UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”
Programa de Pós-Graduação em Ciência da Computação

FABIANA PUPIN MASSON CARAVIERI

SISTEMÁTICA PARA DESENVOLVIMENTO DE MICROSSERVIÇOS A PARTIR DE MODELOS DE PROCESSOS DE NEGÓCIO

Rio Claro – SP
2019

UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”
Programa de Pós-Graduação em Ciência da Computação

FABIANA PUPIN MASSON CARAVIERI

SISTEMÁTICA PARA DESENVOLVIMENTO DE MICROSSERVIÇOS A PARTIR DE MODELOS DE PROCESSOS DE NEGÓCIO

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, área de concentração em Computação Aplicada, linha de Pesquisa Sistemas de Informação, da Universidade Estadual Paulista “Júlio de Mesquita Filho”.

Orientadora: Profa. Dra. Hilda Carvalho de Oliveira

Rio Claro - SP
2019

C262s

Caravieri, Fabiana Pupin Masson
Sistemática para desenvolvimento de microsserviços a partir de modelos de processos de negócio / Fabiana Pupin Masson Caravieri. -- Rio Claro, 2019
130 p. : il., tabs., fotos + 1 CD-ROM

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Geociências e Ciências Exatas, Rio Claro
Orientadora: Hilda Carvalho de Oliveira

1. Computação. 2. Modelagem de Informação. 3. Microsserviços. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Geociências e Ciências Exatas, Rio Claro. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

FABIANA PUPIN MASSON CARAVIERI

SISTEMÁTICA PARA DESENVOLVIMENTO DE MICROSSERVIÇOS A PARTIR DE MODELOS DE PROCESSOS DE NEGÓCIO

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, área de concentração em Computação Aplicada, linha de Pesquisa Sistemas de Informação, da Universidade Estadual Paulista “Júlio de Mesquita Filho”.

Comissão Examinadora

Profa. Dra. Hilda Carvalho de Oliveira
UNESP - Rio Claro
Orientadora

Prof. Dr. Daniel Lucrédio
UFSCar – São Carlos

Prof. Dr. Frank José Affonso
UNESP - Rio Claro

Conceito: Aprovada

Rio Claro - SP

6 de setembro de 2019.

Agradecimentos

Agradeço primeiramente a Deus, ao qual me dá saúde, força e perseverança para superar os entraves ao longo da vida.

Ao meu esposo, Leandro, pelo apoio incondicional, companheirismo, paciência e suporte durante a elaboração deste trabalho e por me proporcionar a vivência da maternidade com nosso filho Arthur (*nosso anjo de luz*).

Aos meus pais, Mercedes e Osmar, minha irmã Viviane, minha sogra Ilza, agradeço o apoio e incentivo em toda a minha vida acadêmica.

Agradeço à minha orientadora, Profa. Dra. Hilda Carvalho de Oliveira, por ter acreditado e investido seu tempo em mim, pelas orientações incentivadoras que servirão para meu crescimento profissional e pessoal. Além do exemplo ímpar de mulher e profissional dedicada.

Ao amigo de Mestrado e de trabalho, Jorge Luís Gregório, pela amizade, brincadeiras, pela construtiva interação durante as disciplinas e outros trabalhos acadêmicos.

Aos colegas de Mestrado Fernando Aparecido Nogueira e Lukas Riehl Figueiredo agradeço a disponibilidade, apoio e discussões construtivas na parte de modelos de processos de negócio.

Meus agradecimentos aos colegas de pesquisa Dhoi de Almeida Cruz, da Unesp, Câmpus de Rio Claro, e Kaique Gazola, da Fatec Jales, pelas interações produtivas relacionadas ao desenvolvimento de microsserviços.

Agradeço à direção e à coordenação da Faculdade de Tecnologia Professor José Camargo (Fatec Jales), pela concessão de afastamento parcial durante o período do Mestrado, bem como à direção da Escola Técnica Estadual “Dr. José Luiz Viana Coutinho”, ambas pertencente ao Centro Paula Souza. Obrigada a todos os colegas de trabalho e colaboradores pelo construtivo e eficiente suporte.

“Tudo parece impossível até estar feito.”

(Nelson Mandela)

RESUMO

Os modelos de processos de negócio têm sido amplamente utilizados por diferentes tipos de organizações para mapear processos de ponta-a-ponta relacionados a produtos ou serviços. A automação desses modelos é normalmente propiciada por sistemas de gerenciamento de processos de negócio, conhecidos como BPMSs (*Business Process Management Systems*), geralmente construídos com arquitetura monolítica e abordagem orientada a serviços (SOA). De modo geral, esses sistemas requerem configurações custosas para automatizar modelos de processos de negócio. Nesse sentido, o principal objetivo deste trabalho é apresentar um processo sistemático para o desenvolvimento de um BPMS dedicado a um modelo específico de processos de negócio "m", desenvolvido integralmente com tecnologias de microsserviços e identificado como "BPMSm". Considerando que uma organização possui “n” modelos de processos de negócio, a integração dos BPMSm´s constitui um BPMS global, identificado como BPMSg. A inovação neste trabalho está no conjunto da proposta apresentada, com destaque à granularidade considerada para a especificação de cada microsserviço: cada microsserviço automatiza uma atividade do modelo de processos de negócio. Todo o trabalho foi contextualizado na abordagem da Engenharia de Software Contínua e DevOps, o que resultou numa solução que permite entregas contínuas, de forma prática, rápida e escalável. O processo sistematizado proposto consiste em três etapas. A validação do modelo de processos de negócio utilizando a notação BPMN v2.0 é feita com base em critérios bem definidos de boas práticas para modelagem gráfica e documentação textual. A especificação dos requisitos para os microsserviços utiliza uma estrutura bem definida e a ferramenta Web “MservSpec” (Especificação de Requisitos para Microsserviços), desenvolvida neste trabalho. A implementação usou a plataforma em nuvem *Google Firebase* e a linguagem Node.js. Três casos foram apresentados para prova de conceito, considerando modelos de processos de negócio com diferentes níveis de complexidade. As análises de desempenho foram feitas com ou sem a utilização de recursos de cache, avaliando o tempo de resposta e o tempo de execução de cada BPMSm. O trabalho também apresenta estudos que abrangem as áreas de modelos de processos de negócio, microsserviços e Engenharia Contínua de Software, essenciais para o desenvolvimento deste trabalho.

Palavras-chave: Modelos de processos de negócio. BPMN. BPMS. Engenharia de Software Contínua. BizDevOps. DevOps. Microsserviços. Computação em nuvem. Google Firebase. Node.js.

ABSTRACT

Business process models have been widely used by different types of organizations to map end-to-end processes related to products or services. The automation of these models is usually provided by Business Process Management Systems (BPMSs), generally built with monolithic architecture and Service Oriented (SOA) approach. Typically, these systems require costly configurations to automate business process models. In this direction, the main objective of this work is to present a systematic process for the development of a BPMS dedicated to a specific model of business processes "m", developed entirely with microservices technology and identified as "BPMSm". Considering that an organization has "n" business process models, the integration of BPMSm´s constitutes a global BPMS, identified as BPMSg. The innovation in this work is in the set of all processes and resources used in the presented proposal, although it can be highlighted the granularity considered for the specification of each microservice: each microservice automates an activity of the business process model. The whole work was contextualized in the approach of Continuous Software Engineering and DevOps, which led to a solution that allows continuous deliveries in a practical, fast and scalable way. The proposed systematized process consists of three stages. The validation of the business process model using a notation BPMN v2.0 is done based on well-defined criteria of good practices for graphic modeling and textual documentation. The specification of the requirements for the microservices uses a well-defined structure and the Web tool "MservSpec" (Requirements Specification for Microservices), developed in this work. The implementation used the Google Firebase cloud platform and the node.js language. Three cases were presented for proof of concept, considering models of business processes with different levels of complexity. Performance analysis were made with or without the use of cache resources, evaluating the response time and the execution time of each BPMSm. The work also presents studies that cover the areas of business process models, microservices and Continuous Software Engineering, essential for the development of this work.

***Keywords**: Business process models. BPMN. BPMS. Continuous Software Engineering. BizDevOps. DevOps. Microservices. Cloud computing. Google Firebase. Node.js.*

LISTA DE FIGURAS

Página

LISTA DE TABELAS

Página

LISTA DE ABREVIATURAS E SIGLAS

ABPMP	*Association of Business Process Management Professionals*
ACM	*Association for Computing Machinery*
ANSI	*American National Standards Institute*
API	*Application Programming Interface*
AWS	*Amazon Web Services*
Baas	*Backend as a Service*
B2B	*Business to Business*
Biz	*Business*
BizDev	*Business and Development*
BizDevOps	*Business, Development and Operations*
BPEL	Business Process Execution Language
BPM	*Business Process Management*
BPMI	*Business Process Management Institute*
BPMN	*Business Process Model and Notation*
BPMS	*Business Process Management Systems*
BPMSg	*Business Process Management Systems global*
BPMSm	*Business Process Management Systems* do modelo
CDM	Critérios para Desenvolvimento de um Microsserviço
CMMN	*Case Management Model and Notation*
CRE	*Continuous Requirement Engineering*
CSS	*Cascading Style Sheets*
CSV	*Comma-Separated Values*
Dev	*Development*
DevOps	*Development and Operations*
DFD	Diagrama de Fluxo de Dados

DNS	*Domain Name System*
DOC	*Document Microsoft Word*
DSL	*Domain Specification Language*
EC2	*Elastic Compute Cloud*
EPC	*Event-driven Process Chain*
ERS	Especificação de Requisitos de Software
ERSa	Especificação de Requisitos de Software de uma atividade
ERSm	Especificação de Requisitos de Software de um modelo
FAQ	*Frequently Asked Questions*
HN	*Heurística de Negócio*
HTTP	*Hypertext Transfer Protocol*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electrical and Electronics Engineers*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
JPEG	*Joint Photographic Experts Group*
JSON	*JavaScript Object Notation*
LPN	Linha de Processos de Negócio
MS	Milissegundos
OMG	*Object Management Group*
OPS	*Operations*
OWL	*Web Ontology Language*
PDF	*Portable Document Format*
PNG	*Portable Network Graphics*
RDF	*Resource Description Framework*
REST	*Representational State Transfer*
SBD	Sistema de Banco de Dados
SBPM	*Semantic Business Process Management*

S-BPM	*Subject-Oriented Business Process Management*
SDK	*Software Development Kit*
SLA	*Service Level Agreement*
SOA	*Service-Oriented Architecture*
SPARQL	*SPARQL Protocol and RDF Query Language.*
SRPD	*Software Requirements from Process Definitions*
SRS	*Software Requirements Specification*
TI	Tecnologia da Informação
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
XMI	*XML Metadata Interchange*
XML	*eXtensible Markup Language*
XPDL	*XML Process Definition Language*
WSBPEL	*Web Services Business Process Execution Language*
YAML	*YAML Ain't Markup Language*

SUMÁRIO

Página

1 INTRODUÇÃO

Modelos de processos de negócio auxiliam na organização, documentação e análise das atividades e eventos, que transformam as matérias primas em serviços e produtos de uma organização, de ponta a ponta. Esses modelos são elaborados segundo a visão estratégica da organização e sua cultura corporativa. As atividades desses modelos podem ser automatizadas, de maneira que todo o processo possa fazer parte da organização de modo orquestrado ou coreografado. Esse contexto faz parte da abordagem de gerenciamento de processos de negócio, conhecida como BPM (*Business Process Management*), que integra as estratégias e objetivos de uma organização com as expectativas e necessidades dos clientes (ABPMP, 2013).

A modelagem dos processos de negócio de uma organização pode ser elaborada por meio de diferentes maneiras, como, por exemplo, notações gráficas, descrições textuais e linguagens. Atualmente, muitas organizações utilizam modelos de processos de negócio na notação BPMN (*Business Process Model and Notation*), devido à sua expressividade, facilidade de uso e especificação bem definida pelo Consórcio OMG (*Object Management Group*). Essa notação possibilita uma ampla descrição visual dos processos de negócio, com possibilidades de documentação textual para cada elemento do modelo, incluindo o conhecimento inerente ao negócio. Apesar do vocabulário de BPMN ter mais de 100 elementos gráficos, Borger (2012) relata que menos de 20% do seu vocabulário é utilizado em mais de 50% dos modelos de processos de negócio. Trata-se de elementos mais comuns na modelagem das atividades, fluxos e eventos, incluindo início e fim.

No mercado, há vários modeladores gratuitos que usam BPMN v2.0 (versão atual), como, por exemplo: *Bizagi Modeler*, *Aris Express*, *BPMN.io*, *Draw.io*, *Yaoqiang BPMN Editor*, *HEFLO!, Modelio*!, *Sydle* e *Bonita BPM* (SGANDERLA, 2017). A portabilidade da parte gráfica tem se mostrado adequada, respeitando-se as especificidades dos sistemas (ex.: cores, espaçamento, etc.). Contudo, como a especificação de BPMN não padroniza os elementos da documentação textual, a portabilidade nessa parte pode ser comprometida na sua forma estrutural de um sistema de modelagem para outro. Nesse contexto, sistemas de gerenciamento de processos de negócio, ou BPMSs (*Business Process Management Systems*), vêm ganhando destaque no cenário de negócios, pois garantem que os processos

estão sendo efetivamente executados como modelados. Normalmente, um BPMS consiste em um conjunto de sistemas que automatizam a gestão de processos de negócio de ponta-a-ponta (modelagem, execução, controle e monitoração), do desenho dos modelos até o monitoramento das atividades em tempo real (SGARDELA, 2018).

Os modelos de processos de negócio, de modo geral, podem ser automatizados por diferentes técnicas, incluindo *Web Services*. Os BPMSs são construídos como uma suíte de programas, comumente projetados com orientação a serviços, na linha de SOA (*Service Oriented Architecture*) e desenvolvidos com arquitetura monolítica. Embora os BPMSs possam constituir uma solução de mapeamento dos processos de ponta-a-ponta para o cenário de negócios, precisam ser configurados de acordo com as necessidades de uma organização, não sendo desenvolvidos para modelos específicos. Atualmente, existem vários BPMSs configuráveis, como, por exemplo, *SoftExpert Excellence Suite*, *Bizagi Automation*, *Orquestra BPMS*, *Sydle BPM*, entre outros. Esses sistemas, embora muito eficientes, são complexos de se configurar, oneram a organização em termos financeiros (valores dispensados com mensalidades e configurações específicas) e de pessoal (treinamento de pessoas dedicadas à configuração e adaptação do sistema) (SGARDELA, 2018).

Por outro lado, é importante observar que modelos de processos de negócio têm sido usados como importante apoio à comunicação entre os profissionais da área de negócios e da área de Tecnologia da Informação (TI). A aproximação entre essas duas áreas tem sido contextualizada na abordagem “BizDevOps” (*Business, Development and Operations*), que incentiva o trabalho conjunto das equipes de negócio, de operações e de desenvolvimento de software. A equipe de negócios define os requisitos e trabalha diretamente com os desenvolvedores para definirem as prioridades para as entregas de software. Essa integração das equipes contribui para entregas mais rápidas das soluções de software, atendendo melhor às demandas dos usuários, com custos menores (FORBRIG, 2018b). A relação entre os desenvolvedores e a equipe de operações, conhecida como “DevOps” (*Development and Operations*), auxilia o gerenciamento do lançamento de novas versões, visando entregas de modo ágil e contínuo (MEDRADO, 2015).

A abordagem BizDevOps faz parte do segmento recente da Engenharia de Software chamado “Engenharia de Software Contínua”, com princípios associados às metodologias ágeis, entregas contínuas e desenvolvimento “enxuto” (*Lean development*) (FRANÇA *et al.*, 2017). A ideia é que todo o ciclo de vida do software possa ser um processo ininterrupto, desde a fase de planejamento até a entrega, garantindo a continuidade da vida do software, impulsionada pela busca incessante por inovação (BOSCH, 2014). Assim, a Engenharia de Software Contínua preza pela agilidade e desenvolvimento contínuo dos produtos, com atenção constante à melhoria da qualidade desses produtos.

Nesse sentido, em vez de sistemas de automação de processos de negócio monolíticos, sistemas compostos por microsserviços tendem a promover uma certa continuidade do ciclo de vida do software. Além disso, Fowler (2017) ressalta a facilidade de manutenção dos códigos dos microsserviços e sua alta escalabilidade, colaborando com o aumento da produtividade no desenvolvimento do software, mesmo com a adoção de novas tecnologias. De certa forma, os microsserviços facilitam o gerenciamento de aplicações de maneira a minimizar as dificuldades de controle dos sistemas, embora requeiram tempo significativo de planejamento (O´CONNOR; ELGER; CLARKE, 2017). Os microsserviços podem ser desenvolvidos separadamente uns dos outros, independentemente de tecnologias, e integrados de maneira ágil (BOSCH, 2014).

Nesse contexto, os objetivos deste trabalho estão apresentados na *Seção 1.1*, desenvolvido com a metodologia apresentada na *Seção 1.2.* A organização dos demais capítulos, por sua vez, encontra-se na *Seção 1.3*.

1.1 OBJETIVOS DO TRABALHO

O principal objetivo deste trabalho é apresentar um processo sistematizado para o desenvolvimento de um BPMS dedicado a um determinado modelo de processos de negócio em BPMN v2.0, desenvolvido totalmente com a tecnologia de microsserviços. Esse sistema dedicado é identificado como BPMSm, onde “m” é a identificação do modelo de processos de negócio. Considerando que uma organização possui *n* modelos de processos de negócio, a integração dos BPMSm´s constituem um BPMS global, identificado como BPMSg. Contudo, neste trabalho, o objetivo é orientar, de modo sistemático, a implementação de um BPMSm.

A implementação de microsserviços requer pequena granularidade de funcionalidades e alta independência de recursos utilizados entre eles, como linguagens de programação, sistemas de bancos de dados, etc. (O’CONNOR *et al.*, 2017). Na prática, isso ainda tem sido discutido e implementado de diferentes modos, alguns não muito aderentes aos aspectos conceituais de microsserviços.

Assim, o que se pretende neste trabalho, é apresentar um conjunto de procedimentos e critérios que orientem o desenvolvimento de microsserviços relacionados a um determinado modelo de processos de negócio. Na abordagem apresentada, cada atividade do modelo “m” é implementada como um microsserviço do BPMSm. Esses microsserviços podem utilizar diferentes recursos, em nuvem ou não.

Com a ideia apresentada neste trabalho, objetiva-se, então, utilizar as técnicas

empregadas no desenvolvimento ágil com a abordagem DevOps para o desenvolvimento e execução de microsserviços. Desse modo, a automação dos modelos de processos de negócio pode ser implementada em menor tempo, com entrega em ciclos curtos e com custos mais acessíveis.

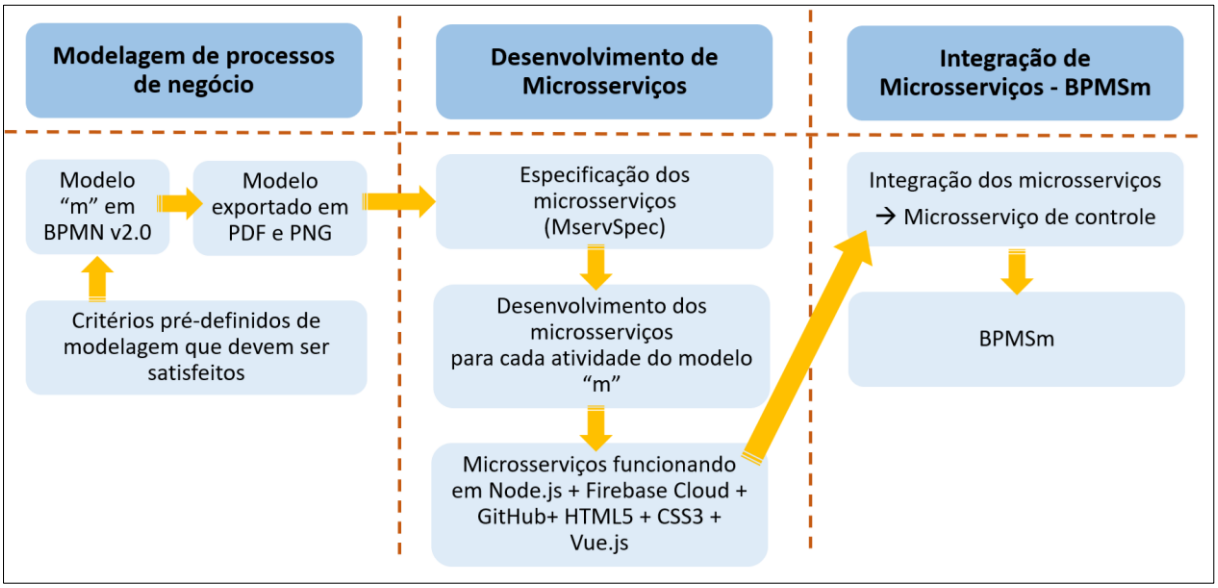
Neste trabalho, é apresentada uma ferramenta Web, MservSpec (*Requirements Specification for Microservices*), que recebe as informações de um modelo de processos de negócio em BPMN v2.0 e gera a especificação de cada microsserviço, com dados relevantes ao processo como um todo e à cada atividade. Essa especificação é baseada na norma ISO/IEC/IEEE 29148:2018, bem como nas recomendações de Fowler (2017) e na documentação para aplicações Web oferecida junto com a suíte de ferramentas *Swagger*. Essas ferramentas auxiliam o desenvolvedor de software na modelagem, documentação e geração de código para APIs (*Application Programming Interfaces*) do estilo REST (*Representational State Transfer*), geralmente utilizada na implementação de *Web Services*.

A partir dessa especificação gerada, este trabalho apresenta procedimentos para implementação, testes e integração dos microsserviços, de modo a compor o BPMSm voltado ao modelo de processos de negócio considerado.

A avaliação do processo proposto neste trabalho tem, como princípio, a aplicação das práticas que envolvem a abordagem DevOps. Essa avaliação consiste em analisar se os modelos de processos de negócio selecionados possuem maior aderência à implementação de um BPMSm. A avaliação também verifica se a documentação desses modelos atende os critérios estabelecidos neste trabalho. No entanto, é importante ressaltar o tempo de desenvolvimento, juntamente com a realização de testes de desempenho (tempo de resposta e tempo de execução), utilizando o recurso de *cache* para o desenvolvimento de cada microsserviço, assim como a sua integração e o uso de boas práticas para aplicações Web. Na **Figura 1**, é ilustrado o esquema do processo sistematizado.

Neste trabalho, para a programação dos microsserviços foi adotado o interpretador *JavaScript Node.js*, no ambiente em nuvem da plataforma *Google Firebase,* no contexto da abordagem DevOps. Observa-se que essa plataforma oferece suporte à programação com diversas linguagens, como, por exemplo: *C++*, *Java*, *Javascript*, *Objective-C* e *Swift*. Por meio do serviço denominado *Cloud Functions*, a plataforma *Firebase* consegue tratar as funcionalidades da aplicação de maneira isolada como um contêiner. Desse modo, as funções podem ser executadas em servidores virtuais distintos, de forma semelhante a usada em tecnologias de orquestração de contêineres (ex.: tecnologias *Docker Swarm* e *Kubernetes*).

Figura 1 – Processo sistematizado para o desenvolvimento de um BPMSm com microsserviços.

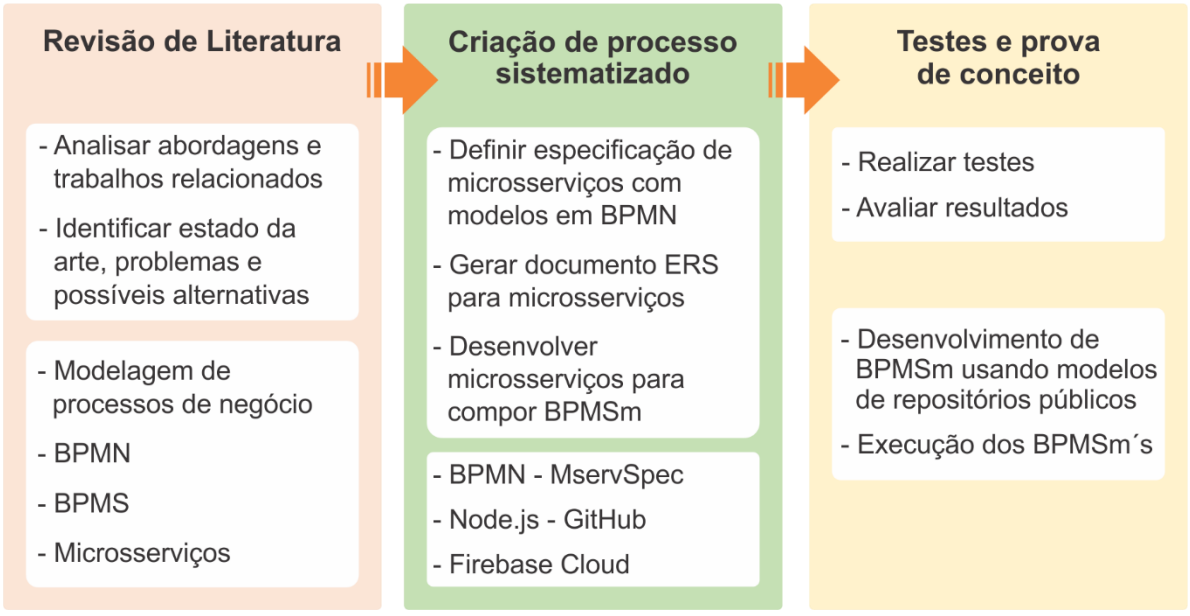


Fonte: elaborado pelo autor.

1.2 METODOLOGIA DE TRABALHO

Uma visão geral das atividades desenvolvidas neste trabalho, para se atingir os objetivos apresentados na *Seção 1.1*, é ilustrada na **Figura 2**.

Figura 2 – Visão geral das atividades desenvolvidas.



Fonte: elaborado pelo autor.

Inicialmente, foi realizado um levantamento bibliográfico por meio de palavras-chave sobre os assuntos: modelos de processos de negócio, BPMN, BPMS, Engenharia de Software Contínua e microsserviços. Essa pesquisa foi realizada em bases científicas, como: *IEEE Xplore Digital Library*, *ACM Digital Library* e *Google Scholar*. Além dos artigos selecionados, a revisão bibliográfica também envolveu livros voltados aos temas abordados.

Em relação aos modelos de processos de negócio em BPMN, destacam-se os experimentos realizados com alguns sistemas de modelagem, como: *Bizagi Modeler*, *BPMN.io* e *Bonita BPM*. Os experimentos incluíram comparações na estrutura da documentação dos modelos, bem como no processo de exportação e importação dos arquivos no formato PDF, PNG e BPMN. A escolha da ferramenta de modelagem *Bizagi Modeler* para este trabalho levou em consideração aspectos como: suporte a BPMN v2.0, exportação para formatos de imagens em PNG e BPMN, estrutura da documentação com atributos estendidos, bem como geração de documentação em formatos PDF e DOC.

No contexto da Engenharia de Software Contínua, foi identificada uma crescente adoção da abordagem DevOps, oriunda de BizDevOps, que proporciona integração entre as equipes de negócio, operações e desenvolvimento, priorizando entregas rápidas e contínuas. Não foram encontrados trabalhos que especificam e desenvolvem microsserviços a partir de modelos de processos de negócio em BPMN. No entanto, foram encontrados trabalhos nessa direção usando a notação S-BPM (*Subject-Oriented Business Process*). Os trabalhos encontrados mais relacionados ao contexto deste trabalho, usando a notação BPMN, envolviam aplicações baseadas em SOA.

Alguns trabalhos estudados mostraram como modelos de processos de negócio em BPMN podem apoiar a especificação de requisitos de software, como é o caso do sistema SPRD (*Software Requirements from Process Definitions*), desenvolvido por Nogueira (2017b). Esse sistema gera um documento de especificação de requisitos de software, incluindo diagramas em UML (*Unified Modeling Language*), a partir de um modelo de processos de negócio em BPMN v2.0. O estudo desse sistema apoiou o processo de extração de requisitos de software neste trabalho. Também ajudou a verificar se as informações contidas nos modelos de processos de negócio podiam, de fato, oferecer informações relevantes para a especificação e desenvolvimento de microsserviços. Contudo, os experimentos com o sistema SRPD mostraram que o documento de requisitos gerado não contemplava as especificações necessárias e adequadas para o desenvolvimento de microsserviços.

Para compreender melhor os aspectos de implementação, funcionamento e gerenciamento dos microsserviços, foram investigadas as possíveis tecnologias que estão sendo adotadas. Uma dessas tecnologias consiste em conteinerização usando a ferramenta

Docker. Com essa ferramenta foi possível desenvolver uma aplicação com múltiplos serviços, onde cada contêiner pode ser tratado como um microsserviço. Também foi estudado o uso de interfaces de programação de aplicações, ou APIs (*Application Programming Interface*), do estilo arquitetural REST (*Representational State Transfer*).

A seleção da plataforma em nuvem para desenvolvimento dos microsserviços envolveu estudos e experimentos usando as plataformas *AWS*, *Google Cloud*, *Microsoft Azure* e *Google Firebase*. Essa última plataforma foi selecionada devido ao modo mais simples e intuitivo de programação de microsserviços em relação às demais.

1.3 ORGANIZAÇÃO DOS CAPÍTULOS

Considerando os objetivos apresentados na *Seção 1.1*, o restante do texto desta dissertação está organizado em mais sete capítulos.

O *Capítulo 2* apresenta aspectos da modelagem de processos de negócio relevantes ao trabalho proposto. O capítulo aborda a notação BPMN, incluindo técnicas de apoio a processos da Engenharia de Requisitos. Informações sobre os elementos gráficos da notação BPMN são apresentadas no ANEXO A. O capítulo também aborda o funcionamento e implementação de BPMSs.

No *Capítulo 3*, são abordados os processos da Engenharia de Requisitos no contexto da Engenharia de Software Contínua, com ênfase na especificação de requisitos. Conceitos da abordagem BizDevOps com foco na metodologia DevOps e tecnologias relacionadas também são apresentados nesse capítulo.

O *Capítulo 4* apresenta aspectos da arquitetura de microsserviços, incluindo formas de elaborar a documentação pertinente, bem como boas práticas e tecnologias de desenvolvimento.

O processo sistematizado para desenvolvimento dos microsserviços usando a plataforma *Firebase* para compor um BPMSm é apresentado no *Capítulo 5*. Esse capítulo também inclui a ferramenta Web “MservSpec” (*Microservice Requirements Specification*), que gera o documento de especificação de requisitos de cada microsserviço, definido neste trabalho.

O *Capítulo 6*, por sua vez, apresenta os testes e a prova de conceito para avaliação do processo apresentado no *Capítulo 5*. Por fim, no *Capítulo 7* são apresentadas as considerações finais e os trabalhos futuros, visando melhorias e extensões deste trabalho.

2 VISÃO GERAL DE MODELOS DE PROCESSOS DE NEGÓCIO

O gerenciamento de processos de negócio envolve modelagem, implementação, monitoramento e controle desses processos, a fim de obter melhorias contínuas (BPMP, 2013). A modelagem de processos de negócio é considerada uma etapa fundamental, envolvendo um conjunto de habilidades e técnicas, que permitem conhecer detalhadamente o gerenciamento dos componentes que fazem parte dos processos. Essa etapa pode ser desenvolvida de diferentes maneiras, podendo-se utilizar notações gráficas, descrições textuais e diferentes linguagens (ABPMP, 2013). A seguir, são apresentados três importantes conceitos envolvidos na modelagem de processos de negócio, segundo a associação de profissionais de gerenciamento de processos de negócio, conhecida como ABPMP (*Association of Business Process Management Professionals*):

a) **Modelos de processos** - representam alguma atividade de negócio, servindo de comunicação entre os diferentes aspectos que os contemplam, resultando em uma documentação que auxilia o alinhamento de requisitos;

b) **Perspectivas** - ajudam a identificar os diferentes escopos e níveis de um mesmo processo para visões distintas, como, por exemplo, para a área de negócios ou de operações;

c) **Notação** - auxiliam na modelagem com a utilização de formas específicas, de acordo com o objetivo e tarefas a serem executadas, podendo combinar mais de uma notação na modelagem.

Os sistemas para automatização e gestão de processos de negócio, conhecidos como BPMSs (*Business Process Management Systems*), permitem a modelagem, execução, controle e monitoração de modelos de processos de negócio (ABPMP, 2013). Geralmente esses sistemas possuem um ambiente de integração que permite, além da modelagem de processos, a definição de fluxos de execução, regras de negócio, eventos e demais especificações necessárias à operação e gerenciamento dos processos de negócio. Essa tecnologia procura uma maior interação entre as atividades humanas e as ações do sistema (VAN DER AALST *et al.*, 2010).

Segundo ABPMP (2013), um BPMS pode simular a execução de processos, além de permitir a criação de aplicações aderentes ao domínio. O sistema *Bizagi Studio* (*www.Bizagi.com*) é um exemplo de BPMS que suporta a criação e distribuição de aplicativos. Atualmente, a notação BPMN (*Business Process Management Notation*) tem sido muito utilizada para modelagem em grande parte dos BPMSs.

Uma tendência que tem ficado cada vez mais evidente nos BPMSs é a migração para serviços em nuvem, disponibilizados por meio da tecnologia *SaaS* (*Software as Service*). Os BPMSs buscam estimular os participantes a contribuírem mais ativamente na elaboração dos modelos de processo e consequentemente no fluxo de trabalho. Esses sistemas possuem telas onde as tarefas são executadas em um processo automatizado (SGARDELA, 2018).

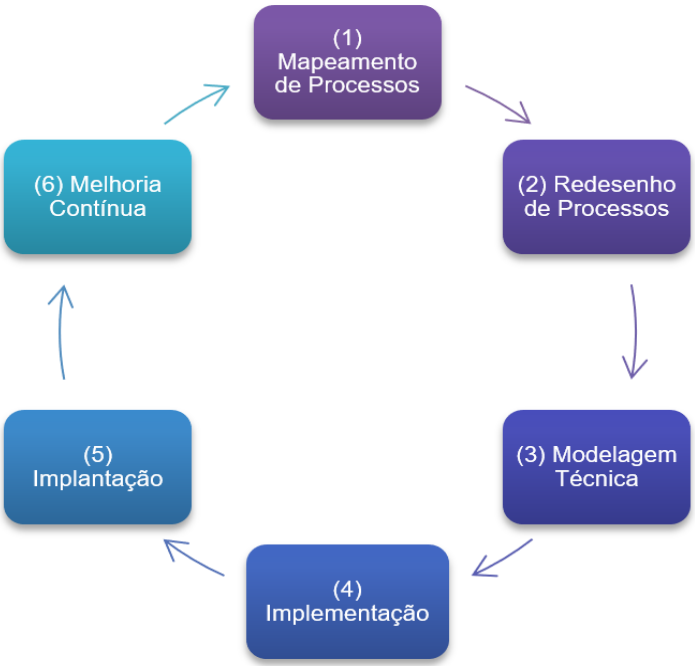
Nesse contexto, a *Seção 2.1* apresenta aspectos referentes ao ciclo de vida do gerenciamento de processos de negócio, ou BPM (*Business Process Management*). *A Seção 2.2*, por sua vez, traz uma visão da notação gráfica BPMN, incluindo aspectos de documentação textual e possibilidades de conversão dos modelos para outras tecnologias, visando a integração com outros sistemas. Na *Seção 2.*3, são apresentados conceitos e processos relacionados a BPMSs, enquanto a *Seção 2.4* apresenta alguns trabalhos voltados à Engenharia de Requisitos apoiada por modelos de processos de negócio. Por fim, a *Seção 2.5* apresenta as considerações finais do capítulo.

2.1 CICLO DE VIDA DOS PROCESSOS NA ABORDAGEM BPM

O ciclo de vida dos processos de negócio, automatizados ou não, visam obter resultados alinhados aos objetivos estratégicos de uma organização (ABPMP, 2013). Na abordagem BPM, o ciclo de vida é definido em seis etapas, conforme ilustrado na **Figura 3**: (1) identificação e mapeamento de processos; (2) redesenho dos processos; 3) modelagem técnica; 4) implementação; 5) implantação; 6) melhoria contínua.

As três primeiras etapas são relativas à identificação de atividades, sendo inicialmente mapeados os processos atuais da organização (modelo "*AS IS*") e depois, sob perspectivas de melhorias (modelo "*TO BE*"), são gerados modelos técnicos para serem implementados (modelos "*TO DO*"). As atividades passam por refinamentos, reavaliando-se questões de negócio, melhorias culturais/organizacionais, eficiência na execução e otimização das tarefas. A intenção é eliminar as interferências humanas e adaptar os processos já automatizados (OMG, 2013).

Figura 3 – Ciclo de vida de BPM.



Fonte: adaptado de Van der Aalst (2013).

As etapas de “implementação” e “implantação” se referem à execução dos processos de negócio. Na etapa de “implementação” se tem o modelo “*TO RUN*”, que utiliza elementos como: controles de transação, compensação e tratamentos de erros, usando suítes de sistemas de BPM. Na etapa de “implantação”, são realizadas as mudanças tratadas nas outras etapas com uma análise de monitoramento dessas mudanças. Na última etapa do ciclo, são realizadas melhorias contínuas no processo de negócio, por meio de metodologias ou técnicas de gestão e aferimento de desempenho desses processos (OMG, 2013).

Neste trabalho, os modelos de processos de negócio podem ser do tipo “AS IS”, “TO BE” ou “TO DO”, desde que sejam modelados usando a notação BPMN v2.0, sendo que o modelo “TO DO” se torna mais adequado para melhores resultados.

2.2 NOTAÇÃO BPMN

Nesta seção, é apresentada uma visão geral da notação BPMN, destacando aspectos relevantes para que modelos de processos de negócio apoiem as fases de elicitação e análise da Engenharia de Requisitos.

A notação BPMN foi desenvolvida pelo grupo BPMI (*Business Process Model Iniciative*) em 2004. A partir de 2006, o consórcio OMG (*Object Management Group*)

padronizou-a, lançando a versão 1.1 em janeiro de 2008 (atualmente encontra-se na versão 2.0). Em 2013, BPMN foi definida como um padrão internacional de modelagem de processos de negócio pela organização ISO (*International Organization for Standardization*), por meio da norma ISO/IEC 19510, idêntica ao padrão OMG BPMN 2.0 (OMG, 2013; ABPMP, 2013).

A notação BPMN tornou-se muito popular no ambiente de negócios pelo fato de responder à demanda de modelagem de processos e de ser apoiada, desde o início, por várias empresas de renome mundial nesse segmento. BPMN pode ser considerada uma notação de orientação a negócios, apresentando recursos para interagir com diferentes linguagens, como: BPEL (*Business Process Execution Language*), XPDL (*Process Definition Language*), e WSBPEL (*Web Services Business Process Execution Language*) (OMG, 2013).

A padronização propiciada pela notação BPMN nos modelos de processos de negócio possibilita diferentes pontos de vista aos participantes do processo, provendo um meio simples de comunicação entre usuários, clientes e fornecedores para a implementação do processo (OMG, 2013). BPMN possui símbolos gráficos de fácil entendimento, com um conjunto de regras baseadas na técnica de fluxogramas, favorecendo a administração e monitoramento dos processos (OMG, 2013). Sua estrutura permite troca de mensagens entre os participantes de uma atividade, além da informação necessária para analisar, simular e implementar um processo de negócio. A semântica de BPMN é bem definida, possibilitando a compreensão por todos os envolvidos, dentro e fora da organização (OMG, 2013).

A estrutura de BPMN permite modelar a troca de mensagens entre os participantes de uma atividade, além de toda informação necessária para analisar, simular e implementar um processo de negócio. A notação tem os seguintes benefícios: pode ser compartilhada e compreendida por todos, possui semânticas ricas e precisas, possui padronização dos processos dentro e fora da organização (*BIZAGI*, 2016).

A notação BPMN é considerada um padrão internacional que representa as melhores práticas para a comunidade de modelagem de negócios, pois define uma notação semântica com diagramas de colaboração, processos e coreografias (OMG, 2013). BPMN tornou-se muito popular no ambiente de negócios, pelo fato de responder à demanda de modelagem de processos, e ser apoiada desde o início por várias empresas de renome mundial nesse segmento. No entanto, outras notações também podem ser utilizadas para modelar processos de negócio, como por exemplo: Diagrama de Atividades em UML (*Unified Modeling Language*), fluxograma, cadeias *EPCs (Event-Process Chains*), diagrama de fluxo de dados (DFD), (BALDAN *et al.*, 2014), etc.

A *Subseção 2.2.1* apresenta o vocabulário gráfico de BPMN, complementado pelo guia de referência para modelar processos, proposto por Bitencourt (2017), apresentado no

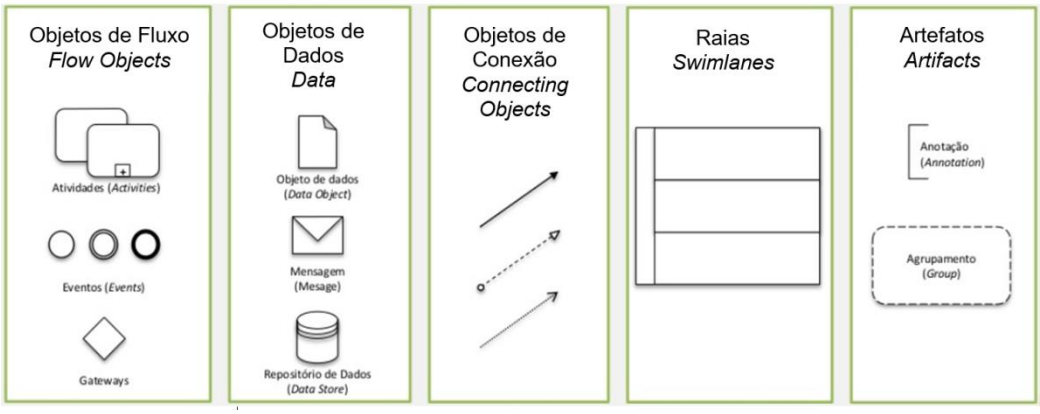
ANEXO A. A *Subseção 2.2.2*, por sua vez, aborda a documentação textual associada aos modelos gráficos.

2.2.1 Vocabulário gráfico de BPMN

A notação BPMN v2.0 possui 132 elementos gráficos, categorizados em cinco classes, exemplificadas na **Figura 4** (OMG,2013):

1. **Objetos de Fluxo** (*Flow* Objects) – classe que contempla os principais elementos gráficos que definem o comportamento dos processos de negócio, classificados como eventos, atividades ou decisões;
2. **Objetos de Dados** (*Data*) – classe composta por objetos de dados, mensagens e repositório de dados;
3. **Objetos de Conexão** (*Connecting Objects)* – classe composta pelos objetos que conectam outros objetos; podem ser do tipo fluxo de sequência, fluxo de mensagens e associação;
4. **Raias** (*Swimlanes*) – classes que representam a separação visual de cada atividade e contêm os objetos de fluxos pertencentes a cada uma. Podem ser do tipo "*pool"* ou "*lane"*.
5. **Artefatos** (*Artifacts*) – classes usadas para colocar informações adicionais no processo, podem representar as entradas ou saídas de uma atividade. Os artefatos podem ser do tipo objetos de dados, grupo ou associações.

Figura 4 – Classes de elementos de BPMN com alguns exemplos.

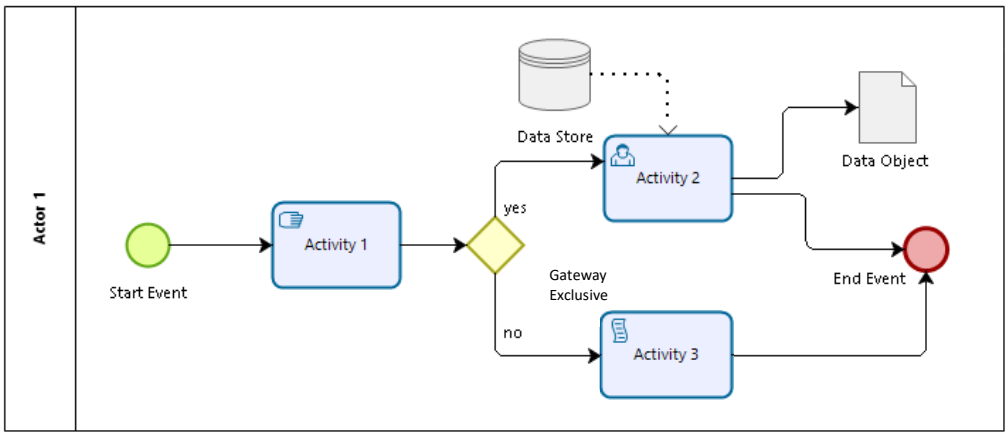


Fonte: iProcess (2013).

Segundo Borger (2012), menos de 20% do vocabulário de BMN são utilizados em mais de 50% dos modelos de processos de negócio. Os elementos mais utilizados são: atividades, subprocessos, eventos, *gateways* (exclusivos, inclusivos e paralelos), artefatos (grupos e anotações), objetos de dados, depósitos de dados, atributos estendidos, piscinas (*pools*), raias (*lanes*), fluxos de sequência e fluxos de mensagem.

Na **Figura 5**, é ilustrado um exemplo genérico simples de um modelo em BPMN v2.0, com uma raia, que identifica um ator (*Actor 1*), e diferentes tipos de atividades (retângulos com bordas arredondadas): "*manual*" (atividade realizada manualmente), "*user*" (realizada por um usuário e auxiliada por um sistema), e "*script*" (executada por um mecanismo que interpreta a linguagem gerada pelo modelador). Os eventos de início e fim são representados por círculos no início e no fim do processo. Os "*gateways*" (losangos) são do tipo "*exclusivo*", ou seja, apenas um dos caminhos é executado após uma decisão resultante da atividade precedente. Os artefatos mostrados são do tipo "Data Store" e "*Data Object*", que representam os dados (persistentes ou não) que são usados na execução do processo.

Figura 5 – Exemplo simples de modelo de processos de negócio na notação BPMN.



Fonte: elaborado pelo autor.

Segundo o consórcio OMG (2013), a especificação de BPMN possibilita a construção de três tipos de diagramas:

a) **Processo** (tradicional) - define a relação entre as atividades, eventos e todos os elementos de apoio ao processo;

b) **Colaboração e Conversação** - representa a comunicação entre as entidades envolvidas no processo (representadas por "*pools*"), onde as mensagens podem ser agrupadas por assunto;

c) **Coreografia** - responsável pela troca de informações (sequência ordenada de

mensagens) entre processos do tipo B2B (*Business to Business*) dentro e fora da organização (agentes externos).

A riqueza de detalhes que se pode expressar na modelagem de processos com a notação BPMN favorece maior aderência aos cenários do mundo real. Em muitos casos, o diagrama de processo já pode ser considerado suficiente, para expressar detalhes (alto grau de refinamento) das informações e interações do processo. Contudo, Kossak *et al.* (2016) apontam que a integração do usuário com a modelagem de dados em BPMN é falha, pois o suporte à modelagem organizacional é restrito. Segundo os autores, mesmo BPMN fornecendo suporte de qualidade aos fluxos do processo, não acontece o mesmo em relação aos atores que modelam os processos. Adicionalmente, mencionam que os recursos de comunicação nem sempre são suficientes na notação. De fato, há outras notações, como, por exemplo, S-BPM (*Subject-Oriented Business Process*), que enfatiza o ponto de vista da comunicação entre os atores, porém não é definida como um padrão internacional (FLEISCHMANN et al., 2015).

2.2.2 Documentação nos modelos com BPMN

Como apresentado na seção anterior, a notação BPMN apresenta um conjunto robusto de símbolos para modelagem de processos de negócio que, à medida que o processo avança nas etapas do ciclo de BPM, a documentação do processo vai sendo construída (ABPMP, 2013).

No entanto, não existe um padrão a ser seguido com relação à documentação textual nos modelos de processos de negócio em BPMN, segundo sua especificação. Sendo assim, cada ferramenta de modelagem define um modo de fazer a documentação associada a cada elemento gráfico de BPM ou a um conjunto deles. Logo, as equipes de negócio devem utilizar os recursos da ferramenta de modelagem adotada. Contudo, se os modelos forem exportados para outras ferramentas de modelagem, a documentação poderá sofrer modificações na sua forma estrutural.

De todo modo, os sistemas de modelagem conseguem armazenar a descrição semântica dos elementos de BPMN e seus atributos em linguagem natural (FIGUEIREDO, 2018). Isso permite documentar informações relevantes para o processo de levantamento de requisitos do software, pois ajudam a compreender o domínio do negócio em ambientes organizacionais.

Para que a documentação gerada pelo modelador possa apoiar o desenvolvimento de microsserviços a partir de modelos de processos de negócio, é necessário que ela contenha uma descrição detalhada de cada elemento que compõe o modelo, além da descrição das atividades e fluxos de mensagens.

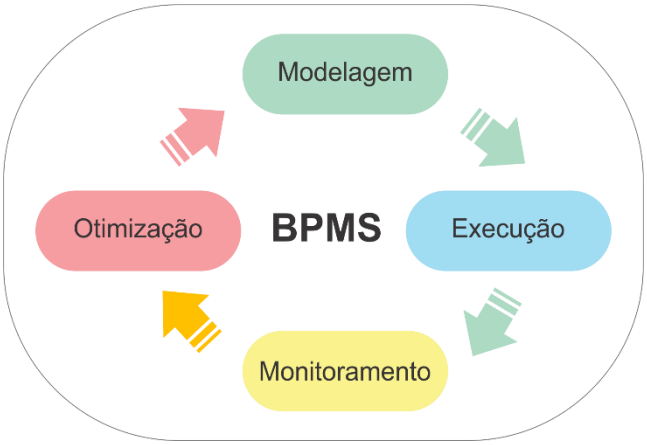
Os seguintes sistemas de modelagem de processos de negócio com BPMN v2.0 foram avaliados neste trabalho: *Bizagi Modeler, Heflo!*, *BPMN.io* e *Bonita Studio*. A intenção foi avaliar o mecanismo de documentação textual provido por cada sistema, considerando a quantidade e a qualidade das informações, uma vez que esse recurso é fundamental para a especificação e desenvolvimento de microsserviços. Os resultados da avaliação estão apresentados no APÊNDICE A.

Geralmente, os sistemas de modelagem de processos de negócio que usam BPMN oferecem recursos para documentação textual da parte gráfica. A documentação pode ser visualizada em diferentes formatos: imagens (ex.: PNG ou JPEG); textos (ex.: arquivos PDF ou DOC); arquivos de metadados, que são utilizados para troca de informações entre as ferramentas de modelagem (ex.: extensões CSV, XML, XMI ou XPDL). Entre os sistemas analisados, o *Bizagi Modeler* apresenta um conjunto maior de recursos de exportação e publicação dos modelos. Mesmo utilizando somente os recursos gratuitos desse sistema, é possível obter uma documentação com maior quantidade de elementos, com descrição mais detalhada. Isso é importante, pois facilita o entendimento do modelo de processos de negócio na elaboração dos microsserviços, motivo pelo qual a ferramenta de modelagem *Bizagi Modeler* foi selecionada para este trabalho.

2.3 SISTEMA DE GERENCIAMENTO DE PROCESSOS DE NEGÓCIOS (BPMS)

Um sistema de gerenciamento de processos de negócio, conhecido como BPMS (*Business Process Management Suite ou System*), consiste de um conjunto de sistemas que automatiza, controla e gerencia os modelos de processos de negócio de uma organização. Esses sistemas também têm recursos de modelagem dos processos e fluxos de trabalho, com mecanismos de definição de regras que permitem a simulação de operações de negócio. As regras de negócio são definidas e adicionadas ao modelo gráfico, fornecendo a lógica para as aplicações que esse sistema gera (ABPMP, 2013). Desse modo, conseguem automatizar os processos, acompanhar e monitorar seu desempenho, além de controlar as atividades que o compõe (ABPMP, 2013). Uma visão geral das funcionalidades de um BPMS pode ser vista na **Figura 6**.

Figura 6 – Visão geral das funções de um BPMS.



Fonte: elaborado pelo autor.

Para um funcionamento adequado de um BPMS, é necessário que todos os módulos sejam instalados e configurados corretamente, propiciando flexibilidade e velocidade de ajustes a mudanças, quando necessário (ABPMP, 2013). Um BPMS deve permitir gerenciar o fluxo dos processos, incluindo acompanhamento, balanceamento da carga de trabalho, identificação de erros, gerenciamento de desempenho, etc.

Observa-se que quando um ator do processo inicia sua atividade de trabalho e faz acesso (*login*) em uma aplicação, o BPMS entra no estado de "*runtime*", quando modelos e regras são executados (ABPMP, 2013). Para Sganderla (2012), quando um processo de negócio é implementado utilizando um BPMS, o modelo "*TO DO*" do processo requer um detalhamento técnico mais apurado, possibilitando que o sistema possa interpretar essas informações no motor do processo, implementando o modelo "*TO RUN*". Esse modelo utiliza um número maior de elementos de BPMN, sem que os usuários percebam que estão sendo utilizados. Tais elementos são necessários para fornecer, ao sistema, detalhes relevantes para execução automatizada do processo, como, por exemplo, controles de transação, compensação e tratamentos de erros.

Os elementos de BPMN que representam os dados do modelo são utilizados pelo BPMS para associação de regras. Dessa maneira, o código de programação de um BPMS tem espaços em branco que são automaticamente preenchidos pelo sistema com as regras associadas a esses modelos de negócio (ABPMP, 2013). Também conseguem associar formulários às tarefas, gerar relatórios e oferecer uma interface amigável.

Segundo ABPMP (2013), os BPMSs possuem duas categorias de ferramentas: as autônomas e as de grupos integrados. As ferramentas autônomas permitem a análise de regras de negócio e definição dos processos e fluxos de trabalho de uma organização, possibilitando a descoberta de inconsistências e conflitos. No entanto, as ferramentas

autônomas não permitem a mudança de novas operações de negócio nos modelos utilizados. Os grupos integrados de ferramentas são capazes de fornecer suporte a lógicas complexas e transações de alto volume de operações. Esses grupos integrados são ambientes de operação de negócios com capacidade para simulações complexas, permitindo que as organizações analisem possíveis alternativas e selecionem as melhores para um modelo de processos de negócio ideal. A seleção do grupo de ferramentas de um BPMS deve ser feita com base no melhor atendimento às regras de negócio específicas de cada organização. A *Subseção 2.3.1* apresenta o funcionamento e a implementação de um BPMS.

2.3.1 Funcionamento e implementação de um BPMS

Os modelos de processos de negócio devem ser modelados seguindo as diretrizes de modelagem do BPMS. Alguns desses sistemas possuem suporte a um conjunto de símbolos de BPMN, com pouca flexibilidade, e a tipos de formulários de captura de dados. Segundo ABPMP (2013), no processo de automação em um BPMS, cada atividade do processo se transforma em uma pequena aplicação, que é disponibilizada ao ator responsável pela sua execução. As informações necessárias para realização de uma atividade e suas regras de negócio são enviadas ao ator do processo. O fluxo do processo controla a sequência de atividades a serem realizadas. O sistema BPMS controla a interação humana por meio de formulários, que são construídos de acordo com cada atividade e suas regras de negócio, respectivamente, fornecendo, ao ator do processo, opções para concluir a atividade.

A aplicação gerada por um BPMS apresenta várias telas sequenciais, de acordo com o fluxo do processo e suas atividades (ABPMP, 2013). Logo que uma atividade é concluída, as informações são repassadas às atividades seguintes, e, desse modo, o fluxo do processo é completado chegando ao seu término (ABPMP, 2013).

Um BPMS pode ser implementado com diferentes tecnologias, como, por exemplo: SaaS (*Software as a Service*), computação em nuvem e redes sociais. Quando se implementa um BPMS utilizando a tecnologia de software como serviços (*SaaS*), a conexão dos clientes é feita independentemente de sua localização. Isso é possível devido a organização estar em um local e seus recursos de hardware, software, aplicações e ferramentas estarem em outro. O meio de comunicação entre as partes é feito pela internet. Esse tipo de implementação apresenta vantagens de baixo custo, bem como facilidade de acesso colaborativo à ferramenta e aos dados do modelo pelas equipes de modelagem, em qualquer lugar do mundo e a qualquer momento (ABPMP, 2013).

Quando a implementação de um BPMS utiliza a tecnologia de computação em nuvem, uma estrutura de conectividade deve ser bem estabelecida, de maneira a prover adaptações a novos protocolos de conexão mais rapidamente (SGANDERLA, 2018). De modo geral, as interfaces de um BPMS em nuvem podem ser atrativas aos usuários do processo. As operações de um BPMS em nuvem podem ser executadas tanto localmente em servidores internos à organização como no modo de redes de serviço baseadas em nuvem. Assim, essa tecnologia proporciona uma arquitetura do BPMS que provê escalabilidade, flexibilidade, custo relativamente baixo, segurança e integridade ao processo. No entanto, a segurança e integridade de dados para muitas organizações ainda é um problema, pois não aceitam, ou não podem deixar, seus dados armazenados em um local externo (ABPMP, 2013). Observa-se que as plataformas de BPMSs em nuvem são rapidamente disponibilizadas a partir de pequenos processos simples, sem integração ou baixa interação com outros sistemas (SGANDERLA, 2018).

A implementação de sistemas BPMS utilizando redes sociais é um desafio para as equipes de TI e de negócios. Algumas aplicações têm sido construídas para minerar dados e relações em redes sociais, de modo a extrair informações de clientes e produtos. Segundo (ABPMP, 2013), essa mineração de dados por meio de redes sociais influencia diretamente mudanças no negócio e operações da área de TI. As organizações necessitam ter flexibilidade para implementar rapidamente as mudanças impulsionadas por dados de redes sociais em seus modelos de processos de negócio.

2.4 TRABALHOS RELACIONADOS

Os modelos de processos de negócio têm auxiliado engenheiros de software a se aproximarem mais do ambiente comercial. Na literatura, há trabalhos que mostram ser possível extrair requisitos funcionais e não funcionais do software a partir de modelos de processos de negócio (manual ou automaticamente), para auxiliar a elaboração do documento de especificação de requisitos. As pesquisas relacionadas à modelagem de processos de negócio e processos da Engenharia de Requisitos. Convém destacar três trabalhos encontrados nessas pesquisas usando a notação BPMN: Nogueira (2017a), Figueiredo (2018) e Bitencourt *et al.* (2016). Outros quatro trabalhos usando a abordagem S-BPM merecem destaque: Lednev e Ivaschenko (2016), Höver *et al.* (2013), Höver e Mühlhäuser (2014) e Geisriegler *et al.* (2017).

Nogueira (2017a) apresenta um processo para extrair requisitos funcionais e não funcionais a partir de modelos de processos de negócio em BPMN v2.0. O trabalho utilizou a

ferramenta de modelagem *Bizagi Modeler* para construção de modelos de processos de negócio, exportando os arquivos em BPMN para XPDL. O autor desenvolveu uma ferramenta em Java, capaz de converter os arquivos em XPDL para XML, usando um conjunto de heurísticas. Essa ferramenta avalia o grau de informação dos requisitos extraídos, de modo que se tiver uma classificação baixa, o analista de negócios pode completar e ou ajustar o modelo. Um documento de especificação de requisitos de software é gerado automaticamente a partir de um modelo de processos de negócio, incluindo diagramas de casos de uso, de classes e de atividades.

Figueiredo (2018) apresenta um processo sistemático usando recursos da Web Semântica (SBPM - *Semantic Bussiness Process Management*), com intuito de integrar as visões de negócios e TI. O objetivo do processo é a extração automática do conhecimento contido em um modelo de processos de negócio em BPMN, representando-o com uma estrutura ontológica em OWL (*Ontology Web Language*). O autor desenvolveu um sistema de consultas sobre as ontologias, provendo variadas informações sobre os respectivos modelos de negócio. O sistema que gera automaticamente a ontologia usou o framework de Web Semântica Apache JENA, que extrai dados e gera gráficos no formato RDF. O sistema de consultas, por sua vez, usou a linguagem SPARQL (*SPARQL Protocol and RDF Query Language*).

Bitencourt *et al.* (2016) objetivam extrair requisitos funcionais e não funcionais a partir de modelos de processos de negócio em BPMN, utilizando Linhas de Processos de Negócio (LPNs) com requisitos associados. A intenção é diminuir o tempo de elicitação de requisitos e aumentar a qualidade dos requisitos em termos de completude e de alinhamento com o negócio. Segundo os autores, a qualidade das LPNs aumenta com a reutilização dos requisitos, já que são testados e aprimorados sempre que outras instâncias de LPN são criadas. Os autores enfatizam a importância de se desenvolver ferramentas computacionais para apoiar essa abordagem.

Lednev e Ivaschenko (2016) usam a abordagem de gerenciamento de processos de negócios orientado ao sujeito, conhecida como S-BPM (*Subject-Oriented Business Process Management*). O objetivo é o desenvolvimento ágil de software utilizando a plataforma CUBA, que foi desenvolvida em Java. Os autores informam que uso da plataforma reduziu esforços de desenvolvimento e custos do projeto. O trabalho mostrou a viabilidade de se desenvolver um sistema de automatização de processos usando a abordagem S-BPM usando a plataforma CUBA. De modo geral, os autores identificaram potencial elevado de utilização dessa plataforma para o desenvolvimento ágil de software voltado ao gerenciamento de processos de negócios.

Höver *et al.* (2013) apresentam uma linguagem específica de domínio para descrever processos S-BPM, denominada de DSL (*Domain Specification Language*). Um fator que motivou à criação de DSL foi que as ferramentas atuais, como *Metasonic Suite*, por exemplo, armazenam os processos S-BPM no formato XML, que não é adequado para edição dos processos quando necessário. Outro fator foi a falta de uma notação baseada em texto adequada para escrever processos S-BPM. A linguagem DSL possui características que permitem escrever processos S-BPM em um estilo similar ao idioma natural e executar esses processos. A linguagem não é idêntica à linguagem natural, mas supre as necessidades atuais e pode ser aprimorada. Os autores propõem a representação dos processos S-BPM por meio de ontologias, usando OWL (*Web Ontology Language)*. Para os autores, as ontologias podem ser usadas para prover suporte à tradução entre diferentes linguagens e representações, sendo uma “interlinguagem” para descrever processos S-BPM.

Höver e Müuhlhäauser (2014) resolveram utilizar a linguagem OWL devido à sua popularidade e por ser um padrão recomendado pelo consórcio W3C. Além disso, os autores enfatizam que OWL é baseada em lógicas de descrição e permite a definição de classes, propriedades e relações entre objetos conceituais, possibilitando o desenvolvimento de uma ontologia S-BPM. A intenção dos autores é inserir regras lógicas e outras restrições para melhorar a semântica da ontologia apresentada, bem como adicionar mais elementos S-BPM e integrar especificações para ações multi-sujeito, multi-envio e multi-recebimento.

Geisriegler *et al.* (2017) apresentam uma plataforma de código aberto que pode ser usada de modo intuitivo para modelar processos S-BPM e exportá-los em arquivos OWL. A plataforma, construída como um aplicativo Web, também executa os processos de negócios independentemente de qualquer sistema operacional. A intenção é a criação de uma arquitetura de referência seguindo a metodologia da abordagem S-BPM. Para os autores, a abordagem S-BPM trata os processos de negócios como uma rede de atores distribuídos e independentes (humanos ou máquinas), que interagem por meio de troca de mensagens. A arquitetura proposta é projetada como uma coleção de microsserviços, já que suportam vários requisitos reais para um software empresarial. A prova de conceito apresentada ajudou a demonstrar que modelos de processos de negócios representados em OWL podem ser executados no protótipo desenvolvido.

2.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo, foi apresentada uma visão geral de modelos de processos de negócio no contexto do ciclo de vida BPM, bem como uma visão geral da notação gráfica BPMN, incluindo seu vocabulário gráfico e documentação textual utilizada nos modelos. Sistemas de gerenciamento de processos de negócio (BPMSs) foram abordados, apresentando-se aspectos de funcionamento e implementação desses sistemas. Trabalhos relacionados aos temas apresentados no capítulo foram suscintamente descritos, evidenciando a importância deste trabalho e quão promissores são os caminhos de pesquisa envolvendo modelos de processos de negócio, processos da Engenharia de Requisitos e microsserviços.

De modo geral, os modelos de processos de negócio auxiliam toda a área de negócio, transversalmente, de ponta a ponta. A execução dos processos controlada por BPMSs de fato traz ganhos em termos de comunicação, custos e otimização dos processos, possibilitando ajustes para melhorias contínuas. Contudo, os BPMSs atualmente disponíveis não são específicos para uma organização, requerendo configurações complexas. Assim, este trabalho contribui com a definição de um processo sistematizado para o desenvolvimento de BPMSs dedicados às reais necessidades de cada organização.

No próximo capítulo, serão apresentados conceitos da Engenharia de Software Contínua, com ênfase nas abordagens BizDevOps, DevOps e tecnologias associadas. Assim, o próximo capítulo está relacionado com o desenvolvimento de um BPMS, segundo os objetivos deste trabalho.

3 ENGENHARIA DE SOFTWARE CONTÍNUA

Com o advento da Web 2.0, a indústria de maneira geral teve a necessidade de atualizar e automatizar seus produtos e serviços cada vez mais. Um exemplo é a indústria automobilística, que faz uso de aplicativos e sistemas de automação em seus veículos. Outros setores também sentiram a necessidade de buscar novas tecnologias e aplicativos que impulsionassem seu crescimento (BOSCH, 2014). Assim, o interesse por aplicativos cada vez mais avançados tecnologicamente impulsionou a busca por novas formas de se desenvolver sistemas de software.

A tecnologia para desenvolvimento de software orientado a serviços, conhecida como SaaS (*Software as a Service*), trouxe benefícios às empresas, pois podem adquirir componentes que realmente precisam, com custos melhores, mais facilidades de atualização e integração com outros sistemas.

Segundo Bosch (2014), um software com abordagem SaaS é capaz de coletar dados sobre o que os clientes realmente usam em sua magnitude, comparativamente a um software instalado no computador local. O apontamento de Bosch (2014) é baseado no relatório "*The Agile Executive*", elaborado em 2001, pela instituição *Standish Group International* - um grupo internacional de consultoria e pesquisa em TI. Apesar da idade desse relatório, ele alerta para o fato de 50% das funcionalidades do software não serem usadas na maioria dos sistemas.

Um dos problemas que a Engenharia de Software enfrenta é o de ajustes a mudanças, ou seja, a adição ou alteração de novos requisitos durante o ciclo de vida do software. Alterações nos requisitos enquanto o sistema está em operação, requerem um paradigma não convencional para que as mudanças possam ir sendo integradas ao sistema de modo transparente aos usuários. Esse tipo de problema tem sido mitigado pela abordagem da Engenharia de Software Contínua, que usa conceitos de *BizDev* (planejamento estratégico e desenvolvimento) e de *DevOps* (desenvolvimento e operação).

Diante do exposto, a *Seção 3.1* apresenta uma visão geral da Engenharia de Software Contínua e seus desafios nos dias de hoje. A *Seção 3.2* aborda aspectos de BizDev e BizDevOps que são importantes para a compreensão dos princípios e práticas de DevOps, apresentados na *Seção 3.3*. Essa seção também apresenta o ciclo de desenvolvimento DevOps e as tecnologias mais utilizadas atualmente. Por outro lado, a S*eção 3.4* aborda

atividades de levantamento e especificação de requisitos na Engenharia de Software clássica, incluindo a documentação gerada. A intenção é possibilitar melhor comparação com as diferenças inerentes à Engenharia de Requisitos Contínua, como parte da Engenharia de Software Contínua, como apresentado na *Seção 3.5*. A *Seção 3.6*, por sua vez, traz uma síntese de alguns trabalhos da literatura relacionados aos temas abordados neste capítulo e que muito contribuíram para o desenvolvimento deste trabalho. Por fim, a *Seção 3.7* apresenta as considerações finais do capítulo.

3.1 CONCEITOS DA ENGENHARIA DE SOFTWARE CONTÍNUA

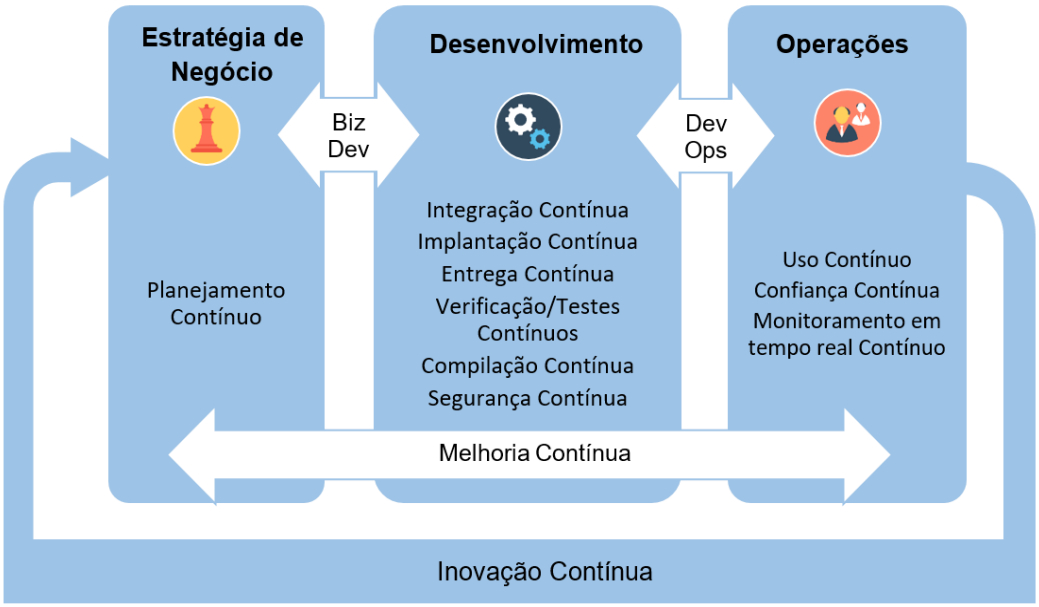
A Engenharia de Software Contínua trata todo o ciclo de vida do software como sendo um processo contínuo, desde seu planejamento até a entrega, garantindo uma continuidade, impulsionada pela busca incessante por inovação de seu produto (BOSCH, 2014). Segundo França *et al.* (2017), trata-se de um segmento recente da Engenharia de Software, cujos princípios estão embasados em metodologias ágeis e desenvolvimento enxuto (*Lean*). Contudo, segundo Forbrig (2016a), termos como “Engenharia de Requisitos Contínuos” e “Engenharia de Software Contínua” já vêm sendo discutidos desde 1998, durante o seminário “Dagstuhl” na Alemanha. Esse seminário discutiu aspectos relacionados a Sistemas de Informação, como complexidade e utilidade para a sociedade atual e futura.

A Engenharia de Software Contínua preza pela atenção constante na melhoria do produto. Com a observação do uso do software, pode-se colecionar dados operacionais e utilizá-los para análises históricas, preditivas e em tempo real. Isso possibilita transformar os dados em conhecimento, que passa a fazer parte do processo de engenharia, melhorando o projeto, permitindo a manutenção preventiva e otimização do desempenho do sistema (SHAMIEH, 2014).

Para França *et al.* (2017), a Engenharia de Software Contínua consiste em atividades que são realizadas em ciclos muito curtos e constantes, nos quais o fluxo de um determinado recurso de software é conduzido desde sua concepção até a sua disponibilidade para uso. Vários ciclos são executados em paralelo e associados às entregas, que são liberadas sempre que os ciclos terminam por uma demanda. A entrega contínua favorece o retorno constante do usuário (*feedback*) em relação ao uso do software, o que facilita a identificação de mudanças. Esse retorno não é restrito só ao final da interação, mas assim que um ciclo independente termina ou um recurso está pronto para ser utilizado (FRANÇA *et al.*, 2017).

Na concepção de Fitzgerald e Stol (2014), a Engenharia de Software Contínua também considera todo o ciclo de vida do software, identificando três fases principais: Estratégia de Negócios, Desenvolvimento e Operações. As fases definidas por Fitzgerald e Stol (2014) se relacionam constantemente, proporcionando a melhoria contínua do software. Ao final da fase de Operações, é realizado o retorno à primeira fase (Estratégia de Negócio), promovendo **Inovação Contínua**, como mencionado na **Figura 7**.

Figura 7 – Fases da Engenharia de Software Contínua.



Fonte: extraído e adaptado de Fitzgerald e Stol (2014).

Na **Figura 7**, aparece o termo “BizDev”, que consiste em uma abordagem utilizada para conectar as atividades das fases de Estratégia de Negócio e de Desenvolvimento e que será abordado na *Seção 3.2*. Já o termo “DevOps”, que faz a transição entre as fases de Desenvolvimento e Operações será abordado na *Seção 3.3*.

A fase de **Estratégia de Negócio** possui a subfase de **Planejamento Contínuo**, que consiste na integração das equipes das fases de Estratégia de Negócio e de Desenvolvimento do software. A relação estreita entre essas equipes é muito importante para que o planejamento contínuo possa ter resultados satisfatórios. Essa subfase possui ciclos paralelos e rápidos, com reuniões à medida que mudanças vão surgindo. Deve possuir elementos fundamentais para a organização de um projeto, como: planejamento organizacional, estratégico e comercial. Também deve incluir fatores de governança, liderança, transparência e desenvolvimento de competências. O planejamento pode incluir itens do produto, como portfólio, estratégias, etc. O tempo para essa subfase de planejamento ser concluída pode

variar, de acordo com cada projeto, sendo medido em horas, dias ou meses (FRANÇA *et al.*, 2017).

A segunda fase, de **Desenvolvimento**, possui seis subfases:

1) **Integração contínua** - é um processo de etapas interligadas, como: compilação de código, execução de testes de unidade e de aceitação, validação de código, controle de codificação de pacotes padrão (implantação, conformidade e construção). A frequência de aplicação dessas etapas é importante, pois ajuda a regular o processo e garantir um retorno rápido aos desenvolvedores. As falhas devem ser registradas, proporcionando uma base histórica para ajudar a agilizar soluções em projetos futuros;
2) **Implantação contínua** - nessa etapa são realizados testes e validações automáticas assim que o software for considerado pronto e disponibilizado para o ambiente de produção, mas não necessariamente para usuários;
3) **Entrega contínua** - garante que o software esteja continuamente pronto para lançamento e implantação para clientes;
4) **Verificação e testes contínuos** - a etapa de verificação adota a inclusão de métodos formais, com várias inspeções ao longo do processo de desenvolvimento até chegar ao seu final. Os testes contínuos são aplicados continuamente desde a descoberta do erro até sua eliminação de forma eficiente; envolvem alguns testes automáticos e alguns específicos para redução do número de erros;
5) **Compilação contínua** - procura satisfazer o desenvolvimento do software com padrões semelhantes de forma contínua, de modo a garantir a conformidade imediatamente antes da liberação do produto;
6) **Segurança contínua** - consiste na preocupação constante com a segurança dos requisitos não funcionais ao longo de todo o ciclo de vida do software, mesmo após a implantação com a identificação de vulnerabilidades de segurança.

A fase de **Operações** possui três subfases:

1) **Uso contínuo** - utiliza estratégia de negócio para a retenção dos usuários, ou seja, preza pela continuidade de uso pelos mesmos usuários ao invés de captar novos;

2) **Confiança contínua** - trata da confiança desenvolvida entre os usuários e o fabricante do software. Dessa forma, o cliente tem expectativas em relação a novos produtos;
3) **Monitoramento em tempo real contínuo** – objetiva a detecção precoce de problemas de qualidade no serviço, como, por exemplo, problemas de desempenho.

O conceito de “melhoria contínua” é baseado na abordagem de “Pensamento Enxuto” (“*Lean Thinking*”), visando sempre a busca pela satisfação do cliente, empregando valor no produto ou serviço oferecido, assim como no processo de desenvolvimento (FITZGERALD; STOL, 2014).

No contexto de negócios, a “inovação contínua” faz uma alusão ao processo pelo qual novas ideias são transformadas, criando valores comerciais para os produtos/serviços. Inovação contínua é o que se busca, por exemplo, quando uma empresa lança um software na versão *beta*, para que os usuários possam comentar erros e falhas do software antes da versão formal (FITZGERALD; STOL, 2014).

Para Fitzgerald e Stol (2017), um dos problemas centrais da Engenharia de Software clássica, mesmo com a utilização de metodologias ágeis, é a descoberta da real necessidade do cliente frente à utilização do software. Os gerentes de negócio tentam coletar os requisitos necessários que o cliente deseja, porém não conseguem saber quais recursos são mais usados. Já na Engenharia de Software Contínua é possível se identificar os recursos que realmente serão utilizados pelos usuários, assegurando o real retorno de investimento no projeto. Isso é feito por meio de análises sistemáticas dos recursos, aliadas a entregas rápidas e retornos (*feedbacks*) constantes do cliente.

No entanto, a Engenharia de Software Contínua encontra alguns desafios e resistências por parte de fábricas de software (FITZGERALD; STOL, 2014). É comum acreditar que só usar metodologias ágeis e técnicas como Kanban já é suficiente para o desenvolvimento contínuo, com entregas rápidas e contínuas (FITZGERALD; STOL, 2014). A Engenharia de Software Contínua procura ter uma visão holística, relacionando as estratégias, atividades, informações e recursos da organização, seja para uma fábrica de software pequena ou para um grupo de organizações diferentes que visam entregar um produto (FITZGERALD; STOL, 2014). O principal objetivo é manter um ciclo contínuo de suas atividades, visto que as tarefas são integradas entre as equipes das fases de Estratégia de Negócio, Desenvolvimento e Operações. A Engenharia de Software Contínua se aplica não

só ao processo de desenvolvimento e entrega do software, mas durante toda a vida do software.

Outro fator de desafio destacado por Fitzgerald e Stol (2017) é o contexto cultural. Em organizações que possuem softwares legados, o processo de atualização é mais demorado, visto que muitos desses softwares são produzidos em linguagens de programação mais antigas (ex.: COBOL) - o que representa uma barreira para implantação de novas funcionalidades e serviços. Fatores de resistência a mudanças também são considerados como uma barreira da Engenharia de Software Contínua. Um desafio adicional a ser considerado é quando o software usa componentes comerciais externos, constituindo mais um obstáculo para atualizações frequentes.

Fitzgerald e Stol (2017) também apontam outro desafio, relativo à velocidade do desenvolvimento do software. O fato da entrega ser rápida não significa que o ciclo de desenvolvimento também deva ser rápido. O fluxo e a continuidade são muito mais importantes em um primeiro momento do que a velocidade de desenvolvimento.

A integração da estratégia de negócio e o desenvolvimento do software são evidenciadas por ciclos ágeis e curtos, com frequência semanal do retorno do usuário (*feedback*), muito similar a metodologia ágil Scrum na fase de definição dos itens que compõem o software (*Product Owner*). Alguns desajustes entre as expectativas e o planejamento de Estratégia de Negócio podem ser gerenciados, desde que aplicadas métricas apropriadas para ajustar e proporcionar rapidez nas interações entre as pessoas envolvidas (FITZGERALD; STOL, 2017).

3.2 VISÃO GERAL DA ABORDAGEM DE BIZDEV E BIZDEVOPS

A abordagem BizDev *(Business Development*), é originária da área da administração que trata do desenvolvimento do negócio, como o próprio nome informa, onde é conhecida pela sigla “BD”. Está presente em diversas áreas como vendas, *marketing*, finanças e engenharia, etc. Essa abordagem atua na identificação e gerenciamento do relacionamento com outras organizações dentro ou fora da empresa, garantindo o cumprimento de suas metas (COHANE, 2015).

Na abordagem BizDev, a modelagem dos negócios e suas atividades estratégicas são integradas com o desenvolvimento de aplicações de software para a organização (FITZGERALD; STOL, 2014). Assim, fornece suporte contínuo à fase de modelagem, auxiliando a Engenharia de Requisitos Contínuos (FORBRIG, 2018a). Segundo Fitzgerald e

Stol (2014; 2017), BizDev pode ser entendida como um meio de interação entre a atividade inicial de Estratégia de Negócios e a atividade de Desenvolvimento do ciclo de vida do software na Engenharia de Software Contínua, como pode ser observado na **Figura 7**, consistindo numa transição anterior à fase DevOps (*Development Operations*).

A abordagem BizDevOps procura reduzir a lacuna de comunicação em relação à filosofia de negócios e o desenvolvimento de softwares relacionados. Proporciona, aos departamentos de negócios, um papel ativo na criação de serviços digitais e redistribui as responsabilidades entre os profissionais de TI (Tecnologia da Informação) e os de negócios (DREWS *et al.*, 2017).

De modo geral, a abordagem BizDevOps pode ser entendida como uma junção das regras de negócio (Biz), com o desenvolvimento (Dev) e as operações (Ops). Usando a abordagem “Biz”, as pessoas envolvidas na área de negócios podem analisar e divulgar os requisitos de maneira prática aos profissionais de TI, sendo definidos ciclos e *feedbacks* rápidos. “Dev” é um conceito empregado no departamento de TI para controlar o desenvolvimento de aplicações que garantam a qualidade necessária do software. O conceito “Ops” se refere ao conjunto de ferramentas utilizadas para integração e automação dos processos no desenvolvimento de um software (GRUHN; SCHÄFER, 2015).

Baumgartner (2014) sugere uma visão mais ampla do domínio ao utilizar a abordagem BizDevOps, como mostrado na **Figura 8**. As atividades de gerenciamento de portfólio, criação de valor, planejamento de negócios, projeto de negócios e *feedback* visam inovar e beneficiar os negócios (arco azul em torno de negócios: “Biz”). O desenvolvimento envolve os aspectos ligados à criação e entrega dos produtos, como, por exemplo, análise, projeto, codificação, testes e entregas (Dev). As operações (Ops) contêm ações de implantação, operação, monitoração e gerenciamento de versões, ou seja, ações de fornecimento e otimização do produto.

Figura 8 – Ciclo contínuo da abordagem BizDevOps.

Fonte: adaptado de Baumgartner (2014).

Para Gruhn e Schäfer (2015), as abordagens atuais para melhoria de processos de software e os métodos ágeis não são suficientes para aprimorar a fronteira entre a área de TI e a de negócios. Neste sentido, a abordagem BizDevOps permite, aos profissionais da área de negócios, participarem ativamente da construção de um software, pois fornecem subsídios detalhados às equipes de TI.

Gruhn e Schäfer (2015) destacam importantes aspectos dos benefícios de BizDevOps no desenvolvimento de software: (a) rápida reflexão de mudanças dos requisitos no produto de software; (b) gerir melhor as incertezas, instabilidades e as frequentes mudanças dos requisitos relacionados à área de marketing/vendas; (c) possibilitar amplo conhecimento sobre o domínio do negócio, considerando a transferência de conhecimento da equipe da área de negócios para a de TI. Cabe observar que a comunicação entre essas equipes deve levar em conta habilidades sociais, culturais e, principalmente, de comunicação (FORBRIG, 2018a).

Forbrig (2018b) propõe uma reflexão sobre a origem da abordagem BizDevOps, partindo dos conceitos BizDev e DevOps da Engenharia de Software Contínua proposta por Fitzgerald e Stol. Também sugere modificações nessa estrutura acrescentando a modelagem de requisitos contínuos e projeto (*design*) centrado no usuário. Além disso, menciona utilizar a modelagem de processos de negócio orientado ao sujeito (S-BPM) (ver *Seção 2.4*), aplicando seus modelos no contexto da abordagem BizDevOps.

Para Forbrig (2018b), os requisitos devem ser especificados com o apoio das pessoas da área de negócios; se isso não for possível, elas devem, pelo menos, conseguir ler a especificação desses requisitos. Para isso, o autor enfatiza a utilização de cenários, histórias de usuários e casos de uso para a modelagem de requisitos contínuos.

3.3 VISÃO GERAL DE DEVOPS

A abordagem DevOps (*Development and Operations*) trata do desenvolvimento e operações do software, combinando o desenvolvimento e a implantação do software com a operacionalização das equipes, de maneira ágil. Essa abordagem procura alinhar todos os esforços que visem garantir a qualidade dos produtos de software em construção. A equipe operacional passa informações para melhorias do software à equipe de desenvolvimento, favorecendo, de forma contínua, a integração, desenvolvimento, processos, monitoramento e entregas do software (WELLER, 2017).

Segundo Medrado (2015), DevOps complementa às abordagens de desenvolvimento ágil, pois adiciona a integração e entrega contínua ao processo de engenharia de software. Assim, DevOps é conhecida como uma abordagem de implementação contínua ou entrega contínua (GAEA, 2015). Segundo Freire (2013), DevOps combina o desenvolvimento ágil com entregas contínuas de software em produção com alto grau de qualidade.

Para Humble e Molesky (2011), DevOps representa a necessidade de alinhar o desenvolvimento do software e sua entrega, mesmo na sua produção. Os softwares do tipo *SaaS* são exemplos de sistemas com disponibilidade sem a necessidade de instalação em um dispositivo local, pois utilizam apenas a Internet, mesmo com a produção do software em andamento e atualizados em tempo real. Esses softwares podem ser executados em qualquer plataforma, como para *smartphone, tablet*, dispositivo *wearable,* etc.

A abordagem DevOps define um conjunto de práticas que são destinadas a reduzir o tempo de se realizar alterações no sistema e efetuá-las durante sua produção. Dessa maneira, garante a qualidade na entrega de novas funcionalidades (BASS *et al.*, 2015). Essa prática alinha os processos, ferramentas e responsabilidades das equipes de desenvolvimento com as equipes de operações.

A abordagem DevOps envolve princípios de colaboração e comunicação entre os departamentos das áreas de desenvolvimento e de operações de uma organização na produção de um software. O trabalho em conjunto das equipes só é possível se a organização promover um ambiente favorável às mudanças, tanto em termos culturais como de comunicação. Segundo Weller (2017), isso pode ser apoiado com atividades que promovam confiança entre as pessoas envolvidas nas equipes, como seminários de engajamento e socialização. Outra maneira sugerida pelo autor é padronizar os ambientes de desenvolvimento, com o gerenciamento de novas versões, controle e documentação do software.

De certa forma, DevOps está relacionada à gestão colaborativa de pessoas, de modo que os profissionais envolvidos possam compartilhar seus conhecimentos, traçar objetivos e dividir as responsabilidades. Medrado (2015) propõe reflexão sobre isso por meio de quatro perguntas, que se respondidas positivamente, caracterizam o emprego da abordagem DevOps na organização: (1) “a equipe de TI conhece os integrantes da equipe de operações e vice-versa?”; (2) “como é o envolvimento da equipe de operações em novos projetos?”; (3) “a equipe de operações conhece a pessoa responsável por desenvolver determinada funcionalidade?”; (4) “na organização são realizados eventos para promover a interação entre equipes?”.

Comparativamente a empresas que adotam processos tradicionais de desenvolvimento de software, empresas que adotam práticas de DevOps mostram dinâmica de trabalho mais ágil e atendimento mais eficaz aos clientes (AWS, 2017). Segundo Bass *et al.* (2015), essas práticas envolvem quatro aspectos de cunho organizacional, cultural e técnico: (1) obtenção rápida de mudanças na fase de produção; (2) utilização de testes automatizados para localização de erros; (3) eliminação ou redução de erros na fase de implantação; (4) correção ao identificar falhas no sistema.

Quando se muda o papel de uma equipe, é realizada uma mudança organizacional (BASS, 2018). No entanto, quando é feita uma adaptação de diferentes papéis de uma equipe, tem-se a mudança cultural. E quando se desenvolve um conjunto de casos de testes, tem-se uma mudança técnica. Já os testes automatizados envolvem aspectos organizacionais, culturais e técnicos.

3.3.1 Ciclo de desenvolvimento DevOps

Segundo Medrado (2015), a abordagem DevOps possui os seguintes princípios fundamentais para descrever seus valores e sua filosofia:

a) **Pensamento sistêmico** - aborda o fluxo do trabalho desde a equipe de desenvolvimento (que faz a implementação das mudanças conforme as regras de negócio) à equipe de operações (que realiza a entrega das mudanças ao cliente);

b) **Ciclos de retorno amplificados** - refere-se ao retorno (*feedback*) dado pela equipe de desenvolvedores à equipe de operação, e vice-versa, bem como à comunicação eficiente entre os clientes internos e externos;

c) **Cultura da experimentação e aprendizado contínuo** - a repetição e a prática são fundamentais para a excelência contínua dos processos. Isso estimula a inovação e a aprendizagem, mantendo a equipe motivada na direção do novo e da mudança.

De modo semelhante, Humble e Molesky (2011) definem quatro princípios para DevOps:

a) **Cultural** - envolve mudanças culturais nas interações entre as equipes de planejamento e as de desenvolvimento. Nas reuniões e tomadas de decisão da

equipe de planejamento deve haver a participação de um representante rotativo da equipe de desenvolvimento - isso pode garantir a integração das equipes no projeto;

b) **Automação** - consiste em mecanismos para implantação, construção e testes que garantam a entrega do software em prazos curtos com *feedback* do usuário. Pode-se obter a automação por meio da elaboração de uma fase da produção até sua execução (*pipeline*) de implantação, onde as pessoas envolvidas podem visualizar o progresso, sua implantação e possíveis mudanças;

c) **Medição** - compreende a capacidade de entrega do software, estabelecendo metas de melhoria por meio do monitoramento de métricas de negócio, como por exemplo a implantação de novas versões do software;

d) **Compartilhamento** - consiste em dividir os recursos tecnológicos (ferramentas, infraestrutura), conhecimento e conquistas da equipe nos lançamentos bem-sucedidos.

Weller (2017) define três fases de DevOps, como mostrado na **Figura 9**, após a fase de Estratégia de Negócios (BizDev na **Figura 7**).

Figura 9 – Principais fases de desenvolvimento com DevOps.

Fonte: adaptado de Weller (2017).

As fases de DevOps definidas por Weller (2017), são descritas da seguinte maneira:

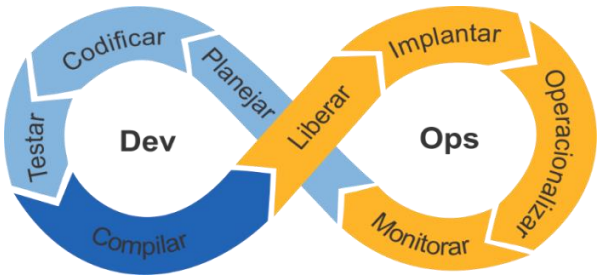
a) **Integração Contínua** - nessa fase é realizada a junção do código liberado pelas equipes de codificação do software. Ela envolve a verificação do código, sua compilação e utilização para realização de testes e validação;

b) **Desenvolvimento Contínuo** - fase que estende a anterior, voltada à automação dos testes, com pouca intervenção humana. Nessa fase o código está quase pronto para ser implantado;

c) **Entrega Contínua** - fase de inclusão dos códigos durante todo o processo de produção, sem a intervenção humana. As equipes que fazem a entrega contínua implantam somente códigos totalmente testados. Os testes, realizados com um número pequeno de usuários, devem ser feitos de modo que se verifique a qualidade e a usabilidade do código antes de ser implantado.

Segundo Weller (2017), as fases de integração e desenvolvimento podem se unir, formando uma fase única, onde o termo "*Dev*" remete à integração contínua. Já a fase de entrega contínua pode ser identificada pelo termo "*Ops*", formando o ciclo DevOps, mostrado na **Figura 10**. A fase *Dev* contempla as atividades de planejamento, codificação, testes e compilação da aplicação, enquanto a fase *Ops* abrange as atividades de liberação, implantação, operacionalização e monitoramento da aplicação.

Figura 10 – Ciclo DevOps e suas fases.



Fonte: adaptado de Weller (2017).

Gruhn e Schäfer (2015) apresentam uma plataforma de software para a abordagem BizDevOps e um estudo de caso realizado em uma grande empresa de resseguros. A plataforma desenvolvida por eles fornece, aos departamentos de negócios, a possibilidade de produzir partes de um software. Assim, as pessoas do departamento de negócios se tornam "programadoras" até certo ponto. Esse recurso é apoiado por uma estrutura conceitual que permite a equipe de TI mitigar os riscos dos "programadores" da área de negócios. Gruhn e Schäfer (2015) apontam, como vantagem, a flexibilidade do departamento de negócios no desenvolvimento dos aplicativos, bem como o controle de qualidade do software, desenvolvido em conjunto, pela equipe de TI.

3.3.2 Tecnologias utilizadas em DevOps

Atualmente, há diversos tipos de infraestrutura para aplicação dos princípios de DevOps, como plataformas de computação na nuvem e virtualização de sistemas operacionais, bem como ferramentas de gerenciamento de configuração, automação de *data centers*, automação de testes e técnicas de integração contínua (WELLER, 2017). Visandoeficiência dos aplicativos de software, na abordagem DevOps, esses aplicativos devem apresentar alto grau de modificabilidade, de capacidade de implementação, de monitorabilidade e de testabilidade (WELLER, 2017).

Na **Tabela 1**, são mencionadas algumas plataformas e recursos que podem ser utilizados para o desenvolvimento de aplicações com a abordagem DevOps, segundo Weller (2017) e Bertram (2017). De modo geral, são tecnologias de computação em nuvem, visualização, conteinerização, testes e gerenciamento de configuração/infraestrutura (codificação, implantação de aplicativos, monitoramento, alertas, supervisores de registro e processamento, segurança, colaboração e orquestração).

Tabela 1 – Plataformas e ferramentas para desenvolvimento DevOps.

Categorias	Ferramentas
Plataformas de computação nas nuvens	*Google Cloud Platform, Pivotal Cloud Platform, e Amazon Web Services.*
Plataformas de virtualização	*Xen, VirtualBox, Vagrant, Amazon Web Services, Microsoft Azure e VMware vCloud*
Ferramentas de contêineres	*LXC, Docker e rkt (para containers de apps no Linux).*
Repositório de código-fonte	*Git, CloudeForce, TFS e Subversion.*
Automação de testes	*Selenium e Water.*
Servidor de compilação de sistemas	*Gradle, Jenkins e Solano Labs, SonarQube e Artifactory.*
Gerenciamento/Configuração	*AWS CloudFormation, Ansible, Puppet, HashiCorp Terraform, Chef, Salt e Capistrano.*
Monitoramento e alertas	*VictorOps, PagerDuty ou Sensu.*
Supervisores de registro e processamento	*Logotries, Sumo Logic, PaperTrail, Loggly e Splunk.*
Segurança	*Sonatype, Snort, Veracode ou Snorby Threat Stack.*
Supervisão	*Monit, Upstart e runit.*
Colaboração	*JIRA, Slack, HipChat, e ServiceNow.*
Orquestração	*AWS Elastic Container Service, Docker Swarm, Kubernetes ou Apache Mesos.*

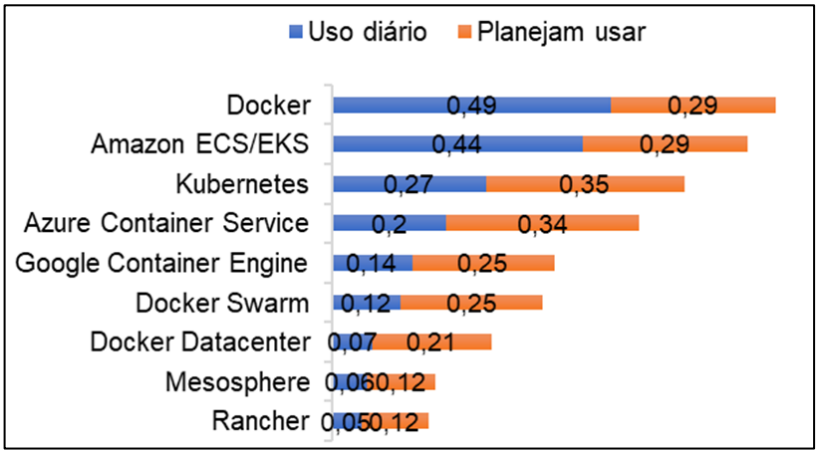
Fonte: adaptado de Weller (2017) e Bertram (2017).

Cabe observar que o desenvolvimento de aplicações usando plataformas em nuvem tem se mostrado como uma forte tendência no desenvolvimento de sistemas usando DevOps. Essas plataformas oferecem cada vez mais serviços e possibilitam gerenciar melhor a carga de trabalho das equipes em função da adição de novos clientes.

Segundo o relatório técnico de 2018 da empresa de gerenciamento de software de computação em nuvem "*RightScale*", denominado "*State of the Cloud Survey* 2018", a adoção do uso dos recursos de nuvem pública (*public cloud*) vem crescendo a cada ano. Nas plataformas de nuvem pública, as empresas ou profissionais de desenvolvimento de software pagam somente os recursos que consomem/utilizam dentro de um determinado período. As pesquisas nesse relatório envolveram 997 pessoas da área TI (executivos técnicos, gerentes, profissionais, etc.) de organizações que estão em todos os estágios de maturidade da computação na nuvem.

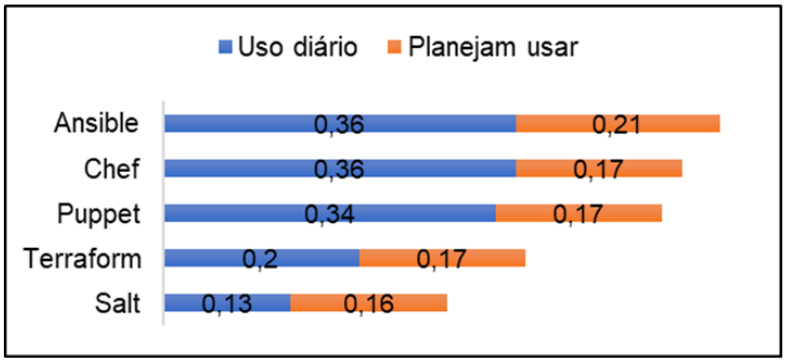
O relatório mencionado mostra as tecnologias de contêineres mais utilizadas (**Figura 11**), onde a ferramenta Docker (49% das empresas fazem uso diariamente) está à frente dos serviços de conteinerização da Amazon ECS/EKS, bem como das ferramentas Kubernetes e Azure Container Service (RIGHTSCALE, 2018). Já entre as ferramentas de configuração classificadas no relatório, como mostrado na **Figura 12**, destacam-se as ferramentas de gerenciamento de configurações Ansible, Chef e Puppet, com nível de utilização na faixa de 30%, praticamente empatadas."

Figura 11 – Ferramentas de contêineres mais utilizadas com DevOps.



Fonte: RightScale (2018).

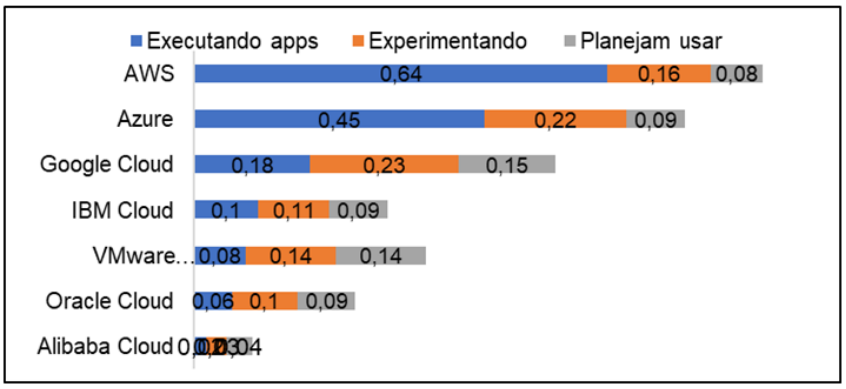
Figura 12 – Ferramentas para gerenciamento de configuração mais utilizadas com DevOps.



Fonte: RightScale (2018).

Em relação aos serviços de nuvem pública, a *AWS* mantém a maioria da execução de *apps*, enquanto a *Azure* continua a crescer em função dos seus clientes corporativos, como mostrado na **Figura 13** (RIGHTSCALE, 2018). Já a plataforma *Google Cloud* mantém a terceira posição, seguida pela *IBM Cloud.* Esse tipo de tecnologia ganha, de fato, cada vez mais adeptos na abordagem DevOps, atendendo também os princípios da Engenharia de Software Contínua.

Figura 13 – Plataformas de nuvem pública mais utilizadas.



Fonte: RightScale (2018).

3.4 ESPECIFICAÇÃO DE REQUISITOS NA ENGENHARIA DE SOFTWARE

A Engenharia de Requisitos é considerada a etapa essencial no processo de desenvolvimento de um software, uma vez que trata as informações sobre as regras de negócio do projeto do software. É nesta etapa que se tem a descoberta, análise e verificação dos serviços e restrições existentes no software a ser construído (SOMMERVILLE, 2011).

Na mesma direção, Vasques e Simões (2016) definem a Engenharia de Requisitos como um conjunto de atividades que envolvem a aquisição, documentação e manutenção de um conjunto de requisitos para construção de um software, visando atender regras de negócio, com foco na qualidade.

Segundo Pressman (2011), a Engenharia de Requisitos constitui o elo entre o projeto do software como um todo e sua construção, de maneira a entender os anseios do cliente, analisando suas necessidades e avaliando a viabilidade econômica para problemas reais.

De modo geral, as atividades desenvolvidas durante as etapas da Engenharia de Requisitos têm, por finalidade, obter uma visão abrangente, clara e precisa dos requisitos necessários para o produto de software (PAULA FILHO, 2012). Para Pressman (2011), as atividades, que podem ocorrer paralelamente, podem ser classificadas e descritas da seguinte maneira:

a) **Concepção** - atividades que definem o software a ser construído, provendo entendimento do problema, elaboração de possíveis soluções e definição das pessoas envolvidas no processo;

b) **Levantamento** - atividades para coleta dos requisitos, buscando-se as fontes de informação e os dados a serem trabalhados durante todo o processo de desenvolvimento do software. Essas atividades, geralmente, são feitas diretamente com os usuários. Alguns problemas que podem ocorrer: problemas de escopo (detalhes técnicos desnecessários), entendimento (dúvidas sobre o que será necessário para o sistema operar de forma correta, incertezas de requisitos e falta de comunicação do usuário com desenvolvedores) e volatilidade (alterações dos requisitos);

c) **Elaboração** - atividades para a criação de um modelo de requisitos, capaz de identificar as funcionalidades e comportamentos do software. Envolvem a descrição dos cenários dos usuários e como esses devem interagir com o sistema proposto. Consistem na definição das classes (atributos e métodos) e diagramas que modelam os requisitos, com observações que elucidam possíveis dúvidas sobre os requisitos levantados;

d) **Negociação** - atividades para solução de conflitos sobre as funcionalidades, custos, prazos e riscos do projeto do software. O processo de negociação possibilita que os clientes, usuários e a equipe desenvolvedora do software possam discutir qual prioridade será considerada, de modo que todos os envolvidos estejam satisfeitos com o acordo realizado;

e) **Especificação** - atividades para a redação de um documento que combina os

diagramas com descrições em linguagem natural, envolvendo elaboração dos cenários de casos de uso para funcionalidades específicas do software;

f) **Validação** - atividades para examinar a especificação dos requisitos, garantindo a declaração de todos os requisitos e identificando inconsistências, omissões e erros variados. Essa avaliação dos requisitos é realizada por uma equipe de revisão técnica;

g) **Gestão** - atividades que auxiliam a equipe do projeto a identificar, acompanhar e controlar as necessidades de mudanças dos requisitos ao longo do projeto.

Por outro lado, Vazques e Simões (2016) consideram que a Engenharia de Requisitos agrega três classes de atividades, descritas como:

1) **Gerência de requisitos** - atividades que identificam a melhor maneira de comunicação relativas aos requisitos, bem como acesso e priorização dos requisitos, prezando pela administração de conflitos e mudanças, de acordo com o escopo do projeto;

2) **Elicitação de requisitos** - atividades que contemplam as técnicas de levantamento de dados, identificando as regras de negócio, partes interessadas e identificação dos requisitos adicionais;

3) **Análise de requisitos** - atividades que tratam dos conflitos não abordados na elicitação dos requisitos, bem como da verificação e validação dos requisitos. Envolve a documentação, organização, modelagem e classificação das informações obtidas.

Segundo Sommerville (2011), em termos de descrição, há requisitos do usuário e requisitos do sistema. Os requisitos do usuário são declarações de alto nível de abstração, ou seja, especificados por meios de diagramas, contendo as especificações e restrições fornecidas pelo usuário do sistema. Já os requisitos do sistema são descrições técnicas detalhadas, contendo o que deve ser implementado no sistema. Dessa forma, os requisitos são utilizados de diversas maneiras e por diferentes leitores envolvidos no desenvolvimento de um software.

Além de auxiliar as equipes durante todo o ciclo de desenvolvimento do software, as atividades de elicitação ou levantamento de requisitos são fundamentais para a elaboração do termo de abertura do projeto (PMI, 2013). O plano de gerenciamento de requisitos de um

software descreve como serão os requisitos serão analisados, documentados e gerenciados, a fim de atender as necessidades do produto a ser desenvolvido.

Vazques e Simões (2016) observam que, antes do levantamento de requisitos, o escopo do projeto deve estar bem definido, envolvendo aspectos internos e externos ao sistema. Em relação à modelagem dos requisitos que fazem parte do escopo, Vazques e Simões (2016) propõem três meios:

1) **Diagrama de contexto** - representa o ambiente no qual o sistema está inserido, como um processo único, identificando os principais fluxos de dados e sem abordar o processamento interno;
2) **Diagrama de casos de uso** - descreve quem faz as ações do sistema, ou seja, define os atores que irão participar dos eventos. Identifica os casos de uso, seus atores, resultado esperado e entidades externas, com as quais o sistema deve interagir;
3) **Modelo de processos de negócio** - descreve o estado do negócio e recursos envolvidos, como: pessoas, informações, instalações, etc. Nesse modelo, são identificados os atores, suas tarefas e respectivos relacionamentos, bem como as tarefas manuais, que não são gerenciadas por sistemas de informação.

Para Paula Filho (2012), uma das formas de aumentar a eficácia das atividades de levantamento de requisitos, com qualidade, é com o uso de protótipos e oficinas de requisitos. A prototipagem de requisitos consiste na elaboração visual de um produto, podendo ser um desenho simples (desenhos das telas feito à mão) ou desenhos mais elaborados feitos com apoio de um software. Assim, os aspectos críticos dos requisitos de um produto podem ser visualizados por elementos de uma interface, com painéis, botões, campos e comandos. O usuário pode, então, de maneira mais visual, aprovar ou não o que se pretende construir (PAULA FILHO, 2012).

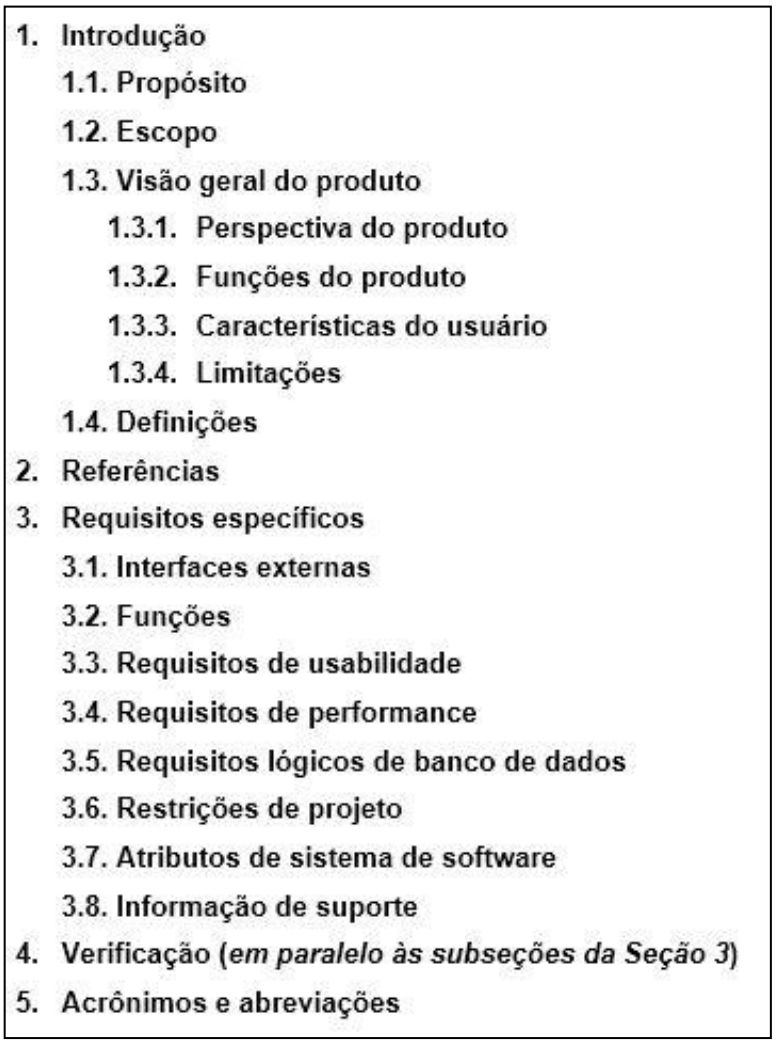
As oficinas de requisitos, mencionadas por Paula Filho (2012), são reuniões para elaborar documentos e materiais utilizados durante diversas atividades do ciclo de vida do software. Essas reuniões contam com a participação de desenvolvedores, gerentes e usuários chaves para a elaboração do levantamento e da negociação dos requisitos. Assim que os artefatos provenientes das reuniões são analisados, é possível analisar o escopo do projeto, assim como suas metas de custos e prazos (PAULA FILHO, 2012).

Os diagramas de casos de uso são muito utilizados para compreensão, especificação e documentação dos requisitos funcionais e identificação dos tipos de usuários (atores) que

interagirão com o sistema (GUEDES, 2011). Em relação aos requisitos não-funcionais, devem ser expressos de forma precisa e quantitativa, segundo normas internacionais vigentes.

O documento denominado “Especificação de Requisitos de Software” (ERS), ou *Software Requirements Specification* (SRS), é um documento que contém as funcionalidades a serem implantadas pelos desenvolvedores, os atores envolvidos e suas permissões de acesso, além dos requisitos não-funcionais (SOMMERVILLE, 2011). Inicialmente, a organização IEEE (*Institute of Electrical and Electronics Engineers*) definiu, juntamente com o Departamento de Defesa dos Estados Unidos, uma estrutura do documento ERS identificada como IEEE/ANSI 830-1998. Atualmente, uma versão revisada dessa norma consiste no padrão internacional ISO/IEC/IEEE 29148-2018, onde a estrutura do documento ERS é organizada em cinco seções, como mostrado na **Figura 14**. Observa-se que não é necessário que sejam utilizados os mesmos nomes das seções e nem a mesma ordem apresentada, mas sim o conteúdo do documento (NOGUEIRA, 2017a).

Figura 14 – Estrutura do documento “Especificação de Requisitos de Software”.



Fonte: ISO (2011).

3.5 ESPECIFICAÇÃO DE REQUISITOS NA ENGENHARIA DE SOFTWARE CONTÍNUA

Segundo Forbrig (2016b), a Engenharia de Software Contínua pode ser aplicada com sucesso se combinar princípios bem da engenharia de software aliados com metodologias ágeis, para sustentar a viabilidade dos sistemas em desenvolvimento. A especificação de requisitos na Engenharia de Software Contínua trata também de novos requisitos que são inseridos no ciclo de desenvolvimento contínuo. Para Forbrig (2016b), as alterações nos requisitos requerem desenvolvimento adicional diferenciado no sistema iterativo.

De certa forma, pode-se dizer que a Engenharia de Requisitos Contínua, ou *Continuous Requirements Engineering* (CRE), combina a Engenharia de Requisitos tradicional com a concepção do produto centrado no usuário, com avaliações e monitoramento dos sistemas em execução.

A Engenharia de Requisitos Contínua pode ser aplicada em grandes empresas que possuam ciclos de mudanças relativamente longos ou em empresas que necessitem de alterações dos requisitos com mais frequência. Já sua utilização em fábricas de software de pequeno porte pode ser mais adequada, visto que essas empresas desenvolvem novos softwares, conforme as oportunidades vão surgindo de maneira mais prática e rápida.

Na Engenharia de Requisitos Contínua, a aplicação de conceitos pertinentes a interfaces centradas no usuário colabora para a participação contínua dos usuários no ciclo de vida do desenvolvimento do software. Isso contribui para melhorar a qualidade do sistema, a usabilidade e a experiência do usuário (FISCHER *et al.*, 2016). Neste sentido, Fischer *et al.* (2016) especificam os seguintes critérios para definir o escopo do projeto no contexto da Engenharia de Requisitos Contínua:

a) **Participação dos usuários** - todas as pessoas interessadas no projeto devem se envolver com o ciclo do desenvolvimento do software, desde a elicitação dos requisitos até a entrega do produto;

b) **Iteração** - a iteração deve ser pautada na elicitação dos requisitos do usuário, envolvendo conhecimentos implícitos e suas necessidades. A quantidade de iterações depende da qualidade desses requisitos;

c) **Tarefas adequadas** - as tarefas dos usuários devem ser definidas corretamente, para que o sistema possa apoiar os usuários no cumprimento de suas tarefas reais;

d) **Prototipagem** - deve ser usada o mais cedo possível, para validar as funcionalidades do software;

e) **Ideia geral** - deve ter uma sincronia dos requisitos do usuário com os da empresa, para que dependências de funcionalidades sejam identificadas com antecedência, promovendo a interação de experiências do usuário;

f) **Arquitetura** - está relacionada com os elementos de interface do usuário por meio de componentes de software reutilizáveis. O sistema deve ser flexível a mudanças, como adição de novos requisitos, por exemplo;

g) **Integração** - a integração deve ser centrada no usuário e aos processos da Engenharia de Software, pois as atividades da Engenharia de Requisitos Contínua não podem ser vistas como atividades paralelas;

h) **Continuidade** - os requisitos devem estar disponíveis e formalizados, de modo que promovam mudanças contínuas.

Fischer *et al.* (2016) observam que a especificação de requisitos acarreta preocupação constante mesmo com os critérios definidos pelos autores. Segundo Forbrig (2016b), a origem desses problemas está na fase de análise, onde são definidas as necessidades e objetivos dos projetos.

Para Forbrig (2016b), uma maneira de reduzir o tempo na especificação de requisitos é a sua reutilização. Para isso, pode-se utilizar uma linguagem de especificação para descrever os requisitos funcionais e não funcionais. Como exemplo dessas linguagens, o autor menciona BPMN e S-BPM, visto que as ferramentas que as implementam, geralmente auxiliam a atualização rápida das especificações de requisitos.

Para Kirikova e Purmalietis (2017), a Engenharia de Requisitos Contínua implica na capacidade de reação rápida às mudanças que possam surgir no sistema, observando seu contexto organizacional ou seu ambiente externo. Nesse sentido, os modelos de processos de negócio surgem como apoio aos “requisitos contínuos” de software, pois as equipes da área de gestão conseguem alterá-los mais facilmente. Portanto, a documentação é extremamente importante para compor as fases de elicitação e análise da Engenharia de Requisitos, seja na abordagem clássica ou na contínua.

3.6 TRABALHOS RELACIONADOS

Entre os trabalhos encontrados na literatura, alguns foram considerados relevantes devido à sua maior contribuição para este trabalho e, por isso, são resumidamente apresentados nesta seção.

Para compreensão da Engenharia de Software Contínua, destacam-se os seguintes trabalhos: Fitzgerald e Stol (2014, 2017), Krusche e Bruegge (2017), Forbrig (2016a), Forbrig (2016b) e França *et al.* (2017).

Fitzgerald e Stol (2014, 2017) apresentam uma revisão de literatura acerca dos conceitos da Engenharia de Software Contínua, com um roteiro bem definido a ser seguido. As etapas do processo da Engenharia de Software Contínua são apresentadas sob uma visão holística.

Krusche e Bruegge (2017) apresentam um metamodelo de Engenharia de Software Contínua denominado CSEPM (*Continuous Software Engineering Process Metamodel*), que descreve as expectativas naturais da Engenharia de Software com a introdução da ideia de fluxos de trabalho que podem ser interrompidos por eventos de mudança. O metamodelo CSEPM trata as atividades de desenvolvimento como fluxos de trabalho paralelos, incluindo dois tipos de modelos: estáticos e dinâmicos. O modelo estático descreve as relações entre conceitos específicos da Engenharia de Software Contínua, incluindo revisões, lançamentos e comentários. O modelo dinâmico retrata como os fluxos de trabalho do desenvolvimento são ativados e interrompidos através de eventos de mudança. O metamodelo permite instanciar modelos de processos lineares e iterativos e foi projetado para modelos de processos ágeis e contínuos, de modo a permitir adaptação, personalização e extensão.

A abordagem da Engenharia de Software Contínua de Forbrig (2016a) tem ênfase especial na modelagem contínua de processos de negócios e projeto (*design*) centrado no usuário, com a adoção de metodologias ágeis (utilizou Scrum no trabalho). Sua ideia é que a manutenção do software integre estratégias empresariais e que o desenvolvimento do software seja um processo contínuo. O autor apresenta a ideia de “modelagem de processos de negócio contínuos” como uma evolução da gestão integrada, que visa permitir que as organizações continuem inovando e aprimorando suas operações. Forbrig (2016a) entende que a Engenharia de Software Contínua deve integrar administração de negócio, onde a adaptação da estratégia do negócio e o monitoramento do sistema em execução são dois pontos importantes para serem tratados conjuntamente. Convém observar que o autor tomou como base o trabalho de Fleischmann *et al.* (2015), em que a especificação e implementação

dos requisitos são oriundas das especificações dos modelos de processos de negócio orientados ao assunto (S-BPM).

Forbrig (2016b) mantém a mesma ideia do trabalho apresentado em Forbrig (2016a), mas acrescentando duas fases à atividade inicial de “Estratégia de Negócios”, denominadas: “Engenharia de Requisitos Contínua” e “Modelagem de Processos de Negócio Contínuos” – ambas com foco na interface contínua centrada no usuário. O autor observa que seu trabalho está inserido na ideologia da visão holística de Fitzgerald e Stol (2017). O desenvolvimento de aplicações segundo o modelo proposto pelo autor pode ser encontrado em Forbrig (2016c).

França *et al.* (2017), por sua vez, apresentam uma revisão bibliográfica sobre “Planejamento Contínuo”, juntamente com estudos de caso relatados pela empresa de software OWSE Informática, situada na cidade Rio de Janeiro. O trabalho compara técnicas de casos de uso encontradas na literatura com as empregadas na referida empresa. Os autores trazem o foco para as Informações organizacionais (tamanho, domínio, serviços/produtos fornecidos e processo de desenvolvimento de software) e para o “Planejamento Contínuo” (motivações, desafios e benefícios observados).

Em relação às pesquisas sobre a Engenharia de Requisitos Contínua, destacam-se três trabalhos que contribuíram para a especificação de requisitos neste trabalho: Finke (2016), Kirikova (2016) e Kirikova e Purmalietis (2016).

Finke (2016) discute os conceitos de continuidade e herança na Engenharia de Requisitos Contínua, apresentando um método/ferramenta com suporte à herança nesse contexto. O autor também apresenta um exemplo prático de caso de uso para ilustrar a complexidade de um projeto, fluxo de informações e análise/manutenção de requisitos contínuos.

Kirikova (2016) analisou a aplicação da Engenharia de Requisitos Contínua no contexto da estrutura FREEDOM, oriunda de pesquisas sobre sistemas de trabalho e viabilidade de sistemas. A estrutura FREEDOM ajudou a identificar as principais unidades de informação a serem tratadas nas atividades da Engenharia de Requisitos Contínua. Kirikova (2016) concentra-se nas relações de propagação e *feedback* entre a função de engenharia de requisitos e outros constituintes da estrutura FREEDOM. Diversas atividades da Engenharia de Requisitos Contínua foram avaliadas pelo autor, que indica que tais atividades devem providenciar possibilidades de propagação, monitoramento, análise e auditoria do conhecimento.

Kirikova e Purmalietis (2016), por sua vez, avaliam as atividades de análise da Engenharia de Requisitos na estrutura FREEDOM. O objetivo é examinar quais são as fontes

potenciais de dados a serem investigadas e quais são os métodos para definir requisitos na Engenharia de Requisitos Contínua.

De modo geral, todos os trabalhos mencionados nesta seção colaboraram para o entendimento da abordagem “Contínua” na Engenharia de Software, de maneira a elucidar os principais aspectos integrados no ciclo de desenvolvimento ágil. Fitzgerald e Stol (2017), por exemplo, contribuíram de maneira significativa para a compreensão de todo o processo contínuo. O metamodelo idealizado por Krusche e Bruegge (2017), por outro lado, serviu de inspiração para o desenvolvimento de outros modelos utilizando a Engenharia de Software Contínua integrada a modelos de processos de negócio. Já Forbrig (2016a) e Forbrig (2016b) contribuíram para melhor identificar o papel dos usuários neste trabalho (são modeladores dos processos de negócio da organização) e como podem atuar de maneira ágil e com enfoque contínuo. Por outro lado, Finke (2016), Kirikova (2016) e Kirikova e Purmalietis (2016) contribuíram com novos métodos para a especificação de requisitos na Engenharia de Requisitos Contínua.

3.7 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Na *Seção 3.1*, foi apresentada uma visão geral sobre a Engenharia de Software Contínua, bem como as atividades que envolvem essa recente subárea da Engenharia de Software. Na *Seção 3.2*, foi apresentada a abordagem BizDevOps, enquanto a *Seção 3.3* abordou *DevOps*, evidenciando fatores importantes para a integração com áreas de negócio, bem como as tecnologias mais utilizadas nessa abordagem. Estendendo a *Seção 3.4,* sobre a Engenharia de Requisitos clássica, a *Seção 3.5* apresentou aspectos diferenciados da Engenharia de Requisitos Contínua, com destaque à satisfação das necessidades e mudanças constantes do usuário, frente ao ambiente de negócio.

Os trabalhos relacionados à Engenharia de Software Contínua e, mais especificamente, à Engenharia de Requisitos Contínua, apresentados na *Seção 3.6*, contribuíram para melhor compreensão dos temas abordados, bem como ver modos de melhorar a elicitação e análise de requisitos de forma integrada com processos de negócio, como tratado neste trabalho. Cabe observar que, nas pesquisas realizadas, a quantidade de trabalhos encontrados sobre a abordagem DevOps foi muito superior à de publicações sobre BizDev.

De modo geral, o conteúdo apresentando neste capítulo foi fundamental para embasar o desenvolvimento do processo sistematizado alvo deste trabalho, voltado à

especificação e implementação de microsserviços, na abordagem *BizDevOps*, a partir de modelos de processos de negócio. Nessa direção, o próximo capítulo aborda a especificação, arquitetura, tecnologias e melhores práticas para o desenvolvimento de microsserviços.

4 ARQUITETURA DE MICROSSERVIÇOS

As expectativas sobre o que se espera das aplicações de software atualmente estão cada vez mais complexas, sendo requerido alto grau de flexibilidade, escalabilidade e desempenho, entre outros aspectos. Segundo Fowler (2017), é importante que se implemente algoritmos de balanceamento de carga, pois à medida que as demandas operacionais aumentam, mais servidores são necessários para hospedar a aplicação. Adicionalmente, quanto mais recursos são utilizados em uma aplicação, o código tende a aumentar e se tornar mais complexo, com mais processamento de tarefas, requerendo servidores com maior capacidade e melhor desempenho (FOWLER, 2017). Nessa direção, a quantidade de testes também aumenta significativamente, para garantir que qualquer alteração não comprometa o sistema.

Nesse contexto, gerenciar todos esses requisitos não é uma tarefa simples no desenvolvimento dos sistemas de software. Muitas vezes, adiamentos de correções cruciais podem ser feitos para se evitar problemas operacionais ainda maiores. No entanto, isso faz com que também se aumente rapidamente a defasagem técnica da aplicação (FOWLER, 2017). Aplicações conhecidas como “monolíticas”, onde todos os serviços e recursos estão contemplados em um código base, requerem grande comprometimento de esforços para manterem-se operacionalmente, sem degradação do desempenho, frente à necessidade de alterações frequentes e aumento da escalabilidade.

Assim, sistemas compostos por microsserviços estão cada vez mais em evidência, devido à independência de cada microsserviço em relação aos demais, possibilitando, de modo mais simples e eficaz, a implementação das alterações requeridas, novas extensões ao sistema e mais facilidade de testes. Cada microsserviço pode ser considerado como uma “microaplicação” para executar uma funcionalidade específica, sendo desenvolvido, testado e implementado de forma independente dos demais (NEWMAN, 2015). Operacionalmente, os microsserviços que compõem um sistema são executados em um ecossistema específico de microsserviços (FOWLER, 2017; O'CONNOR *et al.*, 2017).

Frente ao exposto, este capítulo visa apresentar aspectos e princípios sobre o desenvolvimento de microsserviços, que foram importantes para os propósitos deste trabalho. Assim, a *Seção 4.1* apresenta uma visão geral da arquitetura de microsserviços, abrangendo

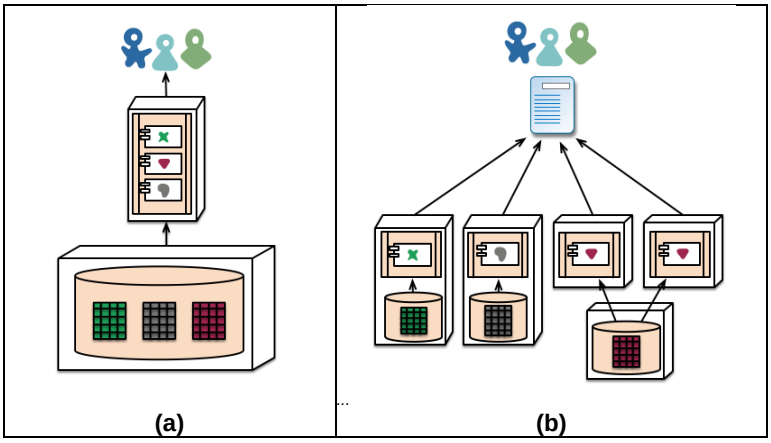
ecossistemas relacionados e a dissociação com sistemas monolíticos e arquiteturas orientadas a serviços, conhecidas como SOA (*Service-Oriented Architecture*). A *Seção 4.2*, por sua vez, aborda essencialmente técnicas para a especificação de microsserviços, enquanto a *Seção 4.3* apresenta as melhores práticas para o desenvolvimento desses microsserviços. Alguns trabalhos da literatura relacionados a microsserviços estão destacados na *Seção 4.4*, devido às contribuições para este trabalho. A *Seção 4.5*, por sua vez, apresenta as considerações finais do capítulo.

4.1 VISÃO GERAL DA ARQUITETURA DE MICROSSERVIÇOS

Segundo Fowler (2017), a adoção da arquitetura de microsserviços pode ser considerada um processo natural de escalonamento de uma aplicação. A organização que deseja implantar a arquitetura de microsserviços deve padronizar seus processos, de modo que garanta a confiança e qualidade dos requisitos de cada microsserviço (FOWLER, 2017).

Na **Figura 15**, pode se ver observar a diferença marcante entre a arquitetura de um sistema monolítico e a de um sistema composto por microsserviços. Na arquitetura monolítica, as funcionalidades estão em uma única aplicação, com acesso a um único sistema de banco de dados. Na arquitetura de microsserviços, as funcionalidades estão distribuídas em pequenas aplicações independentes (microsserviços).

Figura 15 – Arquitetura monolítica (a) *versus* arquitetura de microsserviços (b).



Fonte: Fowler (2014).

Segundo O'Connor *et al.* (2017), cada microsserviço pode operar com um sistema de banco de dados diferente de outro microsserviço. Cada microsserviço executa um processo particular e pode se comunicar com outros microsserviços por meio de uma interface

de programação de aplicativos (API -*Application Programming Interface*) e/ou recursos do protocolo HTTP (*HyperText Transfer Protocol*). Como exemplo de uso dos recursos da Web, Egídio (2017) apresenta uma aplicação de comércio eletrônico com microsserviços integrados o estilo arquitetural REST (*Representation State Transfer*).

Segundo Fowler (2017), as aplicações monolíticas operam, de alguma maneira, envolvendo três camadas: “front-end”, “back-end” e armazenamento de dados. Quando alguma funcionalidade é requisitada, todos os serviços da aplicação são executados, como o carregamento do banco de dados, por exemplo – o que torna o acesso às informações e a própria execução da funcionalidade mais lenta. Para aplicações pequenas, com um ou dois módulos, a arquitetura monolítica é recomendada, pois centraliza as informações em um só local, facilitando e agilizando o desenvolvimento. Contudo, de modo geral, sistemas monolíticos não são facilmente escaláveis, possuem replicação de código (ao invés de reutilização), são dependentes das tecnologias e geram problemas na implementação de mudanças frequentes. No caso de atualização de um recurso utilizado por um só módulo, por exemplo, pode gerar problemas com outras partes da aplicação. Além disso, a implementação com uma só linguagem de programação pode não oferecer suporte necessário a novas funcionalidades a serem implantadas na aplicação (EGIDIO, 2017).

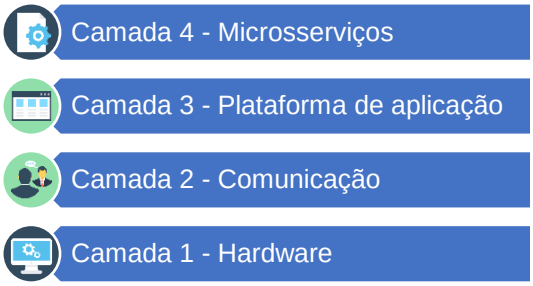
Na arquitetura de microsserviços, uma aplicação é composta por várias unidades de softwares autônomas, onde cada uma pode ser independentemente implantada e escalada. As estruturas e linguagens de programação podem ser diferentes para a criação e implantação de cada microsserviço, pois executam as tarefas de modo independente uns dos outros (AWS, 2017). Porém, a arquitetura de microsserviços também apresenta algumas desvantagens em relação a de sistemas monolíticos: é mais trabalhoso montar a estrutura inicial, exige planejamento relativamente mais elaborado e requer montagem de contêineres.

Apesar da implementação de “serviços”, a abordagem de microsserviços é diferente da arquitetura orientada a serviços, conhecida como SOA (*Service-Oriented Architecture*), embora possuam algumas similaridades. Um sistema segundo a arquitetura SOA é composto por componentes distintos, que fornecem serviços a outros componentes por meio de um protocolo de comunicação em uma rede (ex.: protocolo IP) (DESPODOVSKI, 2017). Essa comunicação pode envolver a passagem de dados simples ou serviços que coordenam a conexão entre os componentes, usando tecnologias e padrões que facilitam a cooperação entre os componentes. Em uma aplicação SOA devem ser bem definidos os papeis de provedores e consumidores de serviços. Exemplos de serviços em uma aplicação SOA: serviços de validação de pagamento, criação de conta de usuário, entre outros (DESPODOVSKI, 2017).

Na implementação da arquitetura de microsserviços, uma aplicação também é dividida em serviços, que podem estar distribuídos em diferentes servidores. Entretanto, cada serviço possui uma responsabilidade bem definida (funcionalidade específica) e é executado em processo isolado dos demais (independentes). Cada unidade de serviço (microsserviço) pode trocar informações com outras unidades, usando, por exemplo, Web Services escritos no padrão REST (DEVMEDIA, 2018). Ueda *et al.* (2016) destaca que os microsserviços não usam um barramento de comunicação comum, como em SOA (ex.: ESB - *Enterprise Service Bus*), pois se comunicam diretamente uns com os outros. Para o desenvolvimento de microsserviços, geralmente se requer um sistema de conteinerização (ex.: Docker) para criar um serviço Web, o que não é necessário em SOA.

Os microsserviços necessitam de um ambiente no qual possam ser construídos, integrados e executados, denominado “ecossistema”. Para Fowler (2017), esse ecossistema é composto por quatro camadas (**Figura 16**): (1) Hardware; (2) Comunicação; (3) Plataforma de aplicação; (4) Microsserviços.

Figura 16 – Modelo de quatro camadas do ecossistema de microsserviços.



Fonte: adaptado de Fowler (2017).

A primeira camada (“Hardware”) contém os equipamentos onde são executados os microsserviços e as ferramentas necessárias para a execução. Essa camada pode conter muitos servidores, que podem ser virtuais ou físicos, próprios ou alugados. Como exemplo, podem ser mencionados os provedores de serviços em nuvem, como: *AWS*/EC2, *Google Cloud Platatform* e *Microsoft Azure*. Nessa camada, também são definidos os seguintes recursos: (1) sistema operacional, que geralmente é *Linux*; (2) ferramentas de apoio para isolar a aplicação do sistema operacional (ex.: *Docker*, *Mesos* e *Kubernetes*); (3) sistemas de gerenciamento e configuração (ex.: *Ansible*, *Chef* e *Puppet*); (4) sistemas de monitoramento e de registros históricos (*logging*) em nível de servidor (ex.: *Nagios*).

A segunda camada (“Comunicação”) faz a ligação entre as outras camadas do ecossistema. É composta por elementos de rede, sistemas de DNS (*Domain Name System*), pontos de comunicação de acesso a uma aplicação (*endpoints*) e serviços de comunicação

entre os microsserviços. Esses serviços de comunicação usam um protocolo específico para que o microsserviço possa enviar, pela rede, alguns dados em formato padronizado para outro serviço ou microsserviço. Segundo Fowler (2017), o método de comunicação mais utilizado entre microsserviços é o HTTP com o estilo de arquitetura de software REST (*Representation State Transfer*). Comumente, os dados são formatados e enviados no formato JSON (*JavaScript Object Notation*). O software *Apache Kafka* é um exemplo de tecnologia utilizada para troca de mensagens entre microsserviços.

A terceira camada (“Plataforma de aplicação”) representa todas as ferramentas e serviços necessários para o correto funcionamento do ecossistema dos microsserviços. De maneira independente, essas ferramentas internas devem ser de “autosserviço”, ou seja, são configuradas na primeira vez e mantidas; os desenvolvedores, então, se preocupem somente com a manutenção dos microsserviços em si e não mais com configurações/atualizações dessas ferramentas. Essa camada também contempla: (1) ambiente de desenvolvimento; (2) ferramentas de testes, empacotamento compilação e liberação; (3) entrega incremental do software (*pipeline deployment*); (3) registros das solicitações do microsserviço (*logging*); (4) monitoramento dos microsserviços.

A quarta camada (“Microsserviços”) é responsável por abrigar os microsserviços e todas as configurações específicas para que eles funcionem corretamente. Desse modo, a arquitetura de microsserviços atua como um sistema distribuído, garantindo o isolamento de falhas, escalabilidade, desempenho e transparência para cada microsserviço (SILVA, 2015).

4.2 ESPECIFICAÇÃO DE MICROSSERVIÇOS

Na literatura, a busca por técnicas e documentos de especificação de microsserviços não foi muito promissora. Porém, foi encontrado um modelo de especificação definido por Fowler (2017), que contribuiu muito para este trabalho. Também foram identificadas algumas ferramentas para a documentação de APIs, como: *Swagger*, *API Blueprint*, *RAML*, entre outras (VAHL, 2015). A documentação de APIs consiste na implementação de um código, gerado de modo automático, com arquivos nos formatos JSON, YAML ou XML. Esses arquivos contemplam uma lista de *endpoints* utilizados e os parâmetros necessários para a correta integração do sistema (VAHL, 2015).

O documento de especificação de requisitos para os microsserviços deve seguir um formato padronizado, de modo claro e bem definido. Apesar da norma ISO/IEC/IEEE 29148:2018 também abordar a documentação dos requisitos de software, o modelo definido

por Fowler (2017) mostrou-se mais específico e, por isso, mais simples e adequado para servir de base para este trabalho.

Segundo Fowler (2017), a documentação para elaboração de um microsserviço deve ser abrangente, mas escrita de forma clara e objetiva, contemplando os seguintes elementos:

a) **Descrição** - é uma definição curta de como deve ser o microsserviço. Por exemplo: “após o cliente efetuar um pedido, o microsserviço “*enviar-recibo*” deve enviar um recibo para o endereço de e-mail do cliente”;

b) **Diagrama de arquitetura** - é um diagrama que inclui, de maneira gráfica, informações detalhadas da arquitetura do microsserviço, como, por exemplo: componentes, fluxos de solicitações, dependências, banco de dados, etc.;

c) **Informações de contato e plantão** - são informações sobre a equipe de desenvolvimento, como, por exemplo: nome, cargo, dados de contato (telefone, endereço de *e-mail, etc.*) e turno de plantão. Dessa forma, é possível entrar em contato com a pessoa responsável pela elaboração do microsserviço (em caso de problemas ou dúvidas) ou com quem está no plantão (casos de emergências);

d) ***Links*** - são os endereços eletrônicos dos repositórios do microsserviço. Por exemplo: *links* de documentos técnicos com comentários sobre o código, *links* de revisões técnicas, *links* de outros microsserviços ou de tecnologias usadas;

e) **Guia de bordo e desenvolvimento** - é um guia de como deve ser elaborado, configurado e mantido o microsserviço, sendo usado principalmente para integração de novos membros à equipe desenvolvedora. Esse guia pode ser dividido em duas partes. A primeira destinada às configurações, com informações sobre como o desenvolvedor deve conferir o código, configurar o ambiente, iniciar e verificar o correto funcionamento do serviço. Na segunda parte, o guia deve conter detalhes técnicos, como os comandos a serem executados e de que forma deve ser a conferência, revisão, alteração, inclusão e implantação do código;

f) **Fluxos de solicitações, *endpoints* e dependências** - a documentação do fluxo de solicitações pode conter um diagrama dos fluxos de solicitações da aplicação, como, por exemplo, um diagrama de arquitetura, representando também a descrição dos tipos de solicitações a serem realizadas para o microsserviço. Os pontos de comunicação (*endpoints*) para um serviço do usuário, em um servidor, devem ser documentados por meio de uma lista, contendo: nome, descrição e resposta. Para casos de falhas, devem ser registradas as informações sobre dependências de serviços, solicitações entre serviços e informações sobre a garantia dos níveis de qualidade (SLA – *Service Level Agreemment*);

g) **Roteiros de plantão** - descrição passo a passo de soluções de contingenciamento, em caso de alguma falha (“alertas”), incluindo como devem ser selecionadas e o problema resolvido ou mitigado. Cada alerta deve conter: nome, descrição da solução, descrição do problema e um guia passo a passo do que deve ser executado;

h) **FAQ** (*Frequently Asked Questions*) - é uma seção com uma lista de perguntas mais frequentes sobre o serviço, de modo a responder questões das equipes de desenvolvimento e de plantão.

Assim, a especificação de um microsserviço pode ser documentada usando-se esse modelo de requisitos proposto por Fowler (2017), juntamente com a documentação gerada automaticamente por ferramentas de desenvolvimento de APIs (ex.: *Swagger*).

4.3 BOAS PRÁTICAS PARA MICROSSERVIÇOS

Equipes da plataforma em nuvem Heroku desenvolveram um manifesto denominado “*The Twelve-Factor App*”[1], que apresenta uma metodologia contemplando doze fatores importantes para o desenvolvimento de softwares-como-serviços, que também se aplica aos microsserviços (SOUZA, 2018). Esses doze fatores podem ser considerados boas práticas a serem seguidas e podem ser descritos, brevemente, como segue (WIGGINS, 2017):

1) **Base de código** - contém a lista de repositórios dos códigos da aplicação;

2) **Dependências** - contém a lista de declarações de todas as dependências, de maneira completa e exatamente como utilizada no código;

3) **Configuração** - contém uma lista de configuração das tecnologias utilizadas, como, por exemplo, recursos para a base de dados, credenciais para serviços externos e quantidades de *deploy* (implantação), como o nome do *host*;

4) **Serviços de apoio** - contém a listagem de serviços de apoio que uma aplicação pode consumir como parte de sua operação normal via rede. Por exemplo: serviços de armazenamento de dados, sistemas de mensagens/filas, serviços SMTP para *e-mails* externos e recurso de *cache*;

5) ***“Build, release, run”*** **(“construa, lance, execute”)** - são as etapas de

[1] *https://12factor.net/pt_br/*

desenvolvimento da aplicação. Cada estágio deve ser bem definido e finalizado. A etapa de **construção** converte um repositório de código em um pacote executável. O estágio de **lançamento** é realizado logo após a etapa de construção e conjuntamente com a atual configuração da implantação. A aplicação está pronta para ser executada imediatamente no ambiente de execução. A etapa de **execução** permite o funcionamento da aplicação com o lançamento de versões;

6) **Processos** - é a execução da aplicação com um ou mais processos que não armazenam estado;

7) **Vínculo de portas** - é a exportação dos serviços que utilizam vínculos com portas que são executadas dentro de contêineres de um servidor Web. Exemplo: aplicações em linguagem PHP que podem utilizar solicitações HTTP;

8) **Concorrência** - é a execução da aplicação em processos. Exemplo: solicitações HTTP podem ser manipuladas para um ou mais processos Web.

9) **Descartabilidade** - os processos podem ser iniciados ou parados a qualquer momento. Desse modo, facilita escalonamento elástico, rápida implantação do código ou mudanças de configuração;

10) **Paridade entre desenvolvimento e produção** - no desenvolvimento da aplicação, a implantação deve ser contínua, deixando uma lacuna pequena entre desenvolvimento (edição de código de um *deploy* local) e produção (um deploy sendo acessado por usuários finais). Por exemplo: o tempo de implantação deve demorar horas em vez de semanas, as pessoas que desenvolvem devem ser as mesmas que implantam e o ambiente de desenvolvimento deve ser o mais similar possível ao ambiente de produção;

11) ***Logs*** - são arquivos que contém informações da execução de uma aplicação. Os *logs* devem ser tratados como fluxos de evento agregados e ordenados por tempo. Devem ser coletados dos fluxos de saída de todos os processos em execução e serviços de apoio;

12) **Processos administrativos** - é a execução de tarefas de administração/gestão em processos pontuais. Exemplo: os desenvolvedores podem executar comandos específicos, de maneira remota, fornecidos pelo ambiente de execução do *deploy* para um determinado processo.

Para Aderaldo *et al.* (2017), o desenvolvimento de aplicações baseadas na arquitetura de microsserviços devem estar em conformidade com boas práticas de desenvolvimento de software, como, por exemplo, utilização de metodologias ágeis e abordagem DevOps. Além disso, deve-se ter meios de avaliar o desempenho dessas aplicações (*benchmarks*).

Assim, Aderaldo *et al.* (2017) propõem uma lista de doze requisitos para serem considerados como boas práticas para desenvolvimento de microsserviços (**Tabela 2**).

Tabela 2 – Categorias e seus requisitos para benchmark de microsserviços.

Categorias	Requisitos	Descrição
Arquitetural	R1 – Visão explícita de topologia	É elaborada uma descrição explícita de seus principais elementos de serviço e suas possíveis topologias de tempo de execução.
	R2 – Arquitetura baseada em padrões	Deve ser projetada com base em padrões arquitetônicos de microsserviços conhecidos.
DevOps	R3 – Facilidade de acesso a partir de um repositório de controle de versão	O repositório de software deve ser facilmente acessível a partir de um sistema de controle de versão pública.
	R4 – Suporte para integração contínua	Deve fornecer suporte para pelo menos uma ferramenta de integração contínua.
	R5 – Suporte para testes automatizados	Deve fornecer suporte para pelo menos uma ferramenta de teste automatizado.
	R6 - Suporte para Gerenciamento de Dependência	Deve fornecer suporte para pelo menos uma ferramenta de gerenciamento de dependências.
	R7 - Suporte para imagens de contêiner reutilizáveis	Deve fornecer imagens de contêineres reutilizáveis para pelo menos uma tecnologia de contêineres.
	R8 - Suporte para implantação automatizada	Deve fornecer suporte para pelo menos uma ferramenta de implantação automatizada.
	R9 – Suporte a orquestração de contêineres	Deve fornecer suporte para pelo menos uma ferramenta de orquestração de contêineres.
Geral	R10 - Independência da tecnologia de automação	Deve fornecer suporte para várias alternativas tecnológicas em cada nível de automação das atividades de DevOps.
	R11 - Versões alternativas	Deve fornecer implementações alternativas em termos de linguagens de programação e / ou decisões de arquitetura.
	R12 - Uso e interesse da comunidade	Deve ser fácil de usar e atrair o interesse de sua comunidade.

Fonte: adaptado de Aderaldo *et al.* (2017).

Os requisitos propostos por Aderaldo *et al.* (2017) são classificados em três categorias:

1) **Arquitetural** – contempla atributos desejáveis na perspectiva da arquitetura de microsserviços;
2) **DevOps** – reflete o desejo da indústria de software no desenvolvimento de aplicações baseadas em microsserviços, de acordo com a abordagem DevOps;
3) **Geral** – contempla atributos gerais de referência e que não são obrigatórios do ponto de vista técnico, mas que são de grande valor para a comunidade de pesquisa de software.

Para Ueda et al. (2016), uma boa prática para desenvolvimento de microsserviços é compreender como medir e balancear a carga de trabalho da aplicação, provendo otimizações sempre que possível. Um dos critérios adotados por Ueda *et al.* (2016) é medir o tempo de resposta das aplicações.

Em outra direção, Alshuqayran *et al.* (2016) observam que um único requisito funcional de negócios pode resultar na orquestração de várias chamadas de serviço juntas, introduzindo, de certa forma, um atraso adicional na experiência do usuário final. Assim, é necessário criar recursos de sincronização de dados para essas chamadas. Isso ajuda a evitar sobrecargas de comunicação causadas pela cópia de dados, que ocorre durante as chamadas de serviço. Os autores propõem que, em alguns casos, as chamadas de serviços sejam feitas por uma única requisição de chamada para facilitar a sincronização dos dados, e, assim, melhorar o desempenho dos microsserviços.

4.4 TRABALHOS RELACIONADOS

Na literatura, podem ser encontrados trabalhos que relacionam arquitetura de microsserviços a modelos de processos de negócio, embora não tenha sido encontrado nenhum que apresente um processo sistematizado para a especificação e desenvolvimento de microsserviços. Nesta seção, são apresentados cinco desses trabalhos, considerados mais relevantes aos propósitos deste trabalho: Alpers *et al.* (2015), Silva (2015), Ueda *et al.* (2016), Geisriegler *et al.* (2017) e Alejandro (2017).

Alpers *et al.* (2015) usam arquitetura de microsserviços para implementar uma ferramenta de modelagem de processos de negócio, com base em Redes de Petri. Essa ferramenta pode ser personalizada para cenários específicos de usuário, projetos ou

propósitos. Segundo os autores, a ferramenta apresenta alto grau de flexibilidade, com suporte à modelagem de processos em BPMN e com funcionalidades de acordo com o tipo de carga de trabalho de cada usuário. Os autores informam que pretendem integrar à ferramenta serviços de importação de modelos de processos em BPMN.

Silva (2015) apresenta como foi o desenvolvimento de uma aplicação com arquitetura de microsserviços para o gerenciamento de ocorrências em órgãos de segurança pública e defesa civil. Para o autor, o sistema deve ser escalável e com alto desempenho. Para o desenvolvimento, foi utilizada uma interface pública, com o objetivo de garantir a integração e interoperabilidade com outros sistemas e plataformas, possibilitando o envio e recebimento de dados através de requisições Web. Para implementação dos microsserviços, o autor utilizou um *framework* com estilo arquitetural *REST*, com as dependências e sistema de contêiner já compilados. O sistema de contêineres foi usado para fornecer e gerenciar uma interface de redes com as dependências necessárias. Nesse contêiner, foi criada uma imagem contendo a base do ambiente de desenvolvimento para manipulação do arquivo de classes e execução do serviço. A comunicação entre os contêineres foi então configurada e os requisitos para tratamento e transferência de dados foram definidos, assim como os tipos de protocolos.

Ueda *et al.* (2016) apresentam técnicas de otimização para cargas de trabalho de microsserviços, em relação ao tempo de execução (código e hardware). Foram analisados o comportamento de duas linguagens de programação amplamente utilizadas: Node.js e Java. Em termos de “sobrecarga de trabalho”, uma aplicação desenvolvida com microsserviços obteve desempenho de 79,2% menor que a versão monolítica na mesma configuração de hardware. Em termos de “tempo de execução”, os autores constataram que tanto o código em Java como em Node.js consumiram muito tempo. O trabalho conclui que, embora microsserviços possam ser desenvolvidos rapidamente usando metodologias ágeis, há um impacto negativo no desempenho da aplicação quando o número de microsserviços aumentar. Não foram apresentados dados de pesquisa empírica sobre a diferença de desempenho entre arquiteturas monolíticas e de microsserviço.

Geisriegler *et al.* (2017) apresentam uma plataforma de modelagem e execução de processos de código aberto, independentemente de qualquer sistema operacional. Essa plataforma pode ser usada para modelar processos na notação S-BPM (orientada ao sujeito) e exportá-los para arquivos em OWL (*Web Ontology Language*). A plataforma é projetada como uma coleção de microsserviços, no domínio da gestão de processos de negócios. Os autores mostram que os modelos de processos de negócios convertidos em OWL podem ser executados no protótipo desenvolvido. Assim, a intenção é que o protótipo seja o ponto de partida para a definição de uma moderna arquitetura de referência para sistemas de BPM

(BPMSs), usando microsserviços para a execução dos processos de negócio. Convém lembrar que, na modelagem usando S-BPM, os processos de negócios são tratados como uma rede de atores distribuídos e independentes (humanos ou máquinas), que interagem por meio de troca de mensagens.

Em outra direção, Alejandro (2017) aplicou conceitos de entrega e integração contínuas em uma arquitetura de microsserviços, ou seja, integrou a abordagem DevOps com a arquitetura de microsserviços, usando a tecnologia de contêineres. A fase de “Integração Contínua” de DevOps abrange as etapas de codificação e construção dos microsserviços. A fase “Entrega e Implementação Contínua”, por sua vez, abrange a construção, testes e implantação dos microsserviços, com contêineres. Segundo o autor, pode-se utilizar a abordagem DevOps em diferentes ambientes de iterações, visando diminuir custos e aumentar a entrega de seus produtos.

4.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo, incialmente, foi apresentada uma visão geral sobre microsserviços, comparando a arquitetura de sistemas compostos por microsserviços com a arquitetura de sistemas monolíticos e de outros sistemas orientados a serviços, na abordagem de SOA. A *Seção 4.2* abordou aspectos importantes para a especificação e documentação de microsserviços enquanto a *Seção 4.3* apresentou boas práticas para o desenvolvimento de microsserviços.

Os trabalhos destacados na *Seção 4.4* contribuíram significativamente para os propósitos deste trabalho, tanto para o processo sistematizado como para o desenvolvimento das ferramentas de apoio. Entretanto, observou-se “lacunas” nas buscas na literatura que os resultados deste trabalho podem vir a “preencher”, com a sistematização de um processo para desenvolver sistemas dedicados à automatização de processos de negócio, como apresentado no próximo capítulo.

5 AUTOMAÇÃO DE MODELOS PROCESSOS DE NEGÓCIO USANDO MICROSSERVIÇOS

Este capítulo apresenta o processo sistematizado definido neste trabalho para apoiar o desenvolvimento de um BPMS dedicado a um determinado modelo de processos de negócio em BPMN v2.0, desenvolvido totalmente com a tecnologia de microsserviços. Esse sistema dedicado é identificado como BPMSm, onde “m” é a identificação do modelo de processos de negócio. Considerando que uma organização possui *n* modelos de negócio, a integração dos BPMSm´s constituem um BPMS global, identificado como BPMSg.

Neste trabalho, o objetivo é orientar, de modo sistemático, a implementação de um BPMSm. Para isso, uma sistemática para construção de microsserviços a partir de modelos de processos de negócio foi definida, de modo que facilite o processo da especificação de requisitos, projeto e implementação de cada microsserviço. Em termos de granularidade, cada atividade do modelo de negócios em BPMN v2.0 será automatizada por um microsserviço, devido à facilidade de mapeamento entre os modelos de negócio e do projeto de software (microsserviço), em conformidade com a teoria de microsserviços.

O modelo de processos de negócio, no entanto, deve ter sido construído levando em consideração boas práticas de modelagem, de modo a evitar interpretações errôneas. Os elementos gráficos devem ser documentados textualmente, de modo que seja possível extrair a especificação dos requisitos de cada microsserviço.

A intenção é que após construído um BPMSm, à medida que sejam necessárias alterações, inserções ou eliminação de microsserviços devido a mudanças no modelo de processos de negócio “m”, isso seja feito de maneira que o sistema continue em operação, com o mínimo de adequações, no contexto da Engenharia de Software Contínua. A granularidade fina considerada auxilia a alcançar esse objetivo.

Foram consideradas técnicas de desenvolvimento ágil com a abordagem DevOps para o desenvolvimento, integração e execução de microsserviços. Desse modo, a automação dos modelos de processos de negócio pode ser implementada em menor tempo, com entrega em ciclos curtos e com custos mais acessíveis.

As tecnologias utilizadas e destinadas à implementação dos microsserviços levaram em consideração sistemas e plataformas de domínio público bem conhecidos, de modo a

facilitar o desenvolvimento de um BPMSm em termos de tempo e custo. Neste trabalho, foi utilizado o ambiente em nuvem da plataforma *Google Firebase* com o interpretador *JavaScript Node.js.*

Frente ao exposto, a *Seção 5.1* apresenta as condições que um modelo de processos de negócio deve satisfazer para ser considerado adequado para constituir uma entrada do processo sistematizado proposto. Como já mencionado, é importante que o modelo tenha sido construído ou tenha sido ajustado a boas práticas de modelagem. Para este trabalho, são indicadas as boas práticas propostas por Figueiredo (2018) e por Nogueira e Oliveira (2017), uma vez que foram definidas após levantamentos de boas práticas de vários autores avaliados.

O processo sistematizado para desenvolvimento dos microsserviços que devem compor o BPMSm, assim como, os procedimentos de implementação e testes dos microsserviços de cada atividade do modelo de processos de negócio e o desenvolvimento do microsserviço de controle é detalhado na *Seção 5.2*. A definição do modelo de especificação de requisitos para um modelo de processos de negócio é apresentado na *Seção 5.3*. Já na *Seção 5.4* é apresentada a ferramenta MservSpec (*Microservice Requirements Specification*) para especificação dos requisitos para microsserviços. As tecnologias utilizadas e recomendadas para a implementação dos microsserviços, por sua vez, estão apresentadas na *Seção 5.5*. Para avaliar o processo sistematizado, foram realizados experimentos para prova de conceito, apresentados no próximo capítulo. Contudo, alguns aspectos contemplados na avaliação estão apresentados na *Seção 5.6*.

5.1 CONDIÇÕES NECESSÁRIAS AOS MODELOS DE PROCESSOS DE NEGÓCIO

Considerando que menos de 20% vocabulário de BPMN é utilizado em mais de 50% dos modelos de processos de negócio, segundo Borger (2012), os modelos de processos de negócio, para este trabalho, devem contemplar as categorias de símbolos mais utilizadas em modelos completos. Os símbolos dessas categorias são considerados suficientes para mapear os processos de negócio reais em grande parte dos casos: atividades, subprocessos, eventos, *gateways* (exclusivos, inclusivos e paralelos), artefatos (grupos e anotações), objetos de dados, depósitos de dados, atributos estendidos, piscinas (*pools*), raias (*lanes*), fluxos de sequência e fluxos de mensagem.

É de extrema relevância que os elementos gráficos do modelo sejam associados à documentação textual. Essa documentação pode seguir o formato do sistema de modelagem

BPMN utilizado, uma vez que não há uma forma padronizada para se estruturar a informação textual dos elementos de BPMN.

Para que o processo sistematizado proposto funcione de maneira satisfatória, o modelo de processos de negócio deve ter sido modelado usando a notação BPMN v2.0, de modo satisfazer os doze critérios definidos por Figueiredo (2018). Observa-se que os critérios de Figueiredo (2018) foram definidos, segundo o autor, levando em consideração também algumas heurísticas propostas por Nogueira e Oliveira (2017).

É importante observar que, caso um dos critérios ou heurística não seja satisfeito, o modelo de processos de negócio deve ser ajustado para satisfazer o critério/heurística. Isso pode ser feito pela própria equipe de modelagem, trazendo ganhos não somente à equipe de TI, para desenvolvimento dos microsserviços, mas também ao próprio ambiente do negócio. Entretanto, alguns ajustes relacionados ao tipo de algumas atividades e à documentação de alguns elementos podem ser realizados com o apoio da equipe de TI, se julgado conveniente.

Tantos os critérios mencionados como as heurísticas consideradas proporcionam melhorias na documentação e na parte gráfica dos modelos de processos de negócio. A aplicação desses critérios e heurísticas não implica em alteração nos fluxos dos modelos de processos de negócio, nem nos fluxos de sequência e de mensagens.

A seguir, são apresentados os doze critérios apresentados por Figueiredo (2018):

C1 - todo processo deve possuir um único evento de início e pelo menos um de fim, com nomes “Início” e “Fim”, respectivamente;

C2 - um processo deve conter pelo menos um participante;

C3 - a documentação textual de cada elemento de BPMN deve conter, no mínimo, uma de suas propriedades descritivas básicas: nome, descrição ou documentação;

C4 - o nome de uma atividade deve utilizar um verbo no infinitivo;

C5 - o tipo de uma atividade deve ser sempre definido;

C6 - elementos do mesmo tipo não devem possuir o mesmo nome;

C7 - uma atividade deve ser executada por pelo menos um ator;

C8 - as ações das atividades não devem ser atribuídas como nomes dos *gateways* (representam a lógica de direcionamento após a tomada de uma decisão);

C9 - os fluxos de sequência de um *gateway* exclusivo devem ser nomeados, com exceção do caso em que só há dois fluxos representando “Sim” ou “Não”. Nesse caso, apenas um dos nomes pode ser explícito e o outro poderá ser deduzido;

C10 - os fluxos de sequência de um *gateway* inclusivo devem ser nomeados;

C11 - atividades que são precedidas por um *gateway* paralelo devem ser unidas no final por outro *gateway* do tipo paralelo;

C12 - atividades que são precedidas por um *gateway* inclusivo devem ser unidas ao final por outro *gateway* do tipo inclusivo.

Como já mencionado, alguns desses critérios são baseados em algumas heurísticas de negócio definidas por Nogueira e Oliveira (2017). Por exemplo, o critério 6 (C6), segundo Figueiredo (2018), foi baseado na heurística de negócio identificada por HNa3, que é uma heurística para atividades em BPMN (item *What*), com a seguinte definição: “o nome da atividade deve ser único no modelo”.

Nogueira e Oliveira (2017) definiram 42 heurísticas denominadas “heurísticas de negócio” (HN). Para este trabalho, os modelos de processos de negócio devem satisfazer também cinco dessas heurísticas, que não foram consideradas nos critérios de Figueiredo (2018). Essas heurísticas de negócio estão apresentadas na **Tabela 3** e se referem aos eventos (HNe), às atividades (HNa), aos fluxos de decisão (*gateways*) (HNg) e aos artefatos e repositórios de dados (HNd). Essas heurísticas contribuem para a identificação de relações entre elementos dos modelos, bem como à descrição textual desses elementos.

Tabela 3 – Heurísticas de negócio utilizadas neste trabalho.

Heurística	Tipo	Descrição
HNa1	Heurística para atividades (item *Who*)	Cada atividade deve possuir a informação de seu executor.
HNa3	Heurística para atividades (item *What*)	O nome da atividade deve ser único no modelo.
HNe5	Heurística para eventos (item *When*)	A informação sobre o início do evento deve ser validada.
HNd2	Heurísticas para artefatos e repositório de dados (item *Who*)	O repositório de dados e os artefatos devem ser relacionados às atividades que manipulam suas informações.
HNd7	Heurísticas para artefatos e repositório de dados (item *When*)	O repositório de dados e os artefatos devem ser relacionados às atividades que manipulam suas informações.

Fonte: adaptado de Nogueira e Oliveira (2017).

A numeração das heurísticas de negócio na **Tabela 3** mantém a mesma numeração encontrada no trabalho de Nogueira e Oliveira (2017). Para cada categoria de HN, os autores

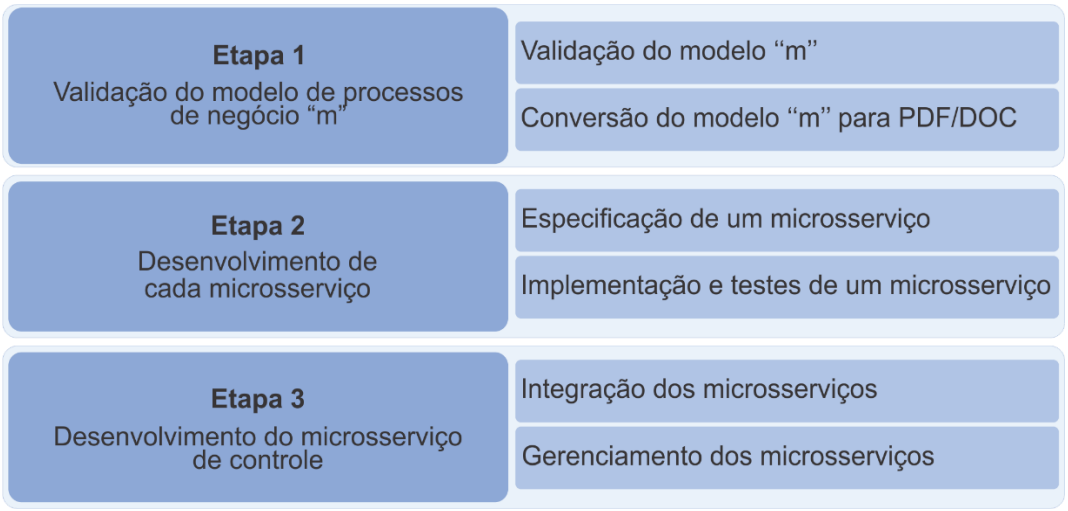
especificaram as informações segundo a técnica gerencial conhecida como 5W1H: *Who* (Quem), *What* (O quê), *Where* (Onde), *When* (Quando), *Why* (Por que), *How* (Como).

5.2 PROCESSO PARA DESENVOLVIMENTO DOS MICROSSERVIÇOS

Considerando um modelo de processos de negócio “m”, esta seção apresenta o processo sistematizado, definido neste trabalho, para desenvolvimento de cada microsserviço correspondente a uma atividade do modelo “m”. Esses microsserviços comporão o núcleo de um BPMS dedicado ao modelo, aqui denominado de BPMSm. O processo é composto por três etapas, como ilustrado na **Figura 17.**

Considere um modelo de processo de negócio “m”. Na **Etapa1**, é feita a validação do modelo de processos de negócio e de sua respectiva documentação. Caso o modelo não satisfaça as condições as condições definidas na *Seção 5.1*, ele deve ser ajustado. A **Etapa 2** se destina ao desenvolvimento de um microsserviço correspondente a uma atividade do modelo “m”, contemplando especificação dos requisitos, implementação e testes. Na **Etapa 3**, é desenvolvido um microsserviço de controle do BPMSm, que integra e gerencia a execução dos microsserviços desenvolvidos na etapa anterior. As Etapas 1, 2 e 3 estão apresentadas nas *Subseções 5.2.1, 5.2.2 e 5.2.3*

Figura 17 – Etapas do processo sistematizado para desenvolvimento de microsserviços.



Fonte: elaborado pelo autor.

5.2.1 Etapa 1 – Validação do modelo e da respectiva documentação

Considere o modelo de processos de negócio “m” em BPMN v2.0. Então, executar as fases E1.1 e E1.2, nessa ordem.

E1.1 – Validação manual do modelo “m”, considerando como “condição” cada um dos 12 critérios e as 5 heurísticas de negócio apresentadas na *Seção 5.1*, onde COND = 17 (12+5).

(a) Para cada condição ‘i” (i = 1..COND), executar os passos a seguir, primeiramente no modelo principal e depois em cada subprocesso, onde a leitura de cada parte do modelo deve ser feita da esquerda para a direita:

(a.1) Se a condição i não for satisfeita, ajustar o modelo “m” de modo a satisfazer a condição i, com o apoio da equipe de negócio, se necessário.

(a.2) Nomear, segundo padrão próprio do modelador, a nova versão do modelo “m”, a qual será a entrada para a próxima iteração.

(b) A versão final do modelo “m” ajustado completamente (com todos seus subprocessos já ajustados) será considerada como “mf” para as próximas fases e etapas.

E1.2 – Geração da documentação que servirá de base para a especificação de cada microsserviço.

(a) Exportar o arquivo da versão “mf” (referente ao modelo “m” validado) para um arquivo de formato textual com extensão “pdf” ou “doc”, usando um sistema de modelagem de processos de negócio com suporte à notação BPMN v2.0. Esse arquivo textual será aqui denominado como “DOCmf”.

(b) Exportar o arquivo da versão “mf” para um arquivo gráfico no formato PNG (*Portable Network Graphics*), usando um sistema de modelagem, caso esta imagem não venha junto da documentação do modelo publicado. Esse arquivo será aqui denominado como “PNGmf”.

5.2.2 Etapa 2 – Desenvolvimento de cada microsserviço

Considere:

- o modelo de processos de negócio “mf” em BPMN v2.0, bem como suas representações “DOCmf” e “PNGmf”, resultantes da Etapa 1;
- ATIV como sendo a quantidade total de atividades no modelo “mf”, onde ATIV é um número inteiro maior que zero;
- o modelo definido para a especificação de requisitos de cada microsserviço apresentado na *Seção 5.3*;
- a ferramenta MservSpec (*Seção 5.4*), desenvolvida para apoiar a geração do documento de Especificação de Requisitos de Software de cada microsserviço, onde cada microsserviço corresponde a uma atividade do modelo “mf” (ERSa).

Então, a equipe de TI deve executar as fases E2.1 e E2.2 nessa ordem.

E2.1 – Especificação dos Requisitos de todos os microsserviços.

(a) Ativar a ferramenta MservSpec.

(b) Na tela inicial:

(b.1) Digitar os seguintes dados sobre o modelo “mf” na tela inicial, obtidos do documento DOCmf: nome, versão do documento, data da criação, ferramenta de modelagem utilizada, autor e descrição).

(b.2) Fazer upload do “PNGmf” na mesma tela inicial.

(b.3) Salvar.

(c) Para cada atividade i (i = 1 .. ATIV) de DOCmf

(c.1) ativar uma nova atividade na ferramenta MservSpec.

(c.2) Preencher, manualmente, os seguintes campos referentes à atividade i a partir das informações sobre a atividade i expressas em DOCmf: *links*, *endpoints*, componentes, segurança, guia de bordo, contato de plantão, roteiro de plantão e FAQ.

(c.3) Salvar.

(d) Gerar o documento de especificação dos requisitos dos microsserviços, denominado ERSmf.

(e) Salvar o documento ERSmf na nuvem ou em formato PDF, utilizando a própria ferramenta MservSpec (disponível em nuvem, na plataforma *Firebase*).

É importante informar que o documento ERSmf contempla assim, a especificação de requisitos de todos os microsserviços. Cada parte dedicada à especificação de uma determinada atividade “a” será aqui identificada como ERSa.

E2.2 - Implementação de cada microsserviço

Considere que:

- o modelo “mf” tenha o processo principal e “n” subprocessos, onde n é um número inteiro maior ou igual a zero;
- “PROC” como sendo a soma do número de subprocessos mais o processo principal, ou seja: PROC = n + 1;
- tanto o processo principal como os subprocessos serão chamados “processos” nesta fase;
- para cada “processo” p, a quantidade total de atividades de p é identificada como ATIVp, onde ATIVp é um número inteiro maior que zero.

Quanto ao projeto e implementação dos microsserviços que comporão o BPMSmf considere que:

- são desenvolvidos na plataforma em nuvem *Google Firebase*, usando a linguagem de programação Node.js e demais tecnologias associadas, conforme apresentado na *Seção 5.5.* Logo, requer conhecimentos prévios de programação nesta plataforma;
- poderão fazer uso do Sistema de Banco de Dados (SBD) nativo da plataforma Firebase ou de qualquer outro SBD externo;
- deverão estar de acordo com o conteúdo do documento ERSmf e, consequentemente, das partes ERSa´s, bem como seguir o fluxo definido no diagrama (gráfico) do modelo mf;
- deverão estar de acordo com os seguintes critérios identificados como CDM (Critérios para desenvolvimento de um Microsserviço) enumerados de 1 a 4:

  CDM1 - Estilo arquitetural REST - a estrutura do código dos microsserviços deve contemplar os métodos básicos do estilo arquitetural REST, como, por exemplo, o conjunto de métodos HTTP (*get, put, post, delete*), etc;

  CDM2 - Categoria de usuários - para cada elemento “raia” do modelo “mf”,

deverá ser criada uma categoria de usuário com identificação de “login” e senha de acesso ao microsserviço (uma categoria de usuário pode representar mais de um usuário);

CDM3 - Usuário administrador - criar uma categoria de usuário “administrador”, para administrar os serviços dos demais usuários, com acesso a todos os recursos da aplicação;

CDM4 - Interface dos usuários - desenvolver uma interface para cada categoria de usuário, de modo a interagirem com o sistema, executando as funcionalidades por meio dos microsserviços.

Para cada processo “p” (p = 1 .. PROC) /*iteração para o processo p */

Para cada atividade “i” (i = 1 .. ATIVp) /* iteração para a atividade i */

(a) desenvolver um microsserviço utilizando os critérios CDM5 a CDM8:

CDM5 - Métodos básicos – cada microsserviço deve contemplar um método de entrada (*post*) e um método para o tratamento de dados (*get*), sendo que esse pode ou não ser de saída de dados;

CDM6 - Metadados - os métodos mencionados em CDM5 devem adicionar “metadados” que serão utilizados pelo microsserviço de controle, abordado na Etapa 3 desse processo sistematizado (*Subseção 5.2.3*). Exemplos desses metadados: data de atualização, data de inserção, usuário responsável pela ação, situação da ação, etc. O método “*post*” armazena esses metadados e os dados de entrada em um banco de dados;

CDM7 - Método path - se a atividade requerer um “caminho” ou destino de uma ação específica, o método “*path*” deve ser utilizado para adicionar os metadados para o microsserviço de controle. Para evitar a utilização de informações “antigas”, é necessário fazer a “invalidação” do *cache* do microsserviço. Isso mantém os dados sempre atualizados. Na **Figura 18**, é apresentado o algoritmo do método “*path*” específico esse caso;

Figura 18 – Algoritmo de atualização do *cache* com o método "*path*".

```
Seja D o dado atualizado recebido na entrada.
Adicionar metadados de controle a D.
Atualizar D no banco de dados do microsserviço.
Se atualizado com sucesso
então {enviar dados relevantes para o microsserviço referente à próxima atividade;
        invocar procedimento de invalidação de cache do microsserviço de controle;
            retornar sucesso.}
    senão retornar falha.
```

Fonte: elaborado pelo autor.

CDM8 – Lista de dados armazenados - possibilitar retornar todos os dados armazenados no banco de dados do microsserviço, usando o método "*get"*, de acordo com o algoritmo da **Figura 19**;

Figura 19 – Algoritmo para listagem de dados armazenados usando o método "*get*".

```
Iniciar L como uma lista vazia.
Para cada dado salvo no banco de dados que satisfaça o filtro:
 {adicionar dado à L.}
Retornar L.
```

Fonte: elaborado pelo autor.

(b) Se a atividade i for do tipo "Manual"

então {implementar o método "*post*" como genérico, segundo o algoritmo apresentado na **Figura 20**}.

Figura 20 – Algoritmo do método "*post*" para atividades manuais.

```
Seja D o dado recebido na entrada.
Adicionar metadados de controle a D.
Salvar D no banco de dados do microsserviço.
Se D salvo com sucesso
   então {chamar o procedimento de invalidação de cache do microsserviço de controle;
            retornar mensagem de "sucesso".}
   senão retornar mensagem de "falha".
```

Fonte: elaborado pelo autor.

(c) Se a atividade i for do tipo "*script*" ou "*serviços*"

então {implementar o método "*post*" de modo específico para essa atividade, segundo o algoritmo apresentado na **Figura 21**}.

Figura 21 – Algoritmo de envio de dados usando método "*post*" (atividades *script*/serviços).

```
Seja D o dado recebido na entrada.
Adicionar metadados de controle a D.
Salvar D no banco de dados do microsserviço.
Se salvo com sucesso
    então {enviar dados relevantes para o microsserviço referente à próxima atividade;
           invocar procedimento de invalidação de cache do microsserviço de controle;
           retornar sucesso.}
senão retornar falha.
```

Fonte: elaborado pelo autor.

(d) Identificar, no diagrama do modelo "mf", as atividades imediatamente subsequentes à atividade i no fluxo do processo, ou seja, sucessoras da atividade i. Essas atividades serão aqui denominadas SUCai.

(d.1) NUM(SUCai) recebe o valor correspondente à quantidade de atividades sucessoras da atividade i, onde NUM(SUCai) é um número inteiro maior ou igual a zero

(d.2) LIST corresponde a uma lista com NUM(SUCai) elementos, onde cada elemento corresponde à identificação de uma atividade SUCai.

(e) Para a atividade LIST(y), onde y = 0 .. NUM(SUCai)

(e.1) Desenvolver um microsserviço para a atividade LIST(y) como um "*Stub*", que corresponde a um pseudo-microsserviço, onde os métodos "*post*" e "*get*" não devem ser usados para realização de processamento adicional (exceção: adição de metadados úteis para o microsserviço de controle, como, por exemplo, data de criação, para controle)

i. Considerar a especificação $ERS_{LIST(y)}$, extraindo para uso apenas a especificação das entradas e das saídas do microsserviço "*Stub"*;

ii. Implementar uma interface simples com as entradas, de modo que o microsserviço para a atividade i faça a chamada (com ou sem envio de dados de entrada) para esse microsserviço "*Stub"* (usar o método "*post*");

iii. Implementar, de modo genérico, as saídas no microsserviço "*Stub"*; como "pseudo-saídas", somente para ver ajudar a implementação ser desenvolvida sempre com um nível à frente no fluxo do diagrama;

iv. Implementar o CDM6 (ver passo (a)).

/*fim da iteração para a atividade i */

/* fim da iteração para o processo p */

Cabe observar que, se em uma iteração para o desenvolvimento de um microsserviço para automatizar uma atividade "i", esse microsserviço foi anteriormente implementado como um "*Stub*", deve ser então substituído por um microsserviço completamente implementado, estendendo o modo com que as entradas e saídas (já testadas) foram implementadas.

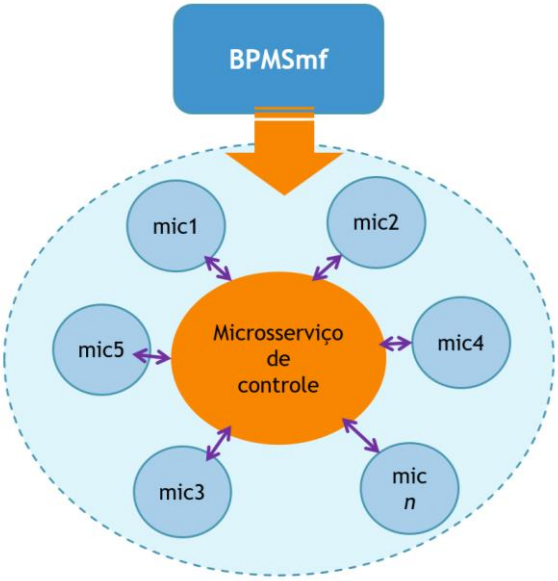
5.2.3 Etapa 3 – Desenvolvimento do microsserviço de controle

Considere que todos os microsserviços que compõem o BMPSmf tenham sido desenvolvidos até a Etapa 2, ou seja, tenham sido desenvolvidos "ATIV" microsserviços, onde ATIV é a quantidade total de atividades no modelo "mf" (ATIV > 0).

Então desenvolver um microsserviço de controle para o BPMSmf, de modo que ele integre e gerencie os microsserviços, em uma arquitetura "estrela", como apresentado na **Figura 22**. Isso é feito por meio de arquivos de "log", que mostram o registro de eventos, processos ou mensagens que ocorrem no sistema, identificando a data/hora, usuário e microsserviço de origem e de destino.

(a) Desenvolver a interface do usuário tipo "Administrador" para acesso ao microsserviço de controle, de modo a visualizar a carga de trabalho e dados que indiquem a sincronização das operações executadas nos microsserviços.

Figura 22 – Microsserviço de controle para administrar o BPMSmf.



Fonte: elaborado pelo autor.

(b) Desenvolver o microsserviço de controle usando todas as orientações cabíveis na Etapa 2.2 para um microsserviço de modo geral e considerar os seguintes passos, adicionalmente:

(b1) verificar continuamente quais microsserviços estão inativos e quais estão ativos (em operação), usando o método "*get*" para todos os microsserviços do BPMSmf,

(b2) manter o *cache* do microsserviço atualizado, de acordo com o algoritmo na **Figura 23**. Quando uma ação é realizada com um método (*post, get, etc*), o *cache* deve ser invalidado para que a aplicação sempre utilize informações atualizadas.

Figura 23 – Algoritmo do microsserviço de controle para atualização do cache.

```
Seja DATA um dicionário vazio.
Para cada microsserviço
  faça {
        se o cache deste microsserviço estiver atualizado
        então {adicionar dados do cache a DATA.}
        senão {fazer requisição ao microsserviço e salvar os dados em D;
                adicionar D a DATA;
                atualizar cache com D.}
       } Retornar data.
```

Fonte: elaborado pelo autor.

(b3) retornar um arquivo JSON com a quantidade de microsserviços do BPMSmf e valores dos metadados e certos dados em cada microsserviço, segundo algoritmo da **Figura 24**. A lista de dados contempla: nome, estado (*status)* atual, data, horário, o usuário que está acessando o microsserviço e usuário/atividade que receberá dados.

Figura 24 – Algoritmo para retornar informações sobre os microsserviços.

```
Seja LM uma lista vazia de microsserviços.
Iniciar LM
Obter lista de microsserviços dos quais quer gerar os dados.
Obter status do cache.
Caso cache esteja atualizado
então {obter dados do cache.
      senão {fazer requisição HTTP para o microsserviço correspondente;
            adicionar na lista de requisições.}
       }
Obter dados requisitados.
Retornar lista de objetos atualizados.
Atualizar todos os objetos da lista.
Retornar LM
 }
```

Fonte: elaborado pelo autor.

5.3 MODELO DE ESPECIFICAÇÃO DOS REQUISITOS

A especificação dos requisitos de cada microsserviço segue um modelo baseado na norma ISO/IEC/IEEE 29148:2018 (ver *Seção 4.2*), nas recomendações de requisitos para microsserviços de Fowler (2017) e na documentação para APIs gerada pela suíte de ferramentas *Swagger*. O modelo é composto por nove seções, descritas na **Tabela 4**.

Cada seção do modelo contempla uma lista de atributos, aos quais devem ser atribuídos valores, de modo a compor o documento de especificação de requisitos (ERS) de cada microsserviço, referente a uma atividade “a” automatizada pelo microsserviço. Como esse documento se destina à automatização das atividades de um determinado modelo “m”, neste trabalho ele é denominado ERSm, enquanto a parte dedicada à especificação de uma determinada atividade “a” é denominada ERSa.

Tabela 4 – Seções que compõem a especificação de requisitos de um microsserviço.

Seção	Nome/assunto abordado	Descrição
1	Cabeçalho e descrição	Informações sobre a origem do modelo de processos de negócio (nome e versão do arquivo XPDL, ferramenta de modelagem utilizada e data de geração do documento), com uma descrição e versão do microsserviço.
2	Links para repositório	Lista de links para repositórios para que os desenvolvedores possam conferir com facilidade o código.
3	Lista de solicitações e *endpoints* e dependências	Descrição qualitativa dos tipos de solicitações que são feitas para o microsserviço e de como são tratadas.
4	Componentes	Contém as tabelas do banco de dados usadas no microsserviço, com seus atributos e tipos.
5	Segurança dos dados	Especificação o recurso de segurança utilizado para tratar a segurança dos dados.
6	Informações de contato e plantão	Informações sobre quem desenvolveu o microsserviço no caso de contatar se houver problemas relativos a dependências.
7	Guia de bordo e desenvolvimento	Especificação sobre as tarefas do microsserviço passo-a-passo para novos membros da equipe a se inteirar do código e dos serviços executados
8	Roteiro de plantão	Contém alertas no roteiro de plantão para acompanhamento das instruções no caso de problemas
9	Faq	Lista de perguntas e respostas para sanar dúvidas da equipe de desenvolvimento com os próprios membros

Fonte: elaborado pelo autor.

A **Tabela 5** apresenta os atributos de cada uma das nove seções do modelo, onde os atributos marcados com asterisco (*) estão presentes na proposta de Fowler (2017). As demais seções foram inspiradas na documentação gerada pela suíte de ferramentas *Swagger*.

As seções de 1 a 6 da **Tabela 5** contêm informações mais direcionadas ao desenvolvimento da aplicação, onde as seções de 2 a 4 são pertinentes às atividades do modelo. As seções de 7 a 9 estão relacionados à manutenção do microsserviço. Por outro lado, as informações referentes ao modelo em si são as seções 1 e as seções 5 a 9.

Observa-se que a atribuição de valores aos elementos de cada seção pode contribuir para avaliar se a documentação da atividade considerada do modelo de processos de negócio está com documentação adequada. Podem ser feitas complementações de modo manual no texto de cada atributo de uma seção, caso a equipe de desenvolvimento ache necessário ou conveniente.

Tabela 5 – Modelo do documento de especificação de requisitos de um microsserviço.

Seção 1: **Cabeçalho e descrição** 1.1. Ferramenta de modelagem: *<nome da ferramenta utilizada>* 1.2. Data criação: *<data de criação do modelo BPMN>* 1.3. Autor: *<nome do modelador>* 1.4. Descrição*: *<texto descrito do microsserviço a ser desenvolvido>* 1.5. Versão do microsserviço: *<versão do microsserviço a ser desenvolvido>*
Seção 2: *Links* para repositório 2.1. Link: *<endereço eletrônico do repositório>* 2.2. Descrição do Link: *<descrição do link, sua funcionalidade>*
Seção 3: Lista de solicitações e *endpoints* e dependências 3.1. Lista*: *<lista contendo o nome do endpoint, descrição e respostas>* 3.2. Relação de Dependências*: *<nome e link das dependências>*
Seção 4: Componentes (esquema de dados, tabelas - data *store* do modelo em BPMN) 4.1. Nome: *<nome da tabela>* 4.2. Atributos: *<atributos com seus tipos de dados>* 4.3. Requerido: *<se o atributo é um campo obrigatório>*
Seção 5: Segurança dos dados 5.1. Tipo: *<tipo de segurança usada>* 5.2. Senha: *<senha de acesso>* 5.3. Link: *<link de conexão>* 5.4. Permissão*: <tipo de permissão – leitura, escrita>*
Seção 6: Informações de contato e plantão 6.1. Nome*: *<nome do desenvolvedor>* 6.2. Cargo*: *<cargo do desenvolvedor>* 6.3. Telefone*: *<telefone do desenvolvedor>* 6.4. Turno do Plantão*: *<turno do desenvolvedor>* 6.5. Horário*: *<horário de entrada e saída>*
Seção 7: Guia de bordo e desenvolvimento 7.1. Configuração do serviço e ambiente*: *<comandos de configuração do serviço e do ambiente>* 7.2. Inicialização e funcionamento*: *<comandos para iniciar e verificar o funcionamento>* 7.3. Ciclo de desenvolvimento*: *<códigos para acesso, inserção, alteração, execução, entrega e implantação>*
Seção 8: Roteiro de plantão 8.1. Requisitos e procedimentos gerais do plantão*: *<lista de alertas de serviços, contendo nome, descrição, problema encontrado, solução dos problemas, nível de gravidade, responsável>*
Seção 9: Faq 9.1. Pergunta*: *<nome da pergunta>* 9.2. Resposta*: *<texto descritivo da resposta>* 9.3. Autor*: *<pessoa que respondeu à pergunta>*

Fonte: elaborado pelo autor.

5.4 FERRAMENTA MSERVSPEC PARA A ESPECIFICAÇÃO DOS REQUISITOS

A ferramenta Web “MservSpec” (*Microservice Requirements Specification*) foi desenvolvida neste trabalho com o objetivo de armazenar, de forma organizada, a especificação de cada microsserviço do modelo de processos de negócio “mf”. É uma ferramenta dedicada à especificação de requisitos de microsserviços a partir de um modelo de processos de negócio em BPMN v2.0, que satisfaça as condições apresentadas na *Seção 5.1*. Isso a diferencia de qualquer outra ferramenta disponível, pois gera um documento ERS específico para microsserviços, contemplando as nove seções apresentadas na **Tabela 5** (*Seção 5.3*).

A ferramenta MservSpec foi desenvolvida com arquitetura de microsserviços e interface responsiva. Para desenvolvimento da interface da ferramenta, foram utilizadas as linguagens *JavaScript e HTML,* folhas de estilo CSS (C*ascading Style Sheets*) e *PureCSS*, plataforma *Firebase*. Para simplificar o desenvolvimento das interfaces, foram utilizados os frameworks *Vue.js* e *Bootstrap*; e a biblioteca *JQuery*. O APÊNDICE B traz o código e a aplicação MservSpec, desenvolvida em Node.js

Quando se utiliza o a ferramenta MservSpec, o modelo de processos de negócio fica armazenado em nuvem, assim como a especificação de cada atividade que o compõe. Isso possibilita a consulta dessas informações via Web, podendo ser armazenadas em arquivos digitais para acesso futuro ou impressão.

Como mostrado na **Figura 25**, a tela inicial é bem simples para o usuário (de TI), contendo apenas o respectivo logotipo e possibilidades de ativação das duas funcionalidades básicas: “Nova documentação” e “Ver documentações”.

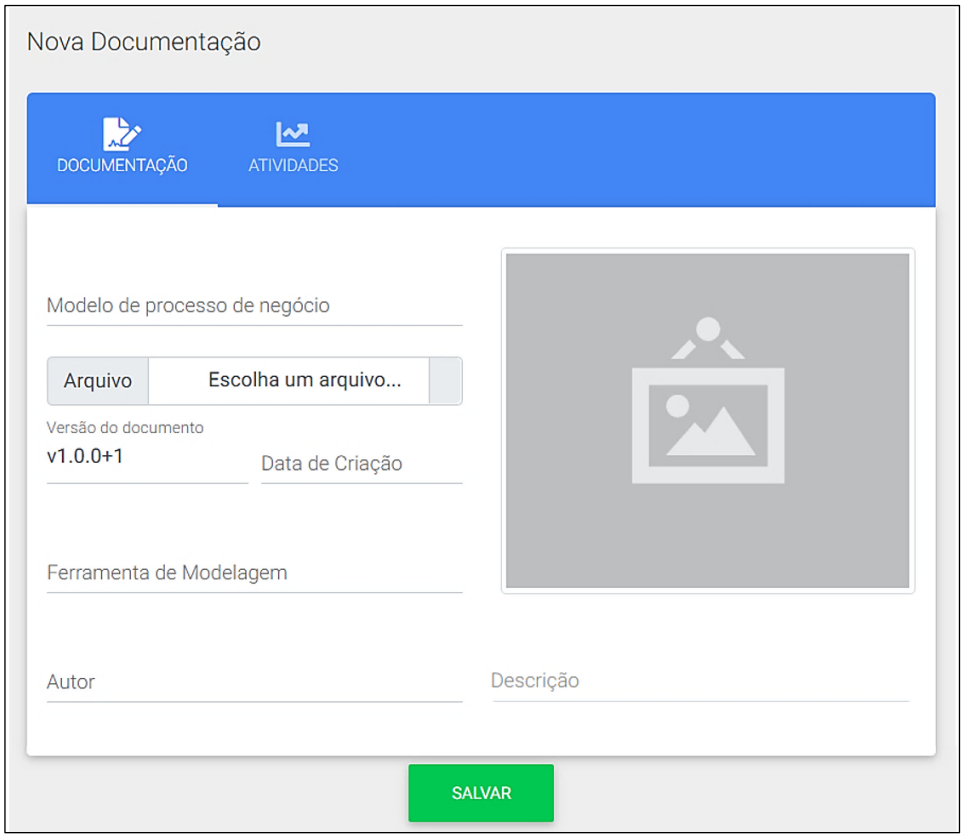
Figura 25 – Tela inicial da ferramenta Web MservSpec.

Fonte: elaborado pelo autor.

Quando ativada a funcionalidade “Nova documentação”, a tela apresenta dois itens

no menu superior, como mostrado na **Figura 26:** "Documentação" e "Atividades".

Figura 26 – Tela do item “Documentação”, referente ao modelo considerado.

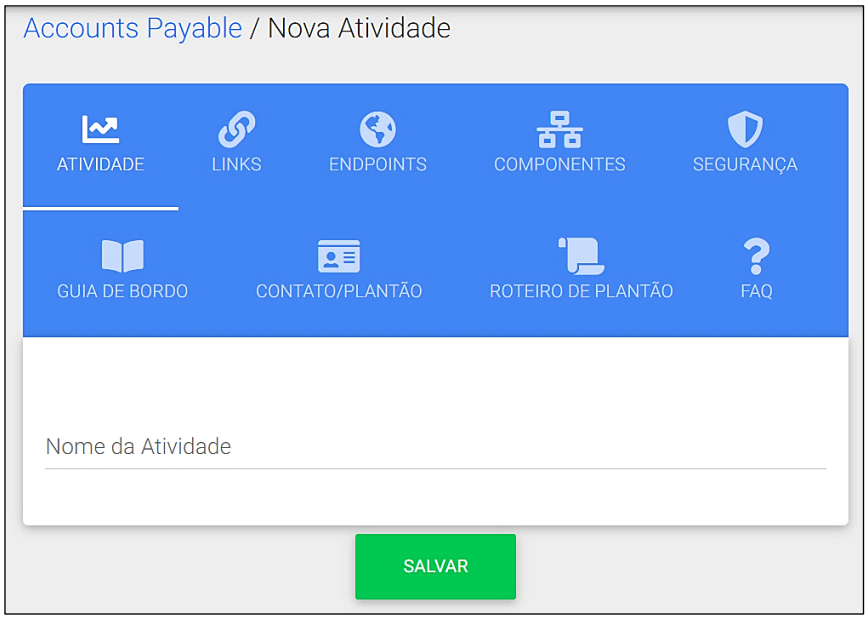


Fonte: elaborado pelo autor.

No item “Documentação”, é solicitado, ao usuário, o arquivo no formato PNG contemplando o modelo e o preenchimento de dados sobre ele: nome do modelo, versão, ferramenta de modelagem utilizada, autor da modelagem e texto descritivo sobre o modelo. No item “Atividades”, devem ser preenchidos campos com os seguintes dados: nome da atividade, *links*, *endpoints*, componentes, segurança, contatos do plantão, roteiros do plantão e perguntas e respostas. Ao final da inserção desses dados, o usuário deve salvá-los, usando comando do próprio sistema. Na **Figura 27**, é mostrada a tela do item “Atividades” para um modelo de processos de negócio denominado “*Accounts Payable*”, como exemplo.

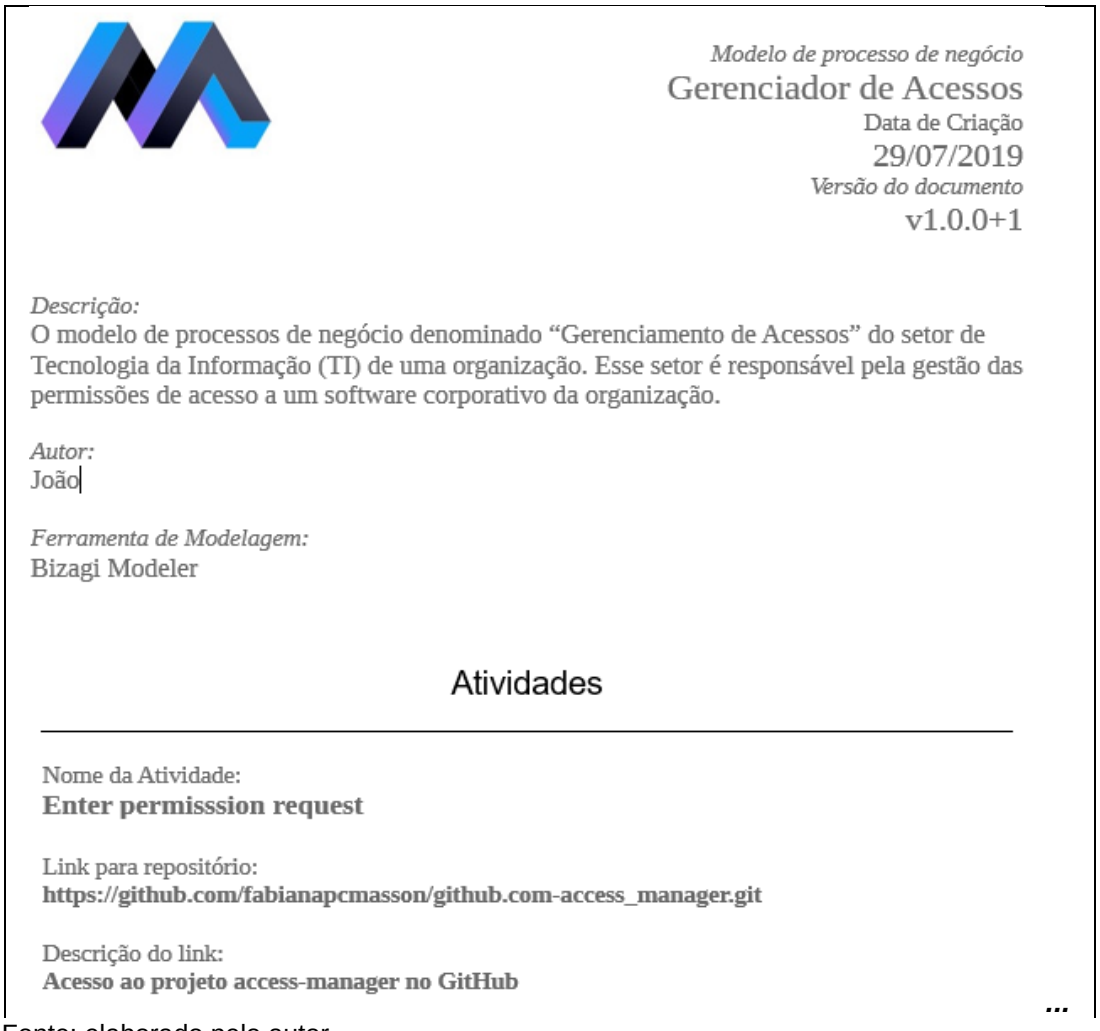
Assim, a ferramenta MservSpec oferece recursos para compor um documento com informações importantes para automatização de um modelo de processos de negócio “m”, compondo um documento denominado ERSm. Esse documento é composto por trechos denominados ERSa, contemplando a especificação de cada microsserviço que irá automatizar a atividade “a”, no formato do modelo de especificação de requisitos definido na *Seção 5.1*. Como exemplo, a **Figura 28** mostra a parte inicial do documento ERSm de um modelo de processos de negócio denominado “Gerenciador de Acessos”.

Figura 27 –Tela inicial do item “Atividades”, referente a um modelo denominado “*Accounts Payable*”.



Fonte: elaborado pelo autor.

Figura 28 – Parte inicial do arquivo que contempla o documento ERS de cada microsserviço.



Modelo de processo de negócio

Gerenciador de Acessos

Data de Criação

29/07/2019

Versão do documento

v1.0.0+1

Descrição:

O modelo de processos de negócio denominado “Gerenciamento de Acessos” do setor de Tecnologia da Informação (TI) de uma organização. Esse setor é responsável pela gestão das permissões de acesso a um software corporativo da organização.

Autor:

João

Ferramenta de Modelagem:

Bizagi Modeler

Atividades

Nome da Atividade:

Enter permisssion request

Link para repositório:

https://github.com/fabianapcmasson/github.com-access_manager.git

Descrição do link:

Acesso ao projeto access-manager no GitHub

...

Fonte: elaborado pelo autor.

5.5 TECNOLOGIAS RECOMENDADAS PARA O PROCESSO SISTEMATIZADO PROPOSTO

Considerando o processo sistematizado apresentado na *Seção 5.*1, para o desenvolvimento dos microsserviços que devem compor o BPMSm, visando a automatização do modelo de processos de negócio “m”, esta seção apresenta as tecnologias recomendadas e que foram utilizadas nos experimentos e nos casos para a prova de conceito apresentada no *Capítulo 6*.

Como este trabalho prezou pelo desenvolvimento do BPMSm em nuvem, a plataforma em nuvem deveria contemplar, gratuitamente, tecnologias abrangendo desenvolvimento de APIs bem definidas e conteinerização, além de possibilitar isolamento de bancos de dados, segundo a abordagem DevOps.

Dentre as tecnologias mencionadas na *Seção 3.3.2,* neste trabalho foi adotada a plataforma de desenvolvimento de aplicativos para dispositivos móveis e aplicações Web *Firebase*, do tipo Baas (*Backend as a Service*). A plataforma *Firebase* é mantida pela empresa Google, que oferece diversos planos de consumo de serviços, de acordo com a necessidade do usuário, integrados com a plataforma *Google Cloud* (AVRAM, 2016). Neste trabalho foi utilizado o plano de cotas gratuito.

A plataforma *Firebase* disponibiliza um *kit* para desenvolvimento de software (*SDK - Software Development Kit*), contemplando as seguintes linguagens de programação: C++, Java, Javascript, Node.js, Objective-C e Swift. Também é possível utilizar *frameworks*, como, por exemplo: Angular, *Backbone*, *React*, entre outros. Algumas bibliotecas disponíveis possibilitam auxiliar o desenvolvimento e a importação de grandes quantidades de dados no formato JSON (*JavaScript Object Notation*). A plataforma *Firebase* também permite a utilização do estilo arquitetural *REST*, para o desenvolvimento das funções em nuvem (FIREBASE, 2019).

No contexto deste trabalho, uma API define requisições do protocolo HTTP que podem ser solicitadas por um aplicativo cliente, para interagir com um serviço Web. A especificação das APIs na plataforma *Firebase* é feita a partir da especificação de *endpoints* e do formato de requisição. Convém lembrar que os *endpoints* (pontos de comunicação de uma aplicação) consistem em URLs (*Uniform Resource Locator*) que permitem a interação com o serviço a ser executado.

Serviços chamados *Cloud Functions* (funções em nuvem) implementam o conceito de contêineres na plataforma *Firebase*. Por meio desse tipo de serviço, é possível desenvolver

funções que tratam de um certo aspecto do sistema de maneira isolada, de modo que possam ser executadas em servidores virtuais distintos (FIREBASE, 2019).

Quando há uma carga maior sobre a aplicação, a plataforma *Firebase* aumenta, automaticamente, o número de servidores e desabilita os inativos, caso a carga diminua. Desse modo, essa plataforma consegue trabalhar de forma semelhante a tecnologias de orquestração de contêineres, como, por exemplo, *Docker Swarm* e *Kubernetes* (FIREBASE, 2019).

A linguagem utilizada para o desenvolvimento dos microsserviços neste trabalho é Node.js, bastante popular no desenvolvimento de serviços da Web, segundo Ueda *et al.* (2016). Foi desenvolvida por Ryan Dahl em 2009 e, atualmente, é mantida pela Fundação Node.js, em parceria com a Fundação Linux (DAYLEY, 2014). Node.js é uma linguagem interpretada, com código aberto, baseado na máquina virtual *JavaScript*, da empresa *Google*. Permite a implementação/execução de código de modo assíncrono e orientado a eventos. Cabe observar que, na implementação dos serviços com Node.js, pode-se utilizar bibliotecas de terceiros, com funções de retorno de chamada.

Os servidores de aplicativos baseados em Java executam suas tarefas uma por vez, não podendo processar novas solicitações dos clientes. No entanto, o modelo de processamento com Node.js possibilita a execução dessas tarefas de maneira singular. Logo, propicia desempenho melhor do que um servidor de aplicativos usando Java (UEDA *et al.*, 2016).

No desenvolvimento de uma aplicação, geralmente é necessário o uso de credenciais ou outras configurações para acesso a certos tipos de serviços (bancos de dados ou API's externas). Essas configurações não devem ser salvas juntas ao código que é compartilhado. Ao invés disso, é necessário usar alguma outra forma de armazená-las. Assim, a plataforma *Firebase* fornece o serviço de armazenamento de configuração de ambiente junto com as bibliotecas utilizadas em uma aplicação, ao invés de armazenar essas configurações em variáveis de ambiente do sistema operacional (FIREBASE, 2019).

As conexões com o banco de dados em uma aplicação Web que utiliza a plataforma *Firebase* é realizada por meio de um "*pool de conexões*". Trata-se de um espaço de *cache* para as conexões de banco de dados, mantido de forma que as conexões possam ser reutilizadas quando requisições futuras, ao banco de dados, forem requeridas (GUIMARÃES, 2019). A cada nova requisição, uma dessas conexões é retirada do "*pool de conexões*" e utilizada. Finalizada a requisição, a conexão volta para o "*pool de conexões*". Essa funcionalidade de reutilização das conexões fica permanentemente aberta e proporciona melhor desempenho e eficiência no uso de recursos da aplicação (FIREBASE, 2019).

A utilização de bancos de dados pela plataforma *Firebase* é feita por meio de uma biblioteca específica ou por uma biblioteca genérica com suporte ao banco de dados desejado. No caso do sistema de banco de dados PostgreSQL, é possível utilizar a biblioteca "*pg"*, específica desse sistema, ou uma biblioteca genérica como "*sequelize*.*js*", que permite o uso de vários bancos de dados, provenientes de vários sistemas, incluindo o PostgreSQL.

Para que um microsserviço, desenvolvido em *Cloud Functions,* utilize um banco de dados externo, é necessário que o banco de dados seja acessível por meio da Internet. Assim, o banco de dados pode ser acessado utilizando as mesmas técnicas que uma aplicação Node.js usaria sem a utilização da plataforma *Firebase*.

5.6 CONSIDERAÇÕES SOBRE A AVALIAÇÃO DO PROCESSO PROPOSTO

O desenvolvimento dos microsserviços contempla os critérios propostos por Wiggins (2017), Aderaldo *et al.* (2017), Ueda *et al.* (2016) e Alshuqayran *et al.* (2016).

Em relação aos doze fatores mencionados por Wiggins (2017), apresentados na *Seção 4.3*, 80% deles são considerados no desenvolvimento dos microsserviços neste trabalho: itens 1, 2, 3, 4, 5, 9 e 11. O item 1 recomenda um repositório para armazenar o código fonte da aplicação, sendo este, implementado na proposta deste trabalho por meio da ferramenta "Git" (controle de versões) e da plataforma "GitHub" (hospedagem de código-fonte). Já os itens 2, 3, 4 e 5 são implementados por recursos disponíveis na plataforma *Firebase*, o que facilita o controle da aplicação de maneira geral. O item 11, que se refere aos *logs* da aplicação, neste trabalho são usados com a finalidade de se obter o tempo de resposta que uma requisição a um serviço demora para executá-lo.

O processo de desenvolvimento dos microsserviços neste trabalho contempla mais de 60% dos doze requisitos de melhores práticas definidos por Aderaldo *et al.* (2017), apresentados na **Tabela 2** da *Seção 4.3*. Somente os requisitos R1, R6 e R8 não estão explicitamente definidos no desenvolvimento da aplicação composta por microsserviços.

Em relação ao desempenho da aplicação, foi adotado o uso de recurso de *cache* apresentado por Ueda *et al.* (2016) e os critérios de avaliação definidos por Alshuqayran *et al.* (2016): tempo de execução para resposta e para tratamento de requisições de chamadas. Convém destacar que poucas literaturas foram encontradas com critérios de desempenho para microsserviços visando a realização de *benchmarking*. Talvez seja devido à abordagem DevOps, com microsserviços, ser algo ainda "recente" e em evolução.

É importante mencionar que o motivo para essa avaliação se deve ao modo de

operação do microsserviço de controle. Durante o primeiro acionamento, o microsserviço de controle efetua as requisições e salva os dados recebidos em seu *cache*. Futuros acionamentos utilizarão os dados salvos em *cache*, se estiverem válidos. Caso os dados do *cache* não sejam mais válidos, eles serão atualizados através de novas requisições aos microsserviços correspondentes. Isso poderia ter um impacto significativo no desempenho do sistema, com alto tráfego de dados e tempo relativamente longo para responder as requisições. Assim, para mitigar esses tipos de problemas, a solução foi utilizar o recurso de *cache*, que armazena os dados necessários no microsserviço de controle.

6 AVALIAÇÃO DO PROCESSO SISTEMÁTICO PROPOSTO

Neste capítulo, são apresentados três casos práticos, selecionados entre os experimentos realizados, como prova de conceito do processo sistematizado apresentado no *Capítulo 5*. A intenção é evidenciar a aplicabilidade e exequibilidade do processo proposto, possibilitando sua avaliação. Em cada caso, será mostrado como foram desenvolvidos os microsserviços para compor o BPMSm dedicado ao modelo considerado.

Os modelos de processos de negócio apresentados mostram complexidades diferentes, com diferentes números de subprocessos. Em todos os modelos, foi verificado se satisfaziam os critérios definidos na Etapa 1 do processo, ou seja, se precisavam ou não de ajustes. A ferramenta Web MservSpec, desenvolvida para este trabalho, mostrou-se adequada para a especificação de todos os microsserviços, que automatizam atividades tipo manual, *script* ou serviço, onde cada raia do modelo de processos de negócio corresponde a um tipo de usuário.

Para o desenvolvimento, testes e integração dos microsserviços, foram seguidos todos os procedimentos definidos na Etapa 2 do processo proposto. Observa-se que cada microsserviço tem sua própria base de dados, se a atividade correspondente tiver um objeto de dados relacionado a ela. Para complementar o BPMSm referente a cada modelo de processos de negócio em cada caso, o microsserviço de controle foi desenvolvido seguindo os procedimentos da Etapa 3.

Os três casos considerados neste capítulo estão apresentados nas *Seções 6.1* a *6.3*, sendo que os respectivos modelos de processos de negócio estão no APÊNDICE C, em formato digital (CD-R anexo), tanto no formato em BPMN v2.0 quanto em PDF e PNG. O código dos microsserviços desenvolvidos estão disponíveis no APÊNDICE D, também em formato digital (CD-R anexo).

Na *Seção 6.1*, o modelo do primeiro caso foi obtido do repositório público de modelos de processos de negócio da empresa de desenvolvimento de software *Bizagi*[2]. Esse repositório conta com modelos de diferentes áreas de negócio e diferentes níveis de complexidades. O modelo selecionado se refere a um processo de contas a serem pagas a

[2] *http://www.Bizagi.com/en/community/process-xchange*.

fornecedores (“*Accounts Payable*”), contemplado atividades que podem ser definidas em várias organizações. Basicamente, o processo valida a documentação enviada por fornecedores, visando a liberação dos respectivos pagamentos.

Na *Seção 6.2*, o modelo de processos de negócio do segundo caso também foi obtido do repositório da empresa *Bizagi* e contempla atividades gerenciais de um setor de Tecnologia da Informação (TI) de uma organização. O processo (“Gerenciamento de Acessos”) gerencia permissões de acesso a um software corporativo.

Na *Seção 6.3*, o modelo de processos de negócio do terceiro caso foi desenvolvido por Jorge Luís Gregório, docente da Faculdade de Tecnologia de Jales “Prof. José Camargo” (Fatec Jales). O modelo mapeia o processo, definido pelos orientadores de estágio da Fatec Jales, para formalização do estágio obrigatório a todos os alunos de todos os cursos superiores da Instituição.

Em cada caso, foi realizada uma avaliação de desempenho do respectivo BPMSm, considerando tempo de resposta e tempo de execução quando se usava ou não o recurso de *cache* da plataforma em nuvem *Firebase*. Desse modo, a *Seção 6.4* apresenta uma avaliação geral dos resultados obtidos nos três estudos de caso.

6.1 CASO 1: MODELO “ACCOUNTS PAYABLE”

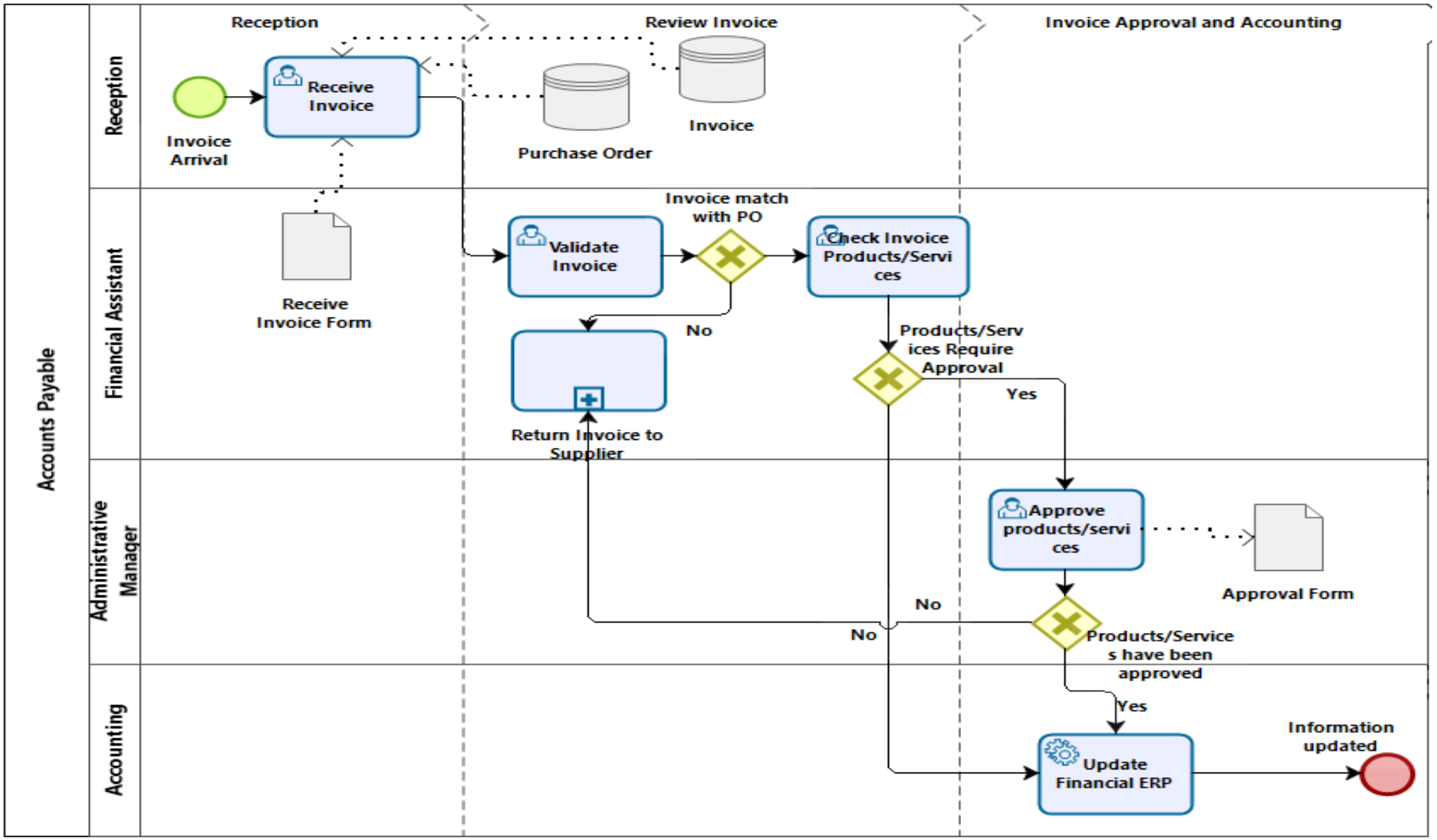
Nesta seção, será considerado o modelo de processos de negócio de contas a pagar denominado “*Accounts Payable*”, que valida a documentação enviada pelos fornecedores, com posterior liberação de pagamento aos mesmos. O processo principal está apresentado na **Figura 29**, mostrando que o modelo contém um subprocesso. O modelo completo está disponível no APÊNDICE C.

Foram verificadas todas as condições requeridas de um modelo, como definido na fase E1.1 da Etapa 1 do processo sistematizado proposto, e não houve necessidade de ajustes. Isso se deve ao fato de que a versão considerada também foi utilizada por Nogueira (2017a). O autor retirou o modelo do repositório público Bizagi (BIZAGI XCHANGE, 2015) e o ajustou, em termos gráficos e textual, segundo as 42 heurísticas de negócio definidas por ele.

O modelo “*Accounts Payable*” foi modelado em BPMN v2.0 e contém:

- 3 eventos (1 evento de início e 2 de fim);
- 5 atividades (4 sem tipo definido, 1 do tipo Serviço);
- 3 *gateways* exclusivos;
- 4 raias.

Figura 29 – Modelo de processos de negócio "*Accounts Payable*".



Fonte: adaptado de Bizagi XChange (2015) por Nogueira (2017a).

O subprocesso *“Return Invoice to Supplier”* não é tratado como subfluxo do processo principal, ou seja, não é invocado a partir do processo principal, pois ao exportar o modelo “*Accounts Payable*” (Etapa 1 do processo sistemático), foi gerado apenas um arquivo BPMN. Este subprocesso possui os seguintes elementos:

- 1 evento de início;
- 2 eventos de fim;
- 3 atividades (1 de tipo indefinido, 1 do tipo Script e 1 do tipo Manual);
- 1 gateway exclusivo.

Na Fase E1.2 da Etapa 1, o modelo completo sem os subprocessos foi convertido para um documento no formato PDF usando-se o sistema *Bizagi Modeler*, gerando o arquivo de nome “*Accounts Payable*.pdf”. De modo similar o sistema foi exportado para o formato PNG, gerando o arquivo “*Accounts Payable*.png”.

Na Etapa 2, a ferramenta Web MServSpec foi utilizada para a geração do documento de especificação dos requisitos de cada microsserviço, referente a cada atividade do modelo. Considerando que há cinco atividades no modelo principal e 3 no seu subprocesso, então o documento ERS do modelo contemplou a especificação de requisitos para o desenvolvimento de cinco microsserviços do modelo principal.

O arquivo “*Accounts Payable*.png” foi submetido como entrada (*upload*) para o sistema MservSpec, preenchendo-se os devidos campos sobre o modelo manualmente. Também foram preenchidos todos os campos referentes a cada atividade do modelo principal.

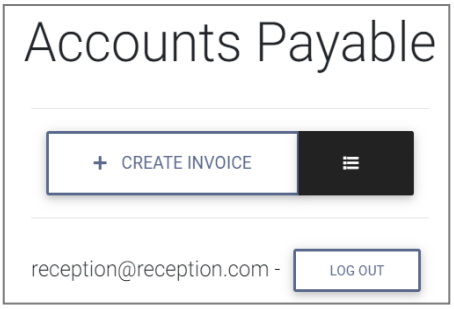
Em relação à fase E2.2 da Etapa 2, para implementação e testes dos microsserviços, observa-se que as atividades do tipo usuário foram implementadas com os métodos *get*, *post* e *path.* As atividades do tipo serviço e *scripts* foram implementadas como *stubs*, conforme algoritmos definidos na *Seção 5.5*.

Foi implementada uma interface para cada um dos usuários do BPMS “*AccountsPayabl*e”. A **Figura 30** apresenta, como exemplo, a tela do usuário da Recepção (“*Reception*”), com as funcionalidades especificadas no documento ERS do modelo. A **Figura 31** apresenta a tela para o usuário Assistente Financeiro (“*Financial Assistant*”), contemplando permissões para validar e visualizar as contas recebidas para sua análise.

Após a Etapa 2, com todos os cinco microsserviços implementados, com as devidas telas de usuário, foi implementado o microsserviço de controle (Etapa 3). As **Figuras 32** e **33** mostram a interface de controle para as atividades “*Receive Invoice*” (tela com parte da listagem das faturas cadastradas) e “*Validate Invoice*” (tela com parte da listagem das faturas enviadas para análise), respectivamente. Após efetuar o cadastro de uma nova fatura

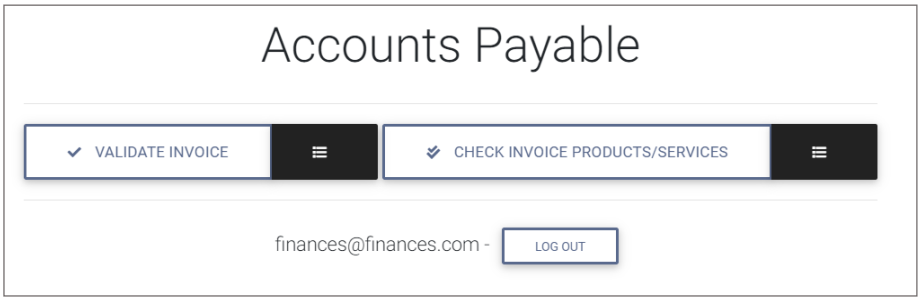
(“*invoice*”), o usuário “*Reception*” pode visualizar quais faturas foram enviadas para serem aprovadas pelo usuário “*Financial Assistant*” (**Figura 32**). O usuário “*Financial Assistant*” tem permissões para validar e visualizar as contas recebidas para sua análise, conforme mostrado nas **Figuras 32** e **33.** O BPMS “*AccountsPayable*” completo encontra-se no APÊNDICE C.

Figura 30 – Tela do usuário “*Reception*”.



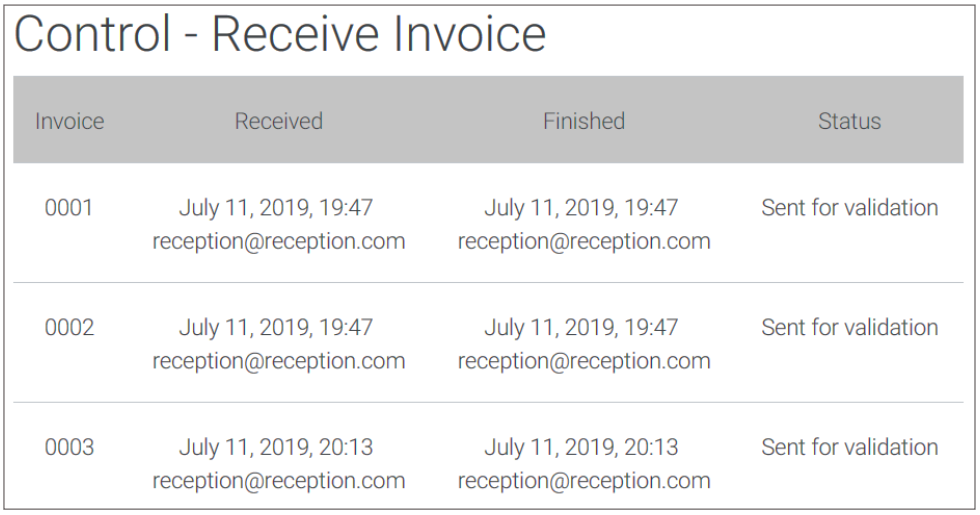
Fonte: elaborado pelo autor.

Figura 31 – Tela do usuário “*Financial Assistant*”.



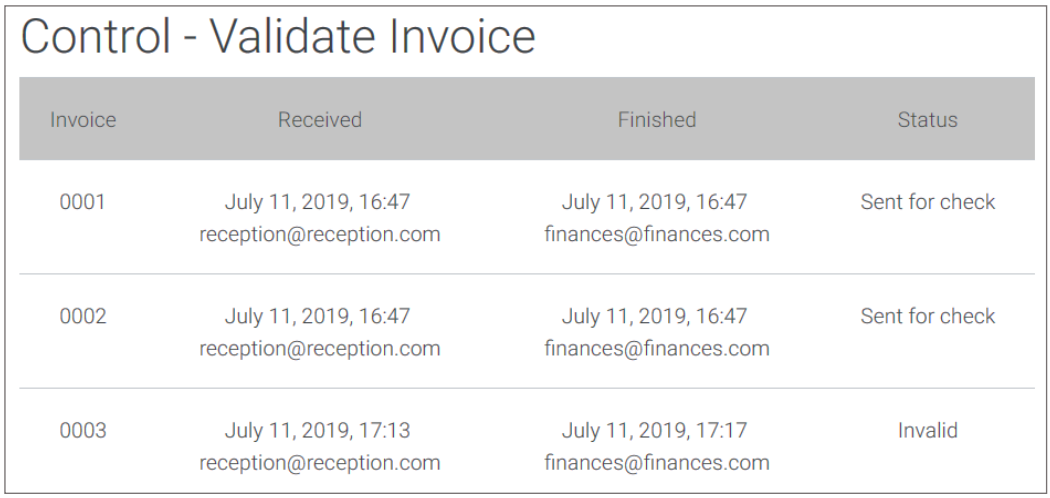
Fonte: elaborado pelo autor.

Figura 32 – Instantâneo da tela do microsserviço de controle para a atividade “*Receive Invoice*”.



Fonte: elaborado pelo autor.

Figura 33 – Instantâneo da tela do microsserviço de controle para a atividade "*Validade Invoice*".



Fonte: elaborado pelo autor.

Uma vez efetuada a implementação das atividades descritas no modelo de processo de negócio, foi implementado o microsserviço de controle, conforme terceiro passo da segunda etapa da *Seção 5.5*. A **Figura 32** e a **Figura 33** mostram a interface de controle para as atividades "*Receive Invoice*" e "*Validate Invoice*" respectivamente.

Avaliação de desempenho do BPMS "*AccountsPayable*"

Após o desenvolvimento dos cinco microsserviços referentes às atividades do modelo de processos de negócio "*Accounts Payable*", foram realizados testes de desempenho na aplicação usando e não usando o recurso de *cache*. Isso foi feito por meio do microsserviço de controle.

Os testes consideraram três critérios: número total de requisições, tempo de resposta do microsserviço de controle e tempo de execução de todos os microsserviços que compõem a aplicação. Os dados coletados nos testes realizados estão na **Tabela 6**, onde pode-se observar que o desempenho do microsserviço de controle é muito superior quando se usa *cache*.

Convém informar os testes realizados em relação aos microsserviços e ao sistema como um todo foram bem sucedidos e evidenciaram conformidade com os critérios de *benchmark* adotados por Ueda *et al.* (2016) e Alshuqayran *et al.* (2016).

Tabela 6 – Desempenho do microsserviço de controle (BPMS para "*Accounts Payable*").

Critérios	Sem utilização de *cache*	Utilizando *cache*
Número total de requisições	7	1
Tempo de resposta do microsserviço de controle	2.569 ms	961 ms
Tempo de execução de todos os microsserviços	5.465 ms	962 ms

Fonte: elaborado pelo autor.

6.2 CASO 2: MODELO "ACCESS MANAGEMENT"

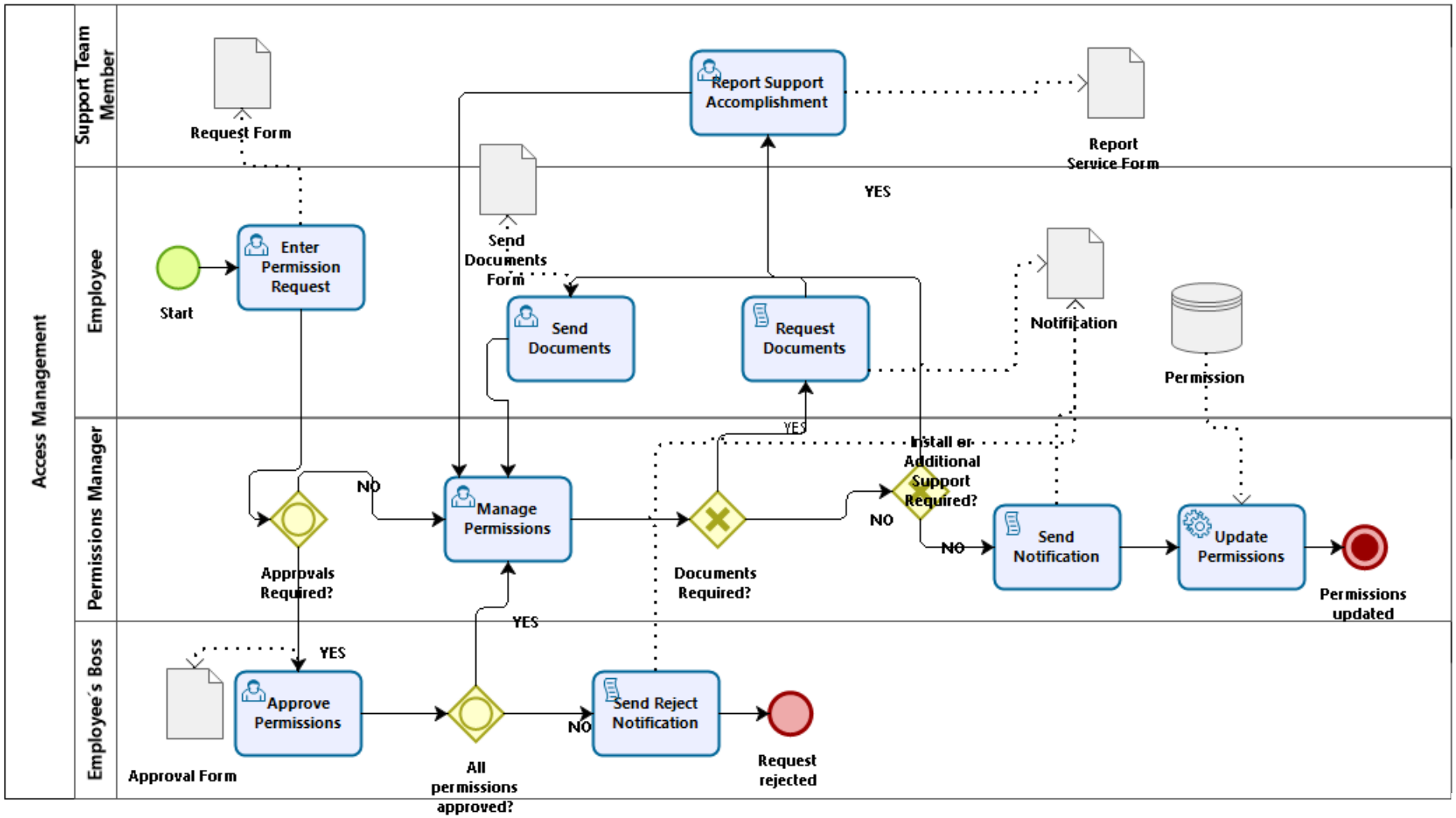
Nesta seção, será considerado o modelo de processos de negócio de gerenciamento de acesso denominado "*Access Management*", do setor de Tecnologia da Informação (TI) de uma organização, responsável pela gestão das permissões de acesso a um software corporativo. Para isso, o processo faz uso de formulários para requisições de acesso, envio de documentos e aprovação das permissões, que são preenchidos e revisados ao longo do fluxo do processo, resultando na concessão ou rejeição da solicitação de permissão. O processo do modelo "*Access Management*" está apresentado na **Figura 34**, observando-se que o modelo não contém nenhum subprocesso. O modelo completo está disponível no APÊNDICE C.

Foram verificadas todas as condições requeridas de um modelo, como definido na fase E1.1 da Etapa 1 do processo sistematizado proposto, e não houve necessidade de ajustes. Isso se deve ao fato de que a versão considerada também foi utilizada por Nogueira (2017a). O autor retirou o modelo do repositório público Bizagi (BIZAGI XCHANGE, 2015) e o ajustou, em termos gráficos e textual, segundo as 42 heurísticas de negócio definidas por ele.

O modelo "*Access Management*" foi modelado em BPMN v2.0 e contém:

- 1 piscina;
- 4 raias;
- 1 evento de início;
- 1 evento de fim;
- 4 participantes;
- 9 atividades (5 sem tipo definido, 1 do tipo Serviço e 3 do tipo *script*);
- 2 *gateways* inclusivos e 2 *gateways* exclusivos.

Figura 34 – Modelo de processos de negócio "*Access Management*".



Fonte: adaptado de Bizagi XChange (2015) por Nogueira (2017a).

Na Fase E1.2 da Etapa 1, o modelo completo sem os subprocessos foi convertido para um documento no formato PDF usando-se o sistema *Bizagi Modeler*, gerando o arquivo de nome "*Access Management*.pdf". De modo similar o sistema foi exportado para o formato PNG, gerando o arquivo "*Access Management*.png".

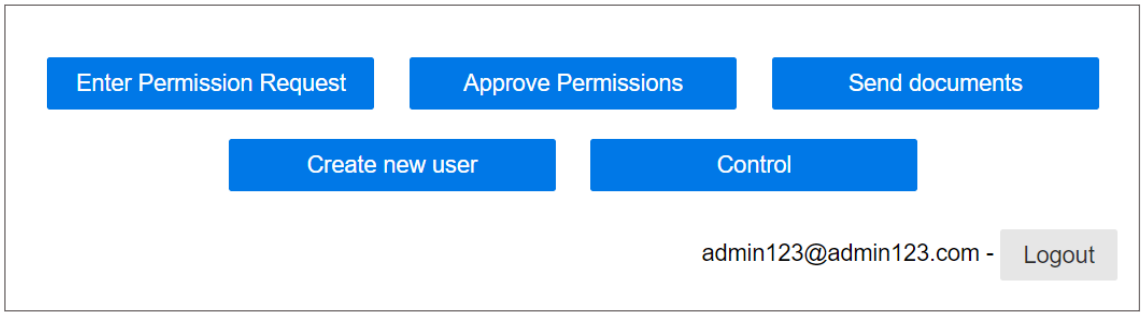
Na Etapa 2, a ferramenta Web MServSpec foi utilizada para a geração do documento de especificação dos requisitos de cada microsserviço, referente a cada atividade do modelo. Considerando que há nove atividades no modelo, então o documento ERS do modelo contemplou a especificação de requisitos para o desenvolvimento de nove microsserviços.

O arquivo "*Access Management*.png" foi submetido como entrada (*upload*) para o sistema MservSpec, preenchendo-se os devidos campos sobre o modelo manualmente. Também foram preenchidos todos os campos referentes a cada atividade do modelo.

Em relação à fase E2.2 da Etapa 2, para implementação e testes dos microsserviços, observa-se que as atividades do tipo usuário foram implementadas com os métodos *get*, *post* e *path.* As atividades do tipo serviço e *scripts* foram implementadas como *stubs*, conforme algoritmos definidos na *Seção 5.5*.

Foi implementada uma interface para cada um dos usuários do BPMS "*Access Management*". A **Figura 35** apresenta, como exemplo, a tela do usuário Administrador ("*Employee´s Boss*"), com as funcionalidades especificadas no documento ERS do modelo. Este tipo usuário possui funcionalidades de requisitar permissões de acesso, aprovar permissões de acesso, enviar documentos, cadastrar novos usuários e exibir um controle de todos as requisições efetuadas. A **Figura 36** apresenta parte da tela do microsserviço de controle sobre o estágio atual das requisições efetuadas pelos usuários do BPMS "*Access Management*".

Figura 35 – Tela do usuário "*Employee´s Boss*".



Fonte: elaborado pelo autor.

Figura 36 – Instantâneo da tela do microsserviço de controle do estágio atual das requisições efetuadas pelos usuários do BPMS "*Access Management".*

Request	Current stage	Last update
1	Update Permissions (Done)	August 20, 2019
3	Update Permissions (Done)	August 20, 2019
4	Update Permissions (Done)	August 22, 2019
5	Manage Permissions	August 22, 2019

Fonte: elaborado pelo autor.

Após a Etapa 2, com todos os nove microsserviços implementados, com as devidas telas de usuário, foi implementado o microsserviço de controle (Etapa 3). A **Figura 37** mostra a interface do microsserviço de controle com listagem das atividades "*Approve Permissions*", "*Request Documents*", "*Send Documents*", "*Report Support Accomplishement*", "*Manage Permissions*", "*Send Notification*", "*Update Permissions"* de um processo de requisição do BPMS "*Access Management*". O BPMS "*Access Management*" completo encontra-se no APÊNDICE C.

Figura 37 – Instantâneo da tela do microsserviço de controle do BPMSm "*Access Management*".

Request 3

Stage	Received	Finished	Status
Approve Permissions	August 20, 2019, 18:07 -	August 20, 2019, 18:08 -	Approved. Sent to Permission Manager.
Request Documents	August 20, 2019, 18:09 -	August 20, 2019, 18:09 -	Documents requested.
Send Documents	August 20, 2019, 18:09 -	August 20, 2019, 18:10 -	Documents sent.
Report Support Accomplishment	August 20, 2019, 18:16 -	August 20, 2019, 18:16 -	Accomplished.
Manage Permissions	August 20, 2019, 18:16 -	August 20, 2019, 18:20 -	Ready. Notification sent.
Send Notification	August 20, 2019, 18:20 -	August 20, 2019, 18:20 -	Notification sent.
Update Permissions (Done)	August 20, 2019, 18:20 -	August 20, 2019, 18:20 -	Permissions updated.

OK

Fonte: elaborado pelo autor.

Avaliação de desempenho do BPMS Access Management

Após o desenvolvimento dos nove microsserviços referentes às atividades do modelo de processos de negócio "*Access Management*", foram realizados testes de desempenho na aplicação usando e não usando o recurso de *cache*. Isso foi feito por meio do microsserviço de controle.

Os testes consideraram três critérios: número total de requisições, tempo de resposta do microsserviço de controle e tempo de execução de todos os microsserviços que compõem a aplicação. Os dados coletados nos testes realizados estão na **Tabela 7**, onde pode-se observar que o número de requisições realizadas pelo microsserviço de controle é menor quando se usa *cache*.

Tabela 7 – Desempenho do microsserviço de controle (BPMS para "*Access Management*").

Critérios	Sem utilização de *cache*	Utilizando *cache*
Número total de requisições	24	2
Tempo de resposta do microsserviço de controle	2.020 ms	942 ms
Tempo de execução de todos os microsserviços	12.386 ms	943ms

Fonte: elaborado pelo autor.

Convém informar os testes realizados em relação aos microsserviços e ao sistema como um todo foram bem sucedidos e evidenciaram conformidade com os critérios de *benchmark* adotados por Ueda *et al.* (2016) e Alshuqayran *et al.* (2016).

6.3 CASO 3: MODELO "ESTÁGIO FATEC JALES"

Nesta seção, será considerado o modelo de processos de negócio denominado "Estágio Fatec Jales", para formalização do estágio que todos os alunos dos cursos superiores da Faculdade de Tecnologia de Jales "Prof. José Camargo" (Fatec Jales) devem realizar. O modelo foi desenvolvido por orientadores de estágio da Fatec de Jales e contempla os aspectos necessários para o desenvolvimento de um sistema informatizado voltado ao setor de estágio da referida Faculdade. Atualmente (final do ano de 2019), não há um sistema informatizado para auxiliar alunos e professores no processo de cadastros, formalização e impressão dos documentos necessários.

O processo principal está apresentado na **Figura 38**, mostrando que o modelo contém dois subprocessos: “Validar informações” e “Verificar Convênio”. O modelo completo está disponível no APÊNDICE C.

Foram verificadas todas as condições requeridas de um modelo, como definido na fase E1.1 da Etapa 1 do processo sistematizado proposto não requereu ajustes. Isso se deve ao fato de que a versão considerada também foi desenvolvida, seguindo as boas práticas de modelagem definidas por Figueiredo 2018 e Nogueira (2017), segundo Gregório (2019).

O modelo “Estágio Fatec Jales” foi modelado em BPMN v2.0 e contém:

- 1 evento de início;
- 2 eventos de fim;
- 2 piscinas;
- 3 raias;
- 12 atividades (6 do tipo “Manual”, 5 do tipo “Usuário”, 1 do tipo “Serviço”);
- 2 subprocessos do tipo “Simples”;
- 5 artefatos (3 do tipo “Repositório de dados” e 2 do tipo “Objeto de dados”);
- 2 *gateways* do tipo “Exclusivo”;
- 3 eventos intermediários (2 do tipo “Mensagem” e 1 do tipo “Condicional”).

O subprocesso “Validar informações” contempla os seguintes elementos:

- 3 eventos (1 do tipo “Início” e 2 do tipo “Fim”);
- 3 atividades (1 do tipo “Manual” e 2 do tipo “Usuário”);
- 3 artefatos (3 do tipo “Repositório de dados”);
- 1 gateway do tipo “Exclusivo”.

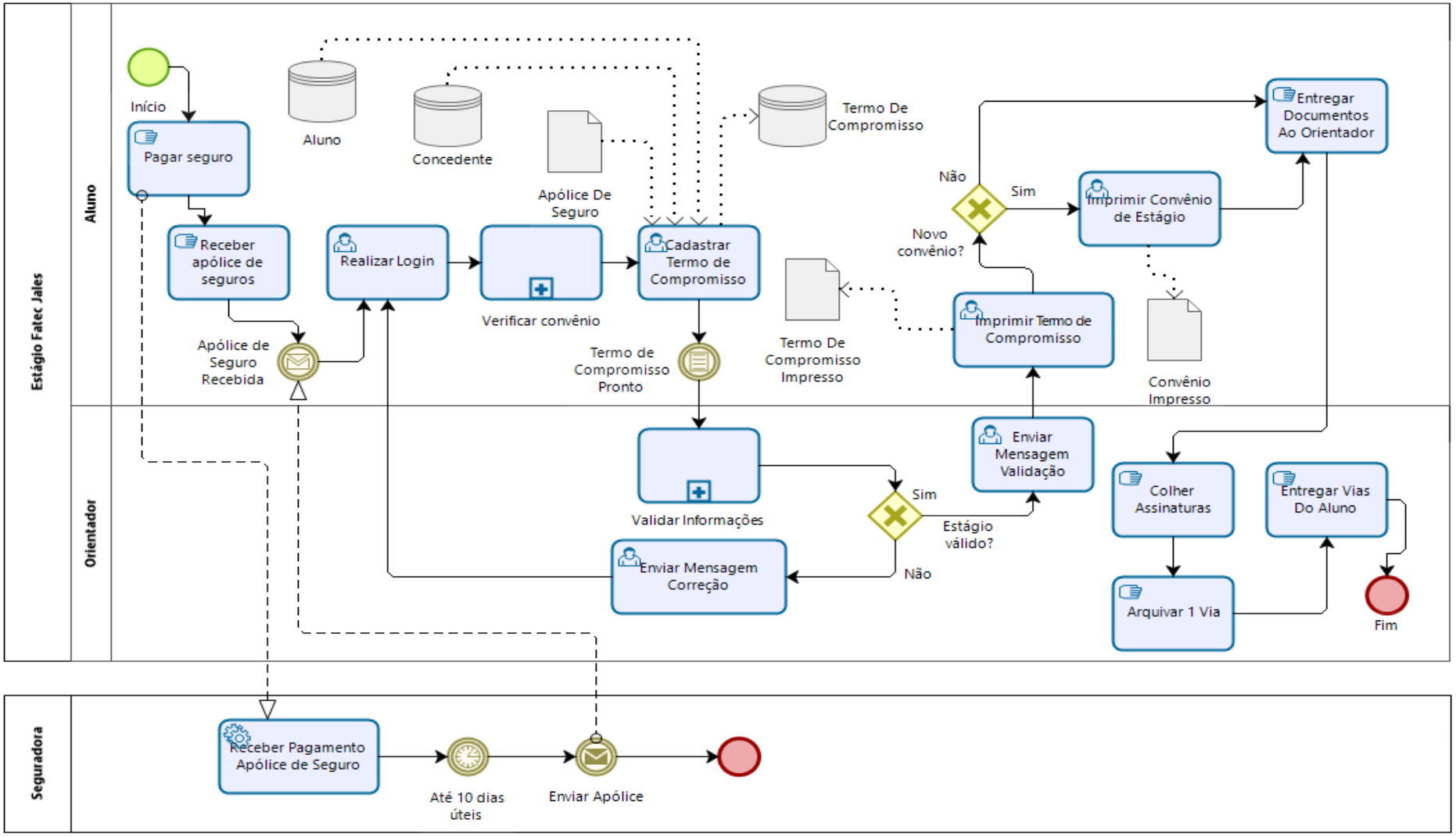
O subprocesso “Validar convênio” contempla os seguintes elementos:

- 2 eventos (1 do tipo “Início” e 1 do tipo “Fim”);
- 4 atividades todas do tipo “Usuário”;
- 2 artefatos ambos do tipo “Repositório de Dados”;
- 2 gateways do tipo “Exclusivo”.

Na Fase E1.2 da Etapa 1, o modelo completo principal, sem subprocessos, foi convertido para um documento no formato PDF usando-se o sistema *Bizagi Modeler*, gerando o arquivo de nome “Estágio Fatec Jales.pdf”. De modo similar o sistema foi exportado para o formato PNG, gerando o arquivo “Estágio Fatec Jales.png”.

Na Etapa 2, a ferramenta Web MServSpec foi utilizada para a geração do documento de especificação dos requisitos de cada microsserviço, referente a cada atividade do modelo. Considerando que há treze atividades no modelo principal e sete nos dois subprocessos, então o documento ERS do modelo contemplou a especificação de requisitos para o desenvolvimento de treze microsserviços.

Figura 38 – Modelo de processos de negócio “Estágio Fatec Jales”.



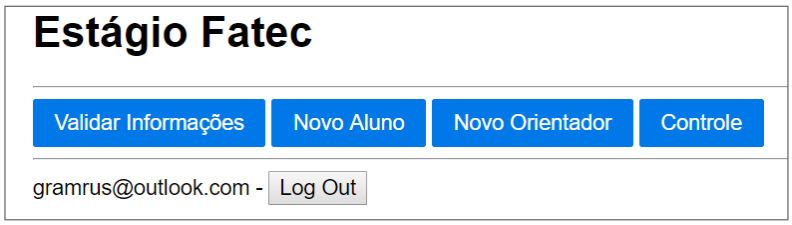
Fonte: Gregório (2019).

O arquivo “Estágio Fatec Jales.png” foi submetido como entrada (*upload*) para o sistema MservSpec, preenchendo-se os devidos campos sobre o modelo manualmente. Também foram preenchidos todos os campos referentes a cada atividade do modelo.

Em relação à fase E2.2 da Etapa 2, para implementação e testes dos microsserviços, observa-se que as atividades do tipo usuário foram implementadas com os métodos *get*, *post* e *path.* As atividades do tipo serviço e *scripts* foram implementadas como *stubs*, conforme algoritmos definidos na *Seção 5.5*.

Foi implementada uma interface para cada um dos usuários do BPMS “Estágio Fatec Jales”. A **Figura 39** apresenta, como exemplo, a tela do usuário “Orientador” do estágio, com as funcionalidades especificadas no documento ERS do modelo. As funcionalidades para esse tipo de usuário se referem ao cadastro de novos alunos e orientadores de estágio, assim como validação das informações passadas pelos alunos.

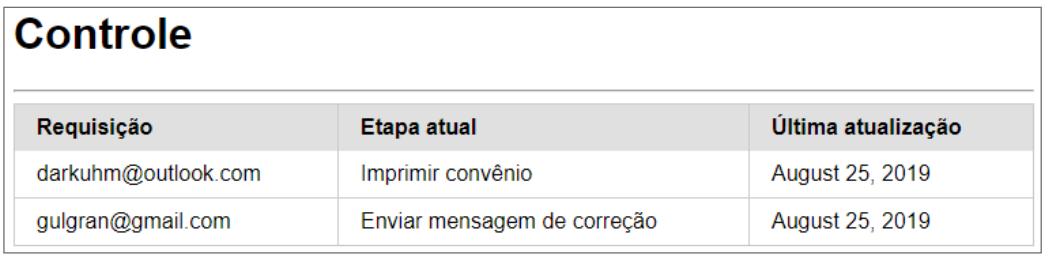
Figura 39 – Tela do usuário “Orientador” do estágio do BPMSm “Estágio Fatec Jales”.



Fonte: elaborado pelo autor.

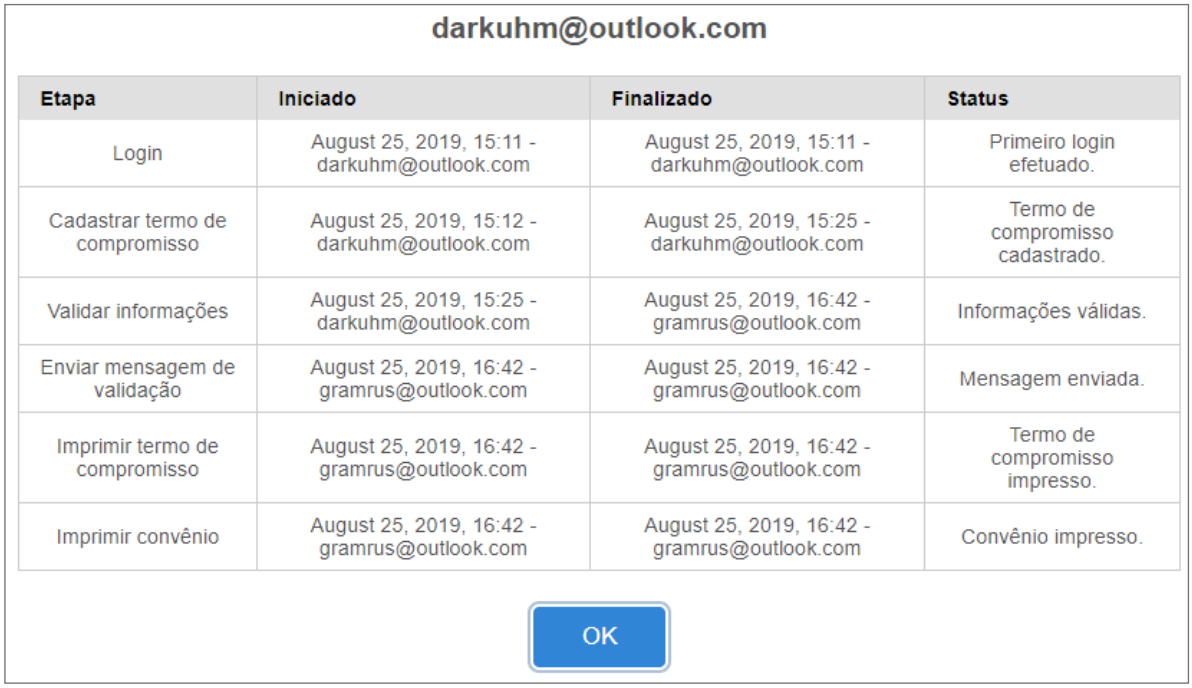
Após a Etapa 2, com todos os treze microsserviços implementados, com as devidas telas de usuário, foi implementado o microsserviço de controle (Etapa 3). Esse microsserviço permite ao orientador do estágio visualizar as solicitações feitas pelos usuários, como apresentado na **Figura 40** (parte da listagem de requisições dos usuários). A **Figura 41** apresenta a interface contemplando parte da listagem com dados sobre uma determinada solicitação de um usuário. O BPMS “Estágio Fatec Jales” completo encontra-se no APÊNDICE C.

Figura 40 – Instantâneo da tela do microsserviço de controle de requisições dos usuários do BPMSm “Estágio Fatec Jales”.



Fonte: elaborado pelo autor.

Figura 41 – Instantâneo da tela do microsserviço de controle de uma solicitação de usuário do BPMSm “Estágio Fatec Jales”.



Fonte: elaborado pelo autor.

Avaliação de desempenho do BPMS “Estágio Fatec Jales”

Após o desenvolvimento dos treze microsserviços referentes às atividades do modelo de processos de negócio “Estágio Fatec Jales” foram realizados testes de desempenho na aplicação usando e não usando o recurso de *cache*. Isso foi feito por meio do microsserviço de controle.

Os testes consideraram três critérios: número total de requisições, tempo de resposta do microsserviço de controle e tempo de execução de todos os microsserviços que compõem a aplicação. Os dados coletados nos testes realizados estão na **Tabela 8**, onde pode-se observar que o número de requisições realizadas pelo microsserviço de controle é menor quando se usa *cache*.

Convém informar os testes realizados em relação aos microsserviços e ao sistema como um todo foram bem sucedidos e evidenciaram conformidade com os critérios de *benchmark* adotados por Ueda *et al.* (2016) e Alshuqayran *et al.* (2016).

Tabela 8 – Desempenho do microsserviço de controle (BPMS para “Estágio Fatec Jales”).

Critérios	Sem utilização de *cache*	Utilizando *cache*
Número total de requisições	9	2
Tempo de resposta do microsserviço de controle	1.395 ms	984 ms
Tempo de execução de todos os microsserviços	3.483 ms	985 ms

Fonte: elaborado pelo autor.

6.4 AVALIAÇÃO GERAL DOS RESULTADOS

Considerando os estudos casos apresentados nas *Seções 6.1* a *6.3*, foi possível validar o processo sistemático definido neste trabalho, apresentado na *Seção 5.2.* A implementação dos BPMSm para cada modelo “m” considerado seguiu literalmente as instruções definidas nas Etapas 1 a 3 do processo proposto.

A **Tabela 9** apresenta uma comparação do desempenho de cada um dos BPMSm implementados, de acordo com os critérios definidos na *Seção 4.3* e com as Tabelas 7, 8 e 9, referentes à avaliação de cada modelo, conforme já apresentado. Os valores foram obtidos durante a execução e resposta dos microsserviços que compõem o BPMSm, por meio do microsserviço de controle com o recurso de utilização de *cache*.

Na **Tabela 9**, pode-se observar que o tempo de resposta e o tempo de execução dos microsserviços dos BPMSm desenvolvidos são muito similares, apesar dos Casos 2 e 3 contemplarem duas requisições de acesso ao aplicativo (quando utilizado o recurso de *cache*).

Tabela 9 – Comparação de desempenho com utilização de cache dos Casos 1, 2 e 3.

Critérios	**“*Accounts Payable*”**	**“*Access Management*”**	**“Estágio Fatec Jales”**
Número total de requisições	1	2	2
Tempo de resposta do microsserviço de controle	853 ms	942 ms	984 ms
Tempo de execução de todos os microsserviços	853 ms	943 ms	985 ms

Fonte: elaborado pelo autor.

Outro fator relevante é a quantidade de atividades em cada modelo de processos de negócio. Considerando os modelos de processos de negócio dos Casos apresentados nas *Seções 6.1 a 6.3,* o número de atividades em cada um deles é: 6, 9 e 15, respectivamente.

Nesses Casos, a diferença de atividades não interferiu nos critérios de desempenho adotados. Isso evidencia escalabilidade de forma horizontal viabilizada pelo processo sistematizado proposto, com custo menor e mais eficiente que a vertical, utilizada em sistemas monolíticos.

Os Casos mostrados nesta seção como prova de conceito, bem como outros experimentos realizados, conduzem as pesquisas na direção da integração de vários BPMSm´s de uma organização, considerando possibilidades de haver recursos comuns entre eles, requerendo estúdios sobre o modo de controle no BPMSg.

7 CONCLUSÃO E CONSIDERAÇÕES FINAIS

A principal contribuição deste trabalho foi apresentar um processo sistematizado para o desenvolvimento de um sistema de gerenciamento de processos de negócio, conhecido como BPMS (*Business Process Management System*), dedicado a um modelo “m”, aqui denominado de “BPMSm”. Esse sistema é composto por microsserviços que automatizam cada atividade do modelo “m”, juntamente com um microsserviço de controle.

O desenvolvimento de BPMSm’s dedicados possibilita que modelos de processos de negócio sejam automatizados e executados em sua plenitude semântica, de forma mais aderente às expectativas funcionais e não funcionais da equipe de negócio. A integração dos BPMSm’s de uma organização compõe o BPMS global (BPMSg), que gerencia, de modo geral, todos os BPMSm’s, embora essa abordagem não faça parte do escopo atual deste trabalho.

O ambiente de desenvolvimento dos microsserviços e de execução de um BPMSm é a plataforma *Google Firebase*. Após a comparação das características de diversas plataformas de computação em nuvem, essa plataforma foi selecionada para hospedagem dos microsserviços, devido ao grande conjunto de recursos gratuitos e ao alto grau de usabilidade apresentado. O serviço de autenticação na plataforma *Firebase* garante que o acesso aos microsserviços seja feito apenas de forma autorizada, de acordo com as categorias de usuários definidas.

A adoção de tecnologias de computação em nuvem possibilitou a construção de interfaces Web para interação com os usuários contextualizadas na abordagem DevOps. A independência em termos de execução e recursos dos microsserviços os tornam relativamente simples para implementação e manutenção, se forem definidas estratégias e tecnologias adequadas, como neste trabalho.

Devido à sistemática e recursos utilizados, o custo de desenvolvimento e implantação de um BPMSm é baixo em relação aos BPMSs tradicionais. Devido ao baixo acoplamento e à independência dos microsserviços, são mais adaptáveis ao desenvolvimento ágil e entregas contínuas (NEWMAN, 2015; SILL, 2016). Um BPMSm permite escalabilidade de forma horizontal, que tem custo menor e é mais eficiente que a vertical, utilizada em sistemas monolíticos.

Como prova de conceito, neste trabalho foram apresentados três casos práticos (*Capítulo 6*). O processo sistematizado proposto mostrou-se exequível e adequado a modelos de processos de negócio com variados níveis de complexidade, no que se refere ao número de atividades e subprocessos, bem como à diversidade de sistemas de banco de dados utilizados.

A especificação de cada microsserviço inclui requisitos funcionais e não funcionais, e são feitas com o apoio de uma ferramenta Web (MservSpec), que também foi construída com arquitetura de microsserviços e interface responsiva. Na solução apresentada para o desenvolvimento de um BPMSm, um microsserviço adicional é responsável pelo monitoramento e controle dos demais: “microsserviço de controle”. Esse microsserviço obtém informações sobre o estado dos dados de todos os outros microsserviços e as apresenta por meio de uma interface unificada. O microsserviço de controle utiliza recursos de *cache*, a fim de evitar custos elevados e congestionamentos na rede, devido ao alto número de requisições que podem ser feitas.

De modo geral, o desenvolvimento de microsserviços tem sido alvo de muitos estudos acadêmicos e aplicações empresariais. Contudo, a inovação proporcionada por este trabalho está no conjunto da proposta, incluindo a granularidade considerada para a especificação de cada microsserviço. Todos os procedimentos definidos no processo sistematizado proposto são originais, com base em várias pesquisas e experimentos práticos. Todo o trabalho foi contextualizado na abordagem da Engenharia de Software Contínua e DevOps, o que levou a uma solução na direção de metodologias ágeis, de modo a possibilitar entregas contínuas de maneira prática, rápida e escalável.

Trabalhos Futuros

Uma análise mais efetiva pode ser feita com a execução de um BPMSm em ambiente real, com testes envolvendo usuários de fato. O terceiro caso apresentado no *Capítulo 6* trata de uma aplicação para automatização do processo de formalização de estágios obrigatórios da “Fatec Jales”. Por se tratar do ambiente de trabalho da autora desta dissertação, um próximo passo é implantar o BPMSm implementado, avaliar seu funcionamento, seu desempenho e usabilidade, entre outros aspectos.

Há a possibilidade de se implementar e avaliar um BPMSm em operação para modelos mais complexos em ambientes reais e acessíveis à autora deste trabalho. Um exemplo é o ambiente da Biblioteca do Câmpus da Universidade Estadual Paulista (Unesp) em Rio Claro, o qual já conta com um modelo de processos de negócio com 2 subprocessos, 90 atividades (vários tipos), 32 eventos, 9 objetos de dados e 32 *gateways* de vários tipos:

exclusivo, inclusivo, paralelo, baseado em evento e complexo. Esse modelo foi implementado e já avaliado quanto à conformidade às heurísticas de negócio/critérios de Nogueira (2017) e Figueiredo (2018) em outros trabalhos. Também possibilitará ampliar o processo sistemático definido para contemplar todos os símbolos de BPMN v2.0 utilizados no modelo. Adicionalmente, o tamanho do modelo possibilitará ampliar o universo da avaliação, bem como analisar a escalabilidade de um BPMSm desenvolvido segundo o processo proposto neste trabalho.

Na Fatec Jales, outros modelos de processos de negócio estão sendo desenvolvidos para outros serviços disponíveis aos alunos, com a devida documentação e segundo as boas práticas consideradas neste trabalho. Assim, o trabalho pode ser estendido para avaliar soluções para integrar os BPMSm's em um BPMSg. Dessa forma, a escalabilidade do processo de integração dos BPMS's também poderá ser avaliada.

Em relação à ferramenta MservSpec, desenvolvida neste trabalho para a especificação dos microsserviços, melhorias podem ser implementadas, inclusive em relação a permissões de acesso dos usuários.

Para facilitar o desenvolvimento dos BPMSm's, um outro trabalho futuro é o desenvolvimento de um *framework* com suporte ao processo sistematizado apresentado neste trabalho. No momento, o processo é aplicado de forma manual. Já com a automatização do processo, a produtividade da equipe de TI seria aumentada e a aplicabilidade da solução proposta também. Isso ajudaria a avaliar entregas contínuas nos BPMSm's, considerando possibilidades de manutenção frequentes nos microsserviços. Esse *framework* proposto poderia integrar a ferramenta Web MservSpec.

REFERÊNCIAS

ABPMP - ASSOCIATION OF BUSINESS PROCESS MANAGEMENT PROFESSIONALS. **BPM CBOK V3.0**: Guia para o Gerenciamento de Processos de Negócio – Corpo Comum de Conhecimento. 1. ed. São Paulo: ABPMP Brasil, 2013.

ADERALDO, C. M., MENDONÇA, N. C., PAHL, C., JAMSHIDI, P. **Benchmark requirements for microservices architecture research**. 2017. IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE).

ALEJANDRO, I., K., F**. Definición de um ambiente de construcción de aplicaciones empresariales através de Deops, microservicios y contenedores**. 2017. 146 fls. Dissertação (Graduação em Engenharia da Informática). Universidad Técnica Particular de Loja. Equador. 2017.

ALPERS, S., BECKER, C., OBERWEIS, A., SCHUSTER, T. Microservice based tool support for business process modelling. In: IEEE INTERNATIONAL ENTERPRISE DISTRIBUTED OBJECT COMPUTING WORKSHOP (EDOCW), 19., 2015, Adelaide. **Proceedings**… Austrália, 2015. p. 71-78.

ALSHUQAYRAN, N.; ALI, N.; EVANS, R**.** A systematic mapping study in microservice Architecture. In: IEEE INTERNATIONAL CONFERENCE ON SERVICE-ORIENTED COMPUTING AND APPLICATIONS (SOCA), 9., Macau. **Proceedingns**…China, 2016. p. 44-51.

AVRAM. A**. Google Firebase: back-end completo para aplicações Web e mobile**. 2016. Disponível em: <https://www.infoq.com/br/news/2016/07/google-Firebase/>. Acesso em: jun. 2019.

AWS - AMAZON WEB SERVICES. **O que é DevOps?** 2017. Disponível em: <https://aws.amazon.com/pt/devops/what-is-devops/?nc1=f_cc>. Acesso em: abr. 2019.

BALDAM, R.; VALLE, R.; ROZENFELD, H. **Gerenciamento de processos de negócio – BPM**: uma referência para implantação prática. 1. ed. Rio de Janeiro: Elsevier, 2014.

BASS, L. The Software Architect and DevOps. **IEEE Software**, v. 35, n 1, January/February 2018. Disponível em: <DOI: 10.1109/MS.2017.4541051>.

BASS, L., WEBER, I. ZHU, L. **DevOps**: a software architect's perspective. Pearson Education: New Jersey, 2015.

BAUMGARTNER, M. **Círculo de liderança de qualidade:** DevOps - uma visão do futuro com sucesso garantido? 2014. Disponível em: <http://www.anecon.com/blog/qlc2-devops/>. Acesso em jun. 2019.

BERTAM, A. **Seis etapas para o sucesso do DevOps**. 2017. Disponível em: <http://cio.com.br/tecnologia/2017/12/19/seis-etapas-para-o-sucesso-do-devops>. Acesso em jun. 2019.

BITENCOURT, A. S.; PAIVA, D. M. B.; CAGNIN, M. I. **Elicitação de requisitos a partir de modelos de processos de negócio em BPMN:** Uma Revisão Sistemática. 2016. In: XII Brazilian Symposium on Information Systems, Florianópolis, Brasil. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbsi/2016/027.pdf>. Acesso em fev. 2019.

BITENCOURT, M. **Guia de referência para modelar processos, casos e decisões em BPMN, CMMN e DMN**. 2017. Disponível em:<http://mauriciobitencourt.com/wp-content/uploads/2017/04/guia-referencia-poster-bpmn20-cmmn11-dmn11-mauricio-bitencourt.pdf>. Acesso em: nov. 2017.

BIZAGI. **Process modeling**. 2016. Vídeo com restrição de cadastramento no ambiente elearning.com. Disponível em: <http://elearning.*Bizagi*.com/my/intermedia.php?lang=en>. Acesso em: out. 2019.

BIZAGI XCHANGE. **Process Models**. 2015. Disponível em: <https://www.bizagi.com/pt/plataforma/xchange>. Acesso em: maio 2019.

BONITA STUDIO. **Bonitasoft process modeling**. 2019. Disponível em: <https://www.bonitasoft.com/>. Acesso em: jan. 2019.

BORGER, E. Approaches to modeling business processes: a critical analysis of BPMN, workflow patterns and YAWL. **Software & Systems Modeling**, v. 3, p. 305-318, 2012.

BOSCH, J. **Continuous Software Engineering:** An Introduction. Springer International Publishing Switzerland. 2014.

BPMN.IO. Web-based tooling for BPMN, DMN and CMMN.2019. Disponível em: <http://*BPMN.IO*/>. Acesso em: jan. 2019.

COHANE, J. **WTF is Biz Dev? (aka what is "business development"?)**. 2015. Disponível em: <http://jamescohane.com/wtf-is-biz-dev-aka-what-is-business-development/>. Acesso em: abr. 2019.

DESPODOVSKI, R. **Microservices vs. SOA – Is there any difference at all?** 2017. Disponível em: <https://dzone.com/articles/microservices-vs-soa-is-there-any-difference-at-al>. Acesso em: jun. 2019.

DEVMEDIA. **Microsserviço é a mesma coisa que web service?** 2018. Disponível em: <https://www.devmedia.com.br/microsservico-e-a-mesma-coisa-que-Web-service/38184>. Acesso em: jun. 2019.

DAYLEY, B. **Node.js, MongoDB, and AngularJS Web Development**. Michigan, EUA: Addison-Wesley Professional. 2014.

DIAS, F. **BPMN 2.0** – Novos diagramas e elementos: Introdução a coreografia. 2013. Disponível em: < http://blog.iprocess.com.br/2013/08/bpmn-2-0-novos-diagramas-e-elementos-introducao-a-coreografia/>. Acesso em: abr. 2019.

DREWS, P.; SCHIRMER, I.; HORLACH, B.; TEKAAT, C. Bimodal Enterprise Architecture Management: The emergence of a new EAM function for a BizDevOps-based fast IT. 2017. In: INTERNATIONAL ENTERPRISE DISTRIBUTED OBJECT COMPUTING CONFERENCE WORKSHOPS. 21. Quebec,**Proceedings**...Canadá, 2017. p. 57-64.

EGIDIO, A, F. **Arquitetura de microsserviços**. 2017. Disponível em: <https://www.java avancado.com/arquitetura-de-microservicos/>. Acesso em jun. 2019.

FIGUEIREDO, L. R. **Mapeamento de modelos de processos de negócio para ontologias, incluindo sistema de consultas.** 2018. 140f. Dissertação (Mestrado em Ciência da Computação) Universidade Estadual Paulista, Rio Claro, 2018.

FINKE, A. Requirements Inheritance in continuous requirements engineering: a position paper. 2016. In: INTERNATIONAL WORKING CONFERENCE ON REQUIREMENTS ENGINEERING: FOUNDATION FOR SOFTWARE QUALITY – REFSQ. 16. Gothenburg, **Proceedings**....Suécia. 2016.

FIREBASE. **Cloud Functions para Firebase.** 2019. Google. Disponível em: <https://Firebase.google.com/docs/functions/?authuser=0#how_does_it_worK>. Acesso em: maio 2019.

FISCHER, H.; ROSE, M.; YIGITBAS, E. Towards a Task Driven Approach Enabling Continuous User Requirements Engineering. In: INTERNATIONAL WORKING CONFERENCE ON REQUIREMENTS ENGINEERING: FOUNDATION FOR SOFTWARE QUALITY – REFSQ. 16. Gothenburg, **Proceedings**...Suécia. 2016.

FITZGERALD, B; STOL, K-J**.** Continuous Software Engineering and beyond: trends and challenges. In: INTERNATIONAL WORKSHOP ON RAPID CONTINUOUS SOFTWARE ENGINEERING, 1, 2014. Hyderabad, India. **Proceedings...** New York: ACM, 2014, p. 1-9.

FITZGERALD, B; STOL, K-J. Continuous software engineering: A roadmap and agenda. **The journal of systems and software**, Amsterdã, v. 123, p. 176–189. Jan. 2017.

FLEISCHMANN, A.; SCHMIDT, W.; STARY, C. **S-BPM in the wild: practical value creation**. (eBook) Heidelberg: Springer, 2015.

FORBRIG, P. Continuous Software Engineering with Special Emphasis on Continuous Business-Process Modeling and Human- Centered Design. In: INTERNATIONAL CONFERENCE ON SUBJECT-ORIENTED BUSINESS PROCESS MANAGEMENT, 2016a. Erlangen. **Proceedings...** Alemanha, 2016.

_____. When Do Projects End? – The Role of Continuous Software Engineering. In: INTERNATIONAL CONFERENCE ON PERSPECTIVES IN BUSINESS INFORMATICS RESEARCH. 15. 2016b. Prague. **Proceedings...** Czech Republic, 2016b, p. 15-16.

_____. BizDevOps and the role of S-BPM. In: INTERNATIONAL CONFERENCE ON SUBJECT-ORIENTED BUSINESS PROCESS MANAGEMENT, 2018a. Linz. **Proceedings** ...Austria 2016.

_____. Use cases, user stories and BizDevOps. In: International Working Conference On Requirements Engineering: Foundation For Software Quality – REFSQ. 18. Utrecht. **Proceedings....** Holanda. 2018b.

FOWLER, M. **Microservices: a definition of this new architectural term**. 2014.Disponível em: <https://martinfowler.com/articles/microservices.html>. Acesso em: abr. 2019.

FOWLER, S. J**. Microsserviços prontos para a produção**. São Paulo: Novatec. 2017.

FRANÇA, B. B. N.; SIMÕES, R. V.; SILVA, V.; TRAVASSOS, G. H. Escaping from the time box towards continuous planning: an industrial experience In: INTERNATIONAL WORKSHOP ON RAPID CONTINUOUS SOFTWARE ENGINEERING, 2017. Buenos Aires. **Proceedings**... Argentina, 2017. p. 43-49.

FREIRE, F. **O que é DevOps?.** 2013. Disponível em: <https://www.ibm.com/developerworks/ community/blogs/rationalbrasil/entry/o_que_devops?lang=en>. Acessado em: abr. 2019.

GAEA. **DevOps**. 2015. Disponível em: <https://gaea.com.br/o-que-e-devops-conceito/>. Acesso em: abr. 2019.

GEISRIEGLER, M., KOLODIY, M., STANI, S., SINGER, R. Actor based business process modeling and execution: a reference implementation based on ontology models and microservices. In: IEEE EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS. 43 ed. 2017. Viena. **Proceedings**... Viena, 2017, p. 359 – 362.

GREGÓRIO, J. L. **Especificação de requisitos de software a partir de ontologias representativas de modelos de processos de negócio**. 2019. 139f. Dissertação (Mestrado em Ciência da Computação) Universidade Estadual Paulista, Rio Claro, 2019.

GRUHN, V.; SCHÄFER, C. BizDevOps: because DevOps is not the end of the story. In: INTERNATIONAL CONFERENCE (SoMet).14. Naples. **Proceedings...** Italy. 2015. p. 15-17.

GUEDES, G. T. A. **UML2:** Uma abordagem prática. 2 ed. São Paulo: Novatec Editora, 2011.

GUIMARÃES, R. **ADO.NET 2.0: A importância do pool de conexões.** Linha de Código. 2019. Disponível em<http://www.linhadecodigo.com.br/artigo/1254/adonet-20-a-importancia-do-pool-de-conexoes.aspx#ixzz5uR5SSVG4>. Acesso em jul. 2019.

HÖVER, K. M., MÜHLHÄUSER, M. S-BPM-Ont: An ontology for describing and interchanging S-BPM Processes. In: S-BPM ONE, 2014. Eichstätt. **Proceedings**... Alemanha. 2014. p. 41-52.

HUMBLE, J.; MOLESKY, J. Why enterprises must adopt devops to enable continuous delivery. **Cutter IT Journal**. Arlington, Ago. 2011. v. 24, nº. 8, p 6–12.

IPROCESS.**Definindo o escopo de modelagem de um processo de negócio**. 2013. Disponível em:<http://blog.iprocess.com.br/2013/09/definindo-o-escopo-de-modelagem-de-um-processo-de-negocio/>. Acesso em: fev. 2019.

ISO - INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC/IEEE/29148:2011. Systems and software engineering - Life cycle processes - Requirements Engineering**. ISO/IEC/IEEE, 2011.

KIRIKOVA, M.: Continuous requirements engineering in FREEDOM framework: a position paper. In: INTERNATIONAL WORKING CONFERENCE ON REQUIREMENTS ENGINEERING: FOUNDATION FOR SOFTWARE QUALITY (REFSQ). 16. Gothenburg, **Proceedings...** Suécia. 2016. p. 1-6.

KIRIKOVA, M.; PURMALIETIS, K Analytics in Continuous Requirements Engineering. In: INTERNATIONAL WORKING CONFERENCE ON REQUIREMENTS ENGINEERING: FOUNDATION FOR SOFTWARE QUALITY (REFSQ). 17. Essen, **Proceedings....** Alemanha. 2017.

KOSSAK, F., ILLIBAUER, C., GEIST, V., NATSCHLÄGER, C., ZIEBERMAYR, T., FREUDENTHALER, B., KOPETZKY, T., SCHEWE, K. D. **Hagenberg business process modelling method (H-BPM).** Springer International Publishing: Switzerland. 2016.

KRUSCHE, S.; BRUEGGE, B. CSEPM - A Continuous Software Engineering Process Metamodel. In: INTERNATIONAL WORKSHOP ON RAPID CONTINUOUS SOFTWARE ENGINEERING, 3. Buenos Aires, **Proceedings**... Argentina. 2017. p. 2-8.

LEDNEV, A.; IVASCHENKO, A. S-BPM implementation in CUBA platform for rapid application development. In: INTERNATIONAL CONFERENCE ON SUBJECT-ORIENTED BUSINESS PROCESS MANAGEMENT. 8. 2016. Erlangen,**Proceedings**... Alemanha. 2016.

LEITÃO, L.M.; FILHO, J.P.S. **Docker: ferramenta essencial para desenvolvedores**. Curso pela internet. Publicado em: 23 jul. 2017. Cod3r Ensino e Consultoria LTDA. Disponível em: <https://www.udemy.com/courses/search/?q=docker&src=ukw>. Acesso em mar. 2019.

LITTLE, M. **Microservices and SOA**. 2014. Disponível em: <http://www.infoq.com/news/2014/03/microservicessoa>. Acesso em: maio. 2019.

MEDRADO, A. **O que é DevOps?:** Colaboração como caminho para entregar valor ao negócio (Edição do Kindle). 2015.

NEWMAN, S. **Building microservices:** Designing Fine-Grained Systems. O'Reilly Media: Estados Unidos. 2015.

NOGUEIRA, F. A. **Levantamento e especificação de requisitos de software utilizando modelos de processos de negócio.** 2017a. 139f. Dissertação (Mestrado em Ciência da Computação) Universidade Estadual Paulista, Rio Claro, 2017.

_____. **Software SRPD – Software requirements from process definitions**. 1.0. Rio Claro, 2017b.

NOGUEIRA, F., A.; OLIVEIRA, H., C. A heuristic-based approach to improve the quality of BPMN business process models. In: IEEE INTERNATION CONFERENCE ON APPLICATION OF INFORMATION AND COMMUNICATION TECHNOLOGIES (AICT), 17., 2017a, Moscow. **Proceedings...** Moscow, 2017a. p. 1-6.

O'CONNOR, R.V.; ELGER, P.; CLARKE, P. M., Continuous software engineering - A microservices architecture perspective. **Journal of software evolution and process**. v. 29, p. 1866. 2017.

OMG - OBJECT MANAGEMENT GROUP**. Business process model and notation (BPMN)**. 2013. Disponível em: <http://www.omg.org/spec/BPMN> Acesso em: fev. 2019.

PAULA FILHO, W. P. **Engenharia de software**: Fundamentos, Métodos e Padrões. 3. ed. Rio de Janeiro: LTC, 2012.

PMI – PROJECT MANAGEMENT INSTITUTE. **Um guia do conhecimento em gerenciamento de projetos**. 5. ed. EUA: Project Management Institute, 2013.

PRESSMAN, R. S. **Engenharia de software**. 7. ed. Porto Alegre: McGraw Hill, 2011.

RIGHTSCALE. **State of the cloud report™.** 2018. Disponível em: <https://www.rightscale.com/lp/state-of-the-cloud>. Acesso em: fev. 2019.

SGANDERLA, K. **A nova geração de BPMS na nuvem** – e como eles podem alavancar a gestão por processos na sua empresa. 2018. Disponível em: <http://blog.iprocess.com.br/tag/bpms/>. Acesso em: fev. 2019.

_____. **7 Ferramentas gratuitas para criar diagramas de processos com BPMN**, 2016. Atualizado em jul. 2017. Disponível em: <http://blog.iprocess.com.br/2016/09/7-ferramentas-gratuitas-para-criar-diagramas-de-processos-com-bpmn/>. Acesso em: fev. 2019.

_____. **BPMN: Introdução ao diagrama de conversação**. 2014. Disponível em: <http://blog.iprocess.com.br/2014/06/novos-diagramas-e-elementos-introducao-a-conversacao/>. Acesso em: abr. 2019.

_____. **Webinares iProcess 2014 – Introdução à notação BPMN**, 2014. Disponível em: <http://blog.iprocess.com.br/2014/09/Webinares-iprocess-2014-introducao-a-notacao-bpmn/>. Acesso em: fev. 2019.

_____. **Um BPMN para cada propósito de modelagem de processos**, 2012. Disponível em: <http://blog.iprocess.com.br/2012/04/um-bpmn-para-cada-proposito-de-modelagem-de-processos/>. Acesso em: jun. 2019.

SHAMIEH, C. **Continuous engineering for dummies:** IBM Limited Edition. EUA: Wiley, 2014.

SILVA, J. L. L. **Sistema de gestão pública para o cadastramento de ocorrências utilizando arquitetura de microsserviços**. 2015. Disponível em: <http://www.uni7.edu.br/ic2015/18-05-2015_182748803.docx>. Acesso em: jul. 2019.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, E. **Twelve factor app – boas práticas para microsserviços**. IMASTERS. 2018. Disponível em: <https://imasters.com.br/desenvolvimento/twelve-factor-app-boas-praticas-para-microsservicos>. Acesso em: jul. 2019.

UEDA.T, NAKAIKE, T. OHARA, M. **Workload characterization for microservices**. 2016. IBM Research. Tokyo, Japan.

VAHL, J. **Documentar é preciso (e as ferramentas estão aí)!**. 2015. Disponível em: <https://sensedia.com/blog/apis/documentar-e-preciso-e-as-ferramentas-estao-ai/>. Acesso em ago. 2018.

VAN DER AALST, W. M. P. **Business process management**: a comprehensive survey. ISRN Software Engineering. 2013. Disponível em: <http://dx.doi.org/10.1155/2013/507984>. Acesso em: mar. 2019.

VAN DER AALST, W.; ADAMS, M.; HOFSTEDE, A.; RUSSELL, N. **Modern business process automation**. Berlin. Springer-Verlag: 2010.

VAZQUES, C. E.; SIMÕES, G.S. **Engenharia de requisitos**: Software Orientada a Negócios. Rio de Janeiro. Brasport: 2016.

WELLER, C. **DEVOPS Handbook**: simple step by step instructions to DevOps. (Edição do Kindle). EUA: Amazon Digital Services LLC. 2017.

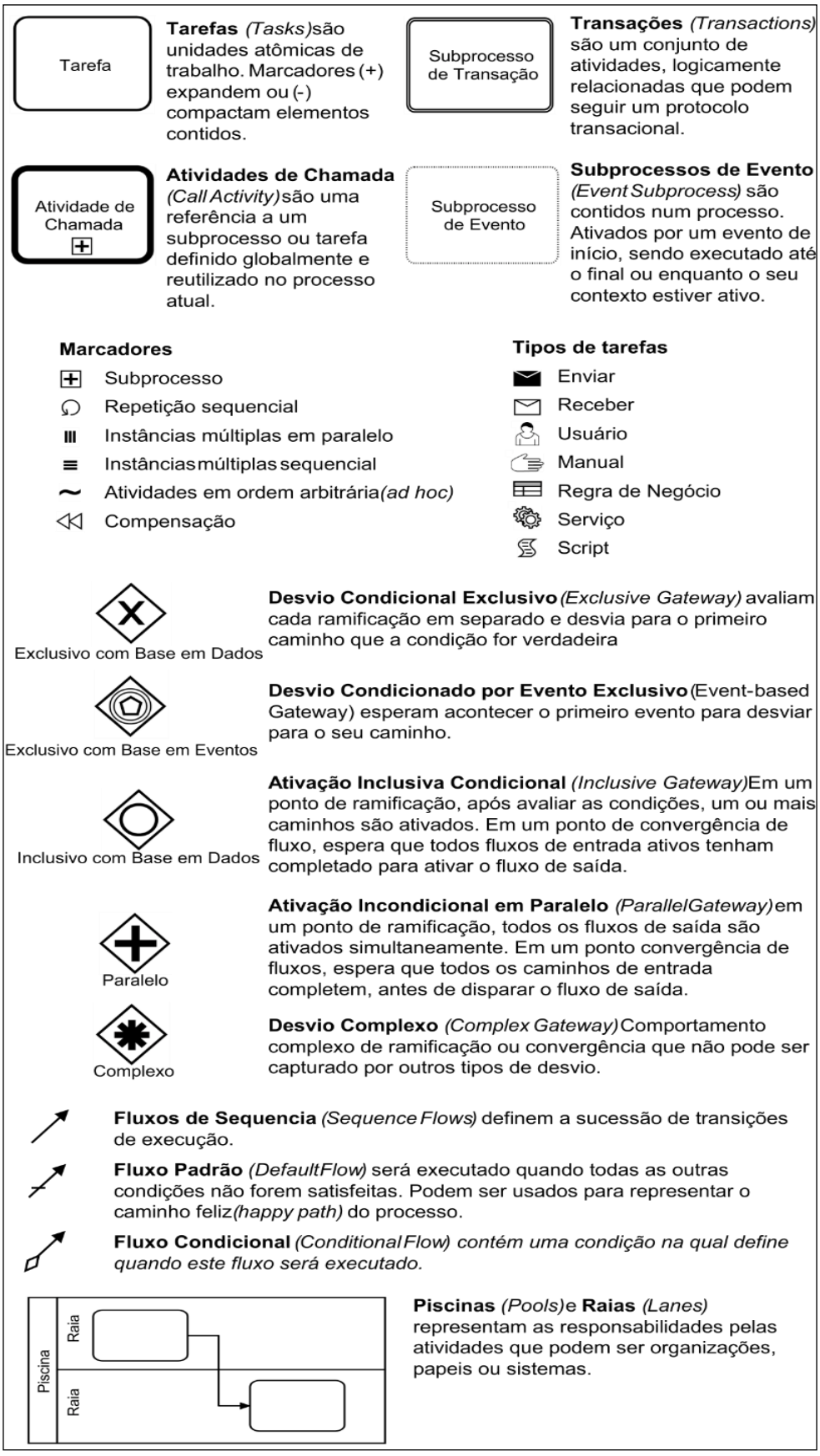
WIGGINS. A. **The twelve-factor app.** 2017. Disponível em:<https://12factor.net/pt_br/>. Acesso em jul. 2019.

ANEXO A – GUIA DE REFERÊNCIA DA NOTAÇÃO BPMN

Neste anexo, é apresentado um guia de referência elaborado por Bitencourt (2017), contendo os principais elementos para modelar processos com BPMN v2.0. Observa-se que, segundo o autor, a representação gráfica foi desenvolvida na ferramenta *Camunda Modeler*.

O guia original foi desmembrado em duas partes para melhor visualização dos elementos de BPMN v2.0. Assim, a **Figura 42** apresenta os elementos: atividades, *gateways*, fluxos e piscinas, enquanto a **Figura 43** mostra os tipos de eventos e suas possíveis ações.

Figura 42 – Elementos básicos da notação BPMN v2.0.



Fonte: extraído de Bitencourt, 2017.

Figura 43 – Elementos do tipo “eventos” da notação BPMN v2.0.

Fonte: extraído de Bitencourt, 2017.

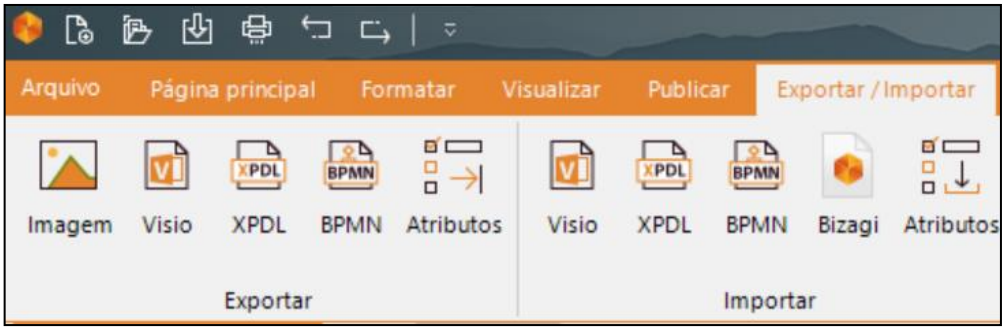
APÊNDICE A – ANÁLISE DAS FERRAMENTAS DE MODELAGEM

Este Apêndice contém uma análise das ferramentas de modelagem de processos de negócio *Bizagi Modeler, BPMN.IO, Heflo!* e *Bonita Studio.*

Na análise foram considerados os itens de exportação dos modelos de processos de negócio para os formatos PDF DOC e PNG. Também considerando a importação desses modelos em formato BPMN v2.0.

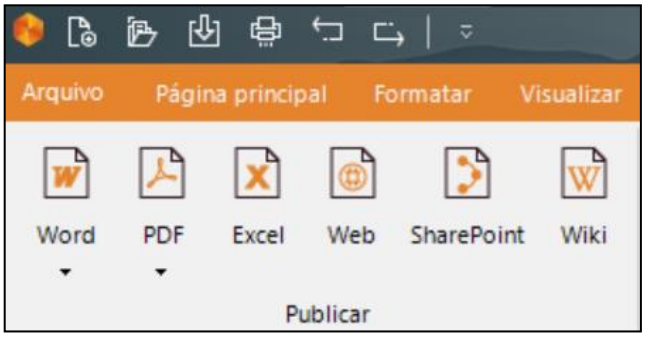
A ferramenta de modelagem *Bizagi Modeler* na versão 3.2.7.242 possui as opções de exportação da documentação referente aos modelos de processos de negócio nos formatos: PNG, VSDX, XDPL, BPMN, XML, e importação nos formatos anteriores com exceção do PNG, conforme é ilustrado pela **Figura 44**. Caso o modelador deseje publicar o modelo, a ferramenta disponibiliza a publicação da documentação nos formatos: DOCX, PDF, XSL, e para Web, como, por exemplo, no formato HTML com compartilhamento via SharePoint e Wiki, conforme é ilustrado pela **Figura 45**.

Figura 44 – Menu de exportação/importação da ferramenta *Bizagi Modeler*.



Fonte: *Bizagi* (2016).

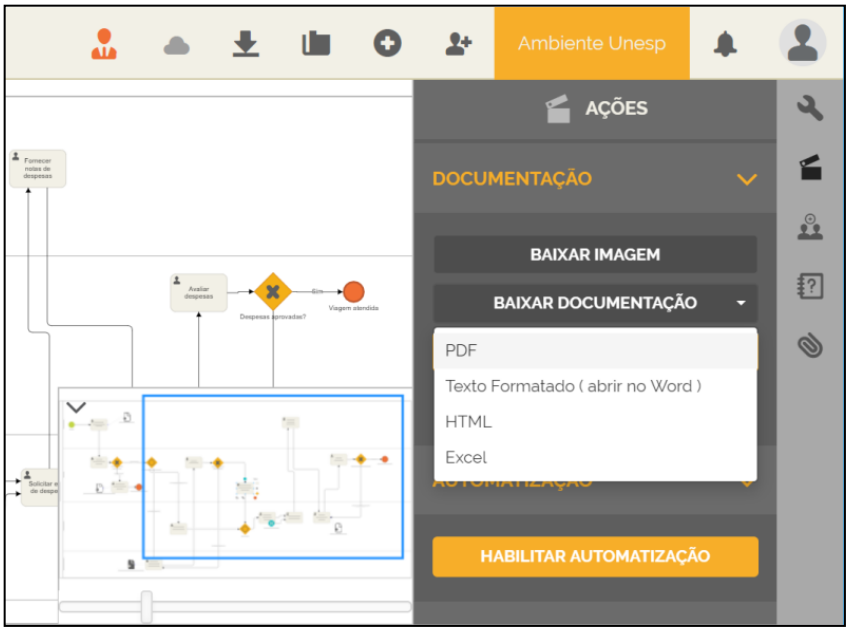
Figura 45 – Menu de publicação da Ferramenta *Bizagi Modeler*.



Fonte: *Bizagi* (2016).

No sistema de modelagem da ferramenta *Heflo!* também é possível fazer a importação/exportação de modelos de processos de negócio. Para o processo de importação só é aceitável no formato BPMN, no entanto, para o procedimento de exportação da documentação a ferramenta tem suporte aos formatos: PNG, PDF, DOCX, HTML e XSL, conforme se é ilustrado pela **Figura 46**.

Figura 46 – Menu de exportação/importação da ferramenta *Heflo!*.

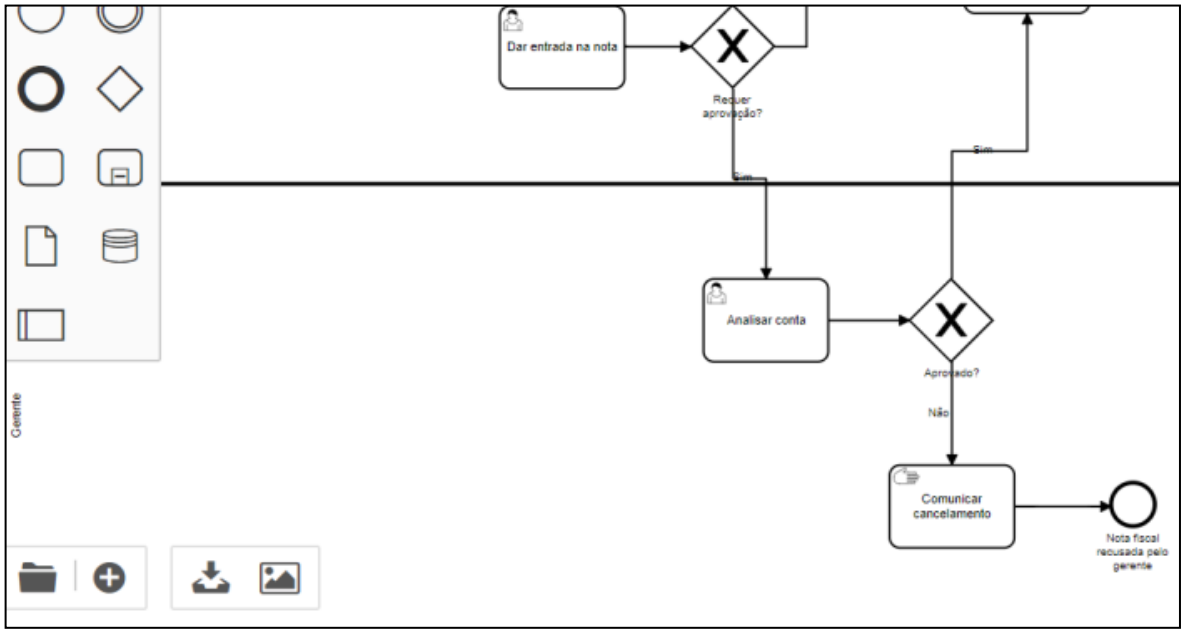


Fonte: *Helfo!* (2019).

O modelador *BPMN.IO* é disponível em uma plataforma Web, nesse sistema possui somente o recurso de exportação da documentação dos modelos de processos de negócio para os formatos: SVG e BPMN. Esse sistema de modelagem não consegue abrir modelos de processos que não estejam na versão 2.0 de BPMN. Na **Figura 47** é ilustrada uma parte de um modelo de processos de negócio com o menu de exportação da ferramenta *BPMN.IO* localizada do lado esquerdo na parte inferior.

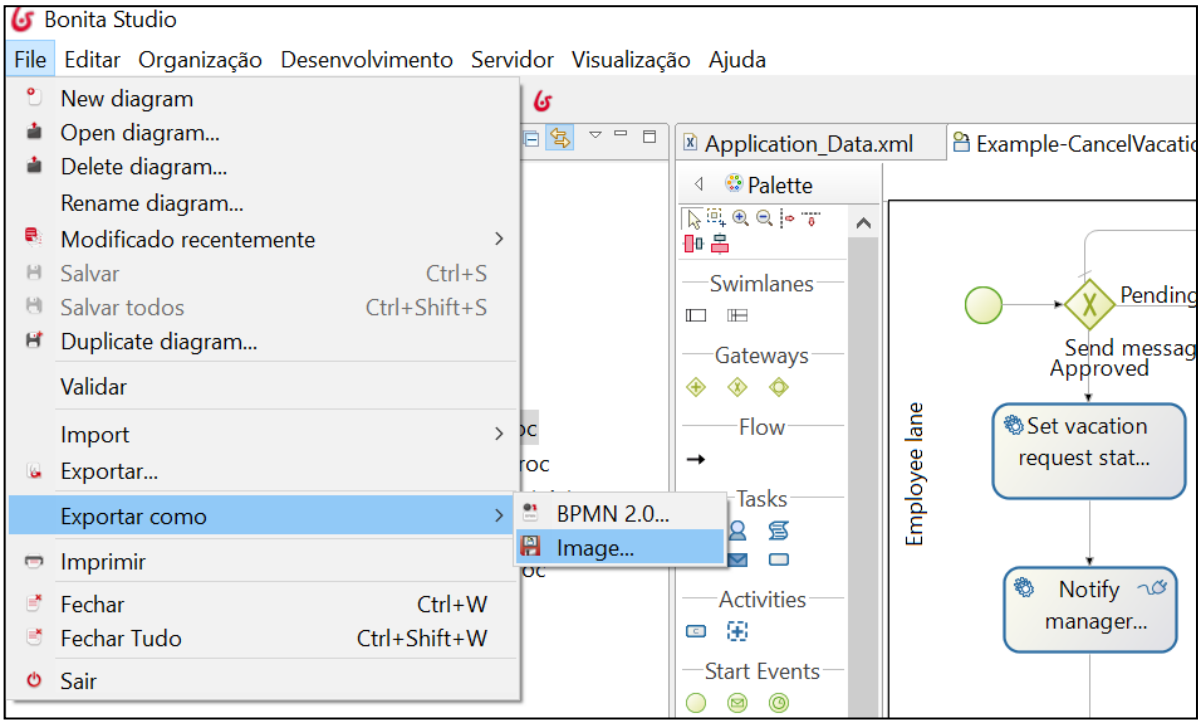
A ferramenta de modelagem *Bonita Studio*, não oferece recursos de importação de modelos de processos de negócio oriundos de outros modeladores em BPMN. Neste sistema de modelagem somente foi possível abrir os arquivos originados na própria ferramenta de ou exemplos de sua biblioteca virtual. Quanto aos recursos de exportação da documentação dos modelos de processos de negócio, esse sistema oferece suporte aos formatos BPMN de PNG, como é ilustrado pela **Figura 48.**

Figura 47 – Menu de exportação da ferramenta *BPMN.io*.



Fonte: *BPMN.io* (2019).

Figura 48 – Menu de exportação da ferramenta *Bonita Studio*.



Fonte: *Bonita Studio* (2019).

APÊNDICE B – FERRAMENTA MservSpec: CÓDIGO E INSTRUÇÕES DE USO (*CD-R anexo*)

Este Apêndice contém o código fonte em Node.js da ferramenta Web MservSpec v1.0, desenvolvida para especificar requisitos de microsserviços a partir de modelos de processos de negócio.

Além disso, este Apêndice contém instruções para a configuração de todos os recursos necessários para execução da ferramenta.

O conteúdo deste Apêndice encontra-se em formato digital, em CD-R anexo a esta dissertação, em pasta denominada “APENDICE B – Ferramenta MservSpec - código e instruções de uso”.

APÊNDICE C – MODELOS EM BPMN PARA OS ESTUDOS DE CASO” (*CD-R anexo*)

Este Apêndice contém os modelos de processos de negócio em BPMN v2.0 utilizados nos três estudos de caso apresentados nas *Seções 6.3* a *6.5*, visando avaliar o processo sistemático apresentado neste trabalho.

Os arquivos desses modelos foram exportados por meio da ferramenta de modelagem *Bizagi Modeler* para arquivos PDF e PNG, os quais também se encontram neste Apêndice.

O conteúdo deste Apêndice encontra-se em formato digital, em CD-R anexo a esta dissertação, em pasta denominada “APENDICE C – Modelos BPMN”.

APÊNDICE D – MICROSSERVIÇOS DESENVOLVIDOS (*CD-R anexo*)

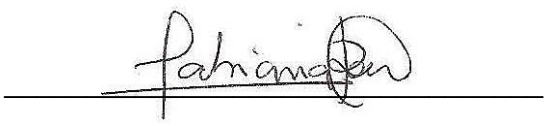
Este Apêndice contém todos os BPMSm compostos por microsserviços estudados nos três estudos de caso apresentados nas *Seções 6.3* a *6.5*, visando avaliar o processo sistemático apresentado neste trabalho.

Os BPMSm foram desenvolvidos em linguagem Node.js, sendo possível visualizá-las por meio da plataforma *Firebase*.

O conteúdo deste Apêndice encontra-se em formato digital, em CD-R anexo a esta dissertação, em pasta denominada “APENDICE D – Microsserviços Desenvolvidos”.

Autorizo a reprodução xerográfica para fins de pesquisa

Rio Claro, 06/09/2019.



Assinatura