Projeto Orientado em Computação II
Relatório Final
Propostas para Melhoria de Processos em um Projeto com Arquitetura de Microsserviços

Ingrid Rosselis Sant Ana da Cunha[1], **Dorgival Olavo Guedes Neto**[1], **Walter do Santos Filho**[1]

[1] Departamento de Ciência da Computação –

{irscunha,dorgival,walter}@dcc.ufmg.br

Tipo de Pesquisa: Científica

Abstract. *Speed, reliability, scalability and security are desirable requirements for software development and distribution nowadays. The DevOps culture and microservice architecture make it easy to implement some of these points in a more fluid way. However, it is challenging to use all the power these approaches can provide without making extensive use of resources, whether personal or financial. This work aims to propose process improvements for a project with microservice architecture that has a complex team and is completely open source, with the purpose of accelerating the development and delivery flow. It is also intended to propose technical improvements to the various services that structure the reviewed project.*

Resumo. *Velocidade, confiabilidade, escalabilidade e segurança são requisitos desejáveis para o desenvolvimento e a distribuição de software nos dias atuais. A cultura DevOps e a arquitetura de microsserviços facilitam a implementação de alguns desses pontos de maneira mais fluída. Contudo, é um desafio utilizar todo o poder que essas abordagens podem fornecer sem fazer uso extensivo de recursos, sejam esses de pessoal ou financeiro. Este trabalho tem como objetivo propor melhorias de processos para um projeto com arquitetura de microsserviços que possui uma equipe complexa e é completamente open source, visando a aceleração do fluxo de desenvolvimento e entrega. Deseja-se também propor melhorias técnicas para os diversos serviços que estruturam o projeto revisado.*

1. Introdução

O processo de desenvolvimento de *software* está em constante evolução, graças a diversas mudanças em cultura, processos, técnicas e ferramentas que vêm acontecendo ao longo do tempo. Essas movimentações têm impactado todo o cenário tecnológico no que diz respeito à criação de aplicações – desde a como estas são pensadas e como são desenvolvidas e disponibilizadas, à forma como empresas e equipes se organizam e à maneira como a comunidade é moldada em torno dessas mudanças. Podemos citar como alguns responsáveis por essas mudanças, dentre outros, o *software as a service (SaaS)*, a computação em nuvem, a arquitetura de microsserviços, as metodologias ágeis e a cultura *DevOps*.

Ao passo em que essas evoluções alteram o cenário da produção de *software*, cada vez mais o mercado atual exige que o desenvolvimento e a distribuição das aplicações sejam velozes, eficientes, confiáveis, seguros e que possibilitem a escalabilidade do *software*. Alguns desses fatores, como escalabilidade e velocidade no desenvolvimento, são efeitos colaterais ganhos quando conceitos como microsserviços e *DevOps* são aplicados de maneira correta.

E apesar de todo o movimento e crescimento proporcionado pelos aspectos anteriormente citados, nem sempre é possível aplicá-los a todos os times e a todos os projetos. Especialmente, algumas das mudanças mais drásticas requerem o uso extensivo de recursos, como pessoal e financeiro, o que pode dificultar ou tornar inviável a obtenção de um estado do *software* considerado adequado de acordo com as diretrizes aplicadas. É importante saber que os novos conceitos e ferramentas não resolvem todos os problemas em um projeto, e agregam novas complexidades ao mesmo.

Dessa forma, o objetivo deste projeto é avaliar a aplicabilidade de práticas amplamente difundidas na cultura *DevOps* sobre o Lemonade, um projeto *open source* com arquitetura de microsserviços mantido pelo Departamento de Ciência da Computação da Universidade Federal de Minas Gerais. Como resultado deste relatório, deseja-se propor um conjunto de melhorias de processos para a equipe que o desenvolve e um conjunto de melhorias práticas para o projeto que permitam a aceleração do fluxo de trabalho, levando em conta as particularidades e o contexto da equipe, e sabendo que o processo de adesão às propostas pode, inclusive, diminuir temporariamente a velocidade do fluxo atual no projeto.

Este relatório discutirá aspectos da contextualização do cenário atual de desenvolvimento de *software*, do movimento *DevOps* e do projeto Lemonade que foram estudados como parte do processo de estruturação das propostas. Em paralelo, será avaliada uma pesquisa qualitativa feita de maneira anônima com os desenvolvedores que contribuem no projeto para entender suas experiências com desenvolvimento do projeto. Por fim, serão discutidas propostas para a aceleração de fluxo e melhoria de processos para o Lemonade baseadas nos estudos feitos e na análise das respostas do questionário.

2. Referencial Teórico

Como citado na introdução deste relatório, dificuldades são comumente enfrentadas ao implantar conceitos da cultura *DevOps* ou da arquitetura de microsserviços em um projeto novo ou existente e que será rearquitetado. Neste âmbito, [Balalaie et al. 2015] traz um relatório de experiências sobre a migração de um monólito para uma arquitetura de microsserviços. A decisão de migrar para a nuvem com serviços distribuídos deu-se pelas necessidades de reusabilidade, de *deploy* automático e de escalabilidade, dentre outros fatores. No relatório feito são descritos os passos da migração e, por fim, são relatadas algumas dificuldades experienciadas e como foram resolvidas como, por exemplo, o *deploy* em ambiente de desenvolvimento, a necessidade de ter desenvolvedores que entendam de serviços distribuídos e de ter processos de desenvolvimento gerais e bem definidos, independentes de serviços ou linguagens de programação.

Em relação à considerações para a adoção de práticas *DevOps*, [Virmani 2015] traz a aplicabilidade desta tecnologia durantes vários passos do ciclo de vida de um *software*, focando em cobrir a lacuna entre a integração contínua e a entrega contínua, e trazendo um exemplo real de manutenção de infraestrutura em forma de *infrastructure as a code (IaaC)*. Ainda, é enfatizado que o movimento *DevOps* traz apenas um conjunto de princípios, e que as empresas e equipes devem estar confortáveis para decidir como adotar estes princípios e quais ferramentas utilizar, dependendo de suas particularidades.

Em [Taibi et al. 2018] foi feito um mapeamento sistemático em 23 artigos publicados para caracterização de princípios e padrões de diversos estilos de arquiteturas de microsserviços, como o padrão arquitetural de *API-Gateway*. Dos padrões considerados, foram feitos estudos comparativos em relação às vantagens e desvantagens de cada modelo. Além disso, os autores mapearam técnicas e ferramentas específicas de microsserviços no processo da cultura *DevOps*, comparando-as em diferentes estágios do processo de entrega contínua.

3. Contribuição

3.1. Metodologia

As atividades executadas neste projeto constituem-se de estudos aprofundados sobre a cultura *DevOps* e sua práticas, com enfoque em seu uso em conjunto com arquiteturas de microsserviços. Foram revisados artigos que trazem visões sobre as tecnologias e suas práticas difundidas, assim como os possíveis conceitos que as definem.

Para a proposição de melhorias, foi necessário buscar um entendimento sobre o projeto Lemonade e sua situação atual. Para isso, foram utilizado artigos, materiais informativos e os códigos que estão disponíveis pelo *Github* [1]. Somado ao estudo de boas práticas, foi possível identificar os principais problemas atuais no projeto, principalmente relacionado à melhoria do fluxo de desenvolvimento e entrega da aplicação.

Ainda, foi feita uma pesquisa qualitativa com alguns contribuidores do Lemonade, de forma anônima, para entender melhor as experiências que estes obtiveram no projeto, a fim de consolidar as propostas geradas neste documento. O questionário foi composto por oito perguntas, focadas em processos que compõe o desenvolvimento, e relacionadas à experiências dos desenvolvedores com conceitos tais quais *DevOps*, *deploy*, revisão de código, fluxo de versionamento, processo de desenvolvimento, documentação, monitoramento e fluxo de correção de *bugs*, dentro outros.

Por fim, foi gerado um compilado contendo propostas de melhoria de processos para o time de desenvolvimento e de melhorias práticas para os serviços que compõe o Lemonade, de maneira generalizada.

3.2. Contextualização

Como citado na introdução, as mudanças tecnológicas ocorreram por consequência da introdução de novos conceitos, ferramentas e metodologias. Em especial, o *software as a service (SaaS)*, a computação em nuvem e as metodologias ágeis foram alguns responsáveis por auxiliar o início de outras tecnologias como a arquitetura de microsserviços e o *DevOps*.

[1] https://github.com/eubr-bigsea?page=1

O *software as a service (SaaS)*, como mostrado em [Turner et al. 2003], é um modelo de distribuição de *software* dinâmico e focado em compor serviços sobre demanda a partir de requerimentos de negócio. Assim, a aplicação se torna um conjunto de serviços distribuídos, autônomos e isolados que podem ser parametrizados e expandidos de maneira independente, mudando a forma como esta aplicação é desenvolvida, entregue e evoluída, facilitando a escalabilidade e a manutenção.

A era da computação em nuvem deu novas dimensões a todo o processo de desenvolvimento de *software*, em conjunto com o conceito de *software as a service*, pois tornou as grandes estruturas físicas e seus times de operações uma decisão opcional para cada projeto. Nos dias atuais, grandes empresas como Amazon e Google, plataformas *SaaS*, fornecem diversos serviços na nuvem, desde o provisionamento de máquinas físicas – onde o usuário define quais especificações de máquina deseja ter a seu dispor por determinado preço fixo, até mesmo serviços do tipo *serveless*, onde o usuário consegue executar uma aplicação sem ter contato com a máquina em que será executada, e sendo cobrado pelo tempo de execução da sua aplicação. Dessa forma, empresas como Netflix e Nubank mantém toda a sua infraestrutura na nuvem utilizando majoritariamente serviços como monitoramento e provisionamento, dentre outros, de uma única plataforma.

A arquitetura de microsserviços é um estilo de arquitetura nativa da nuvem para construção de aplicações distribuídas que herda do conceito de *software as a service* a ideia de que os componentes devem estar separados e distribuídos em serviços menores [Dragoni et al. 2017]. Um microsserviço é um serviço que possui apenas uma funcionalidade e se comunica através da troca de mensagens. Esse padrão de arquitetura facilita a implementação, o teste e a entrega de novas características, além de favorecer a escalabilidade, a conteinerização e a manutenção do *software* como um todo.

As metodologias ágeis também contribuíram para essas mudanças no quesito de processos. Estas visam entregas mais curtas e um desenvolvimento mais fluído, sempre mantendo as ideias de aumento da manutenibilidade, da confiança no código produzido e da diminuição do impacto de novas modificações na experiência do usuário.

3.3. DevOps

Inicialmente como um movimento que visava melhorar a comunicação e a colaboração entre times de desenvolvimento e de operações, o *DevOps* se tornou uma cultura colaborativa e amplamente difundida com propósito de tornar o desenvolvimento e a entrega de código tão rápido, fluído e flexível quanto possível. O movimento teve seu início em meados da década de 2010, possui bases nas metodologias ágeis e tem sofrido um intenso aumento de popularidade nos últimos anos, sendo um dos principais fatores para mudança de paradigma no desenvolvimento de aplicações. Além disso, atualmente é considerado como um conjunto de conceitos, práticas e ferramentas que propõe um novo fluxo do desenvolvimento de *software* que inclui a garantia de qualidade, de manutenção e de entrega.

Os conceitos fundamentais de *DevOps* sofrem constantes alterações dependendo da época e de onde são definidos, mas algumas ideias chaves se mantêm. Em 2015, [Lwakatare et al. 2015] trouxe uma revisão literária para dimensionar o fenômeno do movimento que ainda estava estabelecendo suas bases. Foram analisados artigos e foram feitas entrevistas com praticantes de *DevOps*, identificando quatro elementos que carac-

terizavam a cultura *DevOps*, sendo eles a colaboração entre times através do compartilhamento de informações e de responsabilidades, a automação de tarefas para acompanhar o desenvolvimento ágil e o conceito de integração contínua (CI), a medição para metrificar a eficiência do *software* e o monitoramento contínuo dos serviços para diminuir os impactos de falhas e garantir, junto da medição, que os desenvolvedores tenham *feedback* necessário para planejar futuras melhorias.

Das práticas e conceitos mais referenciados, é válido citar os ”3 caminhos do *DevOps*”, que são os princípios sumarizados no livro *”The phoenix project: a novel about IT, DevOps, and helping your business win.”* [Kim et al. 2013]. O primeiro princípio é a aceleração do fluxo de desenvolvimento, representado na figura 1 como os processos de planejamento, implantação, provisionamento, teste e entrega, no sentido da esquerda para direita. Essas etapas fazem parte dos processos de integração contínua – unir códigos novos à base que executa em produção de maneira constante, e de entrega contínua – gerar valor ao cliente em pequenas e constantes etapas.

O segundo princípio está relacionado à ampliação do processo de *feedback* contínuo, representado na imagem 1 como o passo de monitoramento, da direita para a esquerda. O objetivo desta etapa é avaliar o desempenho da aplicação sendo usada pelo cliente, para prevenir grandes impactos causados por falhas e para gerar análises que se traduzam em melhorias no *software*. Por último, o terceiro princípio é a ideia do aprendizado contínuo, fechando o ciclo do *DevOps*. A partir dos *feedback* recebidos na segunda etapa, um novo planejamento é feito para trabalhar na correção de problemas e na implantação de novas melhorias, voltando ao primeiro princípio.

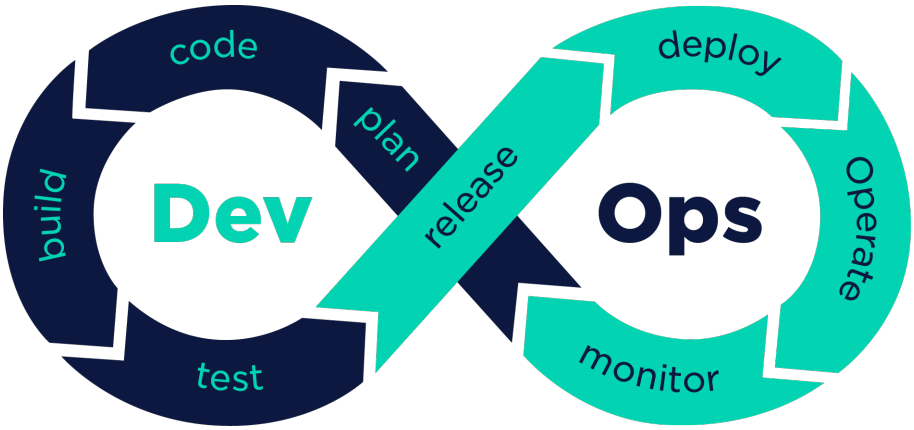

Figura 1. Fluxo *DevOps*. [2]

3.4. Caso de Estudo: Lemonade

O caso de uso a ser tratado neste relatório é referente ao Lemonade (Live Exploration and Mining Of a Non-trivial Amount of Data from Everywhere). Este é uma aplicação *open source* nascida no projeto Big Sea [3], uma parceria entre universidades do Brasil – incluindo a Universidade Federal de Minas Gerais através do Departamento de Ciência da Computação, e da Europa com o foco em processamento de dados em nuvem, provendo

[2]https://www.360logica.com/blog/agile-to-devops/
[3]http://www.eubra-bigsea.eu/

aplicações para gerar maior confiança na pesquisa e produção de *software* usando *Big Data* na nuvem.

Apresentado em [Santos et al. 2018], Lemonade é uma plataforma *web* para a criação de fluxos iterativos de análise e mineração de dados, abstraindo fluxos complexos de processamento e provendo uma solução de alta usabilidade, escalável, robusta, segura e que executa na nuvem. O projeto tem por motivação ajudar grupos que tenham interesse em pesquisas com dados, mas que não possuem conhecimento necessário de programação.

Nascido em uma arquitetura de microsserviços, o Lemonade é composto por oito serviços, como pode ser visto na figura 2, responsáveis por gerir informações sobre bases e operações de mineração de dados, pela autenticação, pelo controle da execução de fluxos e pela visualização de resultados, dentre outros. A aplicação executa via *Kubernetes* em servidores dedicados. O *software* permite ainda que os fluxos de mineração de dados sejam exportados para outras plataformas como *Apache Spark*, *TensorFlow* – usando a biblioteca *Keras* como uma camada de abstração, e *Scikit-Learn*.

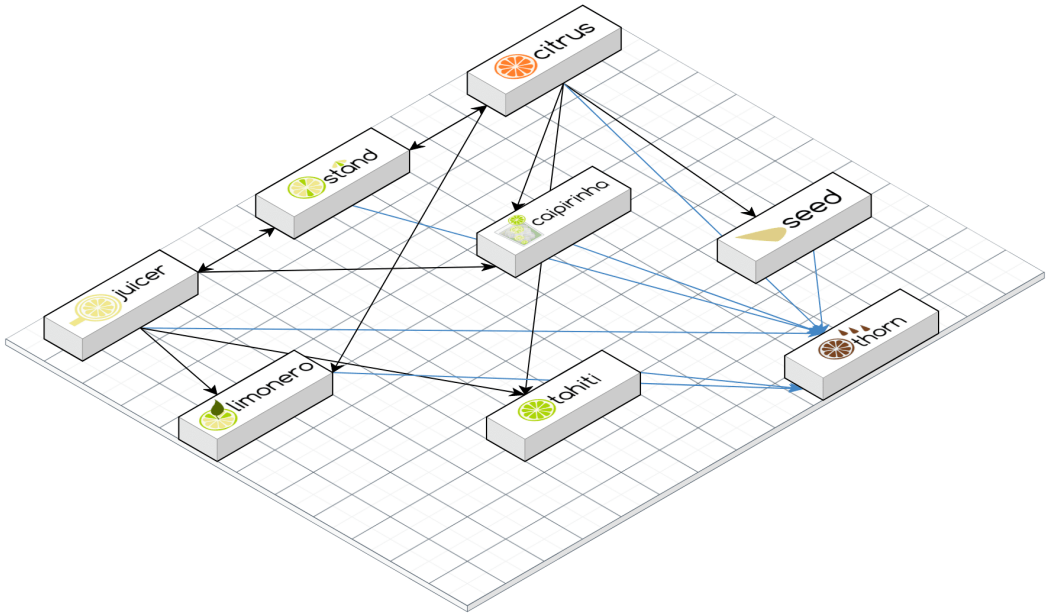

Figura 2. Arquitetura atual do projeto Lemonade simplificada.

O projeto foi selecionado como caso de uso desta análise por ser completamente *open source* – não utilizando serviços providos por plataformas *SaaS*, e por já estar em uma arquitetura de microsserviços, tendo sido implementado de maneira alheias às práticas e padrões sugeridos pela comunidade para esse estilo de arquitetura. Além disso, o Lemonade possui uma equipe de desenvolvedores complexa e descentralizada, com diferentes níveis de experiência e vivências em relação ao desenvolvimento de *software*.

3.5. Problemas Encontrados e Soluções Propostas

Definimos como problemas encontrados no projeto do Lemonade pontos e questões que vão de encontro com o especificado pelos conceitos e práticas das tecnologias *DevOps* e

microsserviços. Destes, vários estão relacionados à melhoria de processos por parte da equipe de desenvolvedores, para obtenção final de um fluxo de desenvolvimento e entrega mais veloz e eficiente. Outros são melhorias que contribuem com a qualidade dos serviços oferecidos e que também irão auxiliar na melhora do fluxo de entrega.

Os próximos parágrafos desta seção são referentes aos problemas encontrados e suas soluções propostas, respectivamente. Serão discutidos as questões em torno destes problemas, quais os benefícios de corrigi-los e uma estimativa de custo em tempo e esforço para correção.

O primeiro ponto em relação à melhorias que deve ser discutido é a definição de um conjunto de processos para cada etapa do desenvolvimento e da entrega do *software*. Analisando as respostas recebidas na pesquisa qualitativa, fica claro que existem níveis variados de experiência e conhecimento entre os desenvolvedores, e algumas perguntas sobre processos receberam respostas muito distintas. Por exemplo, ao serem questionados sobre monitoramento de serviços, alguns participantes responderam que existem *logs* apenas no *Github*, outros disseram que existe a geração de *logs* por parte dos microsserviços, mas que os arquivos de *logs* estão descentralizados e não são analisados, outros disseram que os serviços são monitorados pelo administrador de infraestrutura e alguns não sabiam como funcionava o processo de monitoramento dos serviços.

Estas respostas divergentes também foram percebidas nas questões sobre os processos de revisão de código, de correção de *bugs*, de fluxo de versionamento e de deploy. E, no entanto, as respostas possuem um padrão: alguns participantes seguem processo – não necessariamente igual ao processo de outros, alguns não tem um processo definido e, por diversas vezes, alguns não sabiam como normalmente ocorrem esses processos no escopo do projeto.

Por isso, seria interessante a geração de um documento com processos bem definidos, gerais ou específicos para cada serviço, que sejam seguidos por todos os membros da equipe. O escopo dos processos deve englobar o fluxo de versionamento, de documentação das atividades no quadro do projeto, de revisão de código e de geração de testes, *logs* e documentação. A definição de processos tem um custo relativamente baixo, dependendo mais do empenho da equipe em segui-los, e garantirá um entendimento maior das tarefas que estão sendo desenvolvidas no projeto a cada instante, uma qualidade intrínseca sabendo que os passos mínimos do desenvolvimento foram cumpridos e uma facilidade em poder assistir outros participantes do projeto em suas atividades, visto que os processos serão de conhecimento geral.

Ainda sobre a adoção de processos, outro ponto de suma importância é a criação de um quadro centralizado e digital, podendo, por exemplo, ser baseado em um *Kanban* simples, mas que dê visibilidade às demandas, problemas, débitos técnicos e refatorações necessárias em todos os microsserviços. Algumas das opções *open source* existentes atualmente permitem a divisão das cartas no quadro pelos serviços, pelo seu tipo (*bug*, história, épico ou tarefa, por exemplo), e ainda permitem atribuir as cartas aos desenvolvedores, trazendo um senso de responsabilidade sobre as tarefas e o conhecimento de todos sobre etapas e deveres de cada participante. O custo da implantação desta mudança é baixo e é uma alternativa mais centralizada ao atual processo de definição das tarefas em quadros por microsserviço.

Outro ponto importante é a adoção da etapa de revisão de código. Normalmente, as equipes adicionam um passo entre o desenvolvimento e o *deploy* onde o código, usualmente incluindo testes, é revisado por outros desenvolvedores. Esse tipo de processo é muito simples de ser implementado e ajuda na maior garantia da qualidade do código que está sendo entregue, aumenta o aprendizado dos participantes da revisão e melhora a interação do time.

Atualmente, existe um processo de documentação das entregas feitas no projeto onde um grupo fica responsável por documentar as modificações feitas depois de certo períodos e em lotes. Apesar de ser um processo bem definido, seria interessante que a documentação de melhorias ou correções fosse feita em paralelo à implantação das mesmas para diminuir o esforço do grupo que fica responsável pela documentação e para reduzir a necessidade de designar membros da equipe para esta tarefa.

Uma melhoria prática indispensável ao projeto é a criação de testes, especialmente unitários, de integração, e de contrato – considerando como testar a interação entre serviços distribuídos pode ser um desafio. Os testes garantem o aumento da qualidade na entrega e diminuição do tempo de teste em ambiente de *staging*. O custo de correção pode ser bastante alto, dependendo do estado atual da cobertura de testes, mas a longo prazo é um investimento que compensa pela segurança agregada às entregas, com a garantia que as partes já existentes estão em conforme com o que há de novo, e o aumento na velocidade do fluxo de desenvolvimento e manutenção.

Baseando nos conceitos de integração contínua (CI) e entrega contínua (CD), princípios da cultura *DevOps*, é importante que os microsserviços possuam seus *pipelines* automatizados. Um *pipeline* é uma sequência de tarefas que são executadas em vários estágios depois de uma modificação, por exemplo, ao criar um pedido de *merge* para adicionar novas características à base de código já existente e que executa em produção. Em um exemplo didático, um *pipeline* teria por função executar em sequência os testes unitários do serviço, os testes de contrato, conferir o *quality gate* – ferramenta que será discutida futuramente, e fazer a integração do código à base do serviço.

Vários tipos diferentes de *pipelines* podem ser adicionados ao mesmo projeto com intuitos distintos, como um processo para *deploy* das modificações em servidores de forma automática. O custo de implementação de *pipelines* automatizados não é alto e estes garantem segurança não só da qualidade de código, mas que todos os passos dos processos de desenvolvimento e entrega estão sendo cumpridos na ordem correta e que nenhuma etapa está sendo esquecida. Além disso, deve-se considerar que a atuação feita por humanos é muito propensa a falhas, por isso, os processos de entrega de novas versões devem ser mais automáticos quanto for possível.

Como citado em parágrafos anteriores, o *quality gate* é uma ferramenta extremamente poderosa que garante qualidade no desenvolvimento. Sua função é avaliar diversos pontos da base de código e encontrar *bugs* e débitos técnicos. Além disso, as ferramentas de *quality gate* conseguem comparar entregas baseadas na diminuição da cobertura de testes ou no aumento de problemas técnicos, podendo inclusive bloquear um *pipeline* automático caso algum requisito mínimo definido para o projeto seja violado. O custo da implantação de um *quality gate* é baixo, este não requer um processo de CI/CD bem desenvolvido para funcionar e garante que a qualidade da base de código não será com-

prometida a cada entrega. Existem boas ferramentas de *quality gate open source* hoje em dia, como o *SonarQube*.

O processo de monitoramento de microsserviços é uma boa prática prevista pelo movimento *DevOps* e que deve ser amplamente utilizado, principalmente numa arquitetura com um número maior de serviços distribuídos. O monitoramento através de mensagens de *log* permite averiguar se o funcionamento do serviço está como o esperado, consegue extrair diversas métricas sobre o funcionamento da aplicação e ajuda a detectar problemas assim que estes passam a existir, como falhas de execução e sobrecargas. As ferramentas de monitoramento são amplamente difundidas no mercado e existem várias opções *open source* extremamente utilizadas, como *Prometheus*. E para que o monitoramento seja efetivo, é necessário que cada serviço tenha implementado códigos de *log* de boa qualidade e que permitam a análise pelo sistema de monitoramento, que retornará aos desenvolvedores *feedback* sobre o produto, para assim manter um ciclo de aprendizado contínuo [Chen 2019].

Por último, uma melhoria interessante que pode ser aplicada no projeto Lemonade é a centralização do acesso de *endpoints* do projeto adicionando uma *API Gateway* e transformando a arquitetura no padrão arquitetural *API Gateway*, como mostrado na simulação feita na figura 3. A *API Gateway* simplifica a comunicação entre microsserviços, diminui as chances de sobrecarga por solicitações em um componente e facilita a escalabilidade e a modificação de serviços na *API* [Taibi et al. 2018].

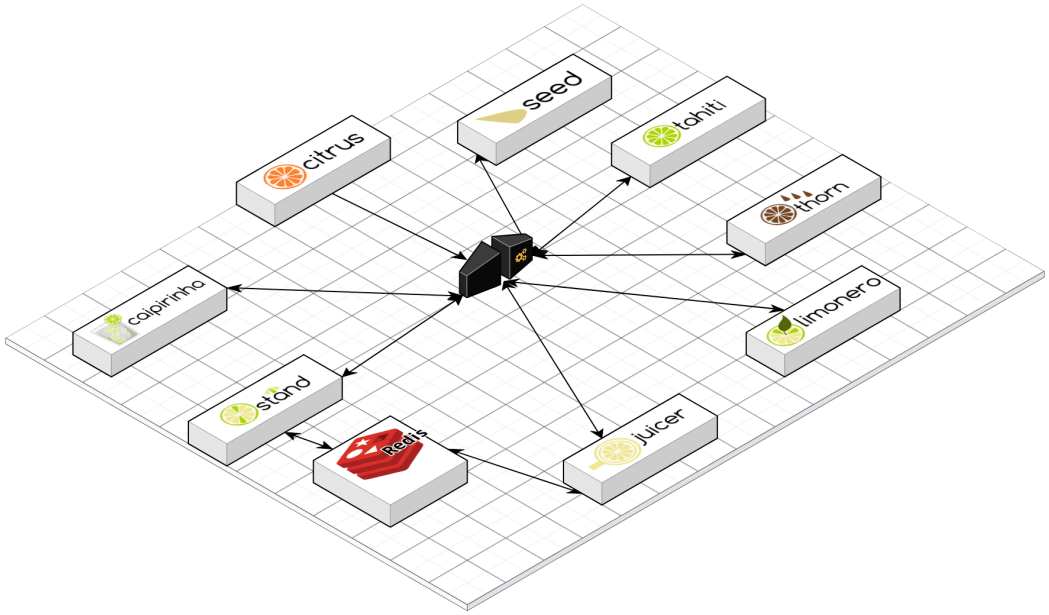

Figura 3. Simulação de arquitetura do projeto Lemonade usando uma API Gateway.

4. Conclusão

O trabalho de conclusão do Projeto Orientado a Computação II tem como objetivo propor melhorias de processos para um projeto com arquitetura de microsserviços. Foi estudado

o Projeto Lemonade, uma plataforma web para mineração e análise de dados desenvolvida dentro do Departamento de Ciência da Computação da Universidade Federal de Minas Gerais, e foram sugeridos diversos pontos de melhoria de processos para o time e de melhorias técnicas para o *software* desenvolvido, com foco principal na aceleração do fluxo de desenvolvimento e de entrega da aplicação e da garantia de qualidade de maneira mais simples.

A partir dos estudos feitos, pode-se perceber uma variedade de problemas com diversos níveis de facilidade de resolução. Além ser gerado um compilado de propostas, foi possível compreender o custo de implementá-las, assim como o tempo necessário para o processo de mudança. Também foi possível avaliar de maneira prática o uso extensivo de conceitos da cultura *DevOps* e de arquiteturas de microsserviços, bem como sua aplicabilidade real.

Os próximos passos a partir deste relatório devem ser a implementação das propostas, de preferência partindo das mudanças mais simples até as mais complexas. Deverá ser realizada, primeiramente, uma discussão com todos os participantes do projeto, para o entendimento geral do propósito, dos passos a serem tomados e de suas priorizações, baseando-se na análise coletiva. Deve-se também definir quais processos serão seguidos pelo time e como serão procedidos. A seguir, a equipe deve se comprometer a executar todos os processos, parte importante da mudança cultural sugerida pelo movimento *DevOps*. Só a partir da mudança de processo e cultura do time, deve-se começar a desenvolver as melhorias propostas para o projeto.

Referências

[Balalaie et al. 2015] Balalaie, A., Heydarnoori, A., and Jamshidi, P. (2015). Migrating to cloud-native architectures using microservices: an experience report. In *European Conference on Service-Oriented and Cloud Computing*, pages 201–215. Springer.

[Chen 2019] Chen, B. (2019). Improving the software logging practices in devops. In *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings*, pages 194–197. IEEE Press.

[Dragoni et al. 2017] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., and Safina, L. (2017). Microservices: yesterday, today, and tomorrow. In *Present and ulterior software engineering*, pages 195–216. Springer.

[Kim et al. 2013] Kim, G., Behr, K., and Spafford, G. (2013). *The phoenix project: a novel about IT, DevOps, and helping your business win*. IT Revolution.

[Lwakatare et al. 2015] Lwakatare, L. E., Kuvaja, P., and Oivo, M. (2015). Dimensions of devops. In *International conference on agile software development*, pages 212–217. Springer.

[Santos et al. 2018] Santos, W. d., Avelar, G. P., Ribeiro, M. H., Guedes, D., and Meira Jr, W. (2018). Scalable and efficient data analytics and mining with lemonade. *Proceedings of the VLDB Endowment*, 11(12):2070–2073.

[Taibi et al. 2018] Taibi, D., Lenarduzzi, V., and Pahl, C. (2018). Continuous architecting with microservices and devops: A systematic mapping study. In *International Conference on Cloud Computing and Services Science*, pages 126–151. Springer.

[Turner et al. 2003] Turner, M., Budgen, D., and Brereton, P. (2003). Turning software into a service. *Computer*, 36(10):38–44.

[Virmani 2015] Virmani, M. (2015). Understanding devops & bridging the gap from continuous integration to continuous delivery. In *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, pages 78–82. IEEE.