LUIS FERNANDO SOARES SENA

DESENVOLVIMENTO DE UMA ARQUITETURA PARA A OBSERVABILIDADE DE LOGS EM UM SISTEMA ON-PREMISSES

São Paulo
2024

LUIS FERNANDO SOARES SENA

DESENVOLVIMENTO DE UMA ARQUITETURA PARA A OBSERVABILIDADE DE LOGS EM UM SISTEMA ON-PREMISSES

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Especialista em Engenharia de Software.

Área de Concentração:

Engenharia de Software

Orientador:

Prof. Dr. Jorge Luís Risco Becerra

Co-orientador:

Prof. Alípio Ferro

São Paulo
2024

Dedico este trabalho a todos e todas que, de alguma forma, cruzaram meu caminho e contribuíram para o meu aprendizado e crescimento.

AGRADECIMENTOS

Agradeço aos meus pais e meus avós que sempre me incentivaram pela busca do conhecimento.

Ao meu namorado, José Roberto, pela parceria e motivação para continuar meus estudos.

Ao Prof. Dr. Jorge Luís Risco Becerra e ao Prof. Alípio Ferro pela orientação e suporte para a elaboração deste trabalho.

À Universidade de São Paulo — USP, à Escola Politécnica da Universidade de São Paulo — EPUSP e ao PECE — Programa de Educação Continuada em Engenharia pela oportunidade e pelo apoio durante meus estudos.

RESUMO

SENA, Luis Fernando S. **Transformação arquitetural para a observabilidade de logs de um sistema on-premisses**. 2024. 44 páginas. Monografia (MBA em Engenharia de Software). Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo. São Paulo. 2024.

A transformação arquitetural para observabilidade de logs em sistemas on-premises aborda a adaptação da arquitetura de um sistema ERP integrado a um middleware orientado a mensagens, visando aprimorar o monitoramento e a análise de logs em tempo real. Utilizando a metodologia TOGAF e seu método de desenvolvimento de arquiteturas (ADM), foram concebidos artefatos representativos das fases A a D, como diagramas de componentes, processos de negócio e interações tecnológicas. O trabalho também integra um módulo de Machine Learning (LogAnomaly), conectado ao sistema Splunk para análise de logs e detecção de anomalias, com suporte a retreinamento automático do modelo. A arquitetura resultante é escalável e adaptável, alinhando objetivos técnicos e de negócios, contribuindo para maior eficiência na gestão de sistemas complexos.

Palavras-Chave: Observabilidade, TOGAF, Machine Learning, Middleware, Logs, Anomalias.

ABSTRACT

SENA, Luis Fernando S. **Transformação arquitetural para a observabilidade de logs de um sistema on-premise**. 2024. 44 páginas. Monografia (MBA em Engenharia de Software). Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo. São Paulo. 2024.

The architectural transformation for log observability in on-premises systems focuses on adapting the architecture of an ERP system integrated with a message-oriented middleware to enhance real-time log monitoring and analysis. Using the TOGAF methodology and its Architecture Development Method (ADM), representative artifacts from phases A to D were developed, including component diagrams, business process models, and technological interaction diagrams. The work also integrates a Machine Learning module (LogAnomaly), connected to the Splunk system for log analysis and anomaly detection, with support for automatic model retraining. The resulting architecture is scalable and adaptable, aligning technical and business objectives, contributing to greater efficiency in managing complex systems.

Keywords: Observability, TOGAF, Machine Learning, Middleware, Logs, Anomalies.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Com o avanço das tecnologias se dando a passos largos, cada vez mais os sistemas de informação assumem papel importante nas atividades humanas, desde como nos comunicamos entre nós ou como fazemos negócios, até mesmo substituindo o ser humano em determinados trabalhos. Tal uso intensivo de sistemas acarreta geração de montantes crescentes de dados, na grandeza de petabytes inclusive, dados que não somente precisam ser armazenados mas também tratados em seus diversos casos de uso.

Dos menores aos maiores sistemas em funcionamento atualmente pode-se destacar uma característica em comum entre eles, são geralmente projetados para produzir, coletar e processar registros de atividades realizadas no sistema, ou seja, realizar a gravação de logs. O gerenciamento eficiente dos logs de um sistema é essencial para garantir que esse esteja operando da melhor forma para qual foi projetado e com isso evitar ao máximo períodos de downtime.

Em uma era onde cada vez mais dependemos dos sistemas de informação, a importância de garantir a continuidade e segurança das operações é imperativo, refletindo também nos esforços de pesquisas nos últimos anos em busca do aprimoramento do monitoramento de logs. Onde vemos o emprego de tecnologias como o aprendizado de máquina e mais recentemente o chamado *Deep Learning* (CHEN et al., 2022).

O registro de logs por um sistema é dos três pilares que compõe o conceito de observabilidade, sendo os outros dois pilares: métricas e rastreabilidade. Teoricamente, a combinação desses três pilares deve prover a capacidade para um observador externo ao sistema de observar o que está acontecendo internamente.

Contudo, esses três elementos são apenas o fundamento da observabilidade, assim sendo, vai muito além de simplesmente registrar logs ou estabelecer algumas métricas, mas sim criar as condições para ser possível a identificação de qualquer estado no qual o sistema possa estar, independentemente do quão novo ele seja (CHARITY MAJORS; MIRANDA, 2022).

1.1 Motivação

Em uma era onde cada vez mais dependemos dos sistemas de informação, a importância de garantir a continuidade e segurança das operações é imperativo, refletindo também nos esforços de pesquisas nos últimos anos em busca do aprimoramento do monitoramento de logs, sendo que em diversas ocasiões a busca pelo grau de observabilidade dos sistemas o elemento norteador. Entretanto, observa-se uma atenção especial a aplicações com arquitetura web distribuídas, embora ainda seja utilizado sistemas *on-premisses* na indústria e por isso me motivei a aplicar os conceitos de observabilidade de software de sistemas distribuídos através da adaptação da arquitetura de um sistema *on-premisses*.

1.2 Objetivo

O objetivo desse trabalho de monografia é realizar a adaptação da arquitetura de um sistema on-premisses para utilização de um módulo de observabilidade baseado em logs que utiliza *Machine Learning*. Para conduzir essa adaptação arquitetural será aplicado o processo ADM (*Architecture Development Method*) do modelo TOGAF (*The Open Group Architechture Framework*).

1.3 Justificativas

Como descrito por Charity Majors e Miranda (2022) Uma vez que as técnicas tradicionais têm caráter reativo e são dependentes de engenheiros que acessam manualmente os logs do sistema e conduzem uma avaliação empírica sobre a falha ou anomalia, técnica propensa a ocorrência de erro. Os sistemas mais modernos exigem que novas metodologias sejam empregadas não só na criação de métricas, mas também no monitoramento dos ambientes desses sistemas.

Há pelo menos vinte anos, observa-se que um domínio de técnicas convencionais que regem a relação entre *hardware* e as pessoas que o operam, a esse conjunto de ferramentas convencionou-se chamar de “monitoramento de sistema”. Por muito tempo essa ideia

de monitoramento imperou entre os desenvolvedores, como sendo a melhor maneira de interpretar o que ocorre no espaço virtual entre o código escrito por eles e o mundo físico, mesmo sabendo que tal abordagem tem suas limitações que acarretam aumento da dificuldade de *troubleshooting* do sistema.

Com o aumento da capacidade de desenvolvimento, os sistemas modernos aumentaram não apenas em tamanho, mas também em complexidade, fazendo com que a tarefa de prever, detectar e tratar anomalias em sistemas se torne muito mais difícil. Aqui que a observabilidade se destaca, uma vez que ela faz com que os dados de telemetria do sistema possam ser trabalhados de forma flexível, permitindo aos times realizar análises e chegar mais rapidamente nas causas raízes dos problemas que ocorrem de maneira singular (CHARITY MAJORS; MIRANDA, 2022).

Como mencionado, a observabilidade de um sistema exige a compreensão de diferentes cenários e por vezes cenários inéditos. Portanto, com base na literatura disponível supracitada, justifica-se transformar a arquitetura de um sistema para buscando a observabilidade em detrimento dos modelos tradicionais de monitoramento. Constatada sua superioridade quando comparando a capacidade de adaptação a cenários de alta complexidade dos sistemas modernos.

1.4 Método de Pesquisa

As possibilidades de tipificação de uma pesquisa são diversas e derivam de aspectos desde sua natureza até os procedimentos técnicos empregados em seu desenvolvimento. Uma pesquisa pode ser entendida como um agrupamento de técnicas, embasado em raciocínio lógico e aplica metodologia científica para produzir conhecimento científico. Os tipos de pesquisa variam conforme o problema que é objeto de estudo, contudo alguns modelos se destacam, como a pesquisa bibliográfica, experimental, documental, qualitativa ou quantitativa, cada uma com metodologias distintas (DE LUNETTA E RODRIGUES GUERRA, 2023).

Para um maior entendimento e esclarecimento sobre o problema objeto de estudo dessa monografia, foi conduzida uma pesquisa bibliográfica. Para a realização do levanta-

mento bibliográfico foram utilizadas as bases de dados “Google Scholar” e também “IEEE Xplore”. Para refinar os resultados e direcionando-os ao objetivo desse trabalho foram aplicados filtros de busca utilizando palavras-chave (como *Log Management*, *Deep Learning* e *Software Observability*) e os trabalhos publicados mais recentemente, preferindo-se artigos com data de publicação nos últimos 3 anos.

1.5 Estrutura do Trabalho

O Capítulo 1 INTRODUÇÃO apresenta as motivações, o objetivo, as justificativas, método de pesquisa e a estrutura do trabalho.

O Capítulo 2 REVISÃO BIBLIOGRÁFICA discorre sobre os temas principais para o desenvolvimento deste trabalho, como:

O Capítulo 3 DESENVOLVIMENTO DA PESQUISA apresenta uma proposta de adaptação da arquitetura de um sistema de detecção de anomalias em logs de um sistema em nuvem para implantação em contexto real de uma indústria, considerando um sistema ERP on-premises com foco em observabilidade.

O Capítulo 4 ANÁLISE DE RESULTADOS apresenta uma análise crítica do capítulo anterior.

O Capítulo 5 CONSIDERAÇÕES FINAIS descreve as conclusões e as contribuições do trabalho e sugestões de trabalhos futuros que poderão ser desenvolvidos a partir deste trabalho.

2 FUNDAMENTOS TEÓRICOS

Neste capítulo são apresentados os conceitos necessários para a adaptação de uma arquitetura de monitoramento de anomalias em logs de sistema utilizando técnicas de gerenciamento de dados de log e deep learning, através de fundamentações teóricas necessárias e relevantes considerando o objetivo desse trabalho.

2.1 Considerações Inicias

O capítulo aborda a utilização de técnicas de aprendizado de máquina no monitoramento de anomalias em logs de sistema na atualidade e traz a descrição das técnicas mais praticadas, avaliando como essas técnicas podem contribuir para aumentar a eficiência do monitoramento, bem como os desafios que podem acarretar.

A exposição dos conceitos será realizada em quatro dimensões, sendo elas: gerenciamento de logs, deep machine learning, redes neurais e detecção de anomalias em ambientes de produção de software.

2.2 Gerenciamento de logs

Antes de adentrar o conceito de gerenciamento é importante definir o que é um log de sistema. O instituto estadunidense NIST (National Institute of Standards and Technology) define como log o registro da ocorrência de eventos nos sistemas e redes de uma organização. São compostos por uma lista entradas, cada uma contendo informações referente a um evento específico que ocorreu no sistema ou rede em questão. Logs estão geralmente relacionados a registros de segurança informática e podem advir de diversas origens, incluindo softwares de segurança, como antivírus, firewalls e sistemas de detecção e prevenção de invasão, bem como sistemas operacionais em servidores, equipamentos de rede e aplicações em geral. (KENT; SOUPPAYA, 2006)

Em suma, os arquivos de logs gerados por um sistema são registros de quais eventos ocor-

reram e quando se deu a ocorrência. Atualmente existem inúmeros tipos de log que servem a diferentes propósitos, portanto não há uma definição comum do que devem conter. Esses arquivos podem conter informações variadas, podendo ser detalhes operacionais, registros de autorizações e mensagens de depuração. As informações contidas em um log são frequentemente definidas pelo sistema responsável pela criação do arquivo e também pelo programador que projetou o sistema de registro dos logs, onde são registrados mensagens e eventos gerados em um programa, podendo conter informações referentes a eventos, erros e alerta de uma aplicação. (HARJUNPÄÄ; SIEKKINEN, 2023)

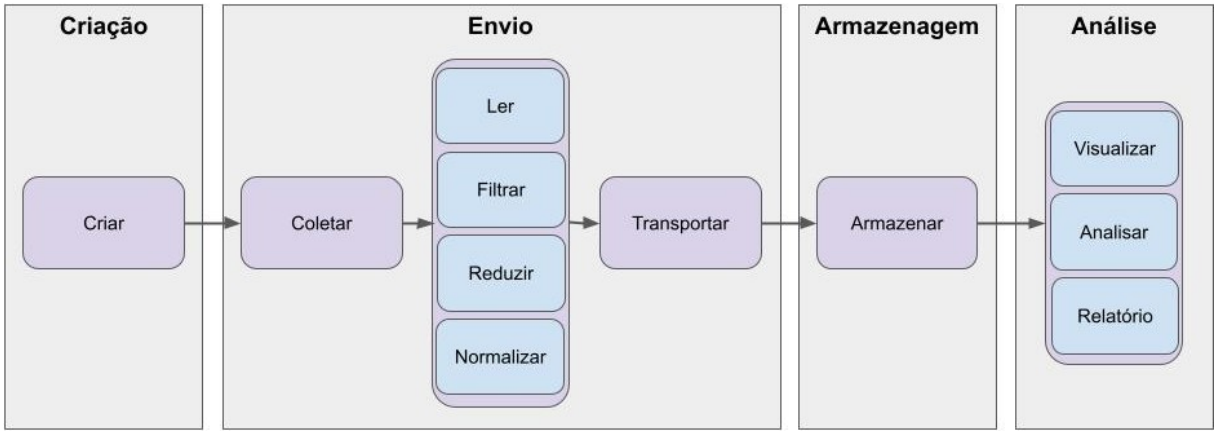
O gerenciamento de logs consiste no estabelecimento de uma metodologia e processos que serão utilizados através do seu ciclo de vida: coleta, processamento, armazenagem e por fim a análise dos arquivos gerados contendo os logs criados pelos diversos sistemas e aplicações. A importância dos sistemas de gerenciamento para as organizações reside não somente em garantir que os sistemas estão trabalhando sem falhas, mas também auxiliando no tratamento de exceções, garantia da segurança da informação ou auditorias.

A arquitetura de um sistema de gerenciamento de logs pode ser concebida como vários elementos que interagem entre si para: criar, coletar, armazenar e analisar os registros nos arquivos de log. Sendo que todos esses elementos são imprescindíveis na arquitetura desse sistema. (HARJUNPÄÄ; SIEKKINEN, 2023)

Comumente, sistemas dessa natureza podem ser separados em quatro componentes distintos:

- **Criação:** É o processo de gerar os arquivos de log. As principais questões que devem ser respondidas nessa etapa estão relacionadas ao local onde fazer o log, o que registrar e como fazer.
- **Envio:** O envio dos logs, é uma atividade executada por componentes de software responsáveis por coletar os registros das diversas origens e enviá-los para o local de armazenamento. Aqui podem ser aplicados processos de seleção e transformação, dessa forma é possível estabelecer um formato preferido e realizar conversões sempre que necessário antes de enviar os arquivos.
- **Armazenamento:** A armazenagem dos logs é um processo que ocorre após a etapa

Figura 1: Exemplo de alto nível do fluxo de um sistema de gerenciamento de logs



Fonte: Adaptado de Harjunpää e Siekkinen (2023).

de envio, nessa etapa os arquivos são enviados para um componente de software cuja responsabilidade é centralizar todas as informações, de forma que possam ser recuperadas a qualquer momento no futuro. Podendo ser um banco de dados relacional, NoSQL ou qualquer outro sistema de armazenamento de dados.

- **Análise:** É o processo de visualização e análise de dados com o intuito de gerar informação de valor. Compreende a identificação de padrões, análise estatística e interconexão entre os dados. O processo consiste essencialmente na interpretação dos dados e extrair valor deles, podendo lançar mão de técnicas avançadas de análise como modelos de aprendizado de máquina.

O processo de gerenciamento de logs ocorre como demonstrado na figura 1.

Como descrito por Harjunpää e Siekkinen (2023), os processos de coleta, transformação e armazenagem num sistema dessa natureza é muito semelhante a um fluxo ETL (*Extract, Transform and Load*), observando-se eventuais etapas que podem ser adicionadas.

2.3 Transformação de Arquitetura

Desfray e Raymond (2014) definem que o processo de transformação de uma arquitetura pode ser descrito como a evolução de um modelo arquitetônico base, para outra modelagem alvo, como o exemplificado na figura 2. A parte mais importante desse processo é definição de qual abordagem será adotada para conduzir essa evolução de arquitetura.

Figura 2: Transformação, da arquitetura base em direção à arquitetura alvo.

Fonte: Adaptado de Desfray e Raymond (2014).

De acordo com Desfray e Raymond (2014), é possível sumarizar o processo de escolha de um *framework* a ser adotado na transformação de uma arquitetura nos quatro pontos a seguir:

- **Conhecimento do ponto de partida:** Nem sempre o conhecimento da arquitetura base está estabelecido, dessa forma na maioria das vezes é necessário empregar esforços em uma reavaliação da arquitetura existente. A importância dessa tarefa reside no fato do *roadmap* da transformação ser definido pelo *gap* entre a arquitetura no ponto de partida e a modelagem final almejada.
- **Determinar um ponto de chegada:** Representa a modelagem arquitetural que se deseja produzir com a transformação. Os objetivos de negócio são o principal fator a se considerar na hora de definir o alvo, contudo também deve-se examinar fatores de ordem técnica, organizacionais e financeiros.
- **Definir o melhor caminho até o objetivo:** Escolher as soluções que serão utilizadas para conduzir a transformação, bem como a definição de um cronograma. Também é muito importante entender como será garantida a continuidade do negócio durante e após a transformação.
- **Concluir a transformação com êxito:** Executar a transformação de uma arquitetura é um processo complexo e delicado e para ser executado com sucesso é mandatório o entendimento absoluto de todas as restrições que se aplicam a essa operação.

Adicionalmente, o escopo coberto na transformação também exerce influência nos cenários que serão encontrados durante o processo. Uma vez que, geralmente, não se trara de uma

reconstrução total do sistema a partir do zero, mas apenas de uma parte específica, ligada a um objetivo de negócio. Ou seja, não há um padrão para o caminho a ser seguido. Embora adaptações ao contexto específico de cada processo de transformação seja necessário, a adoção de um *framework* pode funcionar como o fio condutor da mudança e também ajudar a conferir velocidade (DESFRAY; RAYMOND, 2014).

2.4 Arquitetura middleware

Na engenharia de *software* existe uma questão denominada de *Rendez-vous problem*, que resumidamente descreve a situação onde dois sistemas distintos precisam trocar mensagens entre si e, portanto, precisam estar sincronizados. Atualmente, as possibilidades de conexão entre dois sistemas são diversas, uma delas é o modelo Cliente/Servidor, nessa modalidade de comunicação o *Rendez-vous problem* é resolvido através da determinação de uma dinâmica entre os dois sistemas envolvidos. Necessariamente, nesse modelo a comunicação é estabelecida somente após a inicialização do servidor, o qual aguarda indefinidamente o contato da outra parte, o cliente.

Etzkorn (2017) definiu servidor e cliente da seguinte forma:

- **Servidor:** É um programa que fica disponível na espera de chamadas e quando recebe uma comunicação do sistema cliente, responde executando algum tipo de serviço útil e envio o resultado ao sistema requisitante. AS funções do servidor também incluem: autenticação, autorização e segurança dos dados, portanto servidores necessitam geralmente de privilégios especiais no sistema.
- **Cliente:** A aplicação iniciadora da comunicação, nesse modelo a comunicação sempre será direcionada ao servidor sempre que o cliente necessitar de algum serviço fornecido pelo servidor. Diferentemente dos servidores, clientes comumente não necessitam de privilégios especiais no sistema para operar.

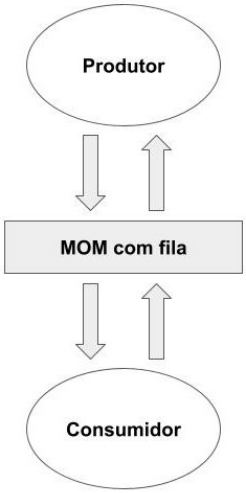
Aos sistemas que habilitam a comunicação entre aplicações distintas, convencionou-se chamar de *middleware*. Na arquitetura, o *middleware* se localiza em uma camada entre o sistema operacional e as aplicações em ambos os lados de uma rede de computadores

distribuída. Geralmente, são utilizados para operacionalizar cenários complexos e sistemas distribuídos (ETZKORN, 2017).

2.4.1 Middleware Orientado a Mensagem

O *middleware* orientado a mensagem (MOM) tem o propósito de atuar como intermediador entre o cliente e o servidor, sedo que dessa forma ambos não precisam se comunicar diretamente. Usualmente, o intermediário gerencia as mensagens enviadas entre o produtor e o consumidor em filas. E aqui tanto o cliente quanto o servidor podem desempenhar o papel de produtor e consumidor de mensagens, conforme exemplificado na figura 3.

Figura 3: Middleware orientado a mensagem com fila de mensagens.



Fonte: Adaptado de Etzkorn (2017).

Nesse paradigma de *middleware* a comunicação é assíncrona, portanto as mensagens são executadas de forma independente. Tomando como exemplo um cenário onde o cliente é o produtor da mensagem e o servidor o consumidor. Nesse caso, a mensagem é criada e enviada ao MOM, que provavelmente a colocaria na fila de mensagens que eventualmente serão recebidas pelo servidor. Esse processo de comunicação mediada por um intermediário pode ocorrer de algumas formas, destacando-se: *push model*, *pull model* e *publisher/subscriber*.

Adicionalmente, um MOM pode executar alguns tipos de processamentos no gerencia-

mento das mensagens, como, por exemplo, colocando mensagens de prioridade mais alta na frente da fila. Podendo também lidar com diversas filas com níveis de prioridade diferentes. Tais capacidades podem ajudar em ganho de desempenho do *middleware* (ETZKORN, 2017).

2.5 Observabilidade de software

Observabilidade é um conceito que atraiu muito interesse nos últimos anos na indústria de desenvolvimento de sistemas, figurando frequentemente nas listas de assuntos do momento. Mesmo que sua adoção ainda seja um desafio. Tal conceito foi proposto primeiramente por Rudolf E. Kálmán no ano de 1960, sendo entendido como a medida do quão bem os estados internos de um determinado sistema podem ser deduzidos a partir do conhecimento externo ao sistema, utilizando como base as saídas produzidas por esse sistema.

Ainda que idealizada para utilização na engenharia mecânica e engenharia de processos, a ideia de observabilidade também pode ser aplicado aos sistemas de informação modernos, contudo quando para fazer isso algumas premissas específicas da engenharia de *software* devem ser seguidas (ETZKORN, 2017).

Para uma aplicação ter observabilidade deve ser possível:

- Ter o entendimento do funcionamento interno da aplicação.
- Ter o poder de entender qualquer estado no qual o sistema eventualmente esteja.
- Constatar os pontos citados acima, utilizando exclusivamente ferramentas externas ao sistema.
- Entender esse estado independentemente de sua severidade ou frequência.

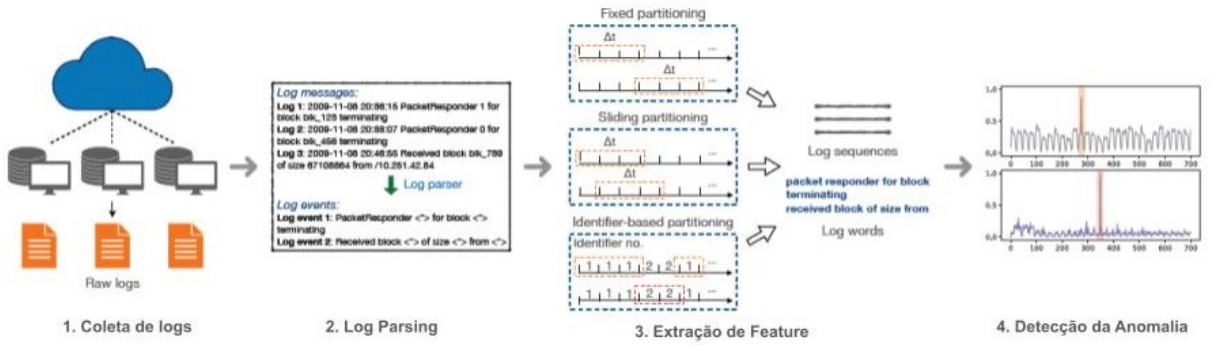
Etzkorn (2017) define observabilidade como sendo a medida na qual é possível entender e elucidar qualquer estado no qual o sistema possivelmente se encontre, sem ter em conta o quanto inédito ele seja. Quando é possível entender esses estados sem a necessidade

de escrever código adicional, considera-se que a aplicação tem observabilidade. Etzkorn (2017) também infere que a observabilidade em sistemas modernos vai além de apenas tipificação de dados ou *inputs*, bem como não é sobre equações matemáticas. Em última estância, observabilidade é sobre a relação de um sistema com as pessoas que são suas usuárias, como elas interagem com ele e o interpretam.

2.6 Detecção de Anomalia

O *framework* de detecção de anomalias em sistemas partindo da análise de logs geralmente é divido em quatro partes, sendo elas: coleta de logs, *log parsing*, extração de *feature* e por fim detecção da anomalia. conforme pode ser visto na figura 4.

Figura 4: Estrutura de detecção de anomalias baseada em log.



Fonte: Adaptado de Chen et al. (2022)

A detecção das anomalias é feita a partir das *features* extraídas na fase anterior, ela visa identificar instâncias anômalas nos logs como, por exemplo, os gerados por exceções. Métodos tradicionais baseados em *machine learning* utilizam toda a sequência de logs para realizar a previsão, utilizando para isso vetores de contagem de eventos. Já os métodos baseados em *deep learning* aprendem padrões normais e avaliam a normalidade de cada evento de log, dessa forma podendo localiza o evento exato que originou a anomalia, melhorando a interpretabilidade dos resultados (CHEN et al., 2022).

Chen et al. (2022) apresentam o método denominado *LogAnomaly*, o qual é um modelo não-supervisionado que utiliza redes neurais e que pode ser utilizados na detecção de anomalias a partir da análise de logs. Com esse método é proposto considerar também a

informação semântica dos logs, através da representação dos registros por meio de uma técnica chamada *template2vec*, onde as palavras dos *templates* de logs são representadas distribuidamente, considerando os sinônimos e antônimos neles contidos. O vetor modelo é então calculado como o peso médio dos vetores das palavras encontradas no *template*. Aqui é adotada uma técnica baseada em *forecasting*, utilizando um modelo de *Long Short--Term Memory*.

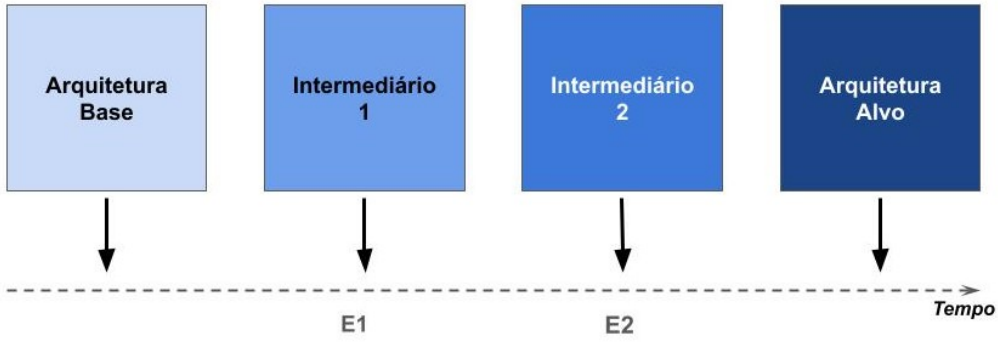
2.7 Adaptação de arquitetura com TOGAF

Como descrito por Desfray e Raymond (2014) a transição entre uma arquitetura existente e um estado futuro com incrementos, está fundamentado no TOGAF e deve fornecido no formato de um caminho a ser seguido. Para que esse caminho seja efetivo, os princípios mencionados a seguir precisam ser atendidos.

- Para ser exitoso, esse caminho precisa considerar todas as facetas da empresa, bem como os efeitos que resultarão de todas as mudanças implementadas.
- O caminho também precisa descrever os estados intermediários da arquitetura, durante o processo de transição.
- Os estados intermediários devem apresentar o valor que agregado a arquitetura.
- Uma análise das lacunas entre a arquitetura base e a arquitetura alvo é o elemento determinante para a definição do caminho a ser seguido.

O caminho em direção à arquitetura alvo, como sugerido no TOGAF (figura 5), pode compreender diversos tipos de projetos: desenvolvimento ou evolução de sistemas, migração de dados, treinamentos ou até mesmo a reorganização do negócio. Já o número de estados intermediários dependerá de outros aspectos específicos de cada transição, como, por exemplo, o domínio no qual a arquitetura está inserida, escopo das mudanças, horizonte temporal e o nível de detalhes, bem como os obstáculos que podem ser encontrados ao longo do caminho (DESFRAY; RAYMOND, 2014).

Figura 5: Caminho para transição de arquitetura



Fonte: Adaptado de Desfray e Raymond (2014)

Ainda que a transição com estados intermediários facilite na gestão da mudança, também é possível, em alguns cenários onde o escopo da mudança é limitado, uma transição direta para a arquitetura alvo. Conforme descrito por Desfray e Raymond (2014), a elaboração de um caminho para a transição da arquitetura base para chegar ao estado desejado é uma das maiores entregas do TOGAF.

2.8 Contexto da Arquitetura Base

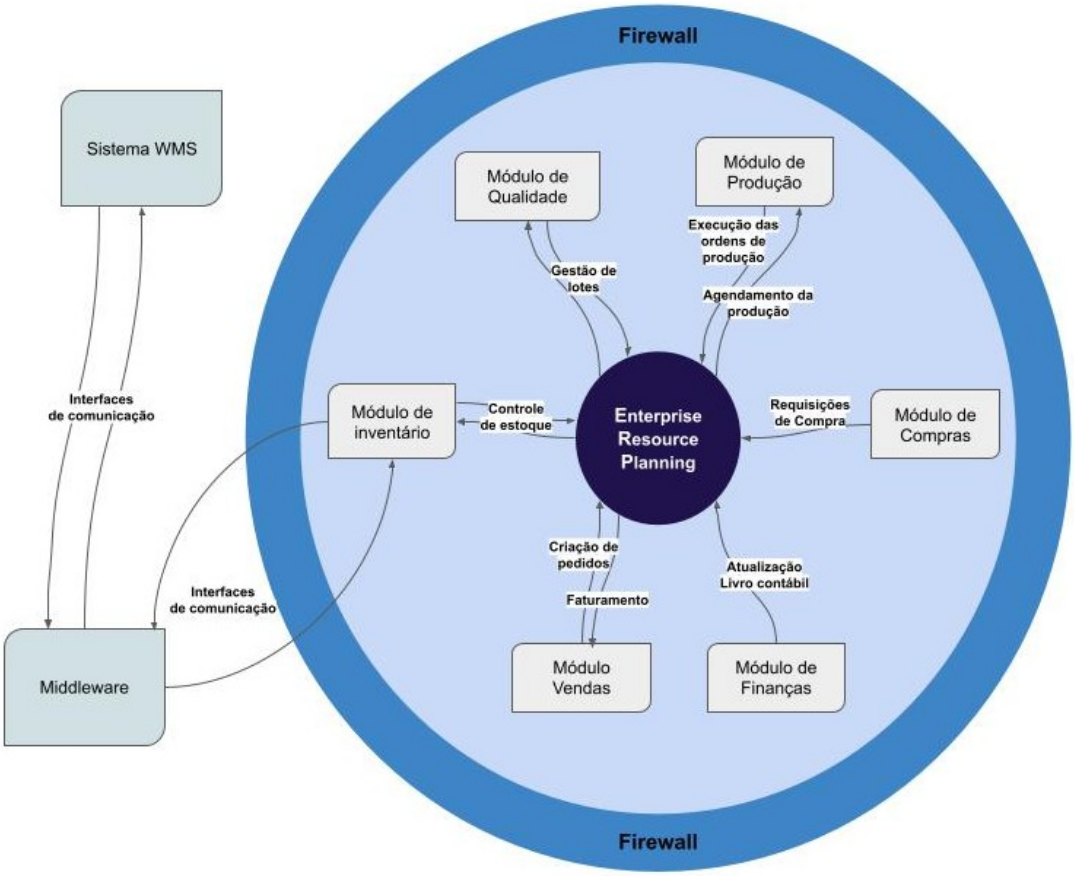
Antes do início de ciclo de alteração da arquitetura está prevista uma etapa especial chama de fase preliminar, nessa fase as atividades geralmente envolvem múltiplas áreas e está ligada aos aspectos gerais da arquitetura. Uma das atividades preliminares mais importantes é a definição do ponto de partida, ou seja, assegurar que todos tenham o correto entendimento do contexto da arquitetura base, e para isso foi elabora um diagrama de contexto (DESFRAY; RAYMOND, 2014).

O diagrama de contexto é uma das maneiras para descrever um processo em alto nível, por conta disso os diagramas geralmente não apresentam muitos componentes. Nesse diagrama estão definidos a organização, representada pelo círculo e os processos executados, bem como as interações com agentes externos.

No diagrama (figura 6) estão representados os principais elementos para a contextualização da arquitetura base. O sistema ERP (*Enterprise Resource Planning*) sendo elemento central conectando os módulos de Qualidade, Produção, Inventário, Vendas e Finanças, também é o iniciador da comunicação com o sistema WMS através de um middleware, os

dois últimos também representados no diagrama como elementos externos.

Figura 6: Diagrama de contexto arquitetura base.



Fonte: Autor.

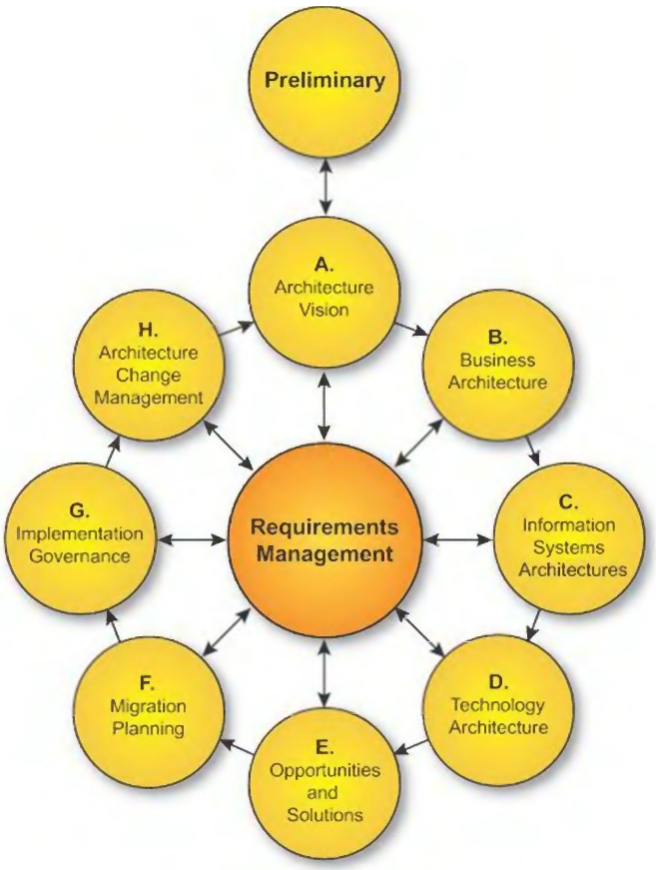
O contexto descrito no diagrama acima (figura 6) caracteriza o ponto de partida a ser considerado durante a execução do TOGAF, que será descrita nas seções seguintes deste capítulo.

2.9 Método ADM

No TOGAF a aplicação da metodologia está estruturada em um ciclo chamado *"TOGAF crop circle"*, onde estão descritas as fases do ADM (*Architecture Development Method*). O método em questão está divido em oito fases sequenciais, iniciando em uma fase preliminar e passando por um ciclo que vai de "A" e indo até a fase "H" (DESFRAY; RAYMOND, 2014).

Conforme descrito por Desfray e Raymond (2014), o framework define em detalhe cada uma das fases do ciclo em detalhes e as entregas de cada uma delas. Na etapa preliminar é onde ocorre a preparação para o trabalho que será realizado na arquitetura, nesse momento são definidos os princípios gerais da mudança, bem como os métodos e ferramentas que serão utilizados no processo e por fim o início do ciclo ADM (figura 7).

Figura 7: Diagrama do ciclo de desenvolvimento de arquitetura TOGAF



Fonte: Reproduzido de Desfray e Raymond (2014)

Com a conclusão da fase preliminar, inicia-se a fase A, a qual é a primeira fase do ciclo. Nessa fase temos dois objetivos distintos, o primeiro deles sendo o aprofundamento dos conceitos que surgiram na fase preliminar, como, por exemplo, os princípios de arquitetura e a organização do trabalho que será executado. Assim sendo que ao fim da fase A, espera-se que haja uma visão comum: dos stakeholders e seus papeis, dos objetivos e principais requisitos, escopo da mudança, o plano para execução do ciclo ADM, bem como os recursos necessários e por fim, uma visão geral da arquitetura base e da arquitetura alvo e os maiores riscos associados ao processo de transição. Em suma, ao final dessa fase deve

estar claro o estado atual, o estado final e como será o caminho a ser seguido entre esses dois estados.

As próximas três fases (B, C e D) irão concentrar a maioria dos esforços no detalhamento da arquitetura base e alvo, identificando a lacuna que existe entre elas e analisando os impactos de mudança nas diversas facetas da organização.

Partindo para a fase B, é nessa fase onde ocorre a formalização dos elementos por parte do negócio como: requisitos, processos, entidades. Em termos de arquitetura, o foco nessa fase se concentra nos seguintes elementos:

- Objetivos do negócio
- Unidades organizacionais
- Processos de negócio
- Papéis e atores do negócio
- Entidades do negócio

Já na fase C do ciclo ADM, essa fase pode ser entendida como a da arquitetura na visão de sistemas de informação, sendo a ponte entre a visão de negócio e sua tradução física. Nessa fase serão definidos os componentes de *software* que irão apoiar a execução das atividades nas funções do negócio. A fase C compreende duas partes distintas, sendo elas: arquitetura de dados e arquitetura de aplicação. Um dos resultados esperados ao final dessa fase é a alocação de cada grupo de dados a um componente da aplicação.

A fase D, *Technology Architecture* é onde acontecerá a correspondência entre tecnologia e mundo físico, baseado nos elementos desenvolvidos nas fases anteriores. Particularmente, nessa fase são definidos as plataformas e ambientes nos quais as aplicações serão executadas. O resultado esperado ao fim dessa fase é a própria arquitetura na visão de tecnologia, ou seja, um conjunto ordenado de componentes de sistemas, infraestruturas e plataformas técnicas.

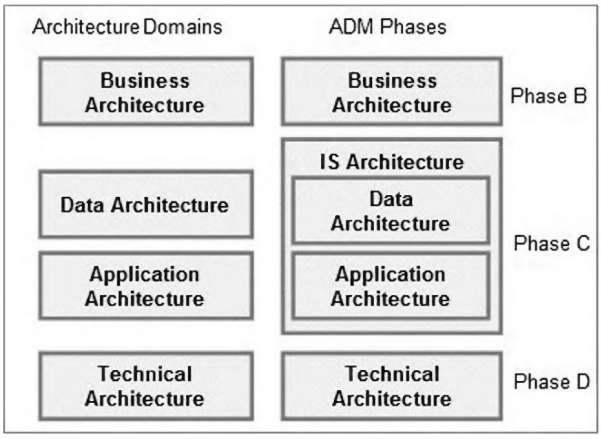
As duas fases que sucedem à fase D, fases E e F, são focadas no gerenciamento do cronograma e organização da implementação da nova arquitetura, com ênfase na definição de um plano de migração. Na fase E, as entregas das fases B, C e D são consolidados (arquiteturas, requisitos e lacunas identificadas) e usados na definição da arquitetura de transição. O plano de migração é então estabelecido na fase F.

A fase G, é a penúltima do ciclo ADM e nela será definida a versão definitiva dos contratos para implementação e recomendações para o *onboarding* da arquitetura. Já na fase H ocorre o gerenciamento da arquitetura implementada, como, por exemplo, a requisição de novas mudanças, que pode acarretar um novo ciclo ADM.

Contudo, apesar da estrutura estabelecida em fases sequenciais, essa sequência pode ser adaptada as necessidades de cada situação, principalmente na forma de iterações no ciclo do ADM. Nesse sentido, a estrutura em fases pode ser entendida como uma referência, muito mais do que uma receita imutável a ser seguida estritamente, sendo até mesmo preferível que sejam aplicados ajustes ao contexto e requisitos específicos (DESFRAY; RAYMOND, 2014).

O presente trabalho focará nas primeiras quatro fases do ciclo ADM, uma vez que as atividades relacionadas a definição arquitetural ocorre dentro dessas fases (figura 8), conforme descrito por Desfray e Raymond (2014).

Figura 8: Domínios de arquitetura e fases do ciclo ADM.



Fonte: Reproduzido de Desfray e Raymond (2014)

Dessa maneira o TOGAF será aplicado nesse trabalho de monografia estruturado da

seguinte maneira, conforme descrito no quadro 3.1:

Quadro 2.1: Aterfatos por fase do ciclo ADM

Fase	Artefatos	Técnica utilizada
A	Modelo de Alto Nível da Solução	Diagrama de componentes
B	Modelo de arquitetura de negócio	Modelagem funcional utilizando BPMN
C	Modelo arquitetura de aplicação	Diagrama de Interação
D	Modelo de estrutura tecnológica	Diagrama de Camadas de Tecnologia

Fonte: Autor

Após a definição dos artefatos e das ferramentas que serão utilizadas no ciclo ADM é possível iniciar a execução das fases, começando pela fase A.

2.10 Considerações do Capítulo

Neste capítulo foram abordados os conceitos e fundamentos de Gerenciamento de Logs, Transformação de Arquitetura, Arquitetura de Middleware, Observabilidade de Software e Detecção de Anomalias. A apresentação de tais assuntos fez-se necessária, uma vez que este trabalho de monografia irá tratar da transformação arquitetural visando o atingimento da observabilidade de um sistema middleware.

No decorrer do capítulo foram apresentados os conceitos de implementação de uma arquitetura alvo a partir de uma arquitetura base, empregando as fases o ciclo ADM, iniciando pela fase preliminar e contemplando as fases onde a arquitetura é produzida (principalmente as fases B, C e D).

Adicionalmente foram estabelecidos os conceitos de MOM e *LogAnomaly*. Sendo MOM um acrograma para *Message-Oriented Middleware*, que descreve uma aplicação que atua como intermediador na comunicação entre dois sistemas em uma relação Produtor/Consumidor. Uma vez que o objeto de estudo desta monografia se caracteriza como um *Middleware* orientado a mensagens. Já o *LogAnomaly* diz respeito a uma técnica de análise de logs utilizando um modelo *Deep Learning* com ênfase em análise semântica.

O TOGAF foi utilizado neste trabalho como a base metodológica para estruturar a adaptação da arquitetura, alinhando-se ao objetivo principal da monografia. Sua aplicação

permitiu abordar a evolução arquitetural de forma sistemática e controlada, adaptando as fases do ciclo ADM às especificidades do contexto proposto.

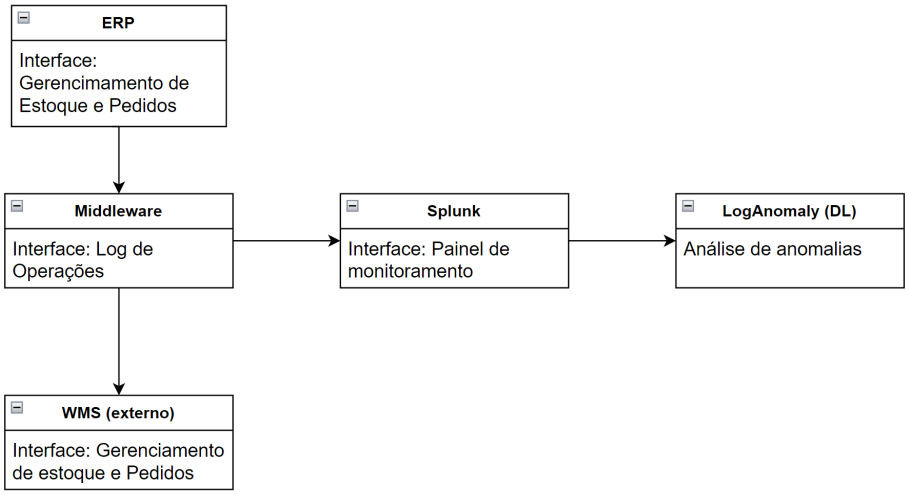
3 ARQUITETURA DE LOGS

Neste capítulo serão abordados temas relacionados a adaptação de uma arquitetura empregando o TOGAF, para uma arquitetura que compreende um sistema ERP on-premisses que está conectado a um middleware orientada a mensagens e serão apresentados os artefatos gerados em cada fase do ciclo ADM conforme o quadro 3.1.

3.1 Fase A

A entrega dessa fase consiste na concepção de um modelo de alto nível da solução e para isso será elaborado um diagrama de componentes do que será arquitetura alvo. No diagrama de componentes (figura 9) o foco é a estrutura dos sistemas e suas interconexões, incluindo o ERP, WMS (*Warehouse Management System*), o *middleware*, a solução Splunk, bem como o módulo de *Machine Learning* responsável pela análise de anomalias.

Figura 9: Diagrama de componentes.



Fonte: Autor

Os componentes e conexões do diagrama apresentado na figura 8 devem ser interpretados da seguinte maneira:

- **Sistema ERP:** Representa o sistema interno da empresa que faz o gerenciamento

de inventário e pedidos. Esse sistema possui interfaces para enviar e receber dados de estoque e pedidos.

- **Sistema WMS:** O componente WMS representa o sistema de gerenciamento de armazém do parceiro logístico externo. Também possuí interfaces para envio e recebimento de dados de estoque e pedidos.
- **Middleware:** Esse componente representa o sistema que faz o intermédio da comunicação entre o sistema ERP da empresa e o WMS do parceiro logístico. Implementa uma arquitetura de middleware orientada a mensagem, garantindo que ambos sistemas não tenham comunicação direta entre si. Além disso, possui uma interface com a solução Splunk para o monitoramento de logs das interfaces.
- **Splunk:** É uma solução WEB para monitoramento e análise de logs, esse componente processa os logs coletados do *middleware*. E possui uma interface que permite gerar visualizações de status dos logs.
- **LogAnomaly:** Esse componente representa o modelo de *Machine Learning* que realiza análise dos logs com anomalias. Conecta-se ao Splunk para consumir logs e realizar análises em tempo real.

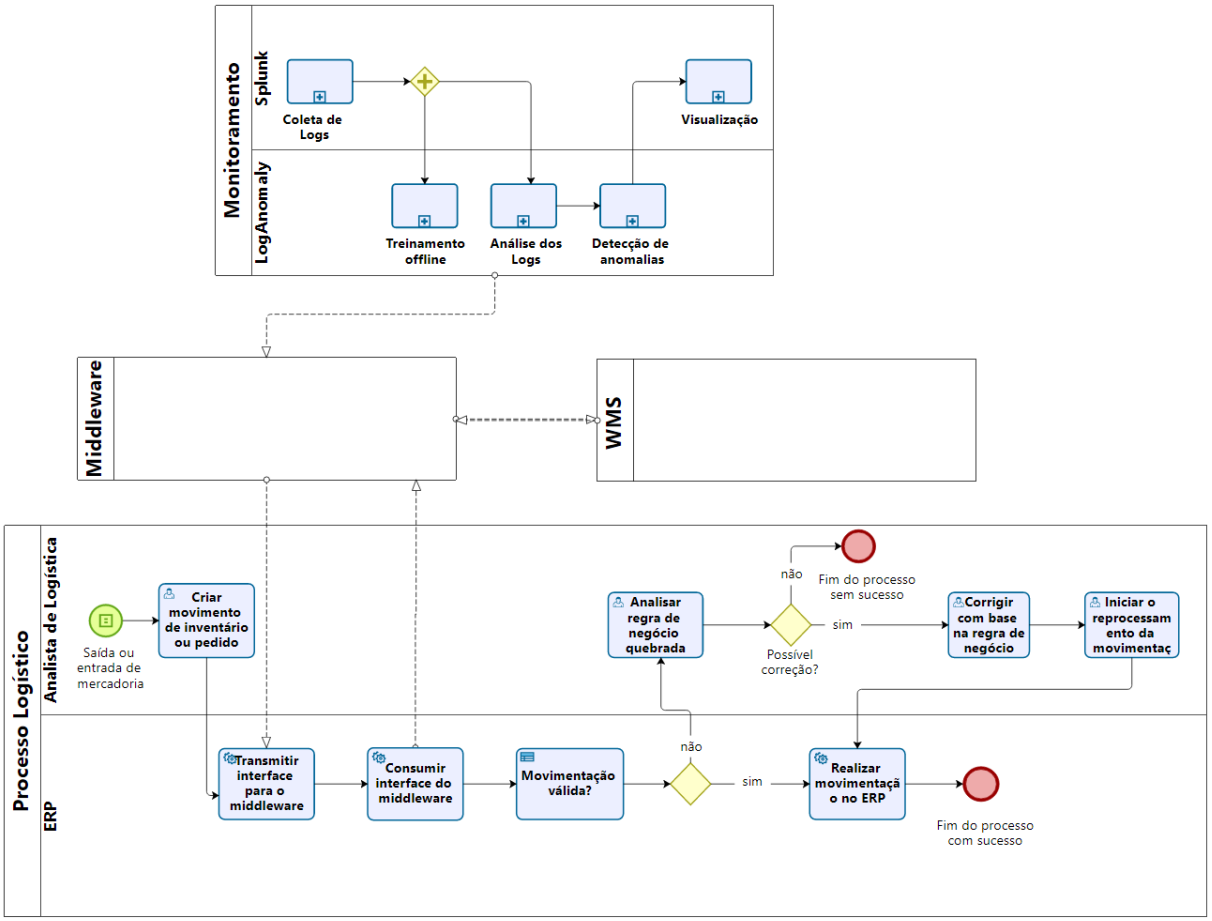
3.2 Fase B

Como descrito por Desfray e Raymond (2014) nessa fase é definida a arquitetura pela visão do negócio, o que será feito através da elaboração de um diagrama funcional utilizando BPMN (*Business Process Model and Notation*).

A BPMN é uma forma de notação utilizada na modelagem de processos de negócio. De acordo com ABPMP (2013), essa metodologia para notação é amplamente empregada para descrever processos de negócio visualmente, visando facilitar o entendimento entre profissionais técnicos e de negócios. No guia de conhecimento em gerenciamento de processos de negócio, publicado pela ABPMP (*Association of Business Process Management Professionals* Brasil em sua terceira versão, a BPMN é exaltada por prover uma notação de fácil compreensão por diferentes stakeholders.

A estrutura da BPMN utiliza elementos visuais que representam: as ativadas, os eventos, *gateways* de decisão e de fluxo de sequência, dessa forma permitindo a elaboração de modelagens que ilustrem a relação entre tarefas e decisões ao longo de um processo. Portanto, a BPMN pode ser entendida como uma linguagem essencial para a documentação e melhoria contínua dos processos de negócio, destacando-se pela clareza e facilidade de interpretação (ABPMP, 2013).

Figura 10: Modelagem utilizando notação BPMN.



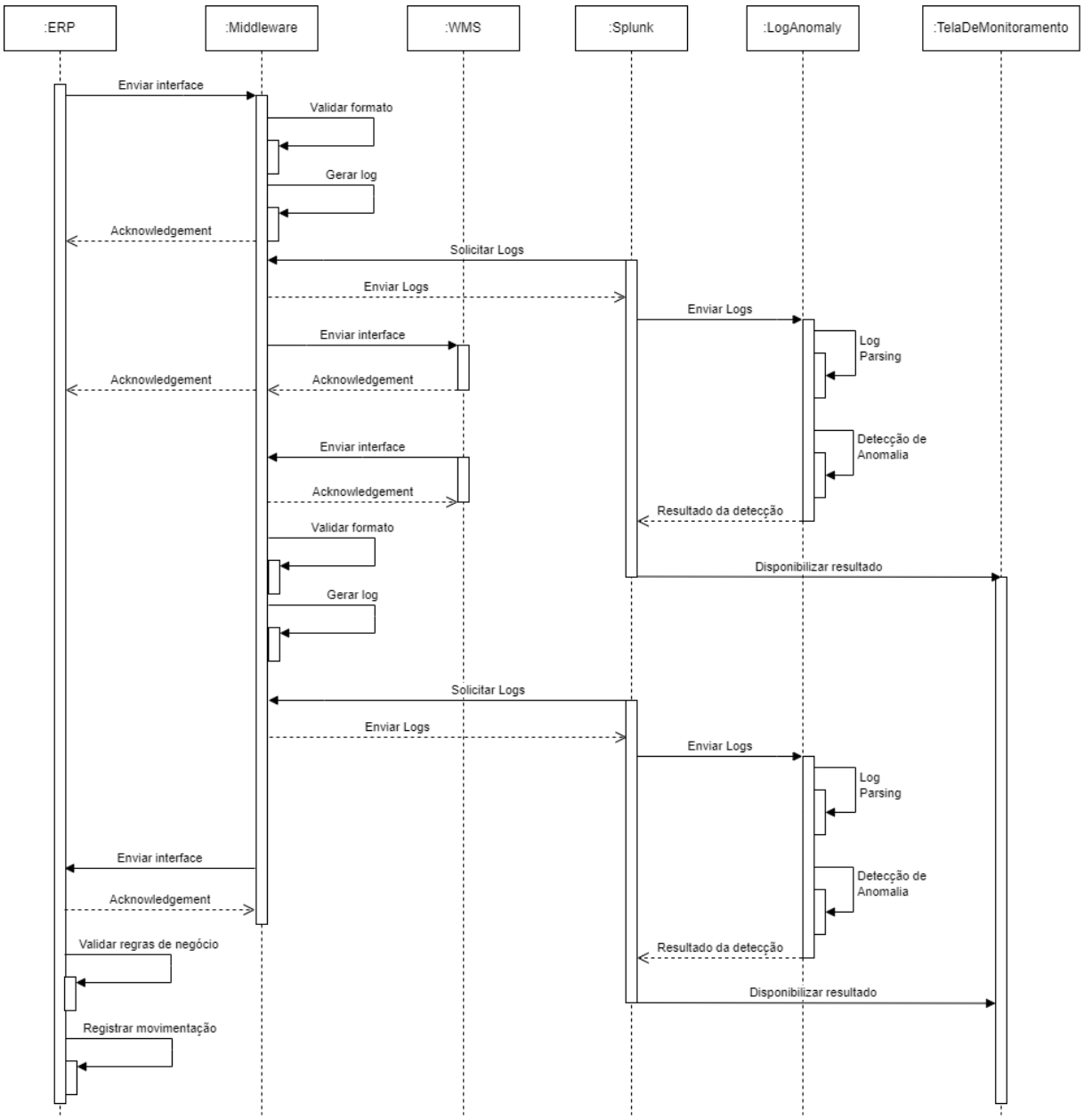
Fonte: Autor.

Na figura 10 pode ser verificada a modelagem BPMN da arquitetura alvo na visão do negócio, retratando os principais elementos do fluxo e oferendo uma visão abrangente dos principais componentes.

3.3 Fase C

A fase C, chamada de fase da arquitetura do sistema de informação, pretende a elaboração de um modelo onde esteja determinada de qual maneira o sistema será utilizado na empresa. Nessa fase o objetivo principal não é o desenho de aplicações, mas sim elaborar uma visão lógica das etapas do processo (DESFRAY; RAYMOND, 2014).

Figura 11: Modelagem do diagrama de interações utilizando UML.



Fonte: Autor.

Segundo Desfray e Raymond (2014), o artefato central da fase C é um diagrama que representa a arquitetura e posicionamento dos componentes da aplicação e a definição de suas interfaces e interconexões. Para gerar esse artefato será empregada a técnica de diagrama de interações, comumente referido como diagrama de sequência (figura 11), onde é possível visualizar as mensagens trocadas entre as ocorrências dos componentes da aplicação.

O diagrama de interações foi elaborando utilizando UML (*Unified Modeling Language*), que serve como forma de notação padronizada amplamente utilizada no desenvolvimento de sistemas, se tornando a língua franca dessa área (WATSON, 2008).

3.4 Fase D

A fase D do ciclo ADM é conhecida como a fase de desenvolvimento da arquitetura tecnológica. Neste estágio, ocorre a determinação a arquitetura do ponto de vista de tecnologia. Esta fase é crucial para garantir a interoperabilidade e a integração de tecnologias com o restante da arquitetura corporativa (DESFRAY; RAYMOND, 2014).

Como artefato dessa fase foi elaborado o diagrama de camadas, utilizando a linguagem UML. O diagrama foi divido em três camadas, a primeira camada é a de interfaces, nessa camada estão representadas as interfaces do usuário com sistema. Na segunda camada estão contemplados os componentes e módulos funcionais. Já na terceira temos a representação dos elementos de dados.

Na camada de interfaces existem três elementos que representam as interfaces dos sistemas ERP, WMS e o Monitor de Logs, pelas quais os respectivos usuários interagem com eles:

- **Interface do Sistema ERP:** Os operadores e gerentes utilizam a interface do ERP para gerenciar dados de pedidos, estoque e outras operações comerciais. O sistema em questão apresenta informações processadas e atualizadas com base nas transações registradas e armazenadas em seu banco de dados enas interações com o sistema WMS por meio do middleware.

- **Interface do Sistema WMS:** Os operadores do armazém externo utilizam a interface do sistema WMS para gerenciar a logística, incluindo a entrada e saída de produtos. O WMS atualiza os dados em tempo real com base nas informações relevantes enviadas pelo ERP através do middleware, refletindo o status do estoque e dos pedidos.
- **Interface Monitor de Logs:** Analistas e operadores de TI (Tecnologia da Informação) utilizam a interface do Splunk para visualizar e monitorar, em tempo real, os logs gerados pelo middleware. Permitindo análises detalhadas e fornece visão do desempenho e possíveis anomalias, com alerta gerado pelo componente LogAnomaly de Machine Learning.

Na segunda camada, denominada camada de serviço, temos representados os seguintes componentes:

- **Processamento de Pedidos e inventário:** Responsável pelo processamento das informações de negócios e comunicação de dados relevantes ao processo ao sistema WMS via o middleware, transacionando dados de pedidos e inventário, utilizadas pelo WMS na atualização de suas operações logísticas, sincronizando os registros nas interfaces do ERP e do WMS.
- **Gerenciamento de Inventário:** Processa os dados de movimentações de estoque e pedidos do ERP por meio do middleware e disponibiliza essas informações na interface para os usuários de logística, dependendo da consistência dos dados enviados pelo ERP para uma visão atualizada das operações logísticas.
- **Middleware:** Um sistema de intermediação da comunicação orientado a mensagens, agindo como agente intermediário entre os sistemas ERP e WMS, transmitindo mensagens de forma assíncrona, registrando logs de transações e comunicações, os quais são enviados ao Splunk. Este middleware é vital para a integração indireta e segura dos sistemas, gerando logs para monitoramento e segurança.
- **Módulo de Coleta de Logs:** Através da aplicação Splunk, os logs gerados pelo middleware são coletados e passam pelo processo de *parsing*, gerando percepções sobre o tráfego de mensagens, desempenho e possíveis problemas. Esses logs são

compartilhados com o LogAnomaly para a identificação de possíveis anomalias em tempo real, contribuindo com os times responsáveis pelo monitoramento a detectar falhas ou atividades incomuns.

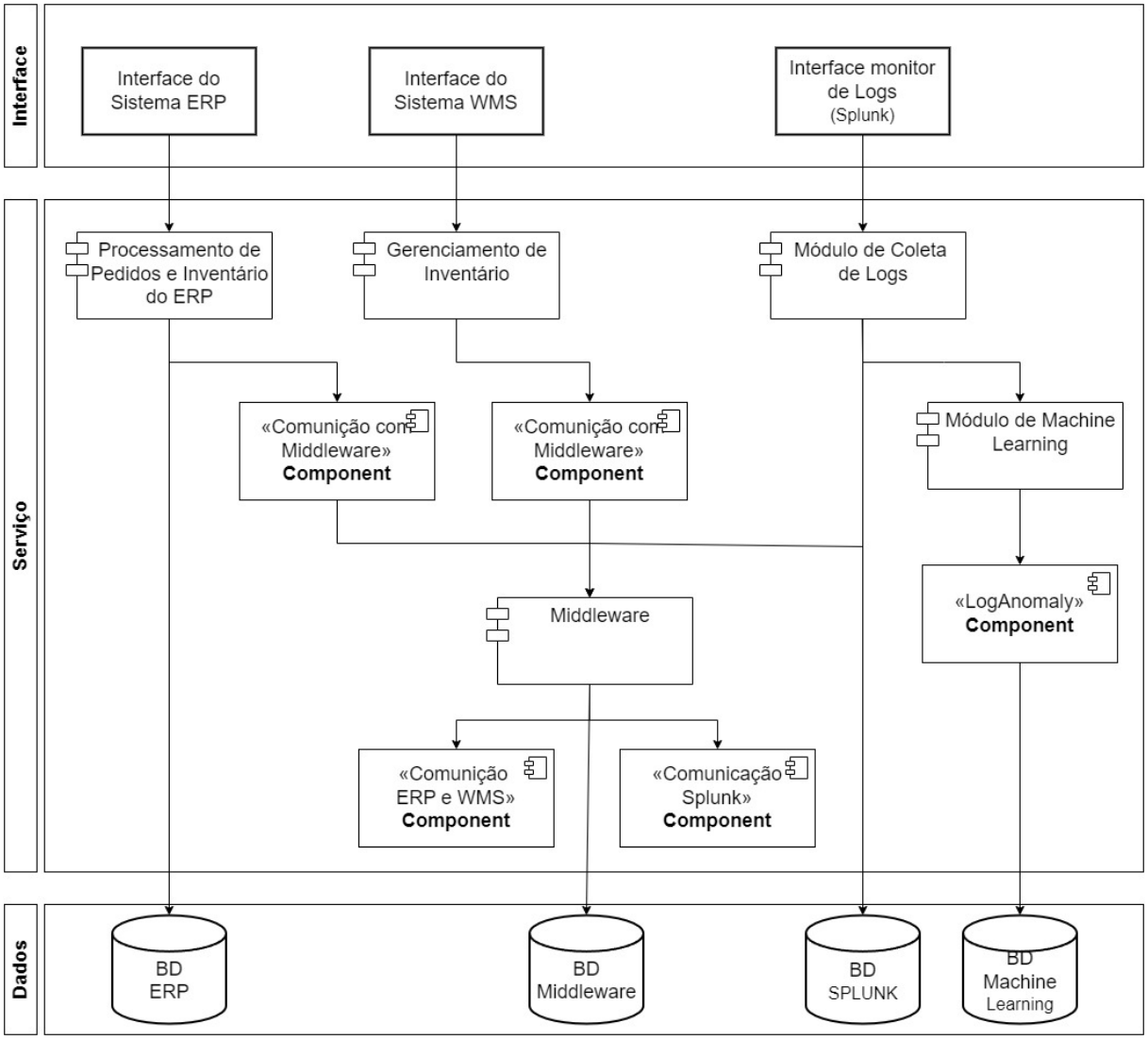
- **Módulo de Machine Learning:** Aplica o modelo de Machine Learning conhecido como LogAnomaly para analisar logs e identificar padrões incomuns que podem indicar problemas, alertando através da interface com o Splunk.

Na terceira camada estão representados os elementos arquiteturais relacionados a persistência dos dados utilizados. Sendo eles:

- **Banco de dados do ERP:** Persiste dados de pedidos, estoque e transações relacionadas ao ERP, enquanto os usuários interagem com o sistema ocorre a atualização dos dados.
- **Banco de dados do Middleware:** O banco de dados de um MOM armazena e registra logs de todas as transações e comunicações entre os sistemas integrados. Permitindo consultas históricas, análise de desempenho e monitoramento em tempo real. Esses dados são essenciais para identificar anomalias e garantir a segurança. Essencial também para a operação fluida e segura dos sistemas conectados.
- **Banco de dados do Splunk:** Registra todos os logs das transações intermediadas pelo middleware entre ERP e WMS, mantendo históricos que possibilitam consultas e análises de desempenho. Esses logs são essenciais para o Splunk e o LogAnomaly, usado para análises de anomalias.
- **Banco de dados de Machine Learning:**Armazena dados históricos de logs para o treinamento e atualização do método de Machine Learning LogAnomaly. Esse banco é crucial para manter o LogAnomaly atualizado, possibilitando assim que ele aprenda com dados históricos e melhore a detecção de anomalias em tempo real.

Essas relações interligadas garantem que a operacionalização e monitoramento dos sistemas ocorram com fluidez, com uma camada dedicada para cada função-chave: interface de usuário, serviço e dados.

Figura 12: Diagrama de Camadas.



Fonte: Autor.

O diagrama de camadas (figura 12) proporciona uma visão detalhada das interações entre os diversos componentes e sistemas, possibilitando o entendimento das dependências. A utilização dessa técnica holística garante que as estratégias de TI estejam alinhadas aos objetivos de negócio, assegurando uma infraestrutura tecnológica robusta e eficiente.

Assim, a fase D, através do diagrama de camadas, não apenas documenta a arquitetura tecnológica como também pode ser utilizada em futuras evoluções e adaptações, fomentando um gerenciamento integrado e eficaz dos recursos empresariais (DESFRAY; RAYMOND, 2014).

3.5 Módulo de Machine Learning

Como mencionado anteriormente, nesse trabalho está sendo considerado a inclusão de um módulo de *Machine Learning* na arquitetura de um processo de logística empresarial e o método escolhido foi o LogAnomaly. A implementação desse módulo foi projetada em conjunto com a solução de monitoramento de logs de sistemas homônimo, desenvolvida pela empresa Splunk. Nessa seção serão será apresentado com mais detalhes como o módulo de *Machine Learning* está integrado na arquitetura.

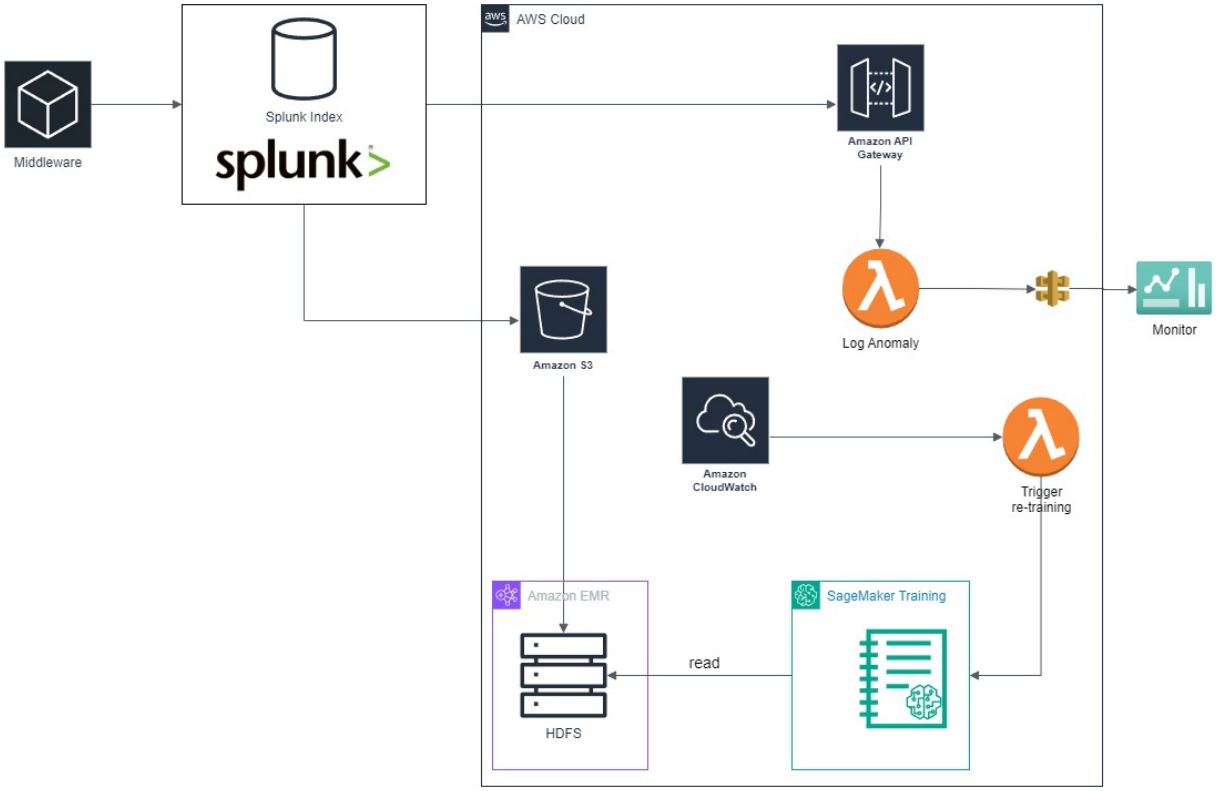
O módulo em si pode ser compreendido em duas partes principais, uma dedicada ao treinamento do modelo e outra responsável pela detecção de anomalias nos logs em tempo real. A origem dos dados para o modelo será o monitor Splunk que fará a coleta dos logs gerados pelo middleware, a ferramenta de monitoramento por sua vez está conectada ao módulo de *Machine Learning* por meio de uma API (*Application Process Interface*) e dessa maneira receberá os arquivos de logs após a atividade de *log parsing* que será executada ainda pela solução Splunk. Em seguida se dará análise de anomalia dos logs, o resultado da análise dos logs e então enviado para a ferramenta de monitoramento que disponibiliza o resultado da análise via uma interface gráfica.

Como mencionado, além da detecção de anomalias em tempo real, o modelo também possui um módulo de treinamento e re-treinamento. Nesse caso os dados brutos de logs são enviados para um repositório de arquivos HDFS (Hadoop Distributed File System), de forma que eles podem ser utilizados tanto em treinamentos como para a avaliação do modelo. No caso da função de retreinamento, um limite deve ser informado para dizer qual nível de degradação de desempenho deve disparar uma necessidade de retreinamento do modelo (CHEN et al., 2022).

Para modelar a arquitetura do modelo de *Machine Learning* foi empregada a notação de arquitetura AWS (*Amazon Web Services*), foram modelados os fluxos de detecção de anomalias em tempo real e o fluxo de treinamento e re-treinamento do modelo. No primeiro, foi representado a conexão entre o middleware, o Splunk, o módulo de *Machine Learning* e o dashboard que disponibiliza os resultados. Já o fluxo de treinamento ilustra o processo de armazenamento dos dados brutos de logs no HDFS, o treinamento inicial e

o disparo para o re-treinamento do modelo com base no estabelecido para a degradação do desempenho.

Figura 13: Módulo de Machine Learning.



Fonte: Autor.

Na elaboração do diagrama ilustrado na figura 13 foram utilizados os seguintes componentes de arquitetura AWS:

- **AWS Lambda:**Para a execução da análise em tempo real dos logs (como inferência do modelo de detecção de anomalia). E para engatilhar o retreinamento quando o limite mínimo de desempenho estabelecido for atingido.
- **Amazon S3:** Para armazenamento temporário de arquivos de log entre Splunk e o HDFS.
- **Amazon EMR:** Para gerenciar o armazenamento em HDFS, além de facilitar o processamento de dados em lote.
- **AWS SageMaker:** Para treinamento e re-treinamento do modelo LogAnomaly.

- **Amazon CloudWatch:** Para monitorar o desempenho do modelo e disparar alarmes quando for necessário um re-treinamento.

A notação na arquitetura AWS foi escolhida pela facilidade no entendimento e ser profusamente utilizada na indústria e na academia para ilustrar arquiteturas.

3.6 Análise e discussão dos resultados

Nesta seção serão discutidos os resultados durante a execução da metodologia TOGAF por meio do *Architecture Development Method*.

3.6.1 Resultado Ciclo ADM

Conforme exposto no capítulo anterior, foram selecionadas no escopo desse trabalho de monografia as quatro primeiras fases do ciclo de desenvolvimento de arquitetura ADM. Na primeira fase, identificada como fase A, como artefato dessa fase foi obtido um diagrama de componentes e suas conexões, fornecendo uma visão alto nível do que seria a arquitetura alvo, a qual se almejava conceber ao final das fases do ciclo.

Após a conclusão do diagrama de componentes, se encerra a fase A e assim iniciando a fase B, caraterizada por ser a etapa onde a arquitetura é apresentada na visão negócio, por esse motivo foi elaborado um diagrama de processos utilizando a notação *Business Process Modeling Notation*, dessa forma foi possível gerar o artefato definido para essa fase, a arquitetura no ponto de vista do negócio. A principal preocupação nessa etapa é garantir que todos envolvidos no possam ter um entendimento adequado do diagrama e consequentemente uma interpretação correta da arquitetura.

Seguidamente a conclusão da fase B vem a fase da arquitetura na visão de sistema da informação, identificada como fase C, aqui o artefato produzido foi um diagrama de interações modelado em UML. O diagrama de interações facilita o entendimento e a deixa clara as diferentes interações entre os componentes de um sistema, auxiliando na detecção de possíveis problemas de comunicação e integração. Na elaboração de um diagrama desse

tipo é necessário em primeiro lugar ter o entendimento da interação entre os diferentes componentes, além de assegurar que o diagrama não deixe margem para ambiguidades de interpretação, o que levaria ao entendimento incorreto do comportamento do sistema.

A última fase do ciclo ADM, que foi executada e exposta no capítulo anterior, é a fase D, o domínio arquitetural abordado nessa etapa é o da tecnologia e consequentemente o artefato gerado nela também está ligado a esse domínio. O diagrama traz uma visão compreensível e sistematizada da arquitetura tecnológica, contribuindo para o planejamento estratégico e a gestão de mudanças durante o ciclo de vida do sistema, além de ter adicionalmente o potencial de melhorar a comunicação entre os times de desenvolvimento e operações.

3.6.2 Arquitetura Machine Learning

A arquitetura elaborada no capítulo anterior mostra como poderia ser feita a integração bem-sucedida de um módulo de *Machine Learning* à arquitetura de monitoramento de logs da empresa, empregando a ferramenta Splunk para a coleta dos logs e *parsing*. Tal integração sustenta a detecção e análise de anomalias me logs de sistema em tempo real.

A dependência de componentes externos e distribuídos, como, por exemplo, o Amazon EMR para realizar o armazenamento no HDFS e o Amazon SageMaker para o treinamento do modelo, agregam complexidade para a gestão de infraestrutura e sincronização dos dados.

Finalmente, a demanda por retreinamento automático acarreta necessidade de que estratégias de detecção eficazes sejam implementadas para identificar variações no desempenho do modelo sem gerar falsos positivos, garantindo que o retreinamento seja acionado sempre que necessário. Nesse ponto destaca-se a importância de uma arquitetura flexível e escalável, bem como de um rastreamento ininterrupto das métricas e desempenho do modelo.

3.7 Considerações do Capítulo

O capítulo destaca a importância do efeito estruturante da metodologia TOGAF, aplicada por meio das fases do ciclo ADM, para a evolução de uma arquitetura. As fases A, B, C e D oferecem um framework robusto que garante o desenvolvimento gradual e controlado da arquitetura, possibilitando iterações entre as fases para gerar incrementos sucessivos. Durante essas etapas, foram produzidos artefatos essenciais, como diagramas de processos de negócios, interações de sistemas e infraestrutura tecnológica, assegurando clareza e coerência na proposta arquitetural.

A integração de serviços como Amazon EMR e SageMaker ao módulo de Machine Learning trouxe valor significativo ao viabilizar o treinamento e re-treinamento automatizado e escalável dos modelos. No entanto, esses componentes introduziram desafios adicionais, como o gerenciamento de infraestrutura e a sincronização de dados. A necessidade de um processo de re-treinamento automático enfatizou a importância de estratégias robustas de monitoramento do desempenho dos modelos, a fim de prevenir alarmes falsos e garantir a eficácia do sistema.

A análise realizada confirma que a arquitetura concebida é funcional, flexível e modular, permitindo ajustes conforme os requisitos evoluam. A aplicação da metodologia TOGAF e das fases do ciclo ADM foi crucial para a criação de uma arquitetura escalável e capaz de evoluir continuamente ao longo do tempo. A definição de um ponto de partida comum, representada pela arquitetura base, e a adaptação do ciclo ADM às necessidades do trabalho foram passos fundamentais nesse processo.

O capítulo também abordou a localização e integração do módulo de *Machine Learning* na arquitetura, evidenciando como ele se conecta com os demais componentes. A apresentação das fases iniciais do ciclo ADM, dos artefatos gerados e dos métodos empregados reforça o caráter adaptável e estruturado da metodologia TOGAF.

4 CONSIDERAÇÕES FINAIS

Este capítulo apresenta as conclusões gerais do trabalho, suas principais contribuições e sugestões para trabalhos futuros.

4.1 Conclusões

A utilização da metodologia TOGAF, trilhando as fases descritas no método de desenvolvimento de arquiteturas, possibilitou o desenvolvimento de uma arquitetura integrada e escalável, que apoia o monitoramento e a análise de logs em tempo real via um módulo de *Machine Learning*. A implementação das fases do ciclo, iniciando na fase A até a arquitetura tecnológica na fase D, proporcionou uma estrutura coerente e bem documentada, facilitando a conformidade entre os objetivos de TI e as metas de negócios.

Empregando o modelo LogAnomaly, aliado ao monitoramento de logs com Splunk, demonstrou-se que uma solução que combina dois processos essenciais, a detecção de anomalias em tempo real e retreinamento automatizado do modelo quando necessário, é viável. Garantindo uma arquitetura que proporciona alto desempenho e agilidade ao sistema.

4.2 Contribuições do Trabalho

Este trabalho de monografia oferece contribuições importantes para a prática e pesquisa em arquitetura corporativa e integração de *Machine Learning*. Em primeiro lugar, expõe a eficácia do TOGAF como um direcionador para o trabalho de construir uma arquitetura modularizada e adaptável em ambientes complexos.

Adicionalmente, a integração do método LogAnomaly com serviços como Amazon EMR e SageMaker adiciona uma camada de inteligência ao sistema de monitoramento, amplificando as capacidades de detecção de anomalias e possibilitando retreinamento contínuo com base no desempenho.

4.3 Trabalhos Futuros

Para pesquisas futuras, recomenda-se explorar a extensão desta arquitetura para incluir análises preditivas mais avançadas, aumentando a capacidade de reconhecer falhas e aprimorar processos logísticos. Outra área de desenvolvimento poderia ser a implementação de uma camada de orquestração seria utilizada na automatização do gerenciamento da infraestrutura e o ciclo de retreinamento com mais precisão.

REFERÊNCIAS

ABPMP, Brasil. **BPM CBOK V3.0**. Association of Business Process Management Professionals, 2013.

CHARITY MAJORS, Liz Fong-Jones; MIRANDA, George. **Observability Engineering: Achieving Production Excellence**. Edição: Kate Galloway. O'REILLY, 2022.

CHEN, Zhuangbin et al. Experience Report: Deep Learning-based System Log Analysisfor Anomaly Detection. **Association for Computing Machinery**, 2022. DOI: <.org/10.1016/j.jss.2022.111537>.

DE LUNETTA E RODRIGUES GUERRA, Avaetê. METODOLOGIA DA PESQUISA CIENTÍFICA E ACADÊMICA. **Revista OWL**, Zenodo, 2023. DOI: <10.5281/ZENODO.8240361>.

DESFRAY, Philippe; RAYMOND, Gilbert. **Modeling Enterprise Architecture with TOGAF**. Edição: Andrea Dierna. Elsevier, 2014.

ETZKORN, Letha Hughes. **Introduction to middleware.pdf**. Taylor & Francis Group, 2017.

HARJUNPÄÄ, Niklas; SIEKKINEN, Matti. **Master's Programme in Computer, Communication and Information Sciences**. 2023. Diss. (Mestrado) – Aalto University School of Science.

KENT, Karen; SOUPPAYA, Murugiah. Guide to Computer Security Log Management. **Special Publication 800-92**, 2006.

WATSON, Andrew. Visual Modelling: past, present and future, 2008.