UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

NATALIA SEBBEN

MODELO DE LOGS PARA IDENTIFICAÇÃO DE FALHAS DE PROCESSO EM ARQUITETURAS DE MICROSSERVIÇOS

CAXIAS DO SUL

2023

NATALIA SEBBEN

MODELO DE LOGS PARA IDENTIFICAÇÃO DE FALHAS DE PROCESSO EM ARQUITETURAS DE MICROSSERVIÇOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Prof. Dr. Helena Graziottin Ribeiro

CAXIAS DO SUL

2023

NATALIA SEBBEN

MODELO DE LOGS PARA IDENTIFICAÇÃO DE FALHAS DE PROCESSO EM ARQUITETURAS DE MICROSSERVIÇOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado(a) em 28/11/2023

BANCA EXAMINADORA

Prof. Dr. Helena Graziottin Ribeiro
Universidade de Caxias do Sul - UCS

Prof. Me. Marcos Eduardo Casa
Universidade de Caxias do Sul - UCS

Prof. Dr. Daniel Luís Notari
Universidade de Caxias do Sul - UCS

Este trabalho é dedicado às crianças adultas que, quando pequenas, sonharam em se tornar cientistas.

"As mais impressionantes façanhas humanas são, na realidade, o agregado de inumeráveis elementos isolados e cada um dos quais, em certo sentido, nada tem de extraordinário."

Dan Chambliss

RESUMO

O presente trabalho tem como objetivo colaborar com a identificação de falhas de processo e de negócios em uma arquitetura de microsserviços. Para isso, foi feita uma revisão sistemática para avaliar qual o cenário atual do tópico, buscando por evidências sobre abordagens de *logging* em arquiteturas de microsserviços. Como resultado, foram levantadas algumas práticas já utilizadas e também dificuldades e lacunas ainda sem solução. A identificação de falhas no processo de negócio foi a maior dificuldade percebida, uma vez que existem grandes dificuldades em extrair informações dos dados de *logs*, utilizados para o rastreamento de falhas. Foi feita uma proposta para a resolução do problema, buscando centralizar e padronizar a estrutura dos *logs* de uma forma que as informações específicas do microsserviço não sejam perdidas e possam ser melhor aproveitadas. Nesta proposta, os *logs* são gravados de modo a corresponder as etapas do processo de negócio para melhor compreensão sobre onde ocorre o problema, e o gerenciamento dos *logs* é feito de forma centralizada por um *middleware*. Essa proposta não limita a utilização de determinadas tecnologias, ela define um modelo padrão de arquitetura e de interfaces de comunicação que podem ser implementadas de diferentes formas, em diferentes tecnologias. Foi criado um protótipo para a realização dos testes, e a abordagem se mostrou efetiva para seu propósito. Em primeiro momento, a abordagem não substitui os mecanismos de *logging* e *tracing* distribuído convencionais, mas complementa com informações relevantes para essa face do problema. Concluiu-se também que é possível continuar avançando com novos estudos nesta abordagem para avaliar a utilizaçao da mesma para suprir também a identificação de falhas técnicas, de *software* ou *hardware*.

Palavras-chave: Arquitetura de Microsserviços. *Logging*. Identificação de Falhas.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

DDD *Domain-Driven Design*

HTTP *HyperText Transfer Protocol*

REST *REpresentational State Transfer*

API *Application Programming Interfaces*

BPMN *Business Process Model and Notation*

IA Inteligência Artificial

MOM *Message-Oriented Middleware*

FTP File Transfer Protocol

BLOB Binary Large OBject

MVC Model - View - Controller

JSON JavaScript Object Notation

SUMÁRIO

1 INTRODUÇÃO

A arquitetura de microsserviços é uma abordagem de desenvolvimento de sistemas distribuídos na qual o sistema é composto por um conjunto de serviços que operam em colaboração para atender aos objetivos do mesmo. Neste modelo, cada serviço é responsável por uma parcela do processo de negócio do grande conjunto ao qual o sistema atende. Este modelo de arquitetura traz grandes benefícios, principalmente para sistemas de alta complexidade, pois permite a divisão de grandes problemas em partes menores, facilitando o crescimento e manutenabilidade do mesmo. Além disso, por sua natureza distribuída, também facilita o processo de paralelização e consequentemente sua escalabilidade (STEEN; TANENBAUM, 2023; NEWMAN, 2021). Em contrapartida, aplicar este tipo de abordagem também agrega grande complexidade ao projeto. O monitoramento e *logging* do sistema, por exemplo, operações essenciais e bastante simples de realizar em sistemas tradicionais monolíticos, precisam de muita atenção dentro de uma arquitetura de microsserviços. Neste cenário não existe apenas um grande serviço e sim um conjunto de serviços menores, com suas particularidades em virtude da porção do processo de negócio ao qual atendem, e que podem inclusive estar em diferentes servidores ou processando mais de um tarefa paralelamente (NEWMAN, 2021; BURNS, 2018; CINQUE; CORTE; PECCHIA, 2021).

Apesar de não existir uma literatura canônica sobre como este problema deve ser resolvido, existem algumas recomendações para coleta e agregação dos dados de monitoramento e *logging* neste modelo de arquitetura. Os principais requisitos são: 1) deve haver uma forma de centralização dos dados de monitoramento e *logging* de todo o sistema, e 2) também deve ser possível identificar qual ou quais microsserviços estão ou originaram algum problema, e compreender o cenário em que isso ocorreu (CERNY *et al.*, 2023; LEE *et al.*, 2023; NEWMAN, 2021). Assim, é possível ter a visão do todo e das partes.

Entretanto, cumprir estes requisitos não é uma tarefa fácil. Existem diversas formas já utilizadas e testadas pela iniciativa acadêmica ou de profissionais da indústria, mas ainda não existe a definição de uma solução para todos os problemas (ZHOU *et al.*, 2023; TAMBURRI; MIGLIERINA; NITTO, 2020). Focando nas questões relacionadas aos *logs*, que é o objeto de estudo do presente trabalho, tem-se como principal problema o grande volume de *logs* gerados e seu conteúdo heterogêneo. Isso ocorre porque os *logs* são gerados por diferentes serviços, que podem ser de tecnologias distintas, que abrangem diferentes partes do negócio e por isso possuem contextos diferentes (BURNS, 2018; NEWMAN, 2021).

Encontrar problemas e sua origem em meio a essa grande quantidade de *logs* com diferentes formatos não é algo simples, principalmente se depender da interpretação humana. Atualmente, ainda existe grande dependência em processos manuais de verificação de *logs*, o que é um processo bastante demorado e muitas vezes ineficiente, mesmo que performado por profissionais experientes (LEE *et al.*, 2023; WANG *et al.*, 2021; ZHOU *et al.*, 2023).

Uma abordagem bastante utilizada é a aplicação de algoritmos de inteligência artificial com a criação de modelos para identificação e localização da origem de falhas. Um dos principais problemas dessa abordagem é que de tempos em tempos esses modelos precisam ser refeitos em virtude da natureza dinâmica dos microsserviços, que estão em constante mudança. Outro problema é que, na maioria dos algoritmos e estratégias de IA (Inteligência Artificial) utilizadas existe uma falta de transparência, já que na maior parte dos casos, apesar da correta identificação, não fica claro o motivo pelo qual foi realizada (SOLDANI; BROGI, 2023).

Outra abordagem também aplicada é utilização de *middlewares adapters* para transformar os *logs* heterogêneos em uma estrutura padrão (BURNS, 2018). A aplicação desta estratégia pode variar bastante conforme as tecnologias e ferramentas empregadas, mas existe grande probabilidade de perda de dados específicos do contexto dos microsserviços. Isso ocasiona perda de especificidade dos dados e consequentemente reduz a quantidade de informações que se pode obter a partir dos mesmos.

Outro fator a ser destacado, é que a arquitetura de microsserviços é fortemente influenciada pelo domínimio de negócio ao qual atende, por ser baseada em DDD (*Domain-Driven Design*) (NEWMAN, 2021; WOLFF, 2017). Sendo assim, as especificidades de contexto dos microsserviços se tornam ainda mais importantes para a identificação de falhas. Os dados de contexto são bastante importantes para a identificação de falhas de *software*, mas se tornam ainda mais importantes para a identificação de falhas relacionadas ao levantamento de requisitos, mapeamento do processo e regras de negócio, que são mais sutis e mais difíceis de identificar (TAMBURRI; MIGLIERINA; NITTO, 2020; CINQUE; CORTE; PECCHIA, 2021).

1.1 OBJETIVOS

Por este motivo, o presente trabalho busca propor um modelo para o registro de *logs* que facilite a identificação de falhas relacionadas ao processo de negócio, e que possa ser aplicado independente da escolha tecnológica dos microsserviços. Sendo assim, os objetivos específicos tratam-se de resolver os seguintes problemas:

- Heterogeneidade dos *logs*: como a arquitetura de microsserviços é composta por diferentes microsserviços, sendo que cada um é bastante específico a um problema ou domínio de negócio, podendo inclusive possuir diferentes tecnologias, muito provavelmente os *logs* não terão a mesma estrutura e informações.
- Descentralização e integração dos *logs*: se cada micosserviço produzirá seus registros de *logs* de forma descentralizada, então para ter a visão total dos processos é necessário de alguma forma integrar estes *logs* de cada microsserviço.
- Retrabalho na codificação de mecanismos de *logs*: evitar que ocorram reimplementações para a gestão de *logs* em cada microsserviço.

- Dificuldade na identificação de falhas de processo ou de negócio via *logs* técnicos tradicionais: tornar as informações relevantes ao processo de negócio mais explícitas em meio ao grande volume de *logs* gerados pelos microsserviços.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está organizado da seguinte maneira:

- Este capítulo apresentou uma introdução sobre o trabalho, incluindo seu cenário, motivações, objetivo geral e objetivos específicos.
- O Capítulo 2 e o Capítulo 3 tem por objetivo apresentar conceitos teóricos sobre a arquitetura de sistemas distribuídos e, mais especificamente, microsserviços.
- O Capítulo 4 apresenta uma pesquisa de revisão sistemática realizada para compreender o cenário do tema e quais suas aplicações atualmente.
- No Capítulo 5 é feita uma proposta de solução para o problema apresentado, juntamente com um exemplo de aplicação e testes para validá-la.
- Na sequência, o Capítulo 6 apresenta os testes da aplicação desenvolvida para prova de conceito.
- Por fim, o Capítulo 7 apresenta as considerações finais do trabalho.

2 ARQUITETURAS DE SISTEMAS DISTRIBUÍDOS

Sistemas distribuídos são sistemas compostos de partes complexas de *software* em que seus componentes estão, por definição, dispersos por várias máquinas. No entanto, para lidar com essa complexidade é essencial que os mesmos sejam organizados de forma apropriada. Essa organização ocorre por meio da especificação da arquitetura lógica e física do sistema (STEEN; TANENBAUM, 2023).

A organização lógica de um sistema, como é definida a arquitetura de *software* (RICHARDS; FORD, 2020; BASS; CLEMENTS; KAZMAN, 2021), deve estabelecer como os vários componentes de *software* do sistema distribuído estarão organizados e como devem interagir entre si. Arquitetura de sistema, por sua vez, define como estes componentes serão instanciados nas máquinas propriamente ditas, ou seja, como e quais estruturas de *hardware* serão utilizadas (STEEN; TANENBAUM, 2023). Questões relacionadas à arquitetura física não serão abordadas neste trabalho, portanto serão tratados apenas aspectos relacionados à arquitetura lógica.

2.1 MIDDLEWARE

O conceito de *middleware* é bastante difundido no desenvolvimento e operação de sistemas distribuídos, é uma estrutura versátil que pode ser aplicada na resolução de diferentes tipos de problemas. Um *middleware* é, basicamente, uma camada de *software* intermediária que pode ter diferentes funcionalidades. Geralmente é utilizado entre as aplicações e o sistema operacional, como um gerenciador de recursos, ou ainda entre uma aplicação e outra, para manter a compatibilidade entre diferentes interfaces. Também pode ser implementado como elemento centralizador contendo funcionalidades e componentes utilizados por um conjunto de aplicações, evitando que estes precisem ser reimplementados em cada uma delas (STEEN; TANENBAUM, 2023).

2.2 MODELOS ARQUITETURAIS

A arquitetura de software provê uma série de técnicas para auxiliar no manejo da complexidade que potencialmente existente no desenvolvimento de sistemas, possibilitando que ideias abstratas possam ser melhor trabalhadas (BASS; CLEMENTS; KAZMAN, 2021).

Um modelo de arquitetura, por sua vez, é composto por algumas definições, tais como: quais serão seus componentes, como esses componentes estarão interligados uns aos outros, quais dados serão trocados por eles, e como esses componentes, em conjunto, configuram um sistema (STEEN; TANENBAUM, 2023). Outro aspecto importante, chamado de conector, é o elemento que fará a conexão entre os componentes. Ou seja, trata-se do canal utilizado para me-

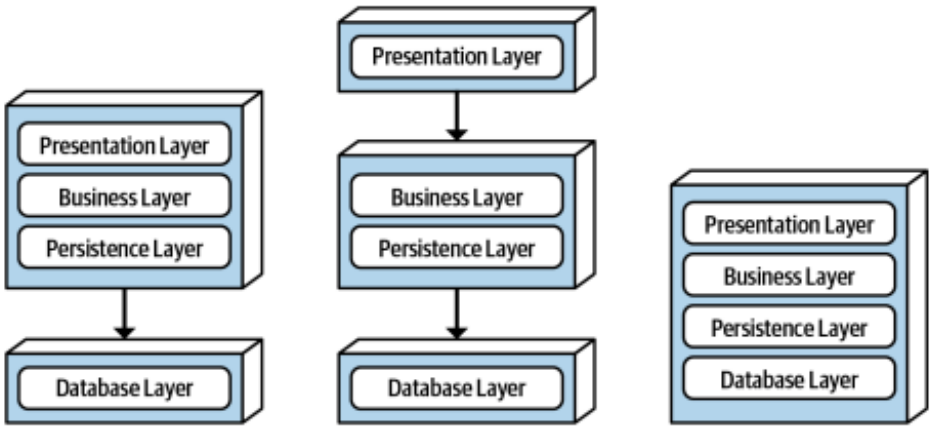
diar a comunicação entre os componentes, possibilitando o fluxo de controle e dados entre eles. (BASS; CLEMENTS; KAZMAN, 2021).

De acordo com Steen e Tanenbaum (2023), existem diversos modelos arquiteturais compostos pela variação dessas especificações, sendo os mais importantes para sistemas distribuídos: a arquitetura em camadas, a arquitetura orientada a serviços e a arquitetura *publish-subscribe*. Entretanto, em grande parte das aplicações em sistemas distribuídos, os estilos podem aparecer de forma combinada.

2.2.1 Arquitetura em Camadas

Nesta arquitetura, os componentes são organizados hierarquicamente em camadas, na qual a camada superior faz chamadas, geralmente aguardando um retorno, para a camada imediatamente inferior. Apenas em casos exepcionais, uma camada inferior realizará uma chamada à sua camada superior. Apesar da grande popularidade deste tipo de arquitetura, um de seus maiores problemas é apresentar alto nível de acoplamento, ou seja, dependência entre as camadas. Ao realizar alterações em uma camada, muito provavelmente as outras camadas também precisarão de ajustes para se adequarem ao novo formato, para que sua comunicação continue funcional (STEEN; TANENBAUM, 2023). A Figura 1 exibe exemplos de arquitetura em camadas com diferentes separações a nível de código fonte.

Figura 1 – Exemplos de arquitetura em camadas



Fonte: Richards e Ford (2020)

Os componentes dentro de uma arquitetura em camadas são organizados de forma horizontal, na qual cada camada datém uma responsabilidade dentro da aplicação como um todo. Não existe uma restrição sobre a quantidade de camadas que podem existir, pois isso pode variar conforme o cenário de cada aplicação (RICHARDS; FORD, 2020). Entretando, uma forma comumente utilizada é a divisão em três níveis lógicos: interface, processamento e dados. O nível de interface é responsável pela interação com o usuário ou com aplicações externas. Já o nível de processamento, por sua vez, é responsável pela execução das funcionalidades do sistema apli-

cando suas regras de negócio. E por fim, o nível de dados, responsável pela comunicação com as fontes de dados do sistema (STEEN; TANENBAUM, 2023).

2.2.2 Arquitetura *publish-subscribe*

Com o objetivo de prover maior flexibilidade ao crescimento e modificação dos sistemas, uma grande parcela dos sistemas distribuídos adotou uma arquitetura conhecida como *publish-subscribe*. Nessa abordagem, que traz forte separação entre o processamento e a coordenação, o sistema é visto como uma coleção de processos autônomos. A coordenação, neste cenário, trata-se da comunicação e cooperação entre esses processos, para formar o sistema como um todo.

O modelo de Cabri, Leonardi e Zambonelli (2000) proveu a taxonomia de modelos de coordenação, que podem ser aplicados a vários tipos de sistemas distribuídos. A partir deste, Steen e Tanenbaum (2023) apresentam uma distinção entre os modelos de acordo com duas dimensões: temporal e referencial, que pode ser visualizada na Figura 2.

Figura 2 – Exemplo de diferentes formas de coordenação

	Temporalmente Acoplado	Temporalmente Desacoplado
Referencialmente Acoplado	Direto	Caixa de Mensagem
Referencialmente Desacoplado	Baseado em Eventos	Espaço de Dados Compartilhados

Fonte: Steen e Tanenbaum (2023)

A coordenação direta caracteriza-se pelo acoplamento referencial e temporal dos processos. O acoplamento referencial ocorre quando os processos só podem se comunicar conhecendo seus nomes ou identificadores. Já o acoplamento temporal ocorre quando ambos processos precisam estar operantes para que a comunicação aconteça.

Um tipo diferente de coordenação ocorre quando os processos, apesar de acoplados referencialmente, não estão acoplados temporalmente. Este modelo chama-se caixa de mensagem, e nessa abordagem não é necessário que os processos estejam operantes no mesmo momento para que a comunicação aconteça. Ao invés disso, o processo remetente deixa uma mensagem na caixa de mensagens para que seja recebida quando o processo destinatário estiver operante. Entretanto, este modelo mantém o acoplamento referencial, uma vez que a caixa de mensagens precisa ser conhecida e referenciada.

Quando os processos são referencialmente desacoplados mas temporalmente acoplados, ocorre a coordenação orientada a eventos. Para que ocorra a comunicação, os processos notificam a ocorrência de eventos a um serviço ou *middleware* responsável por encaminhá-las aos seus respectivos destinatários (MüHL LUDGER FIEGE, 2006). As notificações podem possuir

diversos formatos, e os processos assinam os tipos de notificação nos quais estão interessados. Neste modelo os processos não se conhecem de forma explícita, mas ainda é necessário que ambos estejam operantes para a comunicação ocorrer: os processos remetentes para realizar o envio da notificação, e os processos destinatários para recebê-la.

Por sua vez, processos referencialmente e temporalmente desacoplados caracterizam a coordenação por espaço de dados compartilhados. Nesta proposta os processos comunicam-se através de tuplas, que são registros de dados estruturados compostos por vários campos, bastante similar ao registro de uma tabela no banco de dados. Quaisquer tipos de tuplas podem ser depositadas em um espaço de dados compartilhados, e para que possam ser recuperadas, o processo deve prover um padrão de consulta para ser comparado às tuplas. Quando compatíveis ao padrão informado, as mesmas serão retornadas ao processo que está requisitando-as e são retiradas do espaço de dados compartilhados.

O espaço de dados compartilhados é muitas vezes combinado com coordenação baseada em eventos. Neste cenário, os processos assinam tuplas provendo um padrão de busca, e quando uma tupla compatível com o padrão informado é adicionada no espaço compartilhado, os assinantes que o informaram são notificados.

Um aspecto importante de sistemas *publish-subscribe* é que os processos precisam descrever ao *middlware* em quais eventos estão interessados. Essa descrição normalmente consiste em pares de atributo/valor ou atributo/intervalo, ou seja, os dados não são explicitamente identificados pelos remetentes e destinatários. Para que o *middleware* detecte um evento, espera-se que o atributo especificado exista e apresente, respectivamente, o mesmo valor ou um valor contido no intervalo definido.

2.2.3 Arquitetura Orientada a Serviços

Os problemas de dependência entre componentes levaram a um modelo arquitetural mais flexível com um conjunto de entidades independentes, e então nasceu o conceito de arquitetura orientada a serviços. Neste modelo cada entidade encapsula um serviço, e cada uma delas é executada em processos distintos. Entretanto, esta característica de forma isolada não necessariamente diminui o problema de acoplamento enfrentado pela arquitetura de camadas (STEEN; TANENBAUM, 2023).

Na abordagem arquitetural baseada em objetos, cada objeto corresponde a um componente, e estes são conectados pelo mecanismo de chamada de procedimentos, que no cenário de sistemas distribuídos pode ser feito de forma remota. Esse modelo é atrativo porque provê naturalmente o encapsulamento dos dados (estado do objeto) e suas operações (métodos do objeto) em uma única entidade. A interface fornecida pelo objeto esconde os detalhes de sua implementação, fazendo com que o mesmo possa ser considerado completamente independente do ambiente (STEEN; TANENBAUM, 2023).

Seguindo o conceito de arquitetura orientada a serviços, e inspirado pela orientação a objetos e pelo DDD - *Domain-Driven Design*, a partir do encapsulamento e da presença de contextos limitados (domínios), surgiu a arquitetura de microsserviços. Como resultado, obtiveram-se estruturas modulares de baixo acoplamento, possibilitando a divisão de grandes e complexos sistemas em partes menores, que podem ser implementados em diferentes tecnologias. Cada microsserviço executa como um domínio ou módulo independente, embora o mesmo possa fazer uso de outros serviços ou microsserviços. Essa comunicação, entretanto, ocorre via rede e exige meios que também permitam baixo acoplamento (WOLFF, 2017; STEEN; TANENBAUM, 2023; RICHARDS; FORD, 2020). Na Seção 2.3, essa questão será melhor esclarecida, enquanto no Capítulo 3, a arquitetura de microsserviços será abordado com maior detalhamento.

Um sistema distribuído com arquitetura orientada a serviços é constituído basicamente pela composição de vários serviços. Entretanto, na composição do sistema de microsserviços pode ser que nem todos eles pertençam à mesma organização. Partes do fluxo do sistema podem depender de serviços complementares como integradores de pagamento ou integradores logísticos. Essa condição adiciona problemas similares aos de integração de aplicações, fazendo com que a complexidade de desenvolver esse tipo de arquitetura seja tanto a composição de serviços, como também garantir que todos estes operem em harmonia (STEEN; TANENBAUM, 2023).

2.3 COMUNICAÇÃO

A comunicação é uma função fundamental para os sistemas distribuídos, no qual processos que executam em diferentes máquinas precisam trocar informações para formar uma unidade. Necessário para viabilizar o desenvolvimento distribuído em larga escala, existem diferentes formas dessa comunicação acontecer, variando em características e estratégias. Entretando, a utilização de *middlewares* para a abstração da função de comunicação é amplamente utilizada (STEEN; TANENBAUM, 2023).

Quando um processo precisa comunicar-se com outro, o processo remetente envia uma mensagem para o destinatário, podendo haver um ou mais destinatários. O primeiro aspecto a ser observado é persistência ou não da mensagem. Quando a comunicação ocorre de forma persistente, a mensagem enviada pelo remetente fica armazenada no *middleware* de comunicação até que a mesma seja lida e retirada do mesmo. Já quando ocorre de forma transiente, a mensagem fica disponível apenas enquanto o remetente está enviando-a ou o destinatário está recebendo-a. A comunicação transiente implica na necessidade de ter remetente e destinatários em execução no mesmo momento, o que caracteriza a comunicação síncrona. Enquanto a persistência da mensagem permite que os mesmos executem em momentos diferentes, o que viabiliza a comunicação assíncrona. Quando a abordagem assíncrona é utilizada, o processo remetente continua suas tarefas imediatamente após o envio da mensagem. Em contra partida, quando a abordagem síncrona é utilizada o processo remetente fica bloqueado, aguardando sua liberação. Existem três formas na qual o remetente pode ser liberado: ao ser avisado que a transmissão da

mensagem será realizada; ao ser avisado que a mensagem foi entregue ao destinatário; ou ao receber um retorno do destinatário com a resposta (STEEN; TANENBAUM, 2023).

Existem alguns modelos de comunicação que são bastante utilizados para a comunicação de sistemas distribuídos, entre eles estão os modelos *request-reply* e *publish-subscribe*.

2.3.1 Request-Reply

Este modelo de comunicação baseia-se na arquitetura cliente-servidor, na qual o servidor é o elemento que dispõe de serviços, e o cliente é o elemento que precisa utilizar-se destes serviços. A forma de solicitar estes serviços é através de uma requisição (*request*), que o cliente deve enviar ao servidor. Este, por sua vez, irá processar essa requisição e enviar ao cliente uma resposta (*reply*) (STEEN; TANENBAUM, 2023; FIELDING, 2000).

Uma forma amplamente utilizada para o envio dessa requisição é a utilização do protocolo para transferência de dados na internet, o HTTP (*HyperText Transfer Protocol*), que por sua vez, adota um modelo de arquitetura chamado Rest (Represential State Transfer) (FIELDING, 2000). Este modelo procura padronizar a interface para o tráfego de dados buscando proporcionar maior escalabilidade e assertividade no processamento das requisições. Além de operar em modo cliente-servidor, utiliza-se do conceito *stateless*, no qual o servidor não possui registro do cliente e seu estado, então o mesmo deve enviar todas as informações necessárias para que o servidor identifique o que precisa ser feito e possa realizar sua tarefa (STEEN; TANENBAUM, 2023; FIELDING, 2000).

Web services, por sua vez, são serviços disponíveis através da *web*, construídos com o objetivo de oferecer suporte às necessidades de sistemas clientes que se conectam a eles. Rest APIs tratam-se de um tipo de *web service* que implementa esse modelo de comunicação juntamente ao padrão Rest. Neste modelo, sistemas clientes utilizam-se de APIs (*Application Programming Interfaces* ou Interfaces de Programação de Aplicações), para ter acesso aos dados de um sistema ou a recursos que os modificam (MASSE, 2011).

2.3.2 Publish-Subscribe

Este modelo de comunicação é o mesmo que foi abordado para a coordenação de processos. Um exemplo amplamente utilizado para permitir a comunicação assíncrona entre aplicações são os *middlewares* orientados a mensagens (*Message-Oriented Middleware* - MOM). Neste modelo diversas filas podem ser criadas, e as aplicações devem assinar (*subscribe*) aquelas que forem de seu interesse. Para que ocorra a comunicação, as aplicações enviam mensagens (*publish*) para as filas desejadas, e estas mensagens serão transferidas para as aplicações que assinaram suas filas quando cada uma estiver operante (STEEN; TANENBAUM, 2023).

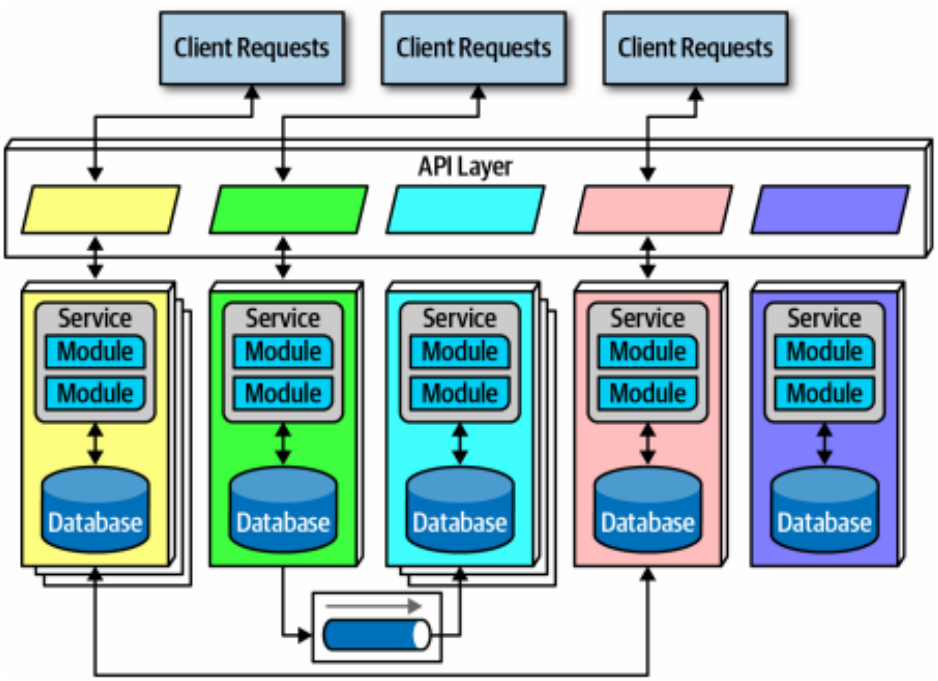
Os *middlewares* de mensageria permitem que a comunicação ocorra de forma persistente, ou seja, quando as filas recebem uma mensagem, a mesma permanece armazenada no

middleware até que seja retirada da fila por alguma aplicação ou configuração. Nesse modelo não existe acoplamento temporal, pois não é necessário que as aplicações destino estejam operantes no momento do envio da mensagem para recebê-la. Neste formato, remetente e destinatários podem executar de forma totalmente independente, permitindo o formato assíncrono (STEEN; TANENBAUM, 2023).

3 MICROSSERVIÇOS E LOGS

Conforme citado anteriormente, a arquitetura de microsserviços é uma abordagem orientada a serviços. Esta defende que a lógica necessária para resolver um grande problema pode ser melhor construída, executada e gerenciada se decomposta em uma coleção de menores partes relacionadas (ERL, 2017). Em outras palavras, é uma abordagem arquitetural para a modularização do software, com o objetivo de dividir grandes sistemas em partes menores (WOLFF, 2017). Tratam-se de serviços independentes modelados em torno de domínios de negócio, mas que podem trabalhar em conjunto. Encapsulam seus dados e complexidades de regras de negócio como uma caixa preta, e disponibilizando uma interface aos outros serviços para que possam utilizar suas funcionalidades (NEWMAN, 2021).

Figura 3 – Exemplo de uma arquitetura de microsserviços



Fonte: Richards e Ford (2020)

3.1 MODELAGEM ORIENTADA A DOMÍNIO

Modelagem orientada a domínio (DDD - *Domain-Driven Design*) trata-se de uma abordagem na qual a modelagem do *software* e organização das equipes envolvidas deve respeitar a estrutura dos domínios de negócio aos quais atende. Deve ser utilizada como uma forma de organizar e abstrair o conhecimento sobre o processo e regras de negócio, além de estabelecer uma linguagem comum entre integrantes com visão técnica e de negócio das equipes (EVANS, 2003).

Essa técnica permite a estrturação do código de forma a melhor representar o domínio do mundo real no qual o *software* opera, e é aplicado aos microsserviços para definir os limites de cada microsserviço. Modelando os serviços em torno dos domínios de negócio, é possível facilitar a liberação de novas funcionalidades no software. Liberar funcionalidades que exijam alterações em mais de um microsserviço é um trabalho difícil, é necessário coordená-lo entre diferentes serviços, e possívelmente entre diferentes equipes. A modelagem deve ser pensada para evitar ao máximo este cenário (NEWMAN, 2021).

3.2 ARQUITETURA ORIENTADA A EVENTOS

Em microsserviços orientados a eventos, a comunicação entre os serviços é feita pelo disparo e consumo de eventos. Esses eventos podem ser consumidos por um ou mais serviços, conforme a necessidade. Ao consumir um evento, os serviços aplicam suas regras de negócio, e a partir disso podem emitir seus próprios eventos ou performar quaisquer outras ações necessárias (BELLEMARE, 2020).

Os eventos são a base da comunicação, no qual todos os dados compartilháveis são publicados em um canal de eventos, pelo qual os serviços se comunicam, formando um histórico detalhado do que aconteceu no sistema. Os eventos são bastante flexíveis e podem comunicar diferentes acontecimentos, já que o evento é o próprio dado. Ou seja, ele contém os detalhes do que ocorreu, caracterizando seu cenário e fornecendo informações para as tratativas necessárias a partir dele. Além de funcionar como armazenamento de dados, é uma forma assíncrona de comunicação entre serviços (BELLEMARE, 2020).

3.3 TOLERÂNCIA A FALHAS

Microsserviços devem ser arquitetados com base no conceito de "feitos para falhar". Isso significa que, nesta arquitetura composta por vários serviços, algum deles vai falhar em algum momento e o sistema precisa estar pronto para se recuperar perante o ocorrido. Sendo assim, recursos de monitoramento e *logging* tornam-se essenciais para aferir a funcionalidade do sistema, localizar e entender mais sobre o cenário de falhas ocorridas para que seja possível corriji-las. Além disso, mecanismos de tolerância a falhas também são necessários, para que os microsserviços se recuperem após a mesma *logging* (NEWMAN, 2021). Falhas, neste caso, são consideradas situações nas quais o sistema não se comporta conforme o esperado. Se não forem tratadas adequadamente, algumas falhas podem causar a completa indisponibilidade do mesmo (KLEPPMANN, 2017).

Existem três tipos de falha que podem acometer sistemas: a falha de *hardware*, a falha de *software* e a falha humana. A falha de *hardware* é aquela em que o problema acontece nos componentes físicos dos computadores. A falha de *software*, por sua vez, ocorre quando o problema é causado pelo código fonte do sistema. Sistemas são feitos e utilizados por hu-

manos, então em todas as partes do processo pode ocorrer falha humana. Um bom exemplo seria pensar em falhas geradas por uma configuração equivocada, neste cenário o *hardware* e o *software* estão funcionando perfeitamente conforme o esperado, mas o erro de configuraçao gera uma falha no sistema (KLEPPMANN, 2017). Outro exemplo pode ser dado por uma regra de negócio que foi estabelecida de forma equivocada, ou etapas necessárias de um processo que não foram devidamente mapeadas. Mesmo que o *hardware* e o *software* funcionem conforme o planejado, ainda haverá falha pois o sistema não conseguirá completar algum processo ao qual foi designado. Este tipo de falha é mais difícil de detectar, pois pode não gerar falha explícita, mas não ter o efeito que o usuário esperava que tivesse. E mesmo que uma falha seja gerada, esta pode passar despercebida se for vista apenas com olhar técnico. Nesse caso, o programador verifica que o sistema está funcionando conforme o que foi previsto, e a falha foi gerada porque foi prevista, enquanto talvez fosse necessário olhar com a visão de processo, para observar uma falha no levantamento de requisitos.

3.4 LOGS E MONITORAMENTO

Para todos os tipos de falha, é essencial que exista uma forma de avaliar o funcionamento do sistema e suas ocorrências, principalmente quando algo não sair conforme o esperado (NEWMAN, 2021; KLEPPMANN, 2017). Para isso, são utilizadas estratégias de monitoramento e *logging*, que consigam extrair e expressar o estado do sistema em tempo real e em momentos específicos (quando ocorreu uma falha, por exemplo). Essa abordagem é fácilmente utilizada em sistemas monolíticos, porém quando aplicada a microsserviços, o problema se torna bem mais complexo. Primeiramente, é preciso monitorar não um, mas diversos serviços. Se apenas um destes serviços falhar, pode ocasionar falhas em vários outros que dependem dele, ou até deixar o sistema indisponível. Os serviços podem estar inclusive em locais físicos distintos (tanto geograficamente quando a nível de *hardware*), com diferentes sistemas operacionais (NEWMAN, 2021; BURNS, 2018). Outro fator é que um sistema construído em arquitetura de microsserviços pode ser composto por diversas tecnologias diferentes, e cada serviço possuirá suas especificidades conforme o domínio de negócio ao qual atende. Isso ocasiona um problema de heterogeinidade de dados, no qual os dados utilizados para monitoramento e *logging* são gerados em diferentes formatos (NEWMAN, 2021; BURNS, 2018).

Um conceito importante dentro do monitoramento, é a observabilidade. Nem sempre monitorar dados e *logs* será efetivo para entender como um sistema está se comportando, principalmente no cenário de microsserviços, em que as informações estão todas dispersas pelos inúmeros serviços que compõe o sistema. Para que o monitoramento realmente funcione, é preciso que sejam empregadas estratégias de observabilidade, e esta por sua vez trata-se do quanto é possível compreender o estado interno do sistema a partir de dados externos (NEWMAN, 2021; MAJORS LIZ FONG-JONES, 2022). Consequentemente, a agregação de métricas de monitoramento e *logging* são essenciais para a construção da observabilidade, uma vez que é

preciso reunir as informações sobre cada uma das partes do sistema para que seja possível ter uma visão sobre o todo. Além disso, também é necessário preservar o que se conhece como *tracing* distribuído. Trata-se de armazenar de alguma forma a relação entre os microsserviços, informação essencial para que seja possível compreender como o sistema está se comportando ao visualizar o fluxo das ocorrências dentro do mesmo, através dos microsserviços que o compõe. Para isso é importante também que a cronologia dos *logs* esteja de acordo com o momento da execução, para que as ocorrências possam ser visualizadas em orde e em decorrência do tempo (NEWMAN, 2021).

Uma estratégia bastante utilizada para unir as informações heterogêneas de diversas fontes é a utilização de *middlewares* com conceito de *adapter*. Para que seja possível monitorar o sistema, é preciso que exista uma interface comum para os dados originados nos vários microsserviços pelo qual o sistema é composto, sendo que essa interface comum depende da escolha da ferramenta ou estratégia de monitoramento e *logging* que será utilizada. A função do adapter, por sua vez, é converter as diferentes interfaces de dados divergentes em uma interface de dados padrão (BURNS, 2018; GAMMA *et al.*, 1994). O ônus dessa operação é que ao transformar *logs* com estruturas específicas relacionadas ao processo e negócio do microsserviço, em uma estrutura genérica, é que parte da informação é perdida no processo. Apesar de permitir a visão geral do todo, as informações de contexto podem ser perdidas, e consequentemente ainda podem haver problemas em identificar questões particulares relacionadas ao domínio de cada microsserviço, ou entender qual o microsserviço que originou um problema.

Outro conceito dentro deste tópico é o monitoramento semântico. Neste aspecto, ao invés de procurar a existência de erros, avalia-se se o sistema está se comportando de forma esperada. É como perguntar, o sistema que foi elaborado e construído atende ao seu propósito? O maior desafio é definir um modelo com quais as expectativas sobre o funcionamento do sistema. Essa definição não abrange o nível técnico, e sim definições a nível de processo e negócio, avaliando os resultados finais do sistema. Após o modelo ser definido, é possível extrair quais informações importantes devem ser monitoradas para que seja possível fazer essa avaliação (NEWMAN, 2021). Esse formato pode viabilizar a identificação de falhas humanas, que são mais sutis e muitas vezes passam despercebidas.

4 TRABALHOS RELACIONADOS

Para melhor avaliação e entendimento do cenário que circunda o assunto deste trabalho, foi realizada uma pesquisa de trabalhos relacionados aplicando a técnica de revisão sistemática (SAMPAIO; MANCINI, 2007). Esta pesquisa busca entender se existem estudos realizados previamente sobre a aplicação de *logs* para a identificação de falhas de negócio e processo em arquitetura de microsserviços. Os termos chave para a realização da pesquisa foram: *microservices*, *logs*, *business failure* e *process failure*. Além disso, a pesquisa foi feita buscando artigos publicados nos últimos cinco anos, entre 2019 e 2023. As fontes utilizadas para a pesquisa, foram o portal de periódicos Capes e o ScienceDirect. O portal Capes retornou 38 artigos, mas com a eliminação das publicações repetidas, restaram 23 resultados. O ScienceDirect, por sua vez, ao filtrar por artigos de revisão e de pesquisa, foram encontrados 122 resultados. Ao todo, somaram 141 artigos não repetidos. O principal critério de seleção dos artigos foi buscar estudos que de alguma forma contribuíssem com o presente trabalho, seja na compreensão dos pontos críticos relacionados ao tema, ou a avaliar diferentes abordagens de solução para problemas semelhantes. A Tabela 1 exibe a relação completa dos artigos selecionados, que serão abordados a seguir.

A Seção 4.1 contém apenas um artigo, no qual Valderas, Torres e Pelechano (2020) defendem o uso de BPMN (*Business Process Model and Notation*) como abordagem de documentação para microsserviços, abrangendo tanto a visão do fluxo como um todo, inclusive a interação entre microsserviços, quanto a visão do fluxo interno de um microsserviço.

A Seção 4.2 é composta por cinco artigos que buscam compreender e comparar o que existe para o estado da arte de monitoria e *logs* para microserviços com os problemas enfrentados e as práticas utilizadas pela indústria.

A Seção 4.3, por sua vez, possui três artigos que trazem diferentes abordagens para monitoria e *logs* em arquiteturas de microsserviços. Estes artigos, entretanto, possuem um viés técnico, diferente do que é proposto neste trabalho.

Tabela 1 – Artigos selecionados pela revisão sistemática

Título	Referência	Fonte	Seção
Composição de microsserviços com abordagem baseada em diagramas BPMN	Valderas, Torres e Pelechano (2020)	ScienceDirect	Seção 4.1
Catálogo e Técnicas de Detecção de *Anti-patterns* e *Bad Smells* de Microsserviços: Um Estudo Terciário	Cerny *et al.* (2023)	ScienceDirect	Seção 4.2
Detecção de Anomalias e Análise de Origem de Falhas em Aplicativos em Nuvem Baseados em (Micro) Serviços: Uma Pesquisa	Soldani e Brogi (2023)	Capes (ACM)	Seção 4.2
Revisitando as Práticas e Dores da Arquitetura de Microsserviços na Realidade: Uma Investigação Industrial	Zhou *et al.* (2023)	ScienceDirect	Seção 4.2
Projeto, Monitoramento e Teste de Sistemas de Microsserviços: A Perspectiva dos Profissionais	Waseem *et al.* (2021)	ScienceDirect	Seção 4.2
Monitoramento de Aplicações em Nuvem: Um Estudo Industrial	Tamburri, Miglierina e Nitto (2020)	ScienceDirect	Seção 4.2
Eadro: Um *Framework* de Solução de Problemas de Ponta-a-Ponta para Microsserviços de Múltiplas Fontes de Dados	Lee *et al.* (2023)	Capes (IEEE)	Seção 4.3
Groot: Uma Abordagem Baseada em Grafos de Eventos para Análise de Origem de Falhas em Ambientes Industriais	Wang *et al.* (2021)	Capes (IEEE)	Seção 4.3
Monitoramento de Microsserviços com *Logs* de Eventos e Rastreamento de Execução de Caixa Preta	Cinque, Corte e Pecchia (2021)	Capes (IEEE)	Seção 4.3

Fonte: O Autor (2023).

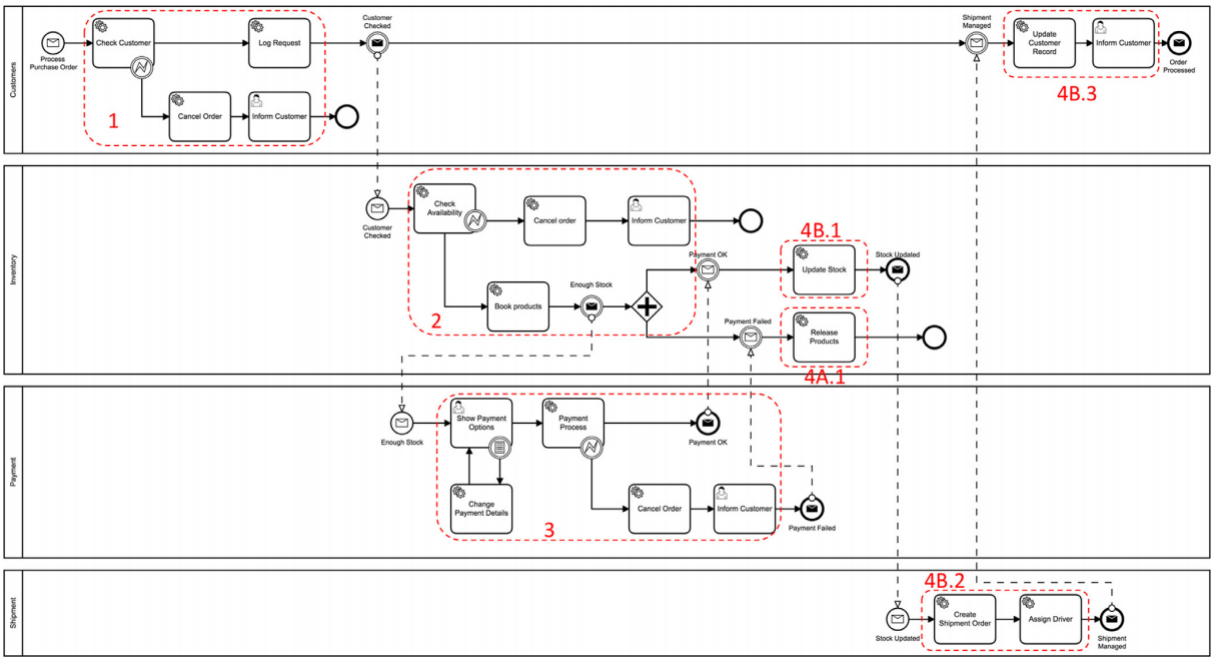
4.1 COMPOSIÇÃO DE MICROSSERVIÇOS COM ABORDAGEM BASEADA EM DIAGRAMAS BPMN

Este artigo defende que a natureza descentralizada (e muitas vezes assíncrona) dos microsserviços faz com que uma abordagem de documentação que realize a junção de passos de todo o processo seja mais apropriada para definir estas composições. Assim, é possível que cada microsserviço envolvido tenha suas operações definidas e junto com os momentos nos quais se conecta com outros microserviços, que por sua vez também possuem seus próprios processos e interações. A maior dificuldade nesta arquitetura é que, além das regras de negócio, as regras de processo também estão dispersas, definidas implicitamente pelas interações entre os microsserviços. Então, para melhorar a visualização dos cenários específicos e geral, o artigo propõe que sejam utilizadas composições de diagramas de processo BPMN (*Business Process Model and Notation*) (VALDERAS; TORRES; PELECHANO, 2020).

A Figura 4 possui um exemplo de BPMN que representa o processo de inclusão de um pedido, que passa pelos microsserviços *Customers*, *Inventory*, *Payment* e *Shipment*. No diagrama, é possível visualizar os processos de cada um dos microsserviços e o momento em que se comunicam via sistema de mensageria. A Figura 5, por sua vez, apresenta mais detalhada-

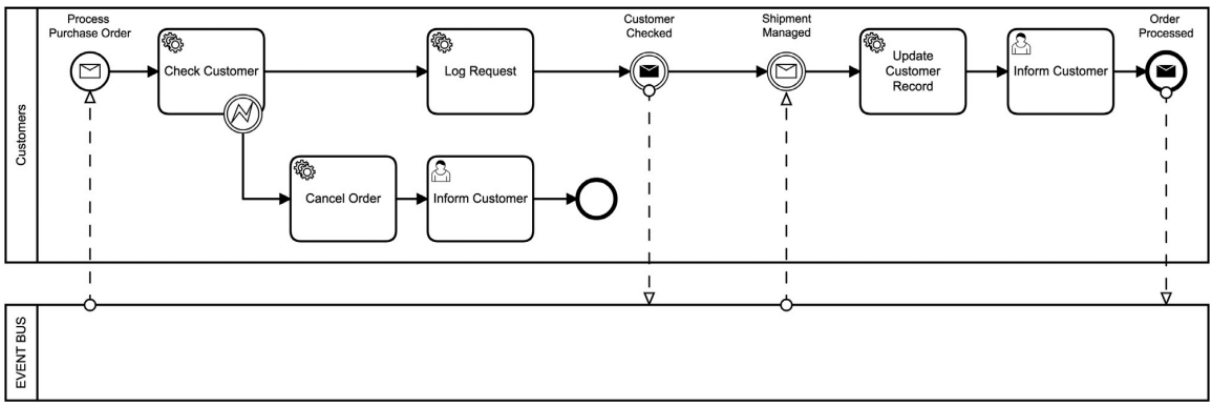
mente o processo do microsserviço *Customers*.

Figura 4 – Diagrama BPMN para visão geral do processo de inclusão de pedido



Fonte: Valderas, Torres e Pelechano (2020)

Figura 5 – Diagrama BPMN para visão detalhada do processo do microsserviço *Customers*



Fonte: Valderas, Torres e Pelechano (2020)

4.2 ARTIGOS DE REVISÃO

De forma resumida, os artigos dispostos nessa seção tem por objetivo avaliar o estado da arte, compará-lo com a aplicação real na indústria, levantar quais problemas ainda estão sem soluções e como a indústria reage a isso, e catalogar más práticas relacionadas a arquitetura de microsserviços e tópicos relacionados. Para cada artigo, foram observadas informações relevantes para as tarefas de monitoramento e *logging* em arquiteturas de microsserviço. A seguir, os artigos serão listados juntamente com suas contribuições para este trabalho.

1. Catálogo e Técnicas de Detecção de *Anti-patterns*[1] e *Bad Smells*[2] de Microsserviços: Um Estudo Terciário (CERNY *et al.*, 2023)

    a) Foi catalogado como *anti-pattern* quando cada microsserviço gerencia de forma independente seu próprio sistema de *log*, sem qualquer agregação ou monitoramento de *log* centralizado. Desta forma, os *logs* podem ser difíceis de agregar e analisar, retardando o processo de monitoramento proporcionalmente ao número de microsserviços e ao tamanho do *log*.

2. Detecção de Anomalias e Análise de Origem de Falhas em Aplicativos em Nuvem Baseados em (Micro) Serviços: Uma Pesquisa (SOLDANI; BROGI, 2023)

    a) Com a grande quantidade de dados obtidos em *logs* técnicos, uma abordagem bastante utilizada é criar modelos de identificação de falhas aplicando algoritmos de IA (Inteligência Artificial). Entretanto, com a constante alteração nos sistemas, pode acontecer a identificação de falsos positivos ou falsos negativos. Isto é, quando o sistema está saudável mas é definido como não saudável, ou quando o sistema não está saudável e é definido como saudável. Para que essa abordagem continue funcionando e acompanhe as frequentes mudanças no software, é preciso recriar os modelos de IA usados para a identificação de anomalias.

    b) Outro ponto negativo desta abordagem é a falta de explicabilidade. A maioria das técnicas de IA avaliadas não são "explicáveis desde a concepção". Isso significa que, apesar de detectarem eficazmente anomalias e as suas causas, não fornecem explicações sobre o motivo pelo qual isto acontece. Essa situação ocorre porque algoritmos de IA são usados para determinar se um serviço é anômalo ou a possível causa raiz de uma anomalia detectada. Entretanto, associar anomalias observadas e suas causas explicitando as razões de estarem interligadas permitiria que os profissionais excluíssem diretamente os falsos positivos. Assim, seria possível eliminar as causas de falhas improváveis e manter apenas as mais prováveis. Isto exige, portanto, técnicas de detecção de anomalias e análise de origem de falhas que sejam "explicáveis por natureza", da mesma forma que a necessidade de explicabilidade é hoje reconhecida na IA.

3. Revisitando as Práticas e Dores da Arquitetura de Microsserviços na Realidade: Uma Investigação Industrial (ZHOU *et al.*, 2023)

    a) A natureza distribuída dos microsservicos traz muitos desafios para o estabelecimento satisfatório dos, recursos de monitoramento e *logging*, fazendo com que

[1] Anti-patterns: Trata-se de um padrão ou solução comumente usado que gera resultados e consequências negativas (BROWN, 2001).

[2] Bad smell: São trecho de código que, apesar de estarem funcionais no momento, indicam potenciais problemas futuros (FOWLER *et al.*, 1999).

exista grande dependência da experiência acumulada de arquitetos de *software*. Algumas operações ainda dependem da avaliação manual de *logs*, que consomem bastante tempo e são ineficientes, sendo que é bastante dificil localizar problemas entre centenas de microsserviços distribuídos, principalmente quando a empresa ainda depende do método tradicional de identificação e correção de falhas, com *logging* e *tracking* distribuido.

4. Projeto, Monitoramento, e Testes de Microserviços: A Perspectiva dos Profissionais (WASEEM *et al.*, 2021)

    a) Gerenciamento de *log* e rastreamento de exceções são práticas de monitoramento amplamente utilizadas.

    b) Foi descoberto que a coleta de métricas e *logs* de monitoramento de contêineres (54,7%), o rastreamento distribuído (45,3%) e a presença de muitos componentes para monitorar (complexidade) (41,5%) são os principais desafios de monitoramento.

5. Monitoramento de Aplicações em Nuvem: Um Estudo Industrial (TAMBURRI; MIGLIERINA; NITTO, 2020)

    a) Nem todas as empresas costumam monitorar seus ativos de software ou têm uma estratégia para fazê-lo. Os procedimentos de monitoramento e medidas de gerenciamento de incidentes surgem organicamente, muitas vezes em torno de soluções de monitoramento personalizadas conforme demanda, com pouca ou nenhuma referência ao estado da arte ou boas práticas.

    b) Incidentes relatados por meio de monitoramento quase sempre envolvem apenas profissionais técnicos. No entanto, quando processos de negócio estão envolvidos, a frequência e tempo de resolução dos incidentes sofrem aumento.

    c) Embora não tenham sido encontradas literaturas sobre o estado da arte, diversas literaturas não oficiais (como a internet) relatam a urgência de conceber formas mais apropriadas e efetivas de mensurar a qualidade de estruturas organizacionais envolvidas na engenharia de software e computação em nuvem, especialmente no que diz respeito a monitoramento.

    d) Dois a cada três incidentes são encontrados por inspeção manual de *logs*, sendo que o manejo subsequente também envolve operações manuais excessivas.

    e) A falta de: (1) monitoramento estratégico do processo de negócio, (2) padronização e (3) de uma cruva de aprendizado mais rápida são percebidos respectivamente como os desafios mais críticos para monitoramento e observabilidade de sistemas em nuvem. Não foi possível encontrar literaturas que tratem essas deficiências, além

de *whitepapers* [3] ou literatura de pesquisas confirmando a urgência e necessidade de investir neste tópico, tanto na perspectiva acadêmica quando prática.

f) Os resultados deste estudo oferecem um vislumbre do potencial inexplorado por trás do uso de abordagens mais estruturadas para monitoramento de aplicações em nuvem e melhoria contínua da qualidade. Conclui-se que a falta de padrões técnicos e organizacionais foi apontada como principais limitações atualmente neste campo.

4.3 ARTIGOS DE APLICAÇÃO

Os artigos apresentados a seguir possuem diferentes abordagens para solução de monitoramento e *logging* para sistemas em arquitetura de microsserviços. Entretanto apresentam um viés técnico, enquanto este trabalho possui uma abordagem voltada para o processo e negócio. A seguir, estes artigos serão citados, juntamente com sua colaboração para o presente trabalho.

1. Eadro: Um *Framework* de Solução de Problemas de Ponta-a-Ponta para Microsserviços de Múltiplas Fontes de Dados (LEE *et al.*, 2023)

    a) Uma das limitações das abordagens existentes é a exploração insuficiente dos dados de monitoramento. Isso é decorrente da alta complexidade da análise de dados de múltiplas fontes, uma vez que estes dados dados são heterogêneos, interagem frequentemente e são bastante grandes. Por outro lado, diferentes tipos de dados podem revelar anomalias de forma colaborativa e trazer mais pistas sobre potenciais falhas.

2. Groot: Uma Abordagem Baseada em Grafos de Eventos para Análise de Origem de Falhas em Ambientes Industriais (WANG *et al.*, 2021)

    a) A tarefa de monitoramento torna-se bastante complexa, pois existe uma grande quantidade de dados de observabilidade que precisa ser monitorada, armazenada e processada, como métricas intra-serviços e entre serviços. Diferentes serviços em um sistema podem produzir diferentes tipos de *logs* ou métricas com diferentes padrões.

3. Monitoramento de Microsserviços com *Logs* de Eventos e Rastreamento de Execução de Caixa Preta (CINQUE; CORTE; PECCHIA, 2021)

    a) As vantagens arquiteturais trazidas pelos microsserviços entram em conflito com as práticas de *logs*. Por exemplo, sistemas de microsserviços (1) resultam da composição de software desenvolvido por equipes com competências diferentes, (2) estão

[3] Whitepapers: documento que usa evidências, fatos e raciocínio para auxiliar no entendimento de um tópico ou problema específico.

distribuídos em plataformas e tecnologias heterogêneas, (3) são fortemente dinâmicos, com microsserviços sendo frequentemente adicionados, atualizados ou replicados para escalabilidade e tolerância a falhas. As implantações atuais criam um *log* distinto por microsserviço, e além disso, a centralização e a coleta de *logs* exigem uma infraestrutura disponível em cada elemento do sistema.

b) Os profissionais enfrentam grandes desafios na manutenção de catálogos de expressões regulares e palavras-chave para análise e monitoramento de *logs*. A composição dos microsserviços exacerba a heterogeneidade semântica e de formato dos *logs* causada pela falta de práticas de codificação padrão. Ainda mais importante, compreender e percorrer os *logs* de diferentes microsserviços exige um trabalho cognitivo substancial por parte de profissionais especialistas.

c) É difícil inferir o contexto de uma chamada de serviço a partir dos *logs*. Por exemplo, por contexto entendemos o serviço chamado e a origem-destino da invocação. O contexto fornece dicas valiosas sobre onde encontrar os problemas.

4.4 ANÁLISE E CONCLUSÕES

Por mais que o viés dos trabalhos citados na Seção 4.3 seja técnico (falhas de *hardware* ou *software*) enquanto o presente trabalho é voltado ao processo de negócio (falha humana no levantamento de regras de negócio e processo ou identificação de melhorias no processo), na arquitetura de microsserviços tratam-se de duas faces de um mesmo problema. Os artigos de revisão expostos na Seção 4.2 possibilitam uma visão mais ampla dos problemas encontrados para monitoramento e *logging* de microsserviços. Trata-se de uma área em crescimento, mas que ainda carece de padrões e boas práticas no estado da arte para resolver os novos desafios trazidos pela arquitetura de microsserviços. A partir dos trabalhos citados nas seções anteriores, foi elaborado um resumo das principais conclusões e contribuições para o presente trabalho:

- Uma boa definição do processo e regras de negócio são essenciais para a elaboração de sistemas em arquitetura de microsserviços. É necessário ter clareza e entendimento sobre onde os domínios de negócio começam, por onde passam e onde terminam, porque isso impacta na construção e manutenção dos microsserviços.

- *Logs* e rastreamento de exceções são práticas de monitoramento amplamente utilizadas em sistemas de microsserviços. Apesar disso, ainda não existe literatura e métodos satisfatórios para apoiar as empresas neste processo.

- Logs são comumente utilizados para objetivos técnicos, mas no cenário de microsserviços o processo e regras de negócio tem grande relevância dentro do sistema. Quando ocorrem problemas que envolvem a área de negócios, a frequência e tempo de resolução

dos incidentes sofrem aumento, porque os *logs* e ferramentas de monitoramento não estão devidamente preparados para dar suporte a esse tipo de problema.

- A utilização de *logs* independentes por microsserviços sem uma forma de integração e agregação dessas informações não é uma boa prática e pode inviabilizar o uso da informação.

- A natureza distribuída dos microsserviços impõe desafios para a atividade de monitoramento e *logging*, fazendo com que algumas operações ainda dependam fortemente do trabalho manual de profissionais experientes. Este trabalho, entretanto, é bastante demorado e ineficiente, uma vez que é bastante difícil localizar problemas em centenas de microsserviços distribuídos.

- Sistemas de *logs* utilizados em microsserviços produzem um grande volume de dados. Muitas vezes, utilizam-se estratégias de IA para que seja possível identificar informações úteis que indiquem problemas e sua origem. Essa técnica possui duas principais desvantagens. Primeiramente, de tempos em tempos os modelos de IA para identificação de problemas precisam ser refeitos para que continuem funcionando e acompanhando a rápida evolução dos sistemas. Além de que, a falta de compreensão sobre a causalidade das falhas impede que sejam tomadas medidas mais assertivas.

- Analisar *logs* de mais de uma fonte é bastante complexo. Os dados são hetergêneos, ou seja, cada microsserviço possui um formato de *log* porque possuem características distintas e podem ser inclusive de diferentes tecnologias. Também é difícil padronizar *logs* já que estas informações específicas não podem ser ignoradas, porque trazem colaborações significativas para a identificação de falhas e entendimento sobre o cenário no qual ocorreu.

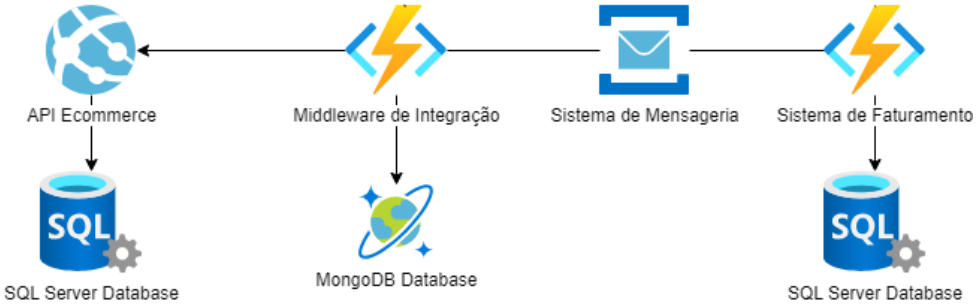
- Apesar das áreas de monitoramento e gestão de *logs* em microsserviços estarem em constante evolução, e da gama de ferramentas existentes no mercado atualmente que buscam suprir este problema, ainda existem lacunas a serem preenchidas no estado da arte para garantir padrões, melhores práticas e maior assertividade nessa atividade, que ainda depende de processos manuais e da experiência de profissionais para a resolução de problemas.

5 PROPOSTA DE SOLUÇÃO

O objetivo deste trabalho é propor um modelo de implementação de *logs* para se adequar às particularidades da arquitetura de microsserviços. Este modelo deve permitir o registro e consulta de *logs* direcionado ao processo e às regras de negócio. Estes *logs* tem por objetivo prover um histórico de ações importantes ao processo de negócio que permitam a identificação de falhas no mesmo.

A seguir, será apresentado o modelo proposto, juntamente um processo elaborado para a implementação do mesmo. Após a definição de cada componente do modelo, será exemplificada a implementação do mesmo para melhor compreensão do funcionamento e possibilidades do mesmo. O cenário de exemplo utilizado caracteriza-se por uma aplicação de vendas *online*, conhecido popularmente como *e-commerce*, que precisa transmitir seus pedidos a um sistema de faturamento, para que as notas fiscais dessas vendas sejam enviadas ao Sefaz. Na Figura 6 encontra-se uma arquitetura hipotética para este cenário, elaborada com base na plataforma Microsoft Azure. Nessa arquitetura, existe uma API para *e-commerce* que se comunica com um *middleware*, que deve intermediar essa integração, e por sua vez comunicar-se com o sistema de faturamento através de um sistema de mensageria.

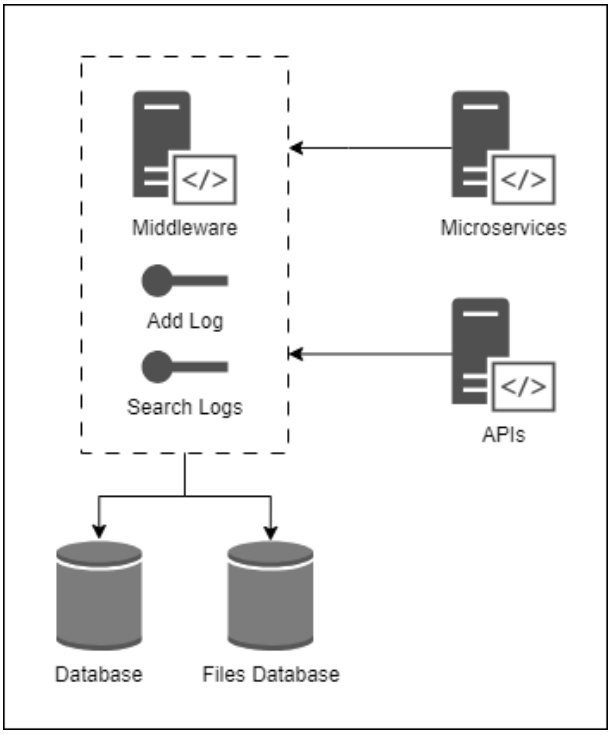
Figura 6 – Arquitetura do Cenário de Testes



Fonte: O Autor (2023)

O modelo proposto neste trabalho é apresentado na Figura 7, e consiste na utilização de um *middleware* para a centralização do gerenciamento de *logs*.

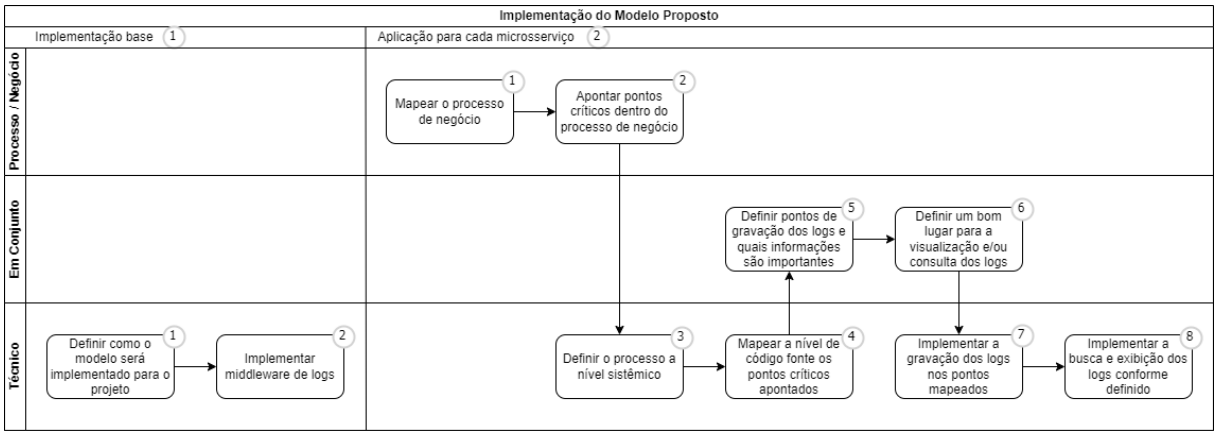
Figura 7 – Proposta Conceito



Fonte: O Autor (2023)

O processo de implantação, por sua vez, pode ser dividido em duas partes: a implementação base e a aplicação para cada microsserviço. A Figura 8 ilustra o passo a passo deste processo, que necessita tanto do conhecimento técnico sobre a arquitetura do projeto e o código fonte, quanto do conhecimento de processo e negócio.

Figura 8 – Processo de Implantação da Proposta

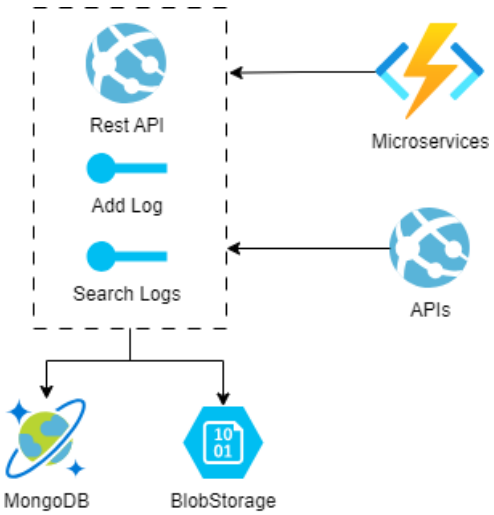


Fonte: O Autor (2023)

5.1 IMPLEMENTAÇÃO BASE

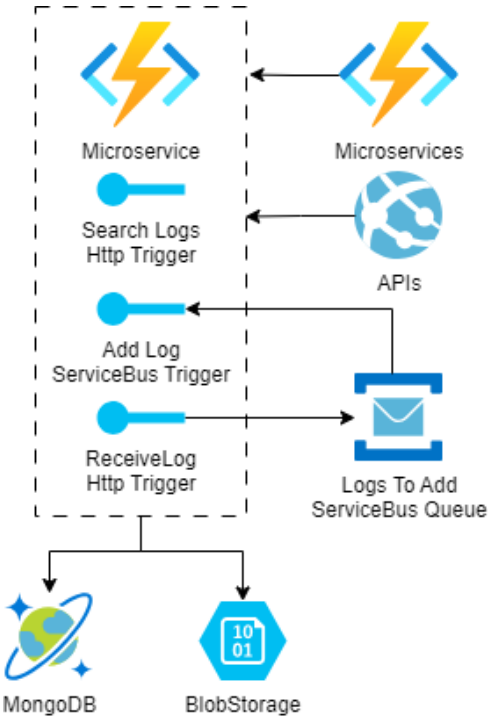
Primeiramente, é necessário definir como será feita a implementação do componente de *middleware* e como ele se encaixará na arquitetura do projeto. O modelo proposto pode ser implementado com as tecnologias e no formato que for mais condizente ao sistema no qual está sendo implantado, desde que mantenha as premissas estabelecidas. Para ilustrar essa versatilidade, foram elaboradas duas possíveis abordagens para a implementação do modelo. A primeira delas, ilustrada na Figura 9, busca implementar o *middleware* por meio de uma API REST. Já a segunda abordagem, ilustrada na Figura 10, aplica o modelo em um microsserviço, fazendo uso também de um serviço de mensageria para a gravação dos *logs*. Ambas as abordagens de implementação usadas como exemplo foram arquitetadas com base na plataforma Microsoft Azure.

Figura 9 – Implementação com API REST



Fonte: O Autor (2023)

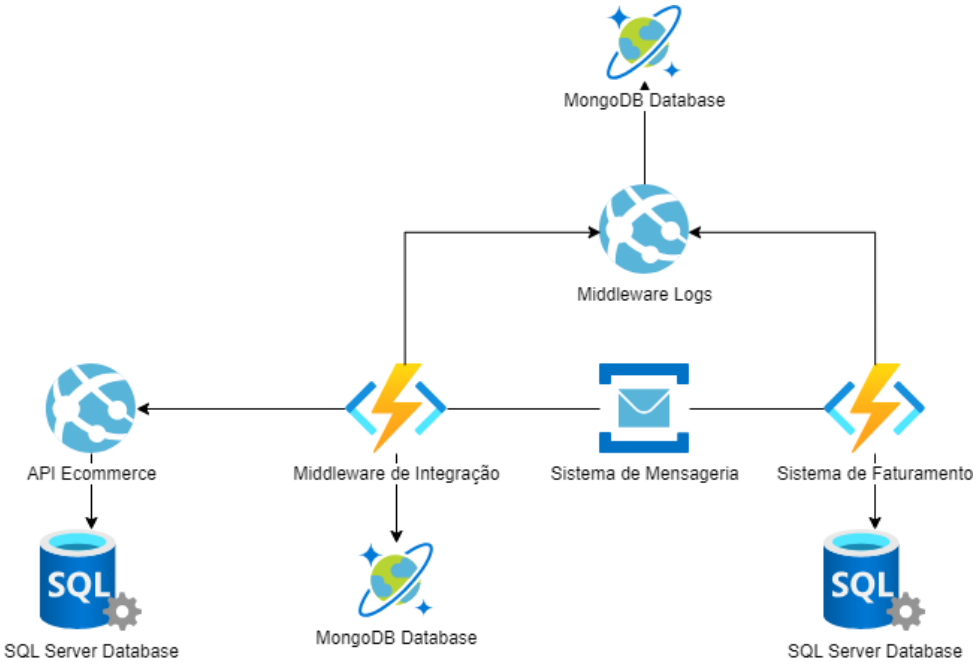
Figura 10 – Implementação com Microsserviço e Serviço de Mensageria



Fonte: O Autor (2023)

Conforme orientado na Figura 8, começando pela etapa de implementação base, o primeiro passo é definir como o modelo será implementado. Considerando o cenário de exemplo estabelecido na Figura 6, foi escolhida a abordagem de API REST, ilustrada na Figura 9, para a implementação do *middleware* de *logs*. Resultando, então, na arquitetura definida na Figura 11.

Figura 11 – Arquitetura Completa do Cenário de Testes



Fonte: O Autor (2023)

5.1.1 Implementando o *Middleware* de Gerenciamento de *Logs*

O *middleware* para gerenciamento de *logs* precisa dispor obrigatoriamente de duas funcionalidades: adição de *logs* e consulta de *logs*. Ambas precisam respeitar as interfaces padronizadas para que o modelo funcione de forma adequada. A partir da estrutura de informações que compõe estas interfaces, qualquer microsserviço poderá fazer a inclusão ou consulta de *logs*. Apesar de generalizada entre os microsserviços do sistema, as interfaces organizam as informações de forma que as especificidades de cada domínio sejam respeitadas. A API REST, para prova de conceito, foi desenvolvida em C# com .NET 6.0. A mesma implementa os dois *endpoints* necessários para a aplicação das funcionalidades previstas no modelo, um para edição e outro para consulta de *logs*, conforme as interfaces definidas.

O Quadro 1 contém a interface da funcionalidade de adição de *logs*, que deve prover a inclusão de novos registros. Nessa operação, os campos obrigatórios devem ser validados para garantir que as informações básicas necessárias sejam informadas. Os campos não obrigatórios, por sua vez, são de caráter situacional. Para dados mais extensos, como o corpo ou o retorno de requisições, indica-se enviá-los na lista de arquivos (*Files*), e armazená-los de forma adequada a sua natureza.

Quadro 1 – Interface para a adição de *Logs*

Campo	**Tipo**	**Definição**
*Classification**	*string*	Classificação do *log*.
*Type**	*string*	Tipo do *log*.
*RegisterID**	*string*	Identificador único do registro ao qual o *log* está relacionado.
*RegisterType**	*string*	Tipo do registro ao qual o *log* está relacionado.
*OccurrenceDate**	*DateTime*	Indica o dia e hora em que o *log* foi gerado (formato UTC).
*OcurrenceAction**	*string*	Ação que gerou o *log*.
GroupID	*string*	Permite agrupamento de *logs* de uma mesma transação ou operação.
RegisterRelatedIDs	*string, string* (lista)	Lista de chave (*RegisterType*) e valor (*OtherRegisterID*) de identificadores relacionados ao registro *RegisterID*.
SpecificInformations	*string, string* (lista)	Lista de chave e valor contendo dados específicos do registro que gerou o *log*.
Observations	*string* (lista)	Lista de observações do *log*, pode ser utilizado para anexar informações sobre o cenário do *log*.
StatusResult	*int*	Código *http* resultante da operação.
MessageResult	*string*	Mensagem resultante da operação.
Files	*string, string* (lista)	Lista de chave (*FileType*) e valor para armazenar dados mais extensos relacionados ao cenário do *log*.

* = campos obrigatórios.
Fonte: O Autor (2023).

Para ilustrar melhor as possibilidades da interface, segue abaixo uma definição de como a mesma pode ser utilizada, com base no cenário de exemplo previamente definido.

- *Classification* ou classificação: trata-se do que o *log* está registrando de forma geral.
  - *Notification*: quando trata-se apenas de um registro.

  - *Warning*: quando não ocorreu erro, mas é algo que precisa de atenção.
  - *Error*: quando de fato ocorreu um erro.

- *Type* ou tipo: trata-se do tipo do *log*, e define o que o mesmo está registrando.
  - *ValidatingOrder*: registra a validação do pedido pelo *middleware* de integração.
  - *SendingOrder*: registra o envio do pedido pelo *middleware* de integração para o sistema de faturamento via sistema de mensageria.
  - *ReceivingOrder*: registra o recebimento do pedido pelo sistema de faturamento via sistema de mensageria.
  - *InvoicingOrder*: registra quando o sistema de faturamento faz o envio das notas fiscais para o Sefaz.

- *RegisterID* ou identificador do registro: será informado o identificador do registro principal ao qual o log está vinculado.
  - Etapa 1: identificador gerado para o pedido a ser importado.
  - Etapa 2: identificador gerado para o pedido a ser importado.
  - Etapa 3: identificador gerado para o pedido importado.
  - Etapa 4: identificador gerado para a nota fiscal do pedido.

- *RegisterType* ou tipo do registro: será informado o tipo do registro principal ao qual o log está vinculado.
  - Etapa 1: *IncomingOrder*, para representar o pedido a ser importado.
  - Etapa 2: *IncomingOrder*, para representar o pedido a ser importado.
  - Etapa 3: *Order*, para representar o pedido importado.
  - Etapa 4: *Invoice*, para representar a nota fiscal do pedido.

- *OccurrenceDate* ou data da ocorrência: dia e horário da ocorrência, em formato UTC para que não exista problema de fuso horário.

- *OccurrenceAction* ou ação de ocorrência: ação (evento, método, procedimento, *endpoint*) que gerou a gravação do *log*, para localizar em que momento do processo a ação ocorreu.
  - *ImportOrdersEvent*: quando o *log* for gerado pelo evento *ImportOrdersEvent*.
  - *ReceiveOrdersEvent*: quando o *log* for gerado pelo evento *ReceiveOrdersEvent*.
  - *InvoiceOrdersEvent*: quando o *log* for gerado pelo evento *InvoiceOrdersEvent*.

- *GroupID* ou identificador do grupo: Identificador para agrupar uma sequência de *logs* relacionados. Será utilizado o identificador do pedido a ser importado, porque o mesmo dá início a sequência de *logs*, mas poderia também se utilizar identificador aleatório.

- *RegisterRelatedIDs* ou identificadores de registros relacionados: Identificadores relacionados ao registro principal. Em todas as etapas foram informados o identificador do cliente e do produto relacionado ao pedido, por serem os registros mais importantes vinculados a um pedido.
- *SpecificInformations* ou informações específicas: Informações específicas ao registro, que são importantes para verificações de *logs* posteriores. Nesse caso será utilizado o número da ordem de compra, pois pode ser útil para conferências com o *e-commerce*.
- *Observations* ou observações: Informações úteis ao leitor do *log*.
- *StatusResult* ou *status* resultante: Quando o *log* possuir ligação ou dependência com uma requisição HTTP, esse campo armazenará o *status* de retorno da solicitação feita.
- *MessageResult* ou mensagem resultante: Quando o *log* possuir ligação ou dependência com uma requisição HTTP, esse campo armazenará a mensagem de retorno da solicitação feita.
- *Files* ou arquivos: Quando o *log* possuir ligação ou dependência com uma requisição HTTP, será enviado nesse campo o corpo da requisição enviada e do retorno recebido.

Para a funcionalidade *Add Log*, foi definido um *endpoint* contendo a interface de dados esperados conforme descrito no Quadro 1. Esta funcionalidade valida os dados informados conforme a interface definida e faz a gravação dos *logs* no banco de dados.

O Quadro 2, por sua vez, apresenta a interface para a busca de *logs*. A interface contém os filtros básicos para a consulta de *logs*, o que não impede a adição de filtros adicionais baseados em outros campos existentes no *log*. E, complementarmente, o Quadro 3 contém o formato de retorno de cada um dos *logs* encontrados na busca, ou seja, uma lista dessa estrutura será retornada.

Quadro 2 – Interface para a busca de *Logs*

Campo	**Tipo**	**Definição**
Classifications	*string* (lista)	Filtrar por classificações dos *logs*.
Types	*string* (lista)	Filtrar por tipos dos *logs*.
RegisterID	*string*	Filtrar pelo identificador de um registro ao qual os *logs* podem estar relacionados.
RegisterType	*string*	Filtrar pelo tipo do registro ao qual o *log* está relacionado.
OccurrenceDateStart	*DateTime*	Filtrar pelo dia e hora (inicial) em que *logs* foram gerados (formato UTC).
OccurrenceDateEnd	*DateTime*	Filtrar pelo dia e hora (final) em que *logs* foram gerados (formato UTC).
OcurrenceActions	*string* (lista)	Filtrar por ações que geraram *logs*.
GroupID	*string*	Filtrar por agrupamento de *logs* de uma mesma transação ou operação.

Fonte: O Autor (2023).

Quadro 3 – Interface para o retorno de *Logs* consultados

Campo	**Tipo**	**Definição**
ID	*Guid*	Identificador gerado na criação do *log*.
Classification	*string*	Classificação do *log*.
Type	*string*	Tipo do *log*.
RegisterID	*string*	Identificador único do registro ao qual o *log* está relacionado.
RegisterType	*string*	Tipo do registro ao qual o *log* está relacionado.
OccurrenceDate	*DateTime*	Indica o dia e hora em que o *log* foi gerado (formato UTC).
OcurrenceAction	*string*	Ação que gerou o *log*.
GroupID	*string*	Permite agrupamento de *logs* de uma mesma transação ou operação.
RegisterRelatedIDs	*string, string* (lista)	Lista de chave (*RegisterType*) e valor (*OtherRegisterID*) de identificadores relacionados ao registro *RegisterID*.
SpecificInformations	*string, string* (lista)	Lista de chave e valor contendo dados específicos do registro que gerou o *log*.
Observations	*string* (lista)	Lista de observações do *log*.
StatusResult	*int*	Código *http* resultante da operação.
MessageResult	*string*	Mensagem resultante da operação.
Files	*string, string* (lista)	Lista de chave (*FileType*) e valor com o link para arquivos armazenados contendo dados mais extensos relacionados ao cenário do *log*.

Fonte: O Autor (2023).

Na funcionalidade de *Search Logs*, por sua vez, foi definida a interface de filtros descrita no Quadro 2. Essa funcionalidade faz uma busca nos *logs* existentes com base nos filtros enviados, e retorna uma lista dos registros encontrados conforme interface definida no Quadro 3. Com isso, é finalizada a etapa de implementação base.

5.1.2 Definindo a Base de Dados para Armazenamento dos *Logs*

Para armazenamento dos dados dos *logs*, é recomendável a utilização de um banco de dados NoSQL orientado a documentos, por ser indicado em casos de grande volume de entrada de dados, e pela necessidade de ter uma estrutura de dados mais dinâmica e expressiva (KLEPPMANN, 2017). Este tipo de banco de dados utiliza o termo coleções para nomear os conjuntos de dados (correspondente a tabelas no banco de dados relacional) e documentos para nomear cada registro gravado em uma coleção. Neste modelo os dados são semi-estruturados, ou seja, os documentos de uma coleção não terão necessariamente a mesma estrutura (SADALAGE, 2012). O Quadro 4 apresenta as informações que a coleção ou tabela deve apresentar.

Quadro 4 – Estrutura do banco de dados dos *Logs*

Campo	**Tipo**	**Definição**
ID	*Guid*	Identificador gerado na criação do *log*.
Classification	*string*	Classificação do *log*.
Type	*string*	Tipo do *log*.
RegisterID	*string*	Identificador único do registro ao qual o *log* está relacionado.
RegisterType	*string*	Tipo do registro ao qual o *log* está relacionado.
OccurrenceDate	*DateTime*	Indica o dia e hora em que o *log* foi gerado (formato UTC).
OcurrenceAction	*string*	Ação que gerou o *log*.
GroupID	*string*	Permite agrupamento de *logs* de uma mesma transação ou operação.
RegisterRelatedIDs	*string, string* (lista)	Lista de chave (*RegisterType*) e valor (*OtherRegisterID*) de identificadores relacionados ao registro *RegisterID*.
SpecificInformations	*string, string* (lista)	Lista de chave e valor contendo dados específicos do registro que gerou o *log*.
Observations	*string* (lista)	Lista de observações do *log*.
StatusResult	*int*	Código *http* resultante da operação.
MessageResult	*string*	Mensagem resultante da operação.
Files	*string, string* (lista)	Lista de chave (*FileType*) e valor com o link para arquivos armazenados contendo dados mais extensos relacionados ao cenário do *log*.

Fonte: O Autor (2023).

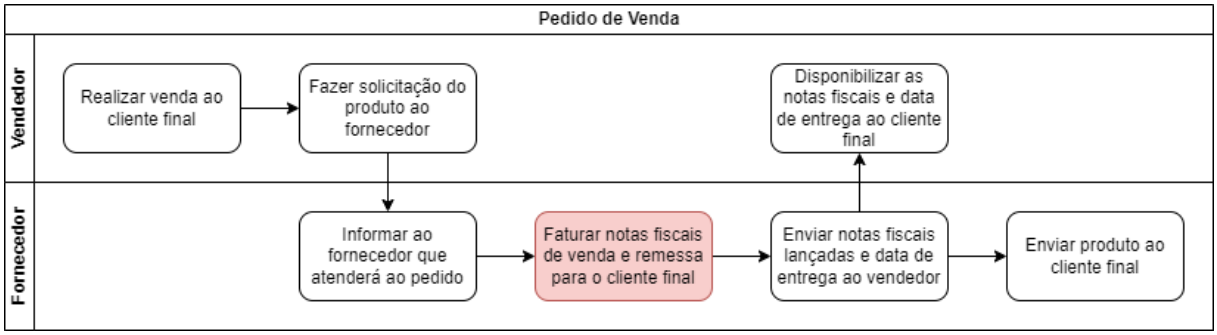
5.1.3 Definindo a Base de Dados para Armazenamento dos Arquivos

Para que o banco de dados não fique sobrecarregado com o corpo ou retorno de requisições, ou com qualquer outra estrutura importante ao *log* que seja mais extensa, estas devem ser armazenadas como arquivo em uma tecnologia apropriada. Sendo assim, o *middleware* deverá receber o arquivo e fazer o *upload* do mesmo conforme ferramenta escolhida, como por exemplo, um diretório de armazenamento FTP (File Transfer Protocol) ou BLOB (Binary Large OBject). O link para *download* ou caminho de acesso aos arquivos de cada *log* devem ficar relacionados ao mesmo, para que sejam de fácil rastreio e acesso quando o mesmo for retornado em consulta.

5.2 APLICAÇÃO PARA CADA MICROSSERVIÇO

Na etapa de aplicação para cada microsserviço, conforme orienta a Figura 8, o primeiro e segundo passos são mapear o processo de negócio do fluxo desejado e indicar quais os pontos críticos do mesmo. Para melhor visualização e entendimento, os fluxogramas BPMN necessários foram elaborados com base no cenário exemplo definido previamente. Começando pelo fluxo geral definido na Figura 12, com seu ponto crítico destacado em vermelho.

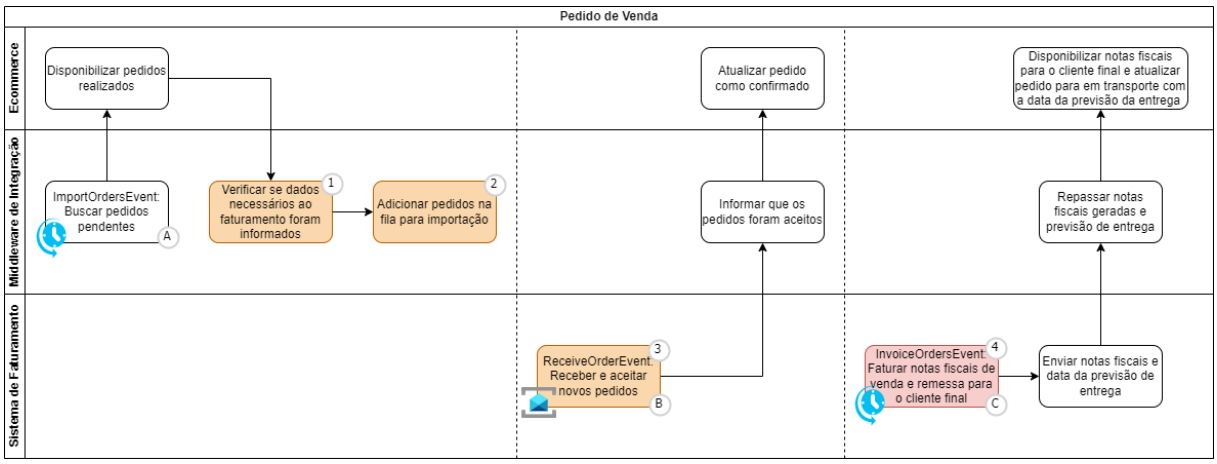
Figura 12 – Processo de Negócio



Fonte: O Autor (2023)

Os passos 3 e 4 tratam-se de definir o fluxo a nível sistêmico para cobrir o processo de negócio indicado, e mapear quais seus pontos críticos correspondentes conforme o apontado no fluxograma do processo de negócio. A Figura 13 contém o processo sistêmico definido, com o ponto crítico também destacado em vermelho. No passo 5, por sua vez, são definidos os pontos de gravação dos *logs*, que estão numerados e destacados em laranja juntamente ao ponto crítico destacado em vermelho. Também cabe destacar que, em uma arquitetura de microsserviços, o mapeamento de interação entre os microsserviços do sistema é uma informação já necessária para o desenvolvimento e manutenção do mesmo.

Figura 13 – Processo Sistêmico



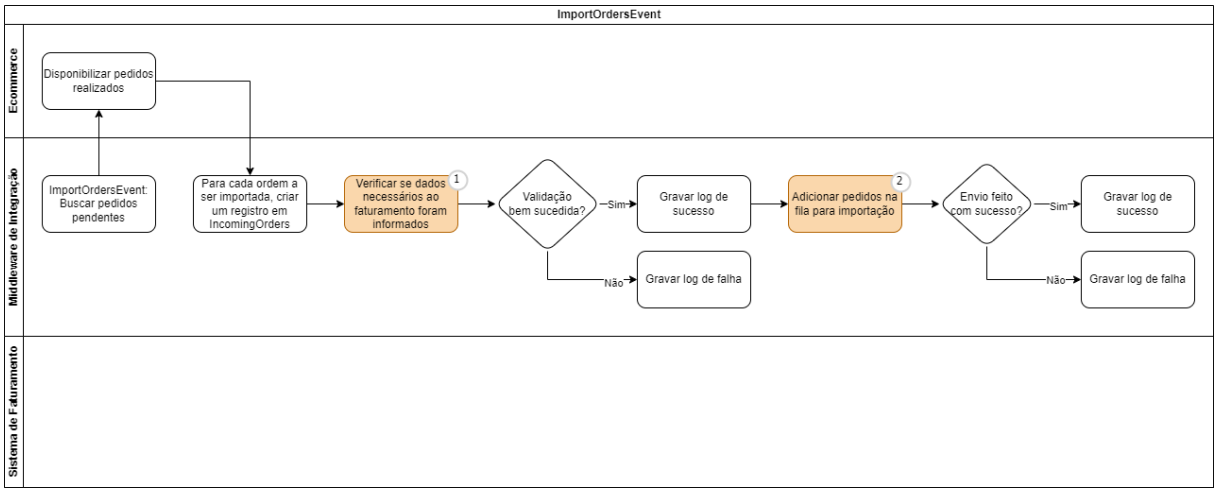
Fonte: O Autor (2023)

Para a definição dos pontos de gravação de *log* buscou-se antecipar os problemas, para que as informações necessárias estejam presentes no momento do envio da nota fiscal, ponto crítico deste processo. Sendo assim, foram elaborados os fluxogramas detalhados da implementação dos três eventos que ocorrem no fluxo de testes: *ImportOrdersEvent*, *ReceiveOrderEvent* e *InvoiceOrdersEvent*.

O evento *ImportOrdersEvent* é disparado por um temporizador a cada 30 minutos. O momento em que essa parte do fluxo ocorre pode ser visto na Figura 13, indicado pela letra A, e está descrito com mais detalhes na Figura 14. Neste evento, o *middleware* de integração

deve buscar na API do *e-commerce* os novos pedidos a serem importados. Os *logs* deverão ser gravados em dois momentos: após a validação dos dados de cada pedido, indicado nas imagens citadas pelo número 1, e após o envio da mensagem contendo cada pedido a ser importado pelo sistema de faturamento, indicado pelo número 2. Em ambos os momentos, deverá ser retratado se houve sucesso ou falha, juntamente com informações importantes para a compreensão do cenário dos mesmos.

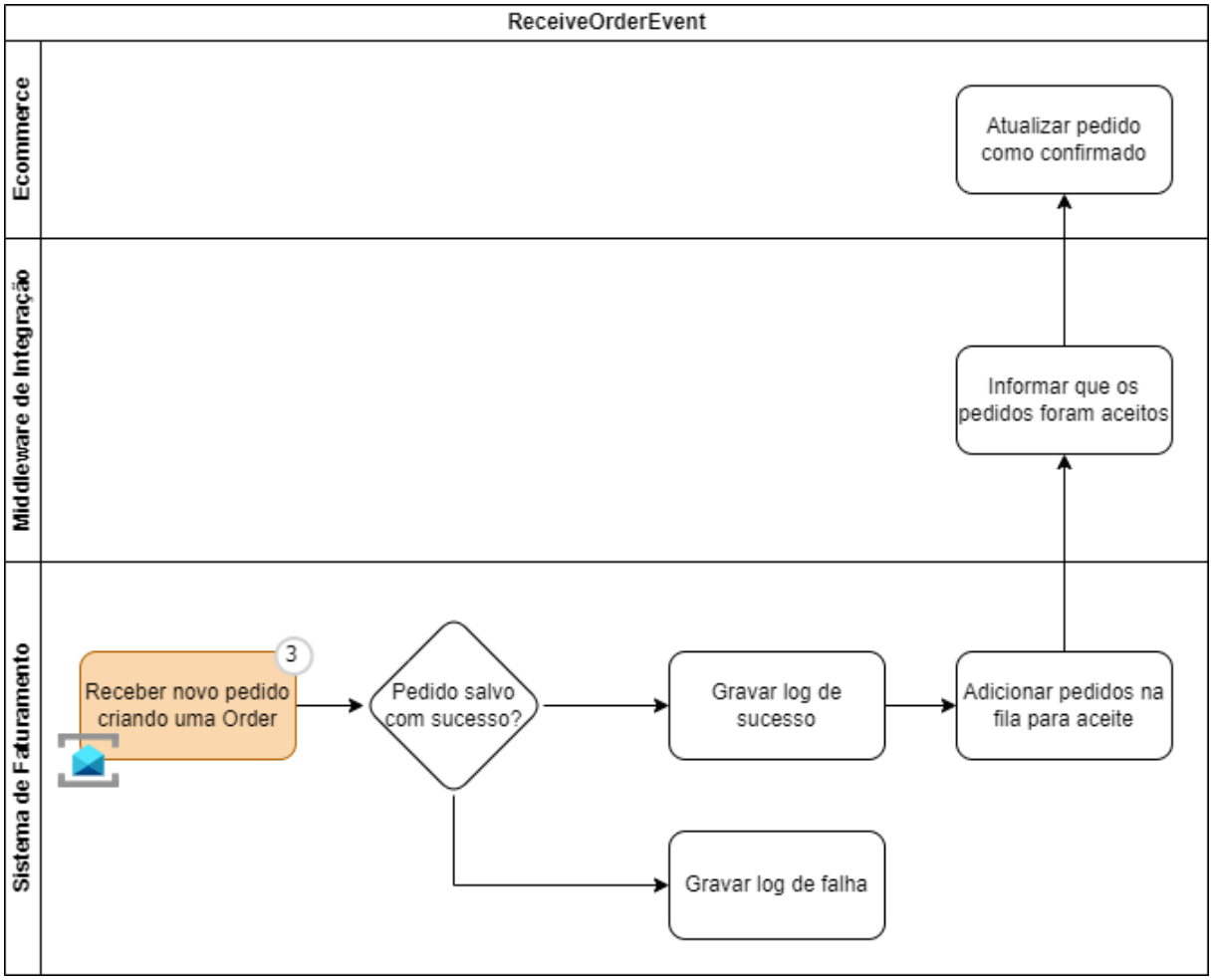
Figura 14 – Fluxo do Evento *ImportOrdersEvent*



Fonte: O Autor (2023)

O evento *ReceiveOrderEvent* é disparado quando existem mensagens a serem lidas no serviço de mensageria do sistema de faturamento, ou seja, quando existem novos pedidos a serem importados, e está indicado na Figura 13 pela letra B. Neste evento, o sistema de faturamento deverá receber cada novo e pedido e informar ao *middleware* de integração que os mesmos foram aceitos. Conforme o detlahamento deste fluxo, apresentado na Figura 15, um *log* deve ser gerado após a importação do pedido (indicado pelo número 3), apresentando seu sucesso ou falha, e contendo informações importantes para a compreensão de seu cenário.

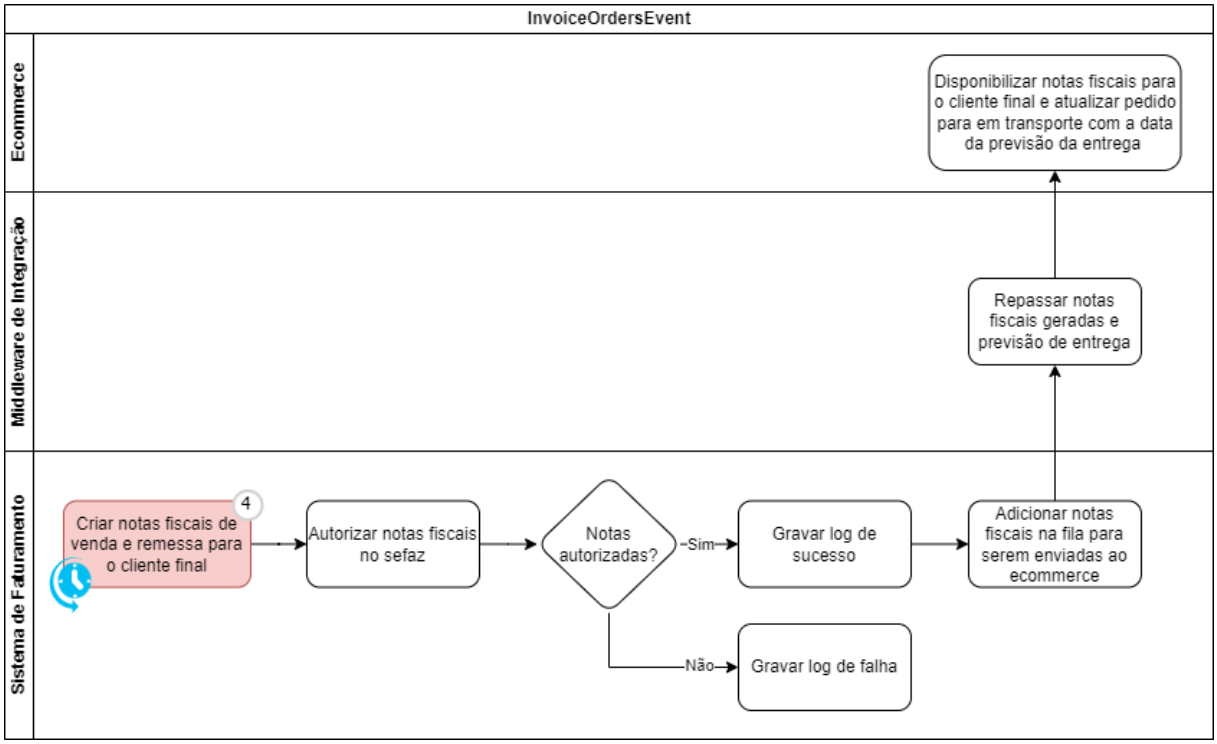
Figura 15 – Fluxo do Evento *ReceiveOrderEvent*



Fonte: O Autor (2023)

O evento *InvoiceOrdersEvent*, por sua vez, é disparado pelo sistema de faturamento por um temporizador a cada 30 minutos, e está indicado na Figura 13 pela letra C. Conforme o detalhamento apresentado na Figura 16, seu objetivo é fazer o faturamento dos pedidos pendentes, enviando-os para o Sefaz. Um *log* deverá ser gravado após o envio (indicado pelo número 4), para retratar se houve falha ou sucesso, juntamente com as informações relevantes ao seu cenário.

Figura 16 – Fluxo do Evento *InvoiceOrdersEvent*



Fonte: O Autor (2023)

Os passos 6, 7 e 8 não serão realizados para a prova de conceito pois não é necessário desenvolver uma outra aplicação que utilize o *middleware* de *logs* para atestar sua funcionalidade. Aplicando o modelo definido, o recurso poderá ser utilizado por qualquer sistema ou serviço que possa realizar requisições via HTTP. Para os testes, o *middleware* será acessado diretamente via requisição. Não faz parte deste trabalho tratar cenários que necessitem de *logs* locais temporários para tratar problemas de perda de conectividade com a *internet* ou acesso ao *middleware* de *logs*. Tratar (ou não) estes cenários e como fazê-los ficam a critério do implementador.

6 TESTANDO O CONCEITO

Foi desenvolvido um protótipo para prova de conceito do *middleware* de *logs* para aplicar testes com base no cenário de exemplo previamente elaborado. Para sua implementação, foi utilizada a arquitetura em camadas com MVC (Model - View - Controller), desenvolvida em ASP.NET 6.0. Complementarmente, o CosmosDB para MongoDB foi utilizado para tecnologia de banco de dados, enquando o Azure BlobStorage foi escolhido para o armazenamento de arquivos.

A seguir, serão realizados experimentos na aplicação desenvolvida para que seja possível visualizar o modelo em funcionamento. Primeiramente, serão inclusos os *logs* para as etapas do processo considerando o exemplo previamente definido, para cenários cinco cenários diferentes. Depois serão realizadas as consultas, para avaliar os filtros em sete cenários para visualização de diferentes formas de filtrar os *logs*.

6.1 INCLUINDO OS *LOGS*

Serão apresentadas cinco simulações de inclusão de *logs*, que consideram o fluxo estabelecido na Figura 13. Dentro deste processo, o registro de *log* ocorre em quatro momentos:

- Etapa 1 - Validação de Dados do Pedido: A primeira etapa que necessita da gravação do *log* é a validação do pedido. A mesma ocorre no *middleware* de integração, em seu evento *ImportOrdersEvent*, descrito na Figura 14.
- Etapa 2 - Envio do pedido: O segundo registro de *log* é feito após o pedido ser enviado pelo *middleware* de integração para o sistema de mensageria do sistema de faturamento, ainda no evento *ImportOrdersEvent*.
- Etapa 3 - Recebimento do pedido: O terceiro *log* é feito pelo sistema de faturamento, em seu evento *ReceiveOrdersEvent*, descrito na Figura 15. Após o sistema de faturamento receber o pedido pelo sistema de mensageria e fazer a gravação do mesmo em seu banco de dados, o mesmo faz o registro do *log*.
- Etapa 4 - Faturamento do pedido: O quarto registro de *log* é feito pelo sistema de faturamento, em seu evento *InvoiceOrdersEvent*, após o faturamento do pedido. Esse processo pode ser visualizado na Figura 16.

6.1.1 Cenário 1

Primeiramente, construiu-se um cenário de sucesso de todo o processo, para que seja possível vê-lo do início ao fim. O Quadro 5 apresenta os dados informados na primeira etapa de

log, após a validação com sucesso dos dados do pedido. Em seguida, o pedido é enviado e o *log* descrito no Quadro 6 é gerado. O sistema de faturamento, por sua vez, recebe o pedido e grava o *log* descrito no Quadro 7. Por fim, o pedido é faturado com sucesso, gerando o *log* descrito pelo Quadro 8. A requisição completa, em formato JSON, pode ser vista na Seção A.1.1.

Quadro 5 – Etapa 1: Validação dos Dados do Pedido

Campo	**Valor**
*Classification**	*Notification*
*Type**	*ValidatingOrder*
*RegisterID**	f0d7d38b-428e-4df4-9f68-8e78c2e764a0
*RegisterType**	*IncomingOrder*
*OccurrenceDate**	2023-10-14T18:56:59.082Z
*OcurrenceAction**	*ImportOrdersEvent*
GroupID	f0d7d38b-428e-4df4-9f68-8e78c2e764a0
RegisterRelatedIDs	*Customer*: d09cb9ca-8188-4139-baf0-48c2a1d1745b, *Product*: 183d99f2-1ec3-4472-95bc-c9b9f4362bf4
SpecificInformations	*PurchaseOrder*: ORDEM1
Observations	Dados validados com sucesso
StatusResult	200 - *Ok*
MessageResult	null
Files	*RequestResponse*: visualizar conteúdo na Seção A.1.1.

* = campos obrigatórios.
Fonte: O Autor (2023).

Quadro 6 – Etapa 2: Envio do Pedido

Campo	**Valor**
*Classification**	*Notification*
*Type**	*SendindOrder*
*RegisterID**	f0d7d38b-428e-4df4-9f68-8e78c2e764a0
*RegisterType**	*IncomingOrder*
*OccurrenceDate**	2023-10-15T18:58:59.082Z
*OcurrenceAction**	*ImportOrdersEvent*
GroupID	f0d7d38b-428e-4df4-9f68-8e78c2e764a0
RegisterRelatedIDs	*Customer*: d09cb9ca-8188-4139-baf0-48c2a1d1745b, *Product*: 183d99f2-1ec3-4472-95bc-c9b9f4362bf4
SpecificInformations	*PurchaseOrder*: ORDEM1
Observations	Pedido enviado com sucesso
StatusResult	null
MessageResult	null
Files	null

* = campos obrigatórios.
Fonte: O Autor (2023).

Quadro 7 – Etapa 3: Recebimento do Pedido

Campo	**Valor**
*Classification**	*Notification*
*Type**	*ReceivingOrder*
*RegisterID**	713b5d2a-bc1d-45be-aca4-313ceb680943
*RegisterType**	*Order*
*OccurrenceDate**	2023-10-15T18:59:59.082Z
*OcurrenceAction**	*ReceiveOrderEvent*
GroupID	f0d7d38b-428e-4df4-9f68-8e78c2e764a0
RegisterRelatedIDs	*Customer*: d09cb9ca-8188-4139-baf0-48c2a1d1745b, *Product*: 183d99f2-1ec3-4472-95bc-c9b9f4362bf4
SpecificInformations	*PurchaseOrder*: ORDEM1
Observations	Pedido recebido com sucesso
StatusResult	null
MessageResult	null
Files	null

* = campos obrigatórios.
Fonte: O Autor (2023).

Quadro 8 – Etapa 4: Faturamento do Pedido

Campo	**Valor**
*Classification**	*Notification*
*Type**	*InvoicingOrder*
*RegisterID**	85be8410-a36e-4ffe-bd7e-a2b6cfe187fa
*RegisterType**	*Invoice*
*OccurrenceDate**	2023-10-15T19:00:00.082Z
*OcurrenceAction**	*InvoiceOrdersEvent*
GroupID	f0d7d38b-428e-4df4-9f68-8e78c2e764a0
RegisterRelatedIDs	*Customer*: d09cb9ca-8188-4139-baf0-48c2a1d1745b, *Product*: 183d99f2-1ec3-4472-95bc-c9b9f4362bf4
SpecificInformations	*PurchaseOrder*: ORDEM1
Observations	Pedido faturado com sucesso
StatusResult	null
MessageResult	null
Files	null

* = campos obrigatórios.
Fonte: O Autor (2023).

6.1.2 Cenário 2

O segundo cenário de teste consiste em uma primeira tentativa de validação do pedido, na qual ocorre falha. Este registro de *log* pode ser observado no Quadro 9. Após certo tempo, com o ajuste das informações do pedido, é feita uma nova tentativa de validação que ocorreu com sucesso. Este registro de *log* por sua vez, pode ser visto no Quadro 10.

Após a validação de sucesso os registros de *log* cessam, provavelmente porque houve algum problema não esperado e não tratado no momento do envio do pedido. O mecanismo proposto neste trabalho não tem como objetivo a identificação de falhas técnicas, e sim de falhas processuais ligadas ao processo e ao negócio. Entretanto, o mesmo pode servir de apoio

fornecendo informações que auxiliam na detecção de qual elemento ou evento de todo o processo em que a falha ocorreu. Isso facilita a busca dos *logs* técnicos, indicando em qual parte do sistema a investigação deve se concentrar. A requisição completa, em formato JSON, pode ser vista na Seção A.1.2.

Quadro 9 – Etapa 1: Validação dos Dados do Pedido

Campo	**Valor**
*Classification**	*Error*
*Type**	*ValidatingOrder*
*RegisterID**	31710a73-35a2-4993-804a-b5ef06eff264
*RegisterType**	*IncomingOrder*
*OccurrenceDate**	2023-10-14T18:56:59.082Z
*OcurrenceAction**	*ImportOrdersEvent*
GroupID	f1b8f1cf-81e2-4edb-9370-53cfade81696
RegisterRelatedIDs	*Customer*: e17ee2ed-2ee1-490a-b1b5-639a0dadbe67, *Product*: 77a3cb8d-1ed7-4149-9912-dbc6ebd636ee
SpecificInformations	*PurchaseOrder*: ORDEM2
Observations	Falha na validação dos dados, CNPJ inválido: 21.811.180/0001-62, IE inválida: 980.790.057.737
StatusResult	200 - *Ok*
MessageResult	null
Files	*RequestResponse*: visualizar conteúdo na Seção A.1.2

* = campos obrigatórios.
Fonte: O Autor (2023).

Quadro 10 – Etapa 1: Nova Tentativa de Validação dos Dados do Pedido

Campo	**Valor**
*Classification**	*Notification*
*Type**	*ValidatingOrder*
*RegisterID**	31710a73-35a2-4993-804a-b5ef06eff264
*RegisterType**	*IncomingOrder*
*OccurrenceDate**	2023-10-15T18:56:59.082Z
*OcurrenceAction**	*ImportOrdersEvent*
GroupID	f1b8f1cf-81e2-4edb-9370-53cfade81696
RegisterRelatedIDs	*Customer*: e17ee2ed-2ee1-490a-b1b5-639a0dadbe67, *Product*: 77a3cb8d-1ed7-4149-9912-dbc6ebd636ee
SpecificInformations	*PurchaseOrder*: ORDEM2
Observations	Dados validados com sucesso
StatusResult	200 - *Ok*
MessageResult	null
Files	*RequestResponse*: visualizar conteúdo na Seção A.1.2

* = campos obrigatórios.
Fonte: O Autor (2023).

6.1.3 Cenário 3

Para o cenário 3, explorou-se uma abordagem um pouco mais genérica para o uso do *log* de validação do pedido. Neste caso, se houver qualquer problema relacionado à estrutura e validação do pedido, um log de erro será gerado. Essa estratégia pode auxiliar na identificação

de problemas na estrutura dos dados esperados mesmo sem uma validação específica, e colabora também para a identificação de regras necessárias ao processo. No exemplo descrito no Quadro 11, o erro ocorre porque a validação é feita considerando que o endereço do comprador virá preenchido, mas por algum motivo o mesmo não está sendo disponibilizado. Isso pode ser causado por uma falha ou falta de validação no *e-commerce*, ou talvez em um cenário não mapeado, o cliente selecionou o envio do produto para seu endereço padrão de cadastro e seja necessário consultar o mesmo em algum momento. A requisição completa, em formato JSON, pode ser vista na Seção A.1.3.

Quadro 11 – Etapa 1: Validação dos Dados do Pedido

Campo	**Valor**
*Classification**	*Error*
*Type**	*ValidatingOrder*
*RegisterID**	4c171f09-8ca5-4d9a-a649-bba5d46fcb7f
*RegisterType**	*IncomingOrder*
*OccurrenceDate**	2023-10-14T18:56:59.082Z
*OcurrenceAction**	*ImportOrdersEvent*
GroupID	4c171f09-8ca5-4d9a-a649-bba5d46fcb7f
RegisterRelatedIDs	*Customer*: 031a6da9-9786-4d1c-a953-c176c44a233c
SpecificInformations	*PurchaseOrder*: ORDEM3
Observations	Erro não catalogado ao validar dados da ordem, Endereco NullPointerException
StatusResult	200 - *Ok*
MessageResult	null
Files	RequestResponse: visualizar conteúdo na Seção A.1.3

* = campos obrigatórios.
Fonte: O Autor (2023).

6.1.4 Cenário 4

No cenário 4, por sua vez, o erro ocorre na última etapa do processo: o envio da nota fiscal para o Sefaz. Neste caso, uma CFOP inválida foi informada. Como o faturamento é feita de forma automática pelo evento *InvoiceOrdersEvent*, é possível que exista um erro no preenchimento das configurações do sistema ou um erro de código em que a regra não foi implementada corretamente. Em ambos os casos, o erro está relacionado ao processo ou a regras de negócio do faturador. Em Quadro 12, Quadro 13 e Quadro 14, encontram-se os *logs* de sucesso gerados pelas etapas antecessoras. No Quadro 15 encontra-se o *log* de falha gerado pela etapa 4. A requisição completa, em formato JSON, pode ser vista na Seção A.1.4.

Quadro 12 – Etapa 1: Validação dos Dados do Pedido

Campo	**Valor**
*Classification**	*Notification*
*Type**	*ValidatingOrder*
*RegisterID**	07abdf46-2d63-4522-b949-20f0ea9a07b0
*RegisterType**	*IncomingOrder*
*OccurrenceDate**	2023-10-14T18:56:59.082Z
*OcurrenceAction**	*ImportOrdersEvent*
GroupID	07abdf46-2d63-4522-b949-20f0ea9a07b0
RegisterRelatedIDs	*Customer*: 4625a8de-7e40-4d6f-ba23-3deb4a79dab5, *Product*: 0eb9bed8-2c0e-458b-8c3e-d0b3e3fadc92
SpecificInformations	*PurchaseOrder*: ORDEM4
Observations	Dados validados com sucesso
StatusResult	200 - *Ok*
MessageResult	null
Files	*RequestResponse*: visualizar conteúdo na Seção A.1.4

* = campos obrigatórios.
Fonte: O Autor (2023).

Quadro 13 – Etapa 2: Envio do Pedido

Campo	**Valor**
*Classification**	*Notification*
*Type**	*SendingOrder*
*RegisterID**	07abdf46-2d63-4522-b949-20f0ea9a07b0
*RegisterType**	*IncomingOrder*
*OccurrenceDate**	2023-10-14T18:59:59.082Z
*OcurrenceAction**	*ImportOrdersEvent*
GroupID	07abdf46-2d63-4522-b949-20f0ea9a07b0
RegisterRelatedIDs	*Customer*: 4625a8de-7e40-4d6f-ba23-3deb4a79dab5, *Product*: 0eb9bed8-2c0e-458b-8c3e-d0b3e3fadc92
SpecificInformations	*PurchaseOrder*: ORDEM4
Observations	Pedido enviado com sucesso
StatusResult	null
MessageResult	null
Files	null

* = campos obrigatórios.
Fonte: O Autor (2023).

Quadro 14 – Etapa 3: Recebimento do Pedido

Campo	**Valor**
*Classification**	*Notification*
*Type**	*ReceivingOrder*
*RegisterID**	77ab97b6-6275-4492-8897-0680ff706a65
*RegisterType**	*Order*
*OccurrenceDate**	2023-10-14T19:59:59.082Z
*OcurrenceAction**	*ReceiveOrdersEvent*
GroupID	07abdf46-2d63-4522-b949-20f0ea9a07b0
RegisterRelatedIDs	*Customer*: 4625a8de-7e40-4d6f-ba23-3deb4a79dab5, *Product*: 0eb9bed8-2c0e-458b-8c3e-d0b3e3fadc92
SpecificInformations	*PurchaseOrder*: ORDEM4
Observations	Pedido recebido com sucesso
StatusResult	null
MessageResult	null
Files	null

* = campos obrigatórios.
Fonte: O Autor (2023).

Quadro 15 – Etapa 4: Faturamento do Pedido

Campo	**Valor**
*Classification**	*Error*
*Type**	*InvoicingOrder*
*RegisterID**	f211e7e7-13ab-4f25-893c-0a5f33e59b48
*RegisterType**	*Invoice*
*OccurrenceDate**	2023-10-14T20:59:59.082Z
*OcurrenceAction**	*InvoiceOrdersEvent*
GroupID	07abdf46-2d63-4522-b949-20f0ea9a07b0
RegisterRelatedIDs	*Customer*: 4625a8de-7e40-4d6f-ba23-3deb4a79dab5, *Product*: 0eb9bed8-2c0e-458b-8c3e-d0b3e3fadc92
SpecificInformations	*PurchaseOrder*: ORDEM4
Observations	Erro ao autorizar nota fiscal no sefaz, Estado origem: RS - Estado destino: SP - CFOP: 5102
StatusResult	400
MessageResult	CFOP inválida para este tipo de operação
Files	null

* = campos obrigatórios.
Fonte: O Autor (2023).

6.1.5 Cenário 5

O quinto e último cenário não retrata um erro, e sim um aviso. Neste caso, o envio da nota fiscal não pôde ser feito pois o serviço do Sefaz estava indisponível no momento. O *log* é de aviso pois não houve um erro do sistema, mas de qualquer forma a operação não pôde ser concluída e é necessário ter um registro do ocorrido. O evento de *InvoiceOrdersEvent* é disparado automaticamente a cada 30 minutos, então não é necessário tomar nenhuma ação. A cada 30 minutos haverá uma nova tentativa de fazer o envio da nota fiscal, até que o Sefaz fique disponível novamente e o envio seja concluído, com sucesso ou falha. Em Quadro 16, Quadro 17 e Quadro 18, encontram-se os *logs* de sucesso gerados pelas etapas antecessoras. No

Quadro 19 encontra-se o *log* de falha gerado pela etapa 4. A requisição completa, em formato JSON, pode ser vista na Seção A.1.5.

Quadro 16 – Etapa 1: Validação dos Dados do Pedido

Campo	**Valor**
*Classification**	*Notification*
*Type**	*ValidatingOrder*
*RegisterID**	016201fa-bd02-4328-bdea-217bc3c01a0b
*RegisterType**	*IncomingOrder*
*OccurrenceDate**	2023-10-14T18:56:59.082Z
*OcurrenceAction**	*ImportOrdersEvent*
GroupID	016201fa-bd02-4328-bdea-217bc3c01a0b
RegisterRelatedIDs	Customer: 45172dd6-c34d-4612-aa16-63f765b082c2, Product: 83b578a4-7669-425d-949a-4fe5eec4166b
SpecificInformations	*PurchaseOrder*: ORDEM5
Observations	Dados validados com sucesso
StatusResult	200 - *Ok*
MessageResult	null
Files	*RequestResponse*: visualizar conteúdo na Seção A.1.5

* = campos obrigatórios.
Fonte: O Autor (2023).

Quadro 17 – Etapa 2: Envio do Pedido

Campo	**Valor**
*Classification**	*Notification*
*Type**	*SendingOrder*
*RegisterID**	5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6
*RegisterType**	*IncomingOrder*
*OccurrenceDate**	2023-10-14T18:59:59.082Z
*OcurrenceAction**	*ImportOrdersEvent*
GroupID	5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6
RegisterRelatedIDs	*Customer*: c0e6fb14-9a43-4b8f-abe0-def92d5fb5be, *Product*: 78b5b43e-8fca-42a6-bd77-e5f26d400f81
SpecificInformations	*PurchaseOrder*: ORDEM5
Observations	Pedido enviado com sucesso
StatusResult	null
MessageResult	null
Files	null

* = campos obrigatórios.
Fonte: O Autor (2023).

Quadro 18 – Etapa 3: Recebimento do Pedido

Campo	**Valor**
*Classification**	*Notification*
*Type**	*ReceivingOrder*
*RegisterID**	020ae440-bb57-4e8c-9b58-cacf45f985ec
*RegisterType**	*Order*
*OccurrenceDate**	2023-10-14T19:56:59.082Z
*OcurrenceAction**	*ReceiveOrdersEvent*
GroupID	5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6
RegisterRelatedIDs	*Customer*: c0e6fb14-9a43-4b8f-abe0-def92d5fb5be, *Product*: 78b5b43e-8fca-42a6-bd77-e5f26d400f81
SpecificInformations	*PurchaseOrder*: ORDEM5
Observations	Pedido recebido com sucesso
StatusResult	null
MessageResult	null
Files	null

* = campos obrigatórios.
Fonte: O Autor (2023).

Quadro 19 – Etapa 4: Faturamento do Pedido

Campo	**Valor**
*Classification**	*Warning*
*Type**	*InvoicingOrder*
*RegisterID**	f8d9b757-f1cc-417d-9330-1e59c1778e6a
*RegisterType**	*Invoice*
*OccurrenceDate**	2023-10-14T20:56:59.082Z
*OcurrenceAction**	*InvoiceOrdersEvent*
GroupID	5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6
RegisterRelatedIDs	*Customer*: c0e6fb14-9a43-4b8f-abe0-def92d5fb5be, *Product*: 78b5b43e-8fca-42a6-bd77-e5f26d400f81
SpecificInformations	*PurchaseOrder*: ORDEM5
Observations	Houve uma falha ao conectar com o SEFAZ, uma nova tentativa será feita em 30 minutos.
StatusResult	503 - *ServiceUnavailable*
MessageResult	Erro Cod 3453 - Serviço indisponível, tente novamente mais tarde
Files	null

* = campos obrigatórios.
Fonte: O Autor (2023).

6.2 CONSULTANDO OS *LOGS*

Serão apresentadas seis simulações de consultas de *logs*, juntamente com o resultado das mesmas. Serão descritos os filtros e valores utilizados para as simulação, baseados nas respectiva interface descrita no Quadro 2. Para indicar os registros retornados, serão indicados os quadros nos quais foram definidos previamente, no momento de inclusão dos *logs*. A única diferença é que na inclusão os arquivos são enviados em formato de texto, e na consulta são retornados os *links* de acesso ao arquivo, conforme interface descrita no Quadro 3. Essas interfaces, entretando, mantem-se genéricas. Ou seja, podem ser utilizadas em diferentes contextos, e os valores utilizados para filtro e dados inclusos nos *logs* é que caracterizam o contexto ao

qual estão sendo aplicadas.

6.2.1 Cenário 1

No primeiro cenário, os *logs* foram consultados pelas classificações de erro e aviso, e pelo intervalo de datas de ocorrência. Essa combinação de filtros pode ser utilizada com o intuito de visualizar todos erros e avisos que ocorreram em determinado período. O Quadro 20 apresenta os valores dos filtros utilizados. Os *logs* retornados nessa consulta foram identificados previamente nos seguintes quadros: Quadro 9, Quadro 11, Quadro 19 e Quadro 15. A requisição e retorno completos podem ser vistos na Seção A.2.1, em formato JSON.

Quadro 20 – Consulta por classificação e intervalo de datas

Campo	**Valor**
Classifications	*Error*, *Warning*
OccurrenceDateStart	2023-10-14T00:00:00.082Z
OccurrenceDateEnd	2023-10-30T23:59:59.083Z

Fonte: O Autor (2023).

6.2.2 Cenário 2

Neste cenário foram utilizados os filtros de classificação e tipo, buscando identificar os *logs* de erro realizados pela operação de validação do pedido. O Quadro 21 apresenta os filtros e valores utilizados, enquanto os registros retonados foram os definidos no Quadro 9 e no Quadro 11. A requisição e retorno completos podem ser vistos na Seção A.2.2, em formato JSON.

Quadro 21 – Consulta por classificação e tipo

Campo	**Valor**
Classifications	*Error*
Types	*ValidatingOrder*

Fonte: O Autor (2023).

6.2.3 Cenário 3

Para o terceiro cenário, foram utilizados os filtros de classificação e ação que gerou a ocorrência, a fim de consultar os registros de erro que ocorreram ao importar os pedidos. No Quadro 22 encontram-se os filtros utilizados, enquanto os registros retornados podem ser vistos no Quadro 9 e no Quadro 11. A requisição e retorno completos podem ser vistos na Seção A.2.3, em formato JSON.

Quadro 22 – Consulta por classificação e ação que gerou a ocorrência

Campo	**Valor**
Classifications	*Error*
OccurrenceActions	*InvoiceOrdersEvent*

Fonte: O Autor (2023).

6.2.4 Cenário 4

Para este cenário foram utilizados os filtros de identificador e tipo do registro, com o objetivo de buscar *logs* relacionados a um registro específico. No Quadro 23 encontram-se os filtros utilizados, e os registros retornados podem ser vistos no Quadro 5 e no Quadro 6. A requisição e retorno completos podem ser vistos na Seção A.2.4, em formato JSON.

Quadro 23 – Consulta por identificador e tipo de registro

Campo	**Valor**
RegisterID	*85be8410-a36e-4ffe-bd7e-a2b6cfe187fa*
RegisterType	*Invoice*

Fonte: O Autor (2023).

6.2.5 Cenário 5

Neste cenário, foram utilizados os filtros de classificação e tipo de registro, buscando identificar *logs* de aviso para as notas fiscais. Os filtros utilizados encontram-se no Quadro 24, e o único registro retornado pode ser visualizado no Quadro 16. A requisição e retorno completos podem ser vistos na Seção A.2.5, em formato JSON.

Quadro 24 – Consulta por classificação e tipo de registro

Campo	**Valor**
Classifications	*Warning*
RegisterType	*Invoice*

Fonte: O Autor (2023).

6.2.6 Cenário 6

Foi utilizado apenas o filtro de identificador do grupo para o sexto cenário, com o intuito de trazer todos os logs relacionados a um mesmo processo, mesmo que estejam relacionados a registros diferentes. Sendo assim, devem ser retornados os registros das quatro etapas vinculadas a este mesmo identificador de grupo. Neste caso, os registros retornados estão descritos nos quadros a seguir: Quadro 16, Quadro 17, Quadro 18 e Quadro 19. O Quadro 25 exibe o valor utilizado para o filtro. A requisição e retorno completos podem ser vistos na Seção A.2.6, em formato JSON.

Quadro 25 – Consulta por classificação e intervalo de datas

Campo	**Valor**
GroupID	5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6

Fonte: O Autor (2023).

7 CONSIDERAÇÕES FINAIS

A partir deste trabalho foi possível compreender que ainda existem pontos a serem estudados e aprimorados no que diz respeito a *logging* e monitoramento de microsserviços. Essa atividade, bastante comum e relativamente simples em sistemas tradicionais monolíticos, se torna bastante complexa quando aplicada em arquiteturas de microsserviços. Atualmente existem algumas orientações no estado da arte sobre padrões e boas práticas que devem ser observadas, entrento ainda existe um vasto campo a ser explorado para aprimorar técnicas e resolver estes problemas de forma mais assertiva.

A solução proposta neste trabalho tem por objetivo suprir a necessidade de *logs* voltados ao processo e regras de negócio, por esse motivo os *logs* são entrelaçados com as etapas do processo, para que seja possível a identificação de falhas ou gargalos no mesmo. Apesar de haverem indícios dessa dificuldade, não foi encontrada abordagem proposta para tal. Em primeiro momento, a solução proposta não substitui *logs* e *tracing* distribuído, mas funciona como um complemento para suprir esse tipo de informação que atualmente fica soterrada sob um grande volume de *logs* necessários para o monitoramento técnico, a nível de *hardware* e *software*.

Além disso, a implementação deste modelo pode reduzir grandemente o tempo de profissionais de alto nível técnico dedicados a busca e rastreamento de falhas por *logs* tradicionais e *tracing* distribuído nos casos em que a situação averiguada esteja ligada a um problema de processo ou negócio. Ao aplicar o modelo, o conjunto de informações necessárias para essa verificação ficam exmplícitas, e disponibilizando-as ao suporte técnico da empresa a partir de uma interface de usuário, por exemplo, é possível que essas situações sejam contornadas sem que seja necessária a intervenção de um analista ou arquiteto de *software*. Desta forma, o modelo funciona também como um filtro, para que esses profissionais só sejam mobilizados quando a necessidade realmente existir.

Ao observar que existem dificuldades em padronizar *logs* sem perder informações específicas de cada microsserviço, a abordagem proposta neste trabalho buscou criar uma estrutura já padronizada de comunicação que respeita a especificidade de cada microsserviço. A interface de consulta também foi pensada para que a busca de registros de *logs* fosse facilitada, já que a tendência é que a quantidade de registros cresça exponencialmente junto com o crescimento natural do sistema.

A padronização da estrutura de dados aliada ao enriquecimento de suas informações também beneficia a aplicação de algoritmos de IA. Assim, possibilita-se que destes dados dados de *logs* possa ser extraído um conjunto também mais rico de informações, que podem inclusive trazer benefícios em nível estratégico para as empresas.

A ideia é bastante semelhante ao padrão Rest: todos precisam se comunicar por uma

via (protocolo HTTP), podem existir diversos sistemas, construídos em diferentes tecnologias, mas ao definir um formato padrão, a complexidade de comunicação entre esses sistemas tão diferentes se torna menos complexa, sem que haja perda de dados. Apesar de que ainda existe uma outra etapa necessário, para a definição dos valores para os campos que essa interface de comunicação demanda, ao partir de uma interface pré definida grande parte da complexidade do problema é absorvida.

A abordagem se mostrou efetiva para seu propósito. Entretanto, ainda pode ser melhor explorada. Os próximos passos seriam avançar com a avaliação da abordagem para que cobrissem também *logs* de falhas de *hardware* e *software*, juntamente a outros dados de monitoramento.

REFERÊNCIAS

BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**. 4. ed. Estados Unidos: Addison-Wesley Professional, 2021. ISBN 9780136885979.

BELLEMARE, A. **Building Event-Driven Microservices: Leveraging Organizational Data at Scale**. Estados Unidos: "O'Reilly Media, Inc.", 2020. ISBN 9781492057840; 1492057843.

BROWN, W. J. **AntiPatterns: refactoring software, architectures, and projects in crisis**. Canadá: John Wiley Sons, 2001. ISBN 0471197130; 9780471197133.

BURNS, B. **Designing distributed systems: patterns and paradigms for scalable, reliable services**. 1. ed. Estados Unidos: O'Reilly Media, 2018. ISBN 9781491983645; 1491983647.

CABRI, G.; LEONARDI, L.; ZAMBONELLI, F. Mobile-agent coordination models for internet applications. **IEEE Computer**, IEEE, v. 33, p. 82–89, 2000.

CERNY, T. *et al.* Catalog and detection techniques of microservice anti-patterns and bad smells: A tertiary study. **Journal of Systems and Software**, ScienceDirect, v. 206, 2023.

CINQUE, M.; CORTE, R. D.; PECCHIA, A. Microservices Monitoring with Event Logs and Black Box Execution Tracing. **IEEE World Congress on Services (SERVICES)**, IEEE, 2021.

ERL, T. **Service-Oriented Architecture: Analysis and Design for Services and Microservices**. 2. ed. Estados Unidos: Prentice Hall, 2017. (The Prentice Hall Service Technology Series from Thomas Erl). ISBN 9780133858587; 0133858588.

EVANS, E. **Domain-Driven Design - Tackling Complexity in the Heart of Software**. 1. ed. Estados Unidos: Addison Wesley, 2003. ISBN 0321125215; 9780321125217.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. Tese (Doutorado) — University of California, 2000.

FOWLER, M. *et al.* **Refactoring - Improving the Design of Existing Code**. 1. ed. Estados Unidos: Addison-Wesley Professional, 1999. ISBN 9780201485677; 0201485672.

GAMMA, E. *et al.* **Design Patterns: Elements of Reusable Object-Oriented Software**. illustrated edition. Estados Unidos: addison-wesley, 1994. ISBN 9780201633610; 0201633612.

KLEPPMANN, M. **Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems**. Estdos Unidos: O'Reilly Media, 2017. ISBN 1449373321; 9781449373320.

LEE, C. *et al.* Eadro: An end-to-end troubleshooting framework for microservices on multi-source data. **ACM 45th International Conference on Software Engineering (ICSE)**, IEEE, 2023.

MAJORS LIZ FONG-JONES, G. M. C. **Observability Engineering: Achieving Production Excellence**. 1. ed. Estados Unidos: O'Reilly Media, 2022. ISBN 1492076449; 9781492076445.

MASSE, M. **REST API Design Rulebook**. Estados Unidos: O'Reilly Media, 2011. ISBN 1449310508; 9781449310509.

MüHL LUDGER FIEGE, P. P. G. **Distributed Event-Based Systems**. 1. ed. Alemanha: Springer-Verlag, 2006. ISBN 9783540326519; 3540326510.

NEWMAN, S. **Building Microservices: Designing Fine-Grained Systems**. 2. ed. [S.l.]: O'Reilly Media, 2021. ISBN 9781492034025; 1492034029.

RICHARDS, M.; FORD, N. **Fundamentals of Software Architecture: An Engineering Approach**. 1. ed. Estados Unidos: O'Reilly Media, 2020. ISBN 1492043451; 9781492043454.

SADALAGE, M. F. P. J. **NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence**. 1. ed. Estdos Unidos: Addison-Wesley Professional, 2012. ISBN 0321826620; 9780321826626.

SAMPAIO, R.; MANCINI, M. Estudos de revisao sistematica um guia para sintese criteriosa da evidencia cientifica. **Revista Brasileira De Fisioterapia**, p. 83–89, 2007. ISSN 1413-3555.

SOLDANI, J.; BROGI, A. Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey. **ACM Computing Surveys**, Association for Computing Machinery, v. 55, p. 1–39, 2023.

STEEN, M. van; TANENBAUM, A. S. **Distributed Systems: Principles and Paradigms**. 4. ed. [S.l.]: Maarten van Steen, 2023. ISBN 9081540637; 9789081540636.

TAMBURRI, D. A.; MIGLIERINA, M.; NITTO, E. D. Cloud applications monitoring: An industrial study. **Information and Software Technology**, ScienceDirect, v. 127, 2020.

VALDERAS, P.; TORRES, V.; PELECHANO, V. A microservice composition approach based on the choreography of bpmn fragments. **Information Software Technology**, ScienceDirect, v. 127, 2020.

WANG, H. *et al.* Groot: An event-graph-based approach for root cause analysis in industrial settings. **36th IEEE/ACM International Conference on Automated Software Engineering (ASE)**, IEEE, 2021.

WASEEM, M. *et al.* Design, monitoring, and testing of microservices systems: The practitioners' perspective. **Journal of Systems and Software**, ScienceDirect, v. 182, 2021.

WOLFF, E. **Microservices Flexible Software Architecture**. Estados Unidos: Addison-Wesley Professional, 2017. ISBN 9780134602417.

ZHOU, X. *et al.* Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry. **Journal of Systems and Software**, ScienceDirect, v. 195, 2023.

ANEXO A – REQUISIÇÕES COMPLETAS DOS TESTES

As imagens deste anexo apresentam as requisições completas utilizadas para execução dos testes apresentados no Capítulo 6.

A.1 REQUISIÇÕES DE INCLUSÃO DE *LOGS*

A.1.1 Cenário 1

Figura 17 – Etapa 1: Validação do Pedido


```
{
    "Classification": "Notification",
    "Type": "ValidatingOrder",
    "RegisterID": "f0d7d38b-428e-4df4-9f68-8e78c2e764a0",
    "RegisterType": "IncomingOrder",
    "OccurrenceAction": "ImportOrdersEvent",
    "OccurrenceDate" : "2023-10-14T18:56:59.082Z",
    "GroupID" : "f0d7d38b-428e-4df4-9f68-8e78c2e764a0",
    "RegisterRelatedIDs": {
        "Customer" : "d09cb9ca-8188-4139-baf0-48c2a1d1745b",
        "Product": "183d99f2-1ec3-4472-95bc-c9b9f4362bf4"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM1"
    },
    "Observations": [
        "Dados validados com sucesso"
    ],
    "StatusResult": 200,
    "MessageResult": null,
    "Files": {
        "RequestResponse" : "{\"Nome\":\"LuaneSophiaConstruçõesME\",\"CNPJ\":\"93.
            398.820/0001-64\",\"IE\":\"078.984.807.749\",
            \"Endereço\":\"PraçaVicenteNunes\",\"Numero\":\"766\",
            \"Bairro\":\"ConjuntoHabitacionalBrigadeiroFariaLima\",
            \"Cidade\":\"SãoPaulo\",\"UF\":\"SP\",\"Produtos\":[\"SKU\":\"PROD321\",
            \"Valor\":110.00,\"Desconto\":10.00,\"Quantidade\":1],
            \"ValorParcial\":110.00,\"Frete\":20.00,\"DescontoTotal\":10.00,
            \"ValorTotal\":120.00}"
    }
}
```


Fonte: O Autor (2023)

Figura 18 – Etapa 2: Envio do Pedido


```
{
    "Classification": "Notification",
    "Type": "SendingOrder",
    "RegisterID": "f0d7d38b-428e-4df4-9f68-8e78c2e764a0",
    "RegisterType": "IncomingOrder",
    "OccurrenceAction": "ImportOrdersEvent",
    "OccurrenceDate" : "2023-10-15T18:58:59.082Z",
    "GroupID" : "f0d7d38b-428e-4df4-9f68-8e78c2e764a0",
    "RegisterRelatedIDs": {
        "Customer" : "d09cb9ca-8188-4139-baf0-48c2a1d1745b",
        "Product": "183d99f2-1ec3-4472-95bc-c9b9f4362bf4"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM1"
    },
    "Observations": [
        "Pedido enviado com sucesso"
    ],
    "StatusResult": null,
    "MessageResult": null,
    "Files": {}
}
```


Fonte: O Autor (2023)

Figura 19 – Etapa 3: Recebimento do Pedido


```
{
    "Classification": "Notification",
    "Type": "ReceivingOrder",
    "RegisterID": "713b5d2a-bc1d-45be-aca4-313ceb680943",
    "RegisterType": "Order",
    "OccurrenceAction": "ReceiveOrderEvent",
    "OccurrenceDate" : "2023-10-15T18:59:59.082Z",
    "GroupID" : "f0d7d38b-428e-4df4-9f68-8e78c2e764a0",
    "RegisterRelatedIDs": {
        "Customer" : "d09cb9ca-8188-4139-baf0-48c2a1d1745b",
        "Product": "183d99f2-1ec3-4472-95bc-c9b9f4362bf4"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM1"
    },
    "Observations": [
        "Pedido recebido com sucesso"
    ],
    "StatusResult": null,
    "MessageResult": null,
    "Files": {}
}
```


Fonte: O Autor (2023)

Figura 20 – Etapa 4: Faturamento do Pedido


```
{
    "Classification": "Notification",
    "Type": "InvoicingOrder",
    "RegisterID": "85be8410-a36e-4ffe-bd7e-a2b6cfe187fa",
    "RegisterType": "Invoice",
    "OccurrenceAction": "InvoiceOrdersEvent",
    "OccurrenceDate" : "2023-10-15T18:59:59.082Z",
    "GroupID" : "f0d7d38b-428e-4df4-9f68-8e78c2e764a0",
    "RegisterRelatedIDs": {
        "Customer" : "d09cb9ca-8188-4139-baf0-48c2a1d1745b",
        "Product": "183d99f2-1ec3-4472-95bc-c9b9f4362bf4"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM1"
    },
    "Observations": [
        "Pedido faturado com sucesso"
    ],
    "StatusResult": null,
    "MessageResult": null,
    "Files": {}
}
```


Fonte: O Autor (2023)

A.1.2 Cenário 2

Figura 21 – Etapa 1: Validação do Pedido


```
{
    "Classification": "Error",
    "Type": "ValidatingOrder",
    "RegisterID": "31710a73-35a2-4993-804a-b5ef06eff264",
    "RegisterType": "IncomingOrder",
    "OccurrenceAction": "ImportOrdersEvent",
    "OccurrenceDate" : "2023-10-14T18:56:59.082Z",
    "GroupID" : "f1b8f1cf-81e2-4edb-9370-53cfade81696",
    "RegisterRelatedIDs": {
        "Customer" : "e17ee2ed-2ee1-490a-b1b5-639a0dadbe67",
        "Product": "77a3cb8d-1ed7-4149-9912-dbc6ebd636ee"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM2"
    },
    "Observations": [
        "Falha na validação dos dados",
        "CNPJ inválido: 21.811.180/0001-62",
        "IE inválida: 980.790.057.737"
    ],
    "StatusResult": 200,
    "MessageResult": null,
    "Files": {
        "RequestResponse" : "{\"Nome\":\"RosaeMarcosViniciusEntulhosME\",
            \"CNPJ\":\"21.811.180/0001-62\",\"IE\":\"980.790.057.737\",
            \"Endereço\":\"AvenidaPenedo\",\"Numero\":\"951\",
            \"Bairro\":\"Catiapoa\",\"Cidade\":\"SãoVicente\",\"UF\":\"SP\",
            \"Produtos\":[\"SKU\":\"PROD123\",\"Valor\":150.00,\"Desconto\":0.00,
            \"Quantidade\":1],\"ValorParcial\":150.00,\"Frete\":35.00,
            \"DescontoTotal\":0.00,\"ValorTotal\":185.00}"
    }
}
```


Fonte: O Autor (2023)

Figura 22 – Etapa 1: Nova Tentativa de Validação do Pedido


```
{
    "Classification": "Notification",
    "Type": "ValidatingOrder",
    "RegisterID": "31710a73-35a2-4993-804a-b5ef06eff264",
    "RegisterType": "IncomingOrder",
    "OccurrenceAction": "ImportOrdersEvent",
    "OccurrenceDate" : "2023-10-15T18:56:59.082Z",
    "GroupID" : "f1b8f1cf-81e2-4edb-9370-53cfade81696",
    "RegisterRelatedIDs": {
        "Customer" : "e17ee2ed-2ee1-490a-b1b5-639a0dadbe67",
        "Product": "77a3cb8d-1ed7-4149-9912-dbc6ebd636ee"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM2"
    },
    "Observations": [
        "Dados validados com sucesso"
    ],
    "StatusResult": 200,
    "MessageResult": null,
    "Files": {
        "RequestResponse" : "{\"Nome\":\"RosaeMarcosViniciusEntulhosME\",
            \"CNPJ\":\"21.811.180/0001-61\",\"IE\":\"980.790.057.735\",
            \"Endereço\":\"AvenidaPenedo\",\"Numero\":\"951\",
            \"Bairro\":\"Catiapoa\",\"Cidade\":\"SãoVicente\",\"UF\":\"SP\",
            \"Produtos\":[\"SKU\":\"PROD123\",\"Valor\":150.00,\"Desconto\":0.00,
            \"Quantidade\":1],\"ValorParcial\":150.00,\"Frete\":35.00,
            \"DescontoTotal\":0.00,\"ValorTotal\":185.00}"
    }
}
```


Fonte: O Autor (2023)

A.1.3 Cenário 3

Figura 23 – Etapa 1: Validação do Pedido


```
{
    "Classification": "Error",
    "Type": "ValidatingOrder",
    "RegisterID": "4c171f09-8ca5-4d9a-a649-bba5d46fcb7f",
    "RegisterType": "IncomingOrder",
    "OccurrenceAction": "ImportOrdersEvent",
    "OccurrenceDate" : "2023-10-14T18:56:59.082Z",
    "GroupID" : "4c171f09-8ca5-4d9a-a649-bba5d46fcb7f",
    "RegisterRelatedIDs": {
        "Customer" : "031a6da9-9786-4d1c-a953-c176c44a233c",
        "Product": "4e04bf27-43b1-4622-a7e9-6c23df1eecae"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM3"
    },
    "Observations": [
        "Erro não catalogado ao validar dados da ordem"
    ],
    "StatusResult": 200,
    "MessageResult": null,
    "Files": {
        "RequestResponse" : "{\"Nome\":\"FranciscoeNinaContábilME\",\"CNPJ\":\"17.
            059.556/0001-56\",\"IE\":\"580.485.047.261\",
            \"Endereço\":\"RuaPauloRodriguesPrado\",\"Numero\":\"\",
            \"Bairro\":\"JardimMendonçaI\",\"Cidade\":\"Bauru\",\"UF\":\"SP\",
            \"Produtos\":[\"SKU\":\"PROD789\",\"Valor\":325.00,\"Desconto\":0.00,
            \"Quantidade\":2],\"ValorParcial\":650.00,\"Frete\":50.00,
            \"DescontoTotal\":0.00,\"ValorTotal\":700.00}"
    }
}
```


Fonte: O Autor (2023)

A.1.4 Cenário 4

Figura 24 – Etapa 1: Validação do Pedido


```
{
    "Classification": "Notification",
    "Type": "ValidatingOrder",
    "RegisterID": "07abdf46-2d63-4522-b949-20f0ea9a07b0",
    "RegisterType": "IncomingOrder",
    "OccurrenceAction": "ImportOrdersEvent",
    "OccurrenceDate" : "2023-10-14T18:56:59.082Z",
    "GroupID" : "07abdf46-2d63-4522-b949-20f0ea9a07b0",
    "RegisterRelatedIDs": {
        "Customer" : "4625a8de-7e40-4d6f-ba23-3deb4a79dab5",
        "Product": "0eb9bed8-2c0e-458b-8c3e-d0b3e3fadc92"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM4"
    },
    "Observations": [
        "Dados validados com sucesso"
    ],
    "StatusResult": 200,
    "MessageResult": null,
    "Files": {
        "RequestResponse" : "{\"Nome\":\"EmanueleThiagoFinanceiraME\",\"CNPJ\":\"78.
            950.238/0001-75\",\"IE\":\"430.396.100.462\",
            \"Endereço\":\"RuaElianaCristinaJardinetti\",\"Numero\":\"750\",
            \"Bairro\":\"VotuporangaI\",\"Cidade\":\"Votuporanga\",\"UF\":\"SP\",
            \"Produtos\":[\"SKU\":\"PROD456\",\"Valor\":550.00,\"Desconto\":0.00,
            \"Quantidade\":1],\"ValorParcial\":550.00,\"Frete\":50.00,
            \"DescontoTotal\":0.00,\"ValorTotal\":600.00}"
    }
}
```


Fonte: O Autor (2023)

Figura 25 – Etapa 2: Envio do Pedido


```
{
    "Classification": "Notification",
    "Type": "SendingOrder",
    "RegisterID": "07abdf46-2d63-4522-b949-20f0ea9a07b0",
    "RegisterType": "IncomingOrder",
    "OccurrenceAction": "ImportOrdersEvent",
    "OccurrenceDate" : "2023-10-14T18:59:59.082Z",
    "GroupID" : "07abdf46-2d63-4522-b949-20f0ea9a07b0",
    "RegisterRelatedIDs": {
        "Customer" : "4625a8de-7e40-4d6f-ba23-3deb4a79dab5",
        "Product": "0eb9bed8-2c0e-458b-8c3e-d0b3e3fadc92"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM4"
    },
    "Observations": [
        "Pedido enviado com sucesso"
    ],
    "StatusResult": null,
    "MessageResult": null,
    "Files": {}
}
```


Fonte: O Autor (2023)

Figura 26 – Etapa 3: Recebimento do Pedido


```
{
    "Classification": "Notification",
    "Type": "ReceivingOrder",
    "RegisterID": "77ab97b6-6275-4492-8897-0680ff706a65",
    "RegisterType": "Order",
    "OccurrenceAction": "ReceiveOrdersEvent",
    "OccurrenceDate" : "2023-10-14T19:59:59.082Z",
    "GroupID" : "07abdf46-2d63-4522-b949-20f0ea9a07b0",
    "RegisterRelatedIDs": {
        "Customer" : "4625a8de-7e40-4d6f-ba23-3deb4a79dab5",
        "Product": "0eb9bed8-2c0e-458b-8c3e-d0b3e3fadc92"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM4"
    },
    "Observations": [
        "Pedido recebido com sucesso"
    ],
    "StatusResult": null,
    "MessageResult": null,
    "Files": {}
}
```


Fonte: O Autor (2023)

Figura 27 – Etapa 4: Faturamento do Pedido


```
{
    "Classification": "Error",
    "Type": "InvoicingOrder",
    "RegisterID": "f211e7e7-13ab-4f25-893c-0a5f33e59b48",
    "RegisterType": "Invoice",
    "OccurrenceAction": "InvoiceOrdersEvent",
    "OccurrenceDate" : "2023-10-14T20:59:59.082Z",
    "GroupID" : "07abdf46-2d63-4522-b949-20f0ea9a07b0",
    "RegisterRelatedIDs": {
        "Customer" : "4625a8de-7e40-4d6f-ba23-3deb4a79dab5",
        "Product": "0eb9bed8-2c0e-458b-8c3e-d0b3e3fadc92"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM4"
    },
    "Observations": [
        "Erro ao autorizar nota fiscal no sefaz",
        "Estado origem: RS - Estado destino: SP - CFOP: 5102"
    ],
    "StatusResult": 400,
    "MessageResult": "Erro Cod 3453 - CFOP inválida para este tipo de operação",
    "Files": {} //anexar XML gerado
}
```


Fonte: O Autor (2023)

A.1.5 Cenário 5

Figura 28 – Etapa 1: Validação do Pedido


```
{
    "Classification": "Notification",
    "Type": "ValidatingOrder",
    "RegisterID": "016201fa-bd02-4328-bdea-217bc3c01a0b",
    "RegisterType": "IncomingOrder",
    "OccurrenceAction": "ImportOrdersEvent",
    "OccurrenceDate" : "2023-10-14T18:56:59.082Z",
    "GroupID" : "016201fa-bd02-4328-bdea-217bc3c01a0b",
    "RegisterRelatedIDs": {
        "Customer" : "45172dd6-c34d-4612-aa16-63f765b082c2",
        "Product": "83b578a4-7669-425d-949a-4fe5eec4166b"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM5"
    },
    "Observations": [
        "Dados validados com sucesso"
    ],
    "StatusResult": 200,
    "MessageResult": null,
    "Files": {
        "RequestResponse" : "{\"Nome\":\"FranciscoeNinaContábilME\",\"CNPJ\":\"17.
            059.556/0001-56\",\"IE\":\"580.485.047.261\",
            \"Endereço\":\"RuaPauloRodriguesPrado\",\"Numero\":\"\",
            \"Bairro\":\"JardimMendonçaI\",\"Cidade\":\"Bauru\",\"UF\":\"SP\",
            \"Produtos\":[\"SKU\":\"PROD789\",\"Valor\":325.00,\"Desconto\":0.00,
            \"Quantidade\":2],\"ValorParcial\":650.00,\"Frete\":50.00,
            \"DescontoTotal\":0.00,\"ValorTotal\":700.00}"
    }
}
```


Fonte: O Autor (2023)

Figura 29 – Etapa 2: Envio do Pedido


```
{
    "Classification": "Notification",
    "Type": "SendingOrder",
    "RegisterID": "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6",
    "RegisterType": "IncomingOrder",
    "OccurrenceAction": "ImportOrdersEvent",
    "OccurrenceDate" : "2023-10-14T18:59:59.082Z",
    "GroupID" : "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6",
    "RegisterRelatedIDs": {
        "Customer" : "c0e6fb14-9a43-4b8f-abe0-def92d5fb5be",
        "Product": "78b5b43e-8fca-42a6-bd77-e5f26d400f81"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM5"
    },
    "Observations": [
        "Pedido enviado com sucesso"
    ],
    "StatusResult": null,
    "MessageResult": null,
    "Files": { }
}
```


Fonte: O Autor (2023)

Figura 30 – Etapa 3: Recebimento do Pedido


```
{
    "Classification": "Notification",
    "Type": "ReceivingOrder",
    "RegisterID": "020ae440-bb57-4e8c-9b58-cacf45f985ec",
    "RegisterType": "Order",
    "OccurrenceAction": "ReceiveOrdersEvent",
    "OccurrenceDate" : "2023-10-14T19:56:59.082Z",
    "GroupID" : "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6",
    "RegisterRelatedIDs": {
        "Customer" : "c0e6fb14-9a43-4b8f-abe0-def92d5fb5be",
        "Product": "78b5b43e-8fca-42a6-bd77-e5f26d400f81"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM5"
    },
    "Observations": [
        "Pedido recebido com sucesso"
    ],
    "StatusResult": null,
    "MessageResult": null,
    "Files": { }
}
```


Fonte: O Autor (2023)

Figura 31 – Etapa 4: Faturamento do Pedido


```
{
    "Classification": "Warning",
    "Type": "InvoicingOrder",
    "RegisterID": "f8d9b757-f1cc-417d-9330-1e59c1778e6a",
    "RegisterType": "Invoice",
    "OccurrenceAction": "InvoiceOrdersEvent",
    "OccurrenceDate" : "2023-10-14T20:56:59.082Z",
    "GroupID" : "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6",
    "RegisterRelatedIDs": {
        "Customer" : "c0e6fb14-9a43-4b8f-abe0-def92d5fb5be",
        "Product": "78b5b43e-8fca-42a6-bd77-e5f26d400f81"
    },
    "SpecificInformation":{
        "PurchaseOrder" : "ORDEM5"
    },
    "Observations": [
        "Houve uma falha ao conectar com o SEFAZ, uma nova tentativa será feita em 30 minutos."
    ],
    "StatusResult": 503,
    "MessageResult": null,
    "Files": { }
}
```


Fonte: O Autor (2023)

A.2 REQUISIÇÕES E RETORNOS DA CONSULTA DE *LOGS*

A.2.1 Cenário 1

Figura 32 – Requisição: Filtro por classificação e intervalo de datas


```
{
    "Classifications": [
        "Error",
        "Warning"
    ],
    "OccurrenceDateStart" : "2023-10-14T00:00:00.082Z",
    "OccurrenceDateEnd" : "2023-10-30T23:59:59.083Z"
}
```


Fonte: O Autor (2023)

Figura 33 – Retorno 1/4: Filtro por classificação e intervalo de datas


```
[
    {
        "classification": "Error",
        "type": "ValidatingOrder",
        "registerID": "31710a73-35a2-4993-804a-b5ef06eff264",
        "registerType": "IncomingOrder",
        "occurrenceDate": "2023-10-14T18:56:59.082Z",
        "occurrenceAction": "ImportOrdersEvent",
        "groupID": "f1b8f1cf-81e2-4edb-9370-53cfade81696",
        "registerRelatedIDs": {
            "Customer": "e17ee2ed-2ee1-490a-b1b5-639a0dadbe67",
            "Product": "77a3cb8d-1ed7-4149-9912-dbc6ebd636ee"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM2"
        },
        "observations": [
            "Falha na validação dos dados",
            "CNPJ inválido: 21.811.180/0001-62",
            "IE inválida: 980.790.057.737"
        ],
        "statusResult": "OK",
        "messageResult": null,
        "files": {
            "RequestResponse": "https:/[redacted]
                [redacted]/RequestResponse-a86f7d3d-8d41-4563-994e-209f09edb0b1.txt"
        }
    },
```


Fonte: O Autor (2023)

Figura 34 – Retorno 2/4: Filtro por classificação e intervalo de datas


```
{
    "classification": "Error",
    "type": "ValidatingOrder",
    "registerID": "4c171f09-8ca5-4d9a-a649-bba5d46fcb7f",
    "registerType": "IncomingOrder",
    "occurrenceDate": "2023-10-14T18:56:59.082Z",
    "occurrenceAction": "ImportOrdersEvent",
    "groupID": "4c171f09-8ca5-4d9a-a649-bba5d46fcb7f",
    "registerRelatedIDs": {
        "Customer": "031a6da9-9786-4d1c-a953-c176c44a233c",
        "Product": "4e04bf27-43b1-4622-a7e9-6c23df1eecae"
    },
    "specificInformations": {
        "PurchaseOrder": "ORDEM3"
    },
    "observations": [
        "Erro não catalogado ao validar dados da ordem"
    ],
    "statusResult": "OK",
    "messageResult": null,
    "files": {
        "RequestResponse": "https:/[redacted]
            [redacted]/RequestResponse-1fb31e11-17fb-4a2c-a210-cf94ada88d15.txt"
    }
},
```


Fonte: O Autor (2023)

Figura 35 – Retorno 3/4: Filtro por classificação e intervalo de datas


```
{
    "classification": "Warning",
    "type": "InvoicingOrder",
    "registerID": "f8d9b757-f1cc-417d-9330-1e59c1778e6a",
    "registerType": "Invoice",
    "occurrenceDate": "2023-10-14T20:56:59.082Z",
    "occurrenceAction": "InvoiceOrdersEvent",
    "groupID": "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6",
    "registerRelatedIDs": {
        "Customer": "c0e6fb14-9a43-4b8f-abe0-def92d5fb5be",
        "Product": "78b5b43e-8fca-42a6-bd77-e5f26d400f81"
    },
    "specificInformations": {
        "PurchaseOrder": "ORDEM5"
    },
    "observations": [
        "Houve uma falha ao conectar com o SEFAZ, uma nova tentativa será feita
            em 30 minutos."
    ],
    "statusResult": "ServiceUnavailable",
    "messageResult": null,
    "files": {}
},
```


Fonte: O Autor (2023)

Figura 36 – Retorno 4/4: Filtro por classificação e intervalo de datas


```
        "classification": "Error",
        "type": "InvoicingOrder",
        "registerID": "f211e7e7-13ab-4f25-893c-0a5f33e59b48",
        "registerType": "Invoice",
        "occurrenceDate": "2023-10-14T20:59:59.082Z",
        "occurrenceAction": "InvoiceOrdersEvent",
        "groupID": "07abdf46-2d63-4522-b949-20f0ea9a07b0",
        "registerRelatedIDs": {
            "Customer": "4625a8de-7e40-4d6f-ba23-3deb4a79dab5",
            "Product": "0eb9bed8-2c0e-458b-8c3e-d0b3e3fadc92"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM4"
        },
        "observations": [
            "Erro ao autorizar nota fiscal no sefaz",
            "Estado origem: RS - Estado destino: SP - CFOP: 5102"
        ],
        "statusResult": "BadRequest",
        "messageResult": "Erro Cod 3453 - CFOP inválida para este tipo de operação",
        "files": {}
    }
]
```


Fonte: O Autor (2023)

A.2.2 Cenário 2

Figura 37 – Requisição: Filtro por classificação e tipo


```
{
    "Classifications": [
        "Error"
    ],
    "Types": [
        "ValidatingOrder"
    ]
}
```


Fonte: O Autor (2023)

Figura 38 – Retorno 1/2: Filtro por classificação e tipo


```
[
    {
        "classification": "Error",
        "type": "ValidatingOrder",
        "registerID": "31710a73-35a2-4993-804a-b5ef06eff264",
        "registerType": "IncomingOrder",
        "occurrenceDate": "2023-10-14T18:56:59.082Z",
        "occurrenceAction": "ImportOrdersEvent",
        "groupID": "f1b8f1cf-81e2-4edb-9370-53cfade81696",
        "registerRelatedIDs": {
            "Customer": "e17ee2ed-2ee1-490a-b1b5-639a0dadbe67",
            "Product": "77a3cb8d-1ed7-4149-9912-dbc6ebd636ee"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM2"
        },
        "observations": [
            "Falha na validação dos dados",
            "CNPJ inválido: 21.811.180/0001-62",
            "IE inválida: 980.790.057.737"
        ],
        "statusResult": "OK",
        "messageResult": null,
        "files": {
            "RequestResponse": "https://[redacted]
                [redacted]/RequestResponse-a86f7d3d-8d41-4563-994e-209f09edb0b1.txt"
        }
    },
```


Fonte: O Autor (2023)

Figura 39 – Retorno 2/2: Filtro por classificação e tipo


```
    {
        "classification": "Error",
        "type": "ValidatingOrder",
        "registerID": "4c171f09-8ca5-4d9a-a649-bba5d46fcb7f",
        "registerType": "IncomingOrder",
        "occurrenceDate": "2023-10-14T18:56:59.082Z",
        "occurrenceAction": "ImportOrdersEvent",
        "groupID": "4c171f09-8ca5-4d9a-a649-bba5d46fcb7f",
        "registerRelatedIDs": {
            "Customer": "031a6da9-9786-4d1c-a953-c176c44a233c",
            "Product": "4e04bf27-43b1-4622-a7e9-6c23df1eecae"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM3"
        },
        "observations": [
            "Erro não catalogado ao validar dados da ordem"
        ],
        "statusResult": "OK",
        "messageResult": null,
        "files": {
            "RequestResponse": "https:/[redacted]
                [redacted]/RequestResponse-1fb31e11-17fb-4a2c-a210-cf94ada88d15.txt"
        }
    }
]
```


Fonte: O Autor (2023)

A.2.3 Cenário 3

Figura 40 – Requisição: Filtro por classificação e ação


```
{
    "Classifications": [
        "Error"
    ],
    "OccurrenceActions": [
        "ImportOrdersEvent"
    ]
}
```


Fonte: O Autor (2023)

Figura 41 – Retorno 1/2: Filtro por classificação e ação


```
[
    {
        "classification": "Error",
        "type": "ValidatingOrder",
        "registerID": "31710a73-35a2-4993-804a-b5ef06eff264",
        "registerType": "IncomingOrder",
        "occurrenceDate": "2023-10-14T18:56:59.082Z",
        "occurrenceAction": "ImportOrdersEvent",
        "groupID": "f1b8f1cf-81e2-4edb-9370-53cfade81696",
        "registerRelatedIDs": {
            "Customer": "e17ee2ed-2ee1-490a-b1b5-639a0dadbe67",
            "Product": "77a3cb8d-1ed7-4149-9912-dbc6ebd636ee"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM2"
        },
        "observations": [
            "Falha na validação dos dados",
            "CNPJ inválido: 21.811.180/0001-62",
            "IE inválida: 980.790.057.737"
        ],
        "statusResult": "OK",
        "messageResult": null,
        "files": {
            "RequestResponse": "https://[redacted]
                [redacted]/RequestResponse-a86f7d3d-8d41-4563-994e-209f09edb0b1.txt"
        }
    },
```


Fonte: O Autor (2023)

Figura 42 – Retorno 2/2: Filtro por classificação e ação


```
{
    "classification": "Error",
    "type": "ValidatingOrder",
    "registerID": "4c171f09-8ca5-4d9a-a649-bba5d46fcb7f",
    "registerType": "IncomingOrder",
    "occurrenceDate": "2023-10-14T18:56:59.082Z",
    "occurrenceAction": "ImportOrdersEvent",
    "groupID": "4c171f09-8ca5-4d9a-a649-bba5d46fcb7f",
    "registerRelatedIDs": {
        "Customer": "031a6da9-9786-4d1c-a953-c176c44a233c",
        "Product": "4e04bf27-43b1-4622-a7e9-6c23df1eecae"
    },
    "specificInformations": {
        "PurchaseOrder": "ORDEM3"
    },
    "observations": [
        "Erro não catalogado ao validar dados da ordem"
    ],
    "statusResult": "OK",
    "messageResult": null,
    "files": {
        "RequestResponse": "https:/[redacted]/RequestResponse-1fb31e11-17fb-4a2c-a210-cf94ada88d15.txt"
    }
}
]
```


Fonte: O Autor (2023)

A.2.4 Cenário 4

Figura 43 – Requisição: Filtro por identificador e tipo de registro


```
{
    "RegisterID": "f0d7d38b-428e-4df4-9f68-8e78c2e764a0",
    "RegisterType": "IncomingOrder"
}
```


Fonte: O Autor (2023)

Figura 44 – Retorno 1/2: Filtro por identificador e tipo de registro


```
[
    {
        "classification": "Notification",
        "type": "ValidatingOrder",
        "registerID": "f0d7d38b-428e-4df4-9f68-8e78c2e764a0",
        "registerType": "IncomingOrder",
        "occurrenceDate": "2023-10-14T18:56:59.082Z",
        "occurrenceAction": "ImportOrdersEvent",
        "groupID": "f0d7d38b-428e-4df4-9f68-8e78c2e764a0",
        "registerRelatedIDs": {
            "Customer": "d09cb9ca-8188-4139-baf0-48c2a1d1745b",
            "Product": "183d99f2-1ec3-4472-95bc-c9b9f4362bf4"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM1"
        },
        "observations": [
            "Dados validados com sucesso"
        ],
        "statusResult": "OK",
        "messageResult": null,
        "files": {
            "RequestResponse": "https:/[redacted]
                [redacted]/RequestResponse-48adf9d5-466d-448c-8d66-0f374a1dfd67.txt"
        }
    },
```


Fonte: O Autor (2023)

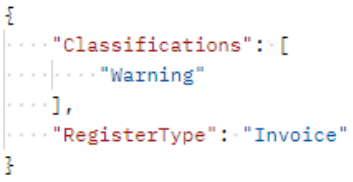
Figura 45 – Retorno 2/2: Filtro por identificador e tipo de registro


```
    {
        "classification": "Notification",
        "type": "SendingOrder",
        "registerID": "f0d7d38b-428e-4df4-9f68-8e78c2e764a0",
        "registerType": "IncomingOrder",
        "occurrenceDate": "2023-10-15T18:58:59.082Z",
        "occurrenceAction": "ImportOrdersEvent",
        "groupID": "f0d7d38b-428e-4df4-9f68-8e78c2e764a0",
        "registerRelatedIDs": {
            "Customer": "d09cb9ca-8188-4139-baf0-48c2a1d1745b",
            "Product": "183d99f2-1ec3-4472-95bc-c9b9f4362bf4"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM1"
        },
        "observations": [
            "Pedido enviado com sucesso"
        ],
        "statusResult": 0,
        "messageResult": null,
        "files": {}
    }
]
```


Fonte: O Autor (2023)

A.2.5 Cenário 5

Figura 46 – Requisição: Filtro por identificador e tipo de registro



Fonte: O Autor (2023)

Figura 47 – Retorno: Filtro por classificação e tipo de registro

Fonte: O Autor (2023)

A.2.6 Cenário 6

Figura 48 – Requisição: Filtro por identificador do grupo


```
{
    "GroupID" : "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6"
}
```


Fonte: O Autor (2023)

Figura 49 – Retorno 1/4: Filtro por identificador do grupo


```
[
    {
        "classification": "Notification",
        "type": "ValidatingOrder",
        "registerID": "016201fa-bd02-4328-bdea-217bc3c01a0b",
        "registerType": "IncomingOrder",
        "occurrenceDate": "2023-10-14T18:56:59.082Z",
        "occurrenceAction": "ImportOrdersEvent",
        "groupID": "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6",
        "registerRelatedIDs": {
            "Customer": "45172dd6-c34d-4612-aa16-63f765b082c2",
            "Product": "83b578a4-7669-425d-949a-4fe5eec4166b"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM5"
        },
        "observations": [
            "Dados validados com sucesso"
        ],
        "statusResult": "OK",
        "messageResult": null,
        "files": {
            "RequestResponse": "https://[redacted]/RequestResponse-82f2a60a-de22-429d-9523-aa71ee574ed6.txt"
        }
    },
```


Fonte: O Autor (2023)

Figura 50 – Retorno 2/4: Filtro por identificador do grupo


```
    {
        "classification": "Notification",
        "type": "SendingOrder",
        "registerID": "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6",
        "registerType": "IncomingOrder",
        "occurrenceDate": "2023-10-14T18:59:59.082Z",
        "occurrenceAction": "ImportOrdersEvent",
        "groupID": "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6",
        "registerRelatedIDs": {
            "Customer": "c0e6fb14-9a43-4b8f-abe0-def92d5fb5be",
            "Product": "78b5b43e-8fca-42a6-bd77-e5f26d400f81"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM5"
        },
        "observations": [
            "Pedido enviado com sucesso"
        ],
        "statusResult": 0,
        "messageResult": null,
        "files": {}
    },
```


Fonte: O Autor (2023)

Figura 51 – Retorno 3/4: Filtro por identificador do grupo


```
    {
        "classification": "Notification",
        "type": "ReceivingOrder",
        "registerID": "020ae440-bb57-4e8c-9b58-cacf45f985ec",
        "registerType": "Order",
        "occurrenceDate": "2023-10-14T19:56:59.082Z",
        "occurrenceAction": "ReceiveOrdersEvent",
        "groupID": "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6",
        "registerRelatedIDs": {
            "Customer": "c0e6fb14-9a43-4b8f-abe0-def92d5fb5be",
            "Product": "78b5b43e-8fca-42a6-bd77-e5f26d400f81"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM5"
        },
        "observations": [
            "Pedido recebido com sucesso"
        ],
        "statusResult": 0,
        "messageResult": null,
        "files": {}
    },
```


Fonte: O Autor (2023)

Figura 52 – Retorno 4/4: Filtro por identificador do grupo


```
    {
        "classification": "Warning",
        "type": "InvoicingOrder",
        "registerID": "f8d9b757-f1cc-417d-9330-1e59c1778e6a",
        "registerType": "Invoice",
        "occurrenceDate": "2023-10-14T20:56:59.082Z",
        "occurrenceAction": "InvoiceOrdersEvent",
        "groupID": "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6",
        "registerRelatedIDs": {
            "Customer": "c0e6fb14-9a43-4b8f-abe0-def92d5fb5be",
            "Product": "78b5b43e-8fca-42a6-bd77-e5f26d400f81"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM5"
        },
        "observations": [
            "Houve uma falha ao conectar com o SEFAZ, uma nova tentativa será feita
                em 30 minutos."
        ],
        "statusResult": "ServiceUnavailable",
        "messageResult": null,
        "files": {}
    }
]
```


Fonte: O Autor (2023)

A.2.7 Cenário 7

Figura 53 – Requisição: Filtro por *status* resultante


```
{
    "StatusResults": [ "ServiceUnavailable" ]
}
```


Fonte: O Autor (2023)

Figura 54 – Retorno: Filtro por *status* resultante


```
[
    {
        "classification": "Warning",
        "type": "InvoicingOrder",
        "registerID": "f8d9b757-f1cc-417d-9330-1e59c1778e6a",
        "registerType": "Invoice",
        "occurrenceDate": "2023-10-14T20:56:59.082Z",
        "occurrenceAction": "InvoiceOrdersEvent",
        "groupID": "5a7d9bfd-ca6c-454b-9d2e-ad1def4937c6",
        "registerRelatedIDs": {
            "Customer": "c0e6fb14-9a43-4b8f-abe0-def92d5fb5be",
            "Product": "78b5b43e-8fca-42a6-bd77-e5f26d400f81"
        },
        "specificInformations": {
            "PurchaseOrder": "ORDEM5"
        },
        "observations": [
            "Houve uma falha ao conectar com o SEFAZ, uma nova tentativa será feita
                em 30 minutos."
        ],
        "statusResult": "ServiceUnavailable",
        "messageResult": null,
        "files": {}
    }
]
```


Fonte: O Autor (2023)