente de hospedagem da Amazon, mesmo estando as mesmas localizadas na mesma região (América Latina – Brasil). A utilização da arquitetura FaaS através do Amazon Lambda cobra 0,20 USD por cada milhão de solicitações acrescentado de 0,0000166667 USD por cada *gigabyte* por segundo processado. Por outro lado, a arquitetura de microsserviços foi instanciada em uma máquina no ambiente virtual do Amazon EC2 e tem seu custo da seguinte maneira: 0,0168 USD por hora, para utilização do servidor t3.micro com dois core e 1 GB de memória RAM.

Estimando-se que a aplicação tenha consumo médio de 15 milhões de requisições e tempo médio de 150ms, a arquitetura FaaS custará mensalmente 2,50 USD. Já o valor para a utilização da arquitetura de microsserviços, com hardware que seja capaz de suportar a mesma quantidade de requisições, será em torno de 5 USD pagos mensalmente. Vale

destacar que essa estimativa foi elaborada de acordo com a quantidade de municípios e escolas que nosso estado possui.

5 CONSIDERAÇÕES FINAIS

Neste trabalho, realizamos uma análise comparativa entre FaaS e microsserviços de forma prática em uma aplicação ERPs. Em seu estado inicial, a aplicação era monolítica e foi realizada a migração para ambas as arquiteturas de FaaS e microsserviços. Ao longo da execução e construção dos *endpoints* percebemos que o reaproveitamento de código foi significativo tanto para a arquitetura de FaaS como para a de microsserviços, fazendo assim com que o trabalho fosse agilizado. Vale ressaltar que a arquitetura de microsserviços mostrou-se mais eficiente no processamento das requisições que foram desenvolvidas para esta aplicação, porém o custo de sua utilização é o dobro da arquitetura de microsserviços.

5.1 Trabalhos futuros

Para trabalhos futuros, buscaremos explorar outros provedores de nuvem como o Microsoft Azure e realizarmos os testes para a mesma aplicação ERP, podendo assim comparar um provedor com o outro na métrica de desempenho e custos para hospedagem. Também podemos explorar outras funcionalidades que as arquiteturas de microsserviços e de FaaS podem nos oferecer.

REFERÊNCIAS

Alshuqayran, N.; Ali, N.; Evans, R. A Systematic Mapping Study in Microservice Architecture. In: **2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)**. [S.l.: s.n.], 2016. p. 44–51. Citado na página 23.

AMAZON, A. **Amazon EC2 T3 Instances, acessado em 10 de janeiro de 2022**. 2022. Disponível em: <https://aws.amazon.com/pt/ec2/instance-types/t3>. Citado na página 29.

AMAZON, A. **AWS Lambda, acessado em 10 de janeiro de 2022**. 2022. Disponível em: <https://aws.amazon.com/lambda>. Citado na página 27.

AMAZON, A. **Elastic Compute Cloud, acessado em 10 de janeiro de 2022**. 2022. Disponível em: <https://aws.amazon.com/pt/ec2/>. Citado 2 vezes nas páginas 27 e 29.

BOGNER, J. et al. Microservices in industry: insights into technologies, characteristics, and software quality. In: IEEE. **2019 IEEE International Conference on Software Architecture Companion (ICSA-C)**. [S.l.], 2019. p. 187–195. Citado na página 25.

BUYYA, R.; VECCHIOLA, C.; SELVI, S. T. **Mastering cloud computing: foundations and applications programming**. [S.l.]: Newnes, 2013. Citado na página 23.

García López, P. et al. Comparison of FaaS Orchestration Systems. In: **2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)**. [S.l.: s.n.], 2018. p. 148–153. Citado na página 23.

IVANOV, V.; SMOLANDER, K. Implementation of a DevOps pipeline for serverless applications. In: SPRINGER. **International Conference on Product-Focused Software Process Improvement**. [S.l.], 2018. p. 48–64. Citado na página 25.

JR, L. F. A. et al. Function-as-a-Service X Platform-as-a-Service: Towards a Comparative Study on FaaS and PaaS. **ICSEA 2017**, p. 217, 2017. Citado na página 25.

POSTMAN, I. **Postman API Platform, acessado em 02 de janeiro de 2022**. 2022. Disponível em: <https://www.postman.com>. Citado na página 31.

SAP. **What Is ERP?, acessado em 5 de janeiro de 2022**. 2022. Disponível em: <https://insights.sap.com/what-is-erp/>. Citado na página 24.

Savchenko, D. I.; Radchenko, G. I.; Taipale, O. Microservices validation: Mjolnirr platform case study. In: **2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**. [S.l.: s.n.], 2015. p. 235–240. Citado na página 23.

SHAHRAD, M.; BALKIND, J.; WENTZLAFF, D. Architectural Implications of Function-as-a-Service Computing. In: **Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture**. [S.l.: s.n.], 2019. p. 1063–1075. Citado na página 25.