Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

AutoMLOps: Arquitetura de Software Autoadaptativa para Internet das Coisas (IoT)

Eduarda Araujo Carvalho

JUIZ DE FORA
JANEIRO, 2026

AutoMLOps: Arquitetura de Software Autoadaptativa para Internet das Coisas (IoT)

Eduarda Araujo Carvalho

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: José Maria Nazar David

Coorientador: Regina Maria Maciel Braga Vilella

JUIZ DE FORA

JANEIRO, 2026

AUTOMLOPS: ARQUITETURA DE SOFTWARE AUTOADAPTATIVA PARA INTERNET DAS COISAS (IOT)

Eduarda Araujo Carvalho

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

José Maria Nazar David
Doutor em Engenharia de Sistemas e Computação – UFRJ

Regina Maria Maciel Braga Vilella
Doutor em Engenharia de Sistemas e Computação – UFRJ

Prof. Ronney Moreira de Castro
Doutor em Informática – UNIRIO

Prov. Victor Stroele de Andrade Menezes
Doutor em Engenharia de Sistemas e Computação –UFRJ

JUIZ DE FORA

22 DE JANEIRO, 2026

Resumo

A Internet das Coisas (IoT) requer sistemas de processamento de dados em tempo real. Nesse cenário, a integração de Inteligência Artificial (IA) e aprendizado de máquina supervisionado viabiliza a análise automatizada de dados. Entretanto, a ocorrência de *concept drift* em fluxos contínuos (*streaming*) ocasiona a degradação de desempenho dos modelos. Para tratar esse problema, este trabalho apresenta a arquitetura AutoMLOps, disponibilizada como um serviço via *API* utilizando *FastAPI* e direcionada a aplicações de IoT, com configuração parametrizável conforme o domínio. A arquitetura realiza a seleção automática de modelos supervisionados por meio de técnicas de *AutoML*, monitora métricas de desempenho e adota o padrão *Champion/Challenger*. Adicionalmente, utiliza um índice de saúde unificado (*Health Index* – HI) para acionar processos de retreinamento e chaveamento de modelos. A avaliação experimental, conduzida com 3.283 registros de monitoramento bovino para emissão de CO_2, evidenciou a capacidade do HI em identificar a degradação de desempenho. Em cenários com ocorrência de *concept drift* induzido por inserção de ruído, a redução do índice acionou os mecanismos de controle e o ciclo *human-in-the-loop*, permitindo o retreinamento automático e a recuperação do desempenho do sistema.

Palavras-chave: Internet das Coisas (IoT), AutoMLOps, Aprendizado de Máquina, *concept drift*, Arquiteturas autoadaptativas

Abstract

The Internet of Things (IoT) requires real-time data processing systems capable of handling continuous data streams. In this context, the integration of Artificial Intelligence (AI) and supervised machine learning enables automated data analysis. However, the occurrence of *concept drift* in streaming data leads to performance degradation of predictive models over time. To address this challenge, this work presents the AutoMLOps architecture, delivered as an API-based service using *FastAPI* and designed for IoT applications with domain-specific parametrization. The architecture performs automatic selection of supervised learning models through *AutoML* techniques, continuously monitors performance metrics, and adopts the *Champion/Challenger* pattern. Additionally, it introduces a unified Health Index (HI) to trigger retraining processes and model switching. The experimental evaluation, conducted with 3.283 records of bovine monitoring data for CO_2 emissions, demonstrated the effectiveness of the Health Index in identifying performance degradation. In scenarios with induced *concept drift* through noise insertion, the reduction in the index activated control mechanisms and the *human-in-the-loop* cycle, enabling automatic retraining and recovery of system performance.

Keywords: Internet of Things (IoT), AutoMLOps, Machine Learning, *concept drift*, Self-adaptive Architectures

Agradecimentos

Inicialmente, agradeço a Deus pela força, sabedoria e perseverança ao longo desta trajetória acadêmica. Paralelamente, reconheço o papel da minha família e parentes pelo encorajamento, apoio e compreensão demonstrados durante o desenvolvimento deste trabalho.

No que diz respeito à orientação, agradeço ao Prof. Dr. José Maria Nazar David pela disponibilidade e pelas contribuições para a conclusão deste projeto. Ainda sob essa perspectiva, destaco a colaboração da coorientadora, Profa. Dra. Regina Maria Maciel Braga Vilella, pelas contribuições e incentivo oferecidos durante as etapas da pesquisa.

Além disso, registro meu agradecimento aos professores e funcionários do Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora pelos ensinamentos e suporte prestados durante a graduação. Por fim, menciono os amigos pelo companheirismo e apoio mantidos em todos os momentos desta jornada.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

API	Interface de Programação de Aplicações
AUC	Área Sob a Curva
AutoML	Aprendizado de Máquina Automatizado
AutoMLOps	Arquitetura de *Software* autoadaptativa para IoT
CI/CD	Integração Contínua e Entrega Contínua
CPS	Sistemas Ciber-Físicos
DCC	Departamento de Ciência da Computação
DSR	Pesquisa em Ciência do Design
HI	Índice de Saúde
HITL	Intervenção Humana no Ciclo
IA	Inteligência Artificial
IoT	Internet das Coisas
M2M	Comunicação Máquina a Máquina
MAPE-K	Monitoramento, Análise, Planejamento, Execução e Conhecimento
MLOps	Operações de Aprendizado de Máquina
MSL	Mapeamento Sistemático da Literatura
PICOC	População, Intervenção, Comparação, Resultados e Contexto
PRISMA	Diretrizes para Relato de Revisões Sistemáticas e Meta-Análises
RF	Requisito Funcional
RNF	Requisito Não Funcional
SDN	Redes Definidas por *Software*
SOA	Arquitetura Orientada a Serviços
UFJF	Universidade Federal de Juiz de Fora
XAI	Inteligência Artificial Explicável

1 Introdução

A Internet das Coisas (IoT) estabelece um paradigma tecnológico fundamentado na interconexão de dispositivos físicos para coleta, transmissão e processamento de dados em tempo real (MANCINI, 2017). Essa capacidade impulsiona aplicações em setores como saúde, agricultura de precisão, cidades inteligentes e Indústria 4.0, ao viabilizar a automação e o suporte à tomada de decisão (ROCHA; KISSIMOTO, 2022). Contudo, o volume de dados gerado, caracterizado por variabilidade e imprevisibilidade, demanda arquiteturas de *software* adaptativas. Nesse contexto, Heinz et al. (2018) definem tais arquiteturas como sistemas capazes de monitorar, analisar e modificar seu comportamento em resposta a mudanças internas e externas, preservando atributos de qualidade.

Sob essa perspectiva, arquiteturas de *software* autoadaptativas distinguem-se pela capacidade de monitorar seu funcionamento e ajustar o comportamento diante de alterações no ambiente operacional (KEPHART; CHESS, 2003). Paralelamente, a Inteligência Artificial (IA) consolidou-se como ferramenta para análise de dados, sendo incorporada em sistemas IoT (AVILA et al., 2022). A integração entre arquiteturas autoadaptativas e técnicas de IA configura uma abordagem para o tratamento da natureza dinâmica dos dados em ambientes IoT, contribuindo para a eficiência sistêmica.

Apesar dos avanços, soluções existentes apresentam limitações. O monitoramento de modelos de aprendizado de máquina depende de intervenção manual, o que compromete a escalabilidade e a autonomia dos sistemas (ROCHA; KISSIMOTO, 2022; SOUZA; FONTANARI, 2024). Ademais, diversas propostas concentram-se em etapas isoladas do ciclo adaptativo, como o monitoramento ou a seleção de modelos, em detrimento de mecanismos integrados de substituição automática baseados em métricas de desempenho (AUDRITO, 2020). Essas lacunas restringem a aplicabilidade das soluções em cenários IoT, nos quais a dinâmica dos dados pode levar à degradação do desempenho dos modelos preditivos em razão de fenômenos como o *concept drift* (GAMA et al., 2014). O *concept drift* manifesta-se quando a relação estatística entre as variáveis de entrada e a variável alvo se altera ao longo do tempo, invalidando a função de mapeamento aprendida du-

rante a fase de treinamento (WIDMER; KUBAT, 1996; QUIÑONERO-CANDELA et al., 2009). Em ambientes de IoT, onde os fluxos de dados são contínuos e não estacionários, esse fenômeno ocorre de forma abrupta, gradual, incremental ou recorrente (GAMA et al., 2014; LU et al., 2018).

Diante desse contexto, o presente trabalho busca responder à seguinte questão de pesquisa: **"Como apoiar a construção de arquiteturas autoadaptativas que utilizam técnicas de IA por meio do monitoramento de modelos?"**

Esta investigação fundamenta-se na arquitetura *ADAPTFlow*, proposta por Soares (2024), cujo objetivo é automatizar as etapas iniciais do ciclo de vida do aprendizado de máquina. A arquitetura original disponibiliza um serviço para ingestão de dados, treinamento de múltiplos algoritmos via *AutoML* e seleção dos modelos com melhor desempenho. Entretanto, essa proposta limita-se às fases de treinamento e seleção, sem contemplar o gerenciamento pós-implantação em produção.

Para mitigar essa lacuna, este trabalho propõe a extensão da arquitetura *ADAPTFlow* por meio da implementação da camada operacional (*Ops*) do ciclo de *MLOps*. O objetivo principal consiste no desenvolvimento de uma *API* parametrizável, escalável e agnóstica ao domínio, capaz de viabilizar a autoadaptação de sistemas IoT por meio do monitoramento contínuo de modelos de aprendizado de máquina em produção.

Para alcançar o objetivo principal, foram definidos os seguintes objetivos específicos: (i) realizar um Mapeamento Sistemático da Literatura (MSL) para compreender o estado da arte e identificar lacunas sobre adaptação em sistemas IoT; (ii) desenvolver a arquitetura *AutoMLOps*, implementando mecanismos de monitoramento contínuo (*Health Index*) e estratégias de substituição de modelos (*Champion/Challenger*); e (iii) avaliar a eficácia da solução proposta por meio de um estudo de caso no domínio da Agropecuária, aplicado ao monitoramento de emissões de CO_2.

A arquitetura proposta, denominada *AutoMLOps*, adota a estratégia *Champion/Challenger* para o gerenciamento do ciclo de vida dos modelos e emprega o padrão de projeto *Observer* como mecanismo central de adaptação. Essa abordagem substitui o monitoramento passivo por um processo de reavaliação proativa, acionado por evidências de degradação do desempenho, tais como anomalias nos dados de entrada ou redução

da confiança nas predições (SOARES, 2024; AUDRITO, 2020). A coleta contínua de evidências em tempo real contribui para a eficiência operacional e a capacidade de resposta do sistema frente a mudanças.

O monitoramento integra o ciclo *Human-in-the-Loop* e baseia-se em uma métrica unificada denominada *Health Index* (HI), a qual orienta as decisões de adaptação. O HI consiste em uma medida composta que avalia a condição do modelo em produção a partir da combinação de três dimensões: o Risco de Deriva (*Drift Risk*), associado a variações estatísticas nos dados; o Risco de Confiança (*Confidence Risk*), relacionado à incerteza das predições; e o Risco de Anomalia (*Anomaly Risk*), responsável por identificar observações fora do padrão do conjunto de dados original (GAMA et al., 2014).

Com base no HI, o sistema gera alertas automáticos para retreinamento e chaveamento de modelos por meio de *endpoints* da *API*. O serviço permite a configuração de métricas de desempenho, como acurácia, AUC, *recall*, precisão e *F1-score*, transformando a seleção de modelos em um processo contínuo de gerenciamento do ciclo de vida. Para apoiar a análise dos dados, a solução integra-se à ferramenta *Grafana* (Grafana Labs, 2025) para visualização dos indicadores.

O restante deste trabalho organiza-se da seguinte forma: o Capítulo 2 apresenta o referencial teórico; o Capítulo 3 discute os trabalhos relacionados; o Capítulo 4 descreve a metodologia baseada em *Design Science Research*; o Capítulo 5 detalha o desenvolvimento da solução; o Capítulo 6 apresenta os resultados experimentais; o Capítulo 7 discute as implicações dos resultados; e o Capítulo 8 apresenta as conclusões e trabalhos futuros.

2 Referencial Teórico

Este capítulo apresenta os conceitos fundamentais que sustentam a compreensão da solução proposta, fornecendo o embasamento teórico necessário para o desenvolvimento e a avaliação do artefato de pesquisa.

2.1 IoT

De acordo com Farooq et al. (2015), a IoT consiste em uma rede de objetos físicos, ou "coisas", equipados com sensores, atuadores, *softwares* e recursos de conectividade, que possibilitam a troca de dados entre dispositivos, fabricantes, operadores e outros sistemas conectados.

Esse conceito vai além da comunicação máquina a máquina (M2M), uma vez que incorpora inteligência e autonomia ao ambiente computacional, permitindo que dispositivos atuem de forma colaborativa e integrada (FAROOQ et al., 2015). A capacidade de gerar, processar e compartilhar grandes volumes de dados em tempo real tem impulsionado processos de transformação digital em diversos setores, como saúde, transporte, indústria, cidades inteligentes e agricultura de precisão (SUN et al., 2025). Esses avanços têm intensificado a demanda por soluções arquiteturais capazes de lidar com sistemas complexos, especialmente no que se refere a atributos de qualidade, como escalabilidade, confiabilidade e desempenho (TREVEIL et al., 2020).

Segundo Ali et al. (2022), a IoT adota, de forma geral, uma arquitetura baseada em camadas, comumente estruturada em:

- **Camada de percepção**, responsável pela captura de informações do ambiente por meio de sensores e atuadores;
- **Camada de rede**, encarregada da comunicação e do transporte de dados, utilizando diferentes tecnologias, como Wi-Fi, 5G, LoRaWAN e Bluetooth;
- **Camada de aplicação**, na qual os dados coletados são processados e transformados

em serviços e aplicações voltadas ao usuário final.

Apesar de seus benefícios, a adoção da IoT impõe desafios à construção de sistemas baseados nessas arquiteturas. O volume massivo de dados gerados requer soluções eficientes para armazenamento, processamento e análise, tornando indispensável a integração com técnicas de IA e aprendizado de máquina (RUDENKO et al., 2022). Essas técnicas possibilitam a extração de conhecimento a partir dos dados, a identificação de padrões e o suporte à tomada de decisão em tempo real.

Outro aspecto crítico refere-se à segurança e à privacidade. A comunicação entre dispositivos conectados amplia a superfície de ataque, expondo vulnerabilidades que podem comprometer tanto informações sensíveis quanto a disponibilidade dos serviços. Nesse sentido, pesquisas recentes têm investigado abordagens que conciliem escalabilidade, confiabilidade e proteção de dados, elementos fundamentais para a consolidação da IoT como uma infraestrutura tecnológica de larga escala (WAKILI; BAKKALI, 2025; SUN et al., 2025).

Por fim, destaca-se que a IoT não se limita a uma tendência tecnológica, mas atua como um catalisador de inovação, viabilizando a construção de sistemas autônomos, adaptativos e inteligentes, capazes de atender às exigências de ambientes altamente dinâmicos e complexos(TANG; QIN, 2023).

2.2 Arquiteturas de *Software* autoadaptativas

A necessidade de adaptação contínua em sistemas de *software* decorre da variabilidade do ambiente de execução, da evolução dos requisitos e das mudanças nas condições operacionais (HEINZ et al., 2018). Para atender a esse contexto, surgem as arquiteturas de *software* autoadaptativas, cujo objetivo é permitir que o próprio sistema identifique variações relevantes e ajuste seu comportamento de forma autônoma. Dessa maneira, a adaptação contínua é viabilizada por mecanismos que observam o estado do sistema, avaliam suas condições e executam ações corretivas ou evolutivas ao longo do tempo (KEPHART; CHESS, 2003).

Segundo Heinz et al. (2018), uma arquitetura autoadaptativa preserva os atribu-

tos de qualidade do sistema por meio de mecanismos de monitoramento, análise e reconfiguração dinâmica, assegurando o atendimento aos requisitos funcionais e não funcionais mesmo em cenários variáveis.

Esse conceito é comumente representado pelo ciclo *MAPE-K* (*Monitor, Analyze, Plan, Execute – Knowledge*), que organiza o processo de adaptação em etapas sistemáticas e interdependentes. Essa estrutura possibilita que os sistemas reajam de maneira autônoma a alterações ambientais ou operacionais.

Conforme ilustrado na Figura 2.1, o ciclo inicia-se na etapa de Monitoramento (*Monitor*), responsável pela coleta contínua de dados sobre o estado do sistema e do ambiente. Em seguida, na etapa de Análise (*Analyze*), os dados coletados são avaliados com o objetivo de identificar desvios, tendências ou violações de requisitos previamente estabelecidos. Com base nessa análise, a etapa de Planejamento (*Plan*) define estratégias de adaptação, selecionando ações capazes de restaurar ou manter os objetivos do sistema. Posteriormente, na etapa de Execução (*Execute*), essas ações são aplicadas por meio de mecanismos de reconfiguração ou ajuste do comportamento do sistema.

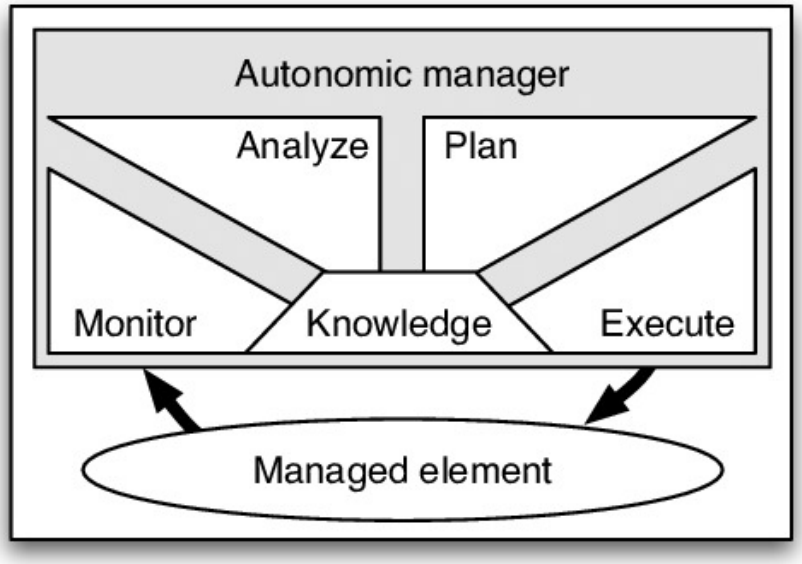

Figura 2.1: Ciclo de adaptação MAPE-K

Fonte: (KEPHART; CHESS, 2003) e (GORLA et al., 2010).

Estudos realizados por Kephart e Chess (2003) e Heinz et al. (2018) indicam que arquiteturas autoadaptativas contribuem para a manutenção da confiabilidade, escalabi-

lidade e autonomia, especialmente em ambientes caracterizados por fluxos contínuos de dados e alta variabilidade contextual. No entanto, para que a adaptação ocorra de maneira sistemática, o sistema deve dispor de mecanismos capazes de analisar dados e apoiar a tomada de decisão.

Nesse contexto, Djennadi et al. (2024) propõem a reconfiguração adaptativa de rotas em edifícios inteligentes por meio de redes definidas por *software* (*Software-Defined Networking – SDN*). A abordagem baseia-se no monitoramento do estado da infraestrutura, na análise das condições operacionais e na execução automática de reconfigurações fundamentadas em políticas previamente definidas.

De forma semelhante, Lam, Haugen e Delsing (2022) discutem a orquestração dinâmica de sistemas industriais baseados em IoT, nos quais a arquitetura observa o comportamento dos dispositivos, avalia o estado do sistema e reconfigura serviços conforme as condições operacionais, caracterizando um ciclo contínuo de decisão e execução voltado à adaptação.

Por sua vez, Dias, Restivo e Ferreira (2021) apresentam mecanismos de autorreparação (*self-healing*) que permitem ao sistema identificar falhas, definir ações de recuperação e aplicar correções de forma autônoma. Esses mecanismos incorporam ciclos de monitoramento, análise e execução, voltados à manutenção do funcionamento do sistema diante de falhas ou degradações.

2.3 Uso da IA com IoT

O mecanismo de decisão requerido por arquiteturas de *software* autoadaptativas pode ser implementado por meio de técnicas de IA. Nesse contexto, a integração entre IA e a IoT possibilita que sistemas baseados em dispositivos conectados não apenas coletem grandes volumes de dados, mas também analisem essas informações e tomem decisões durante a execução do sistema (AVILA et al., 2022). Dessa forma, a adaptação deixa de depender exclusivamente de regras estáticas e passa a ser orientada por dados observados ao longo do tempo (HALLOU et al., 2024).

Segundo Ludermir (2021), técnicas como aprendizado de máquina, redes neurais e algoritmos de classificação têm sido amplamente aplicadas para otimizar processos em

diferentes domínios da IoT, incluindo monitoramento ambiental, manutenção preditiva, gestão energética e controle de tráfego. Essas abordagens contribuem para a construção de sistemas mais eficientes, com maior capacidade de resposta às condições dinâmicas do ambiente.

Além disso, a IA desempenha papel relevante na melhoria da segurança e da privacidade em redes IoT. Algoritmos de detecção de anomalias, por exemplo, são empregados para identificar comportamentos suspeitos e mitigar ataques cibernéticos, conforme discutido por Soares et al. (2024). A análise inteligente dos dados também possibilita a identificação e o tratamento diferenciado de informações sensíveis, reduzindo riscos associados à exposição indevida.

Adicionalmente, a aplicação de técnicas de IA em arquiteturas autoadaptativas amplia a capacidade dos sistemas de reconfigurar seus componentes de forma autônoma, com base em evidências extraídas dos dados (LEE; LEE; SEO, 2022; AMIRI; ZDUN, 2023). Essa integração entre IA e IoT contribui para o desenvolvimento de soluções alinhadas às exigências de ambientes distribuídos e dinâmicos. Entretanto, a simples utilização de modelos de IA não assegura, por si só, a manutenção do desempenho e da confiabilidade ao longo do tempo, tornando necessário o gerenciamento desses modelos durante todo o seu ciclo de vida em produção, a fim de garantir sua adequação contínua às condições do ambiente e aos objetivos do sistema.

2.4 MLOps

Com a crescente adoção de modelos de aprendizado de máquina em ambientes produtivos, especialmente em sistemas complexos e distribuídos, torna-se necessário o gerenciamento estruturado de todo o ciclo de vida desses modelos (LAKSHMANAN; ROBINSON; MUNN, 2020). Nesse contexto, surge a abordagem de MLOps (*Machine Learning Operations*), que reúne práticas voltadas à integração entre o desenvolvimento de modelos de aprendizado de máquina e as operações de *software* (TREVEIL et al., 2020). O objetivo do MLOps é organizar e automatizar as etapas que abrangem desde a preparação dos dados até a implantação e o monitoramento contínuo dos modelos em produção (KREUZBERGER; KÜHL; HIRSCHL, 2023). Essa abordagem combina princípios do DevOps com

técnicas de Ciência de Dados, contribuindo para o aumento do controle, da rastreabilidade e da colaboração entre as equipes envolvidas (KODAKANDLA, 2024).

Conforme apontado por Kreuzberger, Kühl e Hirschl (2023), o MLOps incorpora metodologias de integração e entrega contínuas (CI/CD), que facilitam a atualização de modelos em ambientes produtivos e reduzem o tempo necessário para validar e disponibilizar novas versões. Essas práticas apresentam especial relevância em sistemas de IoT, nos quais os dados são gerados em tempo real e demandam respostas rápidas e adaptativas.

Além disso, o MLOps apoia a manutenção da qualidade dos modelos ao longo do tempo por meio do monitoramento de desempenho, da identificação de desvios e do retreinamento automatizado (KODAKANDLA, 2024; BAYRAM; ALTILAR, 2021). Em ambientes distribuídos, essa abordagem viabiliza a construção de *pipelines* que sustentam aplicações inteligentes com maior estabilidade e eficiência.

A implementação das práticas de MLOps pode ocorrer de forma manual ou automatizada, conforme a maturidade dos processos e a infraestrutura disponível (KREUZBERGER; KÜHL; HIRSCHL, 2023). Quando realizada manualmente, cada etapa do ciclo de vida dos modelos de aprendizado de máquina empregados em sistemas de IoT, como coleta de dados, treinamento, validação, implantação e monitoramento, exige intervenção direta das equipes técnicas (TREVEIL et al., 2020). Essa abordagem demanda maior esforço operacional, aumenta o risco de inconsistências e dificulta a rastreabilidade das versões dos modelos (LAKSHMANAN; ROBINSON; MUNN, 2020). Além disso, a replicação de experimentos e a resposta a mudanças nos dados tornam-se mais lentas, o que pode comprometer a capacidade de adaptação de sistemas de IoT em cenários de variação contínua dos dados (BAYRAM; ALTILAR, 2021).

Por outro lado, a automação dos *pipelines* de aprendizado de máquina permite integrar as etapas de coleta e preparação de dados, treinamento, validação, implantação e monitoramento dos modelos utilizados no ambiente de produção em fluxos contínuos e auditáveis, reduzindo a dependência de intervenções manuais. A utilização de ferramentas de CI/CD, monitoramento e orquestração de *workflows* contribui para a atualização sistemática dos modelos, o retreinamento com novos dados e a detecção de degradação de desempenho. Essa abordagem automatizada de MLOps favorece a escalabilidade das

soluções, a padronização dos processos e a redução do tempo entre o desenvolvimento e a disponibilização em produção (TREVEIL et al., 2020).

2.4.1 Data Drift

No contexto de *pipelines* automatizados de MLOps, a manutenção do desempenho de modelos de aprendizado de máquina em produção está diretamente relacionada à estabilidade dos dados utilizados durante a fase de inferência. Em sistemas de IoT, nos quais os dados são gerados de forma contínua e refletem ambientes sujeitos a mudanças frequentes, é comum que as características estatísticas desses dados se modifiquem ao longo do tempo, dando origem ao fenômeno conhecido como *data drift* (GAMA et al., 2014).

O *data drift* refere-se à alteração na distribuição dos dados de entrada observados em produção em relação à distribuição dos dados utilizados na fase de treinamento do modelo (WIDMER; KUBAT, 1996). Essas mudanças podem ocorrer em função de variações nas condições ambientais, no comportamento dos usuários, no funcionamento de sensores ou na dinâmica operacional do sistema. Como consequência, os dados processados passam a representar padrões distintos daqueles considerados durante o treinamento, o que pode impactar negativamente o desempenho e a confiabilidade do modelo.

Em ambientes de IoT, o *data drift* constitui um desafio recorrente, uma vez que os fluxos de dados são contínuos, distribuídos e sujeitos a variações temporais. Mesmo quando a relação entre as variáveis de entrada e saída permanece inalterada, alterações na distribuição dos dados podem comprometer a capacidade de generalização do modelo. Nesse contexto, a detecção e o monitoramento de *data drift* constituem elementos centrais das práticas de MLOps, permitindo avaliar continuamente a adequação dos dados utilizados em produção.

A incorporação de mecanismos de monitoramento de *data drift* possibilita a adoção de estratégias de adaptação, como o retreinamento de modelos, o ajuste de parâmetros ou a substituição controlada de modelos em produção. Essas estratégias podem ser integradas a abordagens como o padrão *Champion/Challenger* e práticas de *Human-in-the-Loop*, nas quais decisões automatizadas são validadas ou complementadas por especialistas. Dessa forma, o gerenciamento do *data drift* contribui para a manutenção

da qualidade, da robustez e da confiabilidade dos modelos em ambientes distribuídos e dinâmicos.

2.4.2 Concept Drift

Enquanto o *data drift* diz respeito à mudança na distribuição dos dados de entrada, o *concept drift* representa um desafio ainda mais crítico para a confiabilidade de sistemas inteligentes: a alteração na relação fundamental entre as variáveis de entrada e a variável alvo. Em outras palavras, o "conceito"que o modelo aprendeu, regra de decisão que mapeia X para Y, deixa de ser válido devido a mudanças na dinâmica do ambiente real (GAMA et al., 2014).

Diferentemente de falhas de *software* tradicionais, o *concept drift* frequentemente ocorre de maneira "silenciosa". O modelo continua recebendo dados e gerando predições sem erros de execução, porém, a acurácia dessas predições degrada-se progressivamente, uma vez que a fronteira de decisão aprendida no treinamento não reflete mais a realidade atual (WIDMER; KUBAT, 1996; LU et al., 2018). Em cenários de IoT, onde os dados são inerentemente não estacionários, essa obsolescência do modelo é praticamente inevitável se não houver mecanismos de adaptação.

A literatura classifica esse fenômeno conforme a velocidade e o comportamento da mudança:

- **Drift Abrupto:** Mudança repentina no conceito, comum em falhas de sensores ou alterações drásticas no processo monitorado;
- **Drift Gradual:** Transição lenta onde o conceito antigo e o novo coexistem por um período;
- **Drift Incremental:** Pequenas variações contínuas que acumulam desvios ao longo do tempo;
- **Drift Recorrente:** Reaparecimento de conceitos antigos, típico de padrões sazonais.

A persistência de modelos estáticos em ambientes sujeitos a *concept drift* compromete diretamente a utilidade da solução de IoT. Portanto, a detecção desse fenômeno

exige mais do que apenas monitorar a distribuição dos dados; requer a avaliação contínua da incerteza do modelo e da sua performance em relação a novos dados rotulados ou *feedbacks* do ambiente (BAYRAM; ALTILAR, 2021). É nesse contexto que se torna imperativo adotar mecanismos arquiteturais capazes não apenas de identificar a degradação, mas de operacionalizar a substituição segura dos modelos afetados por novas versões mais aderentes ao conceito atual, utilizando abordagens consolidadas de MLOps.

2.4.3 Padrão *Champion/Challenger*

O padrão *Champion/Challenger* consiste em uma estratégia de MLOps voltada à avaliação contínua de modelos de aprendizado de máquina em ambientes sujeitos a variações nos dados. Nesse contexto, o *champion* corresponde ao modelo atualmente implantado em produção, enquanto o *challenger* refere-se a um ou mais modelos alternativos avaliados em paralelo. A substituição do modelo ocorre quando um *challenger* apresenta desempenho superior ao *champion*, de acordo com métricas previamente definidas e critérios de validação estabelecidos.

Essa abordagem tem sido aplicada em sistemas de IoT com o objetivo de preservar a qualidade dos modelos diante da variabilidade dos dados e das condições operacionais, conforme discutido por Hutter, Kotthoff e Vanschoren (2019). A execução simultânea de múltiplos modelos, característica do padrão *Champion/Challenger*, permite a comparação de desempenho sob as mesmas condições de operação, contribuindo para decisões mais consistentes em sistemas distribuídos, como aqueles baseados em sensores e redes inteligentes.

Além disso, o padrão *Champion/Challenger* integra-se às práticas de MLOps ao apoiar a automação dos ciclos de validação, promoção e substituição de modelos em ambientes produtivos, conforme destacado por Kodakandla (2024). Essa integração favorece a detecção e a mitigação de deriva de dados, contribuindo para a estabilidade de sistemas que operam de forma contínua.

Em aplicações que envolvem *deep learning*, estratégias baseadas em múltiplos modelos concorrentes podem ser combinadas com mecanismos de adaptação à deriva conceitual (WANG et al., 2020). Essa combinação amplia a capacidade de resposta dos sistemas

frente a cenários complexos e ao processamento de dados não estruturados (RESTUCCIA; MELODIA, 2020).

Dessa forma, o padrão *Champion/Challenger* favorece a construção de *pipelines* adaptativos capazes de sustentar aplicações em ambientes distribuídos e dinâmicos, atendendo às exigências operacionais da IoT e às práticas contemporâneas de engenharia de aprendizado de máquina (TREVEIL et al., 2020; KREUZBERGER; KÜHL; HIRSCHL, 2023). Além disso, essa abordagem fornece a estrutura técnica necessária para mitigar os efeitos do *concept drift*, permitindo a manutenção da acurácia por meio da competição contínua entre modelos. Contudo, a automação integral da substituição de modelos em domínios sensíveis pode introduzir riscos operacionais quando as decisões se baseiam exclusivamente em métricas estatísticas. Nesse contexto, torna-se fundamental integrar mecanismos de supervisão que assegurem a robustez e a confiabilidade do processo de adaptação.

2.4.4 Human-in-the-Loop (HITL)

O conceito de *Human-in-the-Loop* (HITL) refere-se à integração da intervenção humana em processos automatizados de aprendizado de máquina, especialmente em etapas críticas de monitoramento, validação e tomada de decisão (AMERSHI et al., 2014; WU et al., 2024). Nessa abordagem, operadores ou especialistas são acionados para validar, corrigir ou complementar as predições realizadas pelo sistema, garantindo maior confiabilidade em cenários caracterizados por incerteza, ambiguidade ou condições não previstas durante o treinamento dos modelos (CIGNARELLA et al., 2023; ZHAO et al., 2023).

Em ambientes de MLOps, o HITL atua como um componente de supervisão contínua, operando em conjunto com métricas automatizadas de desempenho, como o HI. Quando essas métricas indicam risco de *data drift* ou redução na confiança das predições, o sistema pode emitir alertas e solicitar a intervenção humana antes da execução de ações automáticas, como retreinamento, chaveamento ou substituição de modelos (WU et al., 2024). Essa estratégia contribui para que decisões críticas não dependam exclusivamente de mecanismos automatizados, promovendo um equilíbrio entre autonomia do sistema e controle humano (AMERSHI et al., 2014).

A aplicação do HITL mostra-se particularmente relevante em sistemas críticos ou que lidam com dados sensíveis, como aqueles inseridos nos domínios da saúde, das finanças e da IoT. Nesses contextos, decisões incorretas podem acarretar impactos operacionais, éticos ou econômicos significativos, reforçando a necessidade de validação humana em pontos estratégicos do processo (DENG; WANG; LI, 2023).

Além de aumentar a segurança e a confiabilidade dos sistemas, o HITL contribui para a melhoria contínua dos modelos de aprendizado de máquina. O *feedback* oriundo da intervenção humana pode ser incorporado aos ciclos de atualização e adaptação dos *pipelines* de MLOps, promovendo o refinamento progressivo dos modelos e fortalecendo a robustez dos processos de automação (WU et al., 2024; CIGNARELLA et al., 2023).

3 Trabalhos Relacionados

O MSL consiste em uma abordagem metodológica estruturada voltada à organização, classificação e análise da produção científica sobre um determinado tema. Seu objetivo principal é identificar lacunas, tendências e oportunidades de pesquisa em uma área de conhecimento específica (COELHO; DERMEVAL, 2019).

A realização do MSL neste trabalho teve como objetivo compreender o estado da arte sobre estratégias de detecção e mitigação de *concept drift* em sistemas auto-adaptativos, bem como identificar abordagens aplicáveis ao contexto da IoT.

Para isso, este capítulo descreve de forma detalhada as etapas conduzidas durante o processo de mapeamento, que incluem o planejamento, a formulação das questões de pesquisa, a definição dos critérios de inclusão e exclusão e a estratégia de busca adotada para a seleção dos estudos relevantes.

A etapa inicial do MSL consistiu na elaboração de um protocolo de pesquisa, responsável por orientar todas as fases subsequentes do estudo. Esse protocolo contemplou os seguintes elementos:

- Objetivo do mapeamento;
- Questões de pesquisa (QPs);
- Fontes de dados selecionadas;
- *String* de busca utilizada;
- Critérios de inclusão e exclusão.

Com o intuito de ampliar a cobertura da revisão, foram selecionadas as bases de dados Scopus e IEEE Xplore. A Scopus, de caráter multidisciplinar, reúne publicações em áreas como ciência da computação, engenharia e IA. A IEEE Xplore, por sua vez, é especializada em engenharia elétrica, ciência da computação e tecnologias emergentes, sendo frequentemente utilizada em estudos sobre IoT, arquiteturas de *software* e aprendizado de

máquina. A combinação dessas duas bases visa reduzir possíveis vieses de seleção e ampliar a representatividade dos estudos incluídos, favorecendo a incorporação de pesquisas relevantes, tanto em periódicos quanto em anais de conferências.

Para apoiar a execução e o gerenciamento do processo, foi utilizada a ferramenta PARSIFAL (PARSIFAL, 2021), que auxiliou na organização dos resultados e na aplicação sistemática do protocolo definido.

3.1 Questões de Pesquisa

As questões de pesquisa foram formuladas com o objetivo de orientar a análise dos estudos primários e manter o alinhamento com os objetivos do mapeamento. A Tabela 3.1 apresenta as questões definidas, sendo a QP a questão central, que fornece o contexto para as demais.

Tabela 3.1: Questões de Pesquisa

Identificador	**Tipo de questão**	**Descrição**
QP	Questão Principal	Como apoiar a construção de arquiteturas autoadaptativas que utilizam técnicas de IA?
QP1	Questão Secundária	Quais são os principais desafios enfrentados no desenvolvimento dessas arquiteturas?
QP2	Questão Secundária	Como a integração de padrões de projetos pode viabilizar a adaptação contínua em arquiteturas de IoT?
QP3	Questão Secundária	Quais métodos de avaliação são utilizados na literatura para validar a eficácia de arquiteturas autoadaptativas no tratamento de *concept drift*?

Fonte: Elaborado pelo autor.

3.2 Critérios de Inclusão e Exclusão

Com base nos objetivos do estudo, foram definidos critérios específicos para a seleção dos estudos. As Tabelas 3.2 e 3.3 apresentam, respectivamente, os critérios de inclusão e exclusão adotados.

Tabela 3.2: Critérios de Inclusão

Identificador	**Descrição**
CI1	Estudos que abordem arquiteturas de software autoadaptativas em cenários de IoT, com foco no uso de IA para suporte à adaptação. O trabalho deve discutir desafios de desenvolvimento, propor ou utilizar padrões arquiteturais e/ou *frameworks* para adaptação, ou apresentar métodos de avaliação relacionados à eficácia da adaptação (como tratamento de *concept drift*).

Fonte: Elaborado pelo autor.

Tabela 3.3: Critérios de Exclusão

Identificador	**Descrição**
CE1	Estudos não escritos em inglês ou português.
CE2	Estudos cujo texto completo não esteja disponível.
CE3	Publicações que não se caracterizam como artigos científicos primários.
CE4	Estudos substituídos por versões mais recentes do mesmo trabalho.

Fonte: Elaborado pelo autor.

3.3 Estratégia de Busca

A definição da estratégia de busca teve como objetivo assegurar a identificação abrangente e sistemática dos estudos primários relevantes ao tema desta pesquisa. Para isso, a construção da *string* de busca foi conduzida de forma estruturada, a partir da decomposição da questão de pesquisa em seus conceitos centrais, utilizando uma adaptação do *framework* PICOC (Population, Intervention, Comparison, Outcome e Context), amplamente empregado em estudos de mapeamento e revisão sistemática da literatura (HIGGINS; GREEN,

2011).

O uso do *framework* PICOC possibilitou organizar os principais elementos conceituais do estudo, assegurando coerência entre os objetivos da pesquisa e os termos utilizados na busca. A Tabela 3.4 apresenta os elementos do PICOC considerados neste trabalho e sua respectiva aplicação no contexto da pesquisa.

Tabela 3.4: Elementos do *framework* PICOC utilizados na estratégia de busca

Elemento	**Descrição**	**Aplicação neste estudo**
P	População	Sistemas de IoT
I	Intervenção	Análise de modelos e arquiteturas que permitem adaptação dinâmica em tempo de execução
C	Comparação	–
O	Resultado	Arquiteturas que promovem melhorias de eficiência, desempenho ou adaptabilidade dos sistemas
C	Contexto	Arquitetura de *Software*

Fonte: Elaborado pelo autor.

A partir desses eixos conceituais, foram definidos termos-chave, sinônimos e variações terminológicas associadas a cada elemento do PICOC, com o objetivo de ampliar a abrangência da busca e capturar estudos relevantes que utilizassem diferentes nomenclaturas. A combinação desses termos resultou na *string* de busca final apresentada a seguir:

```
((‘‘self-adaptive system*’’ OR ‘‘self-adaptiv*’’ OR ‘‘adaptive software’’
OR ‘‘autonomic computing’’ OR ‘‘runtime adaptation’’ OR ‘‘dynamic
reconfiguration’’ OR ‘‘AI-Assisted Software Architecture’’) AND (‘‘Internet
of Things’’ OR ‘‘IoT’’) AND (‘‘software arch*’’ OR ‘‘software design’’
OR ‘‘architectural pattern*’’ OR ‘‘framework’’))
```

A *string* foi aplicada diretamente nas bases de dados Scopus e IEEE Xplore, sem a necessidade de adaptações específicas para cada plataforma, uma vez que ambas suportam operadores booleanos e consultas complexas. A escolha dessas bases justifica-se

por sua ampla cobertura de publicações científicas nas áreas de ciência da computação, engenharia de *software*, IoT e IA.

A avaliação da *string* de busca foi realizada por meio da utilização de artigos de controle, previamente selecionados por sua relevância e aderência ao domínio da pesquisa. Esse procedimento teve como finalidade verificar se a *string* era capaz de recuperar tais estudos nas bases selecionadas, assegurando que os principais trabalhos relacionados ao tema fossem contemplados pela estratégia de busca adotada.

3.4 Procedimentos de Busca e Análise dos Estudos

As buscas nas bases de dados foram conduzidas durante o primeiro semestre de 2025. Inicialmente, a aplicação da *string* de busca detalhada na seção anterior resultou na identificação de 262 estudos na base Scopus e 187 estudos na IEEE Xplore, totalizando 449 publicações relacionadas a propostas de arquiteturas de *software* autoadaptativas aplicadas a sistemas de IoT. Após a remoção das duplicatas, restaram 366 artigos únicos. O processo de busca e seleção dos estudos foi conduzido em três etapas sequenciais.

Na primeira etapa, foram analisados os títulos, resumos e palavras-chave dos 366 artigos identificados. Com base nos critérios de exclusão previamente definidos (ver Tabela 3.3), 305 estudos foram eliminados, resultando na seleção de 61 artigos para a fase seguinte.

Na segunda etapa, realizou-se a leitura dos títulos, resumos, introduções e conclusões dos 61 artigos remanescentes. A partir da reaplicação dos critérios de inclusão e exclusão, 38 estudos foram excluídos, sendo a indisponibilidade de acesso ao texto completo um fator adicional para eliminação. Ao final dessa etapa, 23 artigos avançaram para a fase final de análise.

Na terceira etapa, procedeu-se à leitura integral dos 23 estudos selecionados. Após nova verificação dos critérios estabelecidos, todos os 23 artigos foram considerados relevantes e incluídos na análise final do mapeamento sistemático.

A Figura 3.1 apresenta uma visão geral do processo de busca, filtragem e seleção dos estudos, conforme o protocolo PRISMA.

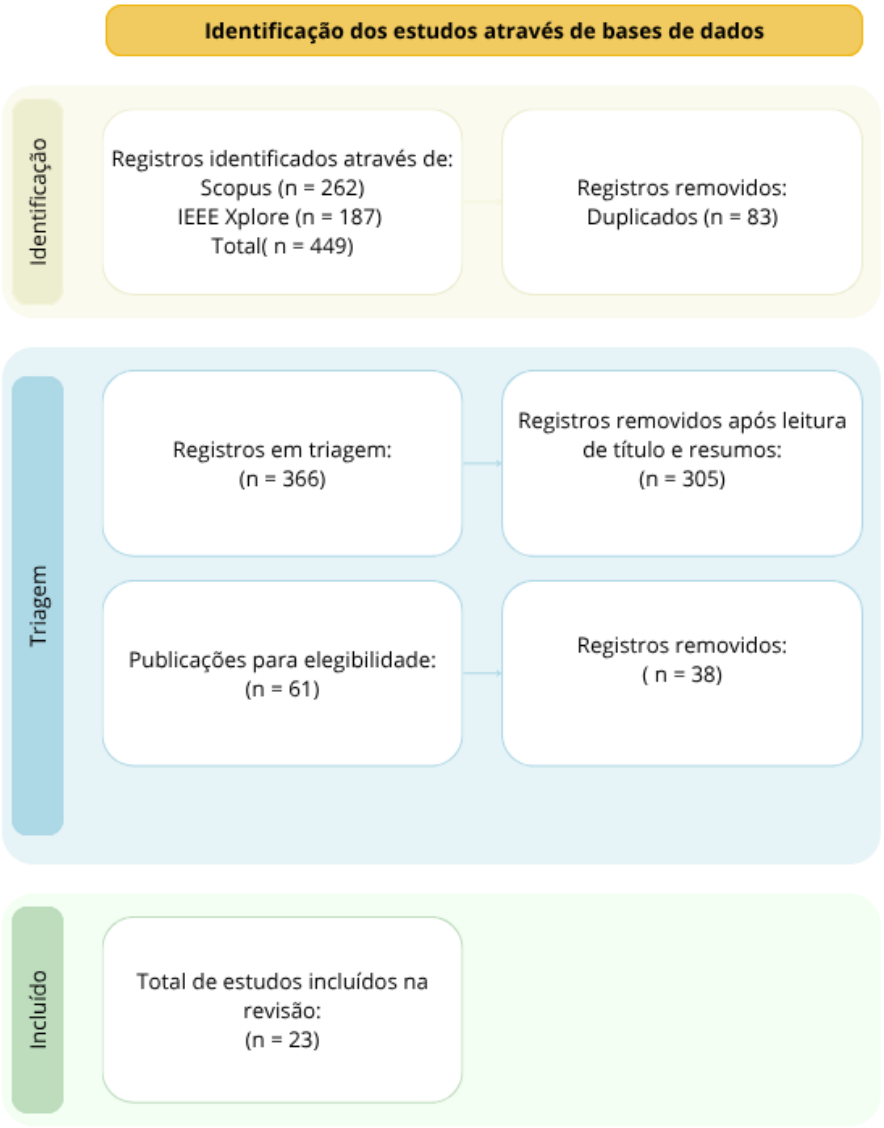

Figura 3.1: Fluxograma do protocolo PRISMA

Fonte: Elaborado pelo autor.

3.5 Resultados e Discussões

Esta seção apresenta os principais achados obtidos a partir da análise dos estudos selecionados, os quais estão listados na Tabela 3.5. Os 23 artigos incluídos atenderam aos critérios de inclusão definidos no protocolo do mapeamento sistemático da literatura e compõem a base empírica utilizada para responder às questões de pesquisa estabelecidas.

Tabela 3.5: Estudos incluídos na fase de análise

ID	**Autor(es)**	**Título**	**Ano**
E1	HALLOU, Amal et al.	*Context-Aware IoT System Development Approach Based on Meta-Modeling and Reinforcement Learning: A Smart Home Case Study.*	2024
E2	HACHICHA, Marwa; BEN HALIMA, Riadh; HADJ KACEM, Ahmed.	*Modeling and specifying formally compound MAPE pattern for self-adaptive IoT systems.*	2022
E3	ALKHABBAS, Fahed et al.	*Assert: A blockchain-based architectural approach for engineering secure self-adaptive IoT systems.*	2022
E4	LEE, Euijong; LEE, Sukhoon; SEO, Young-Duk.	*Deep learning based self-adaptive framework for environmental interoperability in IoT.*	2022
E5	DI MENNA, Federico; MUCCINI, Henry; VAIDHYANATHAN, Karthik.	*FEAST: a framework for evaluating implementation architectures of self-adaptive IoT systems.*	2022
E6	GULDNER, Achim et al.	*A framework for AI-based self-adaptive cyber-physical process systems.*	2023
E7	KARADUMAN, Burak; TEZEL, Baris Tekin; CHALLENGER, Moharram.	*Enhancing BDI agents using fuzzy logic for CPS and IoT interoperability using the JaCa platform.*	2022

Continua na próxima página...

Tabela 3.5 – continuação da página anterior

ID	Autor(es)	Título	Ano
E8	FONSECA, Adrilene et al.	*Dealing with IoT defiant components.*	2021
E9	DJENNADI, Liticia et al.	*SDN-based approach for adaptive reconfiguration of routing in IoT for smart buildings.*	2024
E10	LAM, An Ngoc; HAUGEN, Oystein; DELSING, Jerker.	*Dynamical orchestration and configuration services in industrial IoT systems: An autonomic approach.*	2022
E11	HE, Xing et al.	*Redefinition of digital twin and its situation awareness framework designing toward fourth paradigm for energy internet of things.*	2024
E12	NIKKHAH, Shayan Tabatabaei et al.	*A Deployment Framework for Quality-Sensitive Applications in Resource-Constrained Dynamic Environments.*	2021
E13	TANG, Lin; QIN, Hang.	*Divisible task offloading for multiuser multiserver mobile edge computing systems based on deep reinforcement learning.*	2023
E14	AMIRI, Amirali; ZDUN, Uwe.	*Smart and Adaptive Routing Architecture: An Internet-of-Things Traffic Manager Based on Artificial Neural Networks.*	2023
E15	RULLO, Antonino et al.	*Kalis2.0 — A SECaaS-Based Context-Aware Self-Adaptive Intrusion Detection System for IoT.*	2023
E16	WANG, Xiaofei et al.	*Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching.*	2020

Continua na próxima página...

Tabela 3.5 – continuação da página anterior

ID	Autor(es)	Título	Ano
E17	RESTUCCIA, Francesco; MELODIA, Tommaso.	*DeepWiERL: Bringing deep reinforcement learning to the internet of self-adaptive things.*	2020
E18	BASSENE, Avewe; GUEYE, Bamba.	*A self-adaptive QoS-management framework for highly dynamic IoT networks.*	2022
E19	FRANGOUDIS, Pantelis A.; REISINGER, Matthias; DUSTDAR, Schahram.	*Recursive design for data-driven, self-adaptive IoT services.*	2021
E20	DIAS, Joao Pedro; RESTIVO, André; FERREIRA, Hugo Sereno.	*Empowering visual Internet-of-Things mashups with self-healing capabilities.*	2021
E21	DURÁN, Francisco et al.	*Seamless reconfiguration of rule-based IoT applications.*	2021
E22	XIAO, Wenjing et al.	*Collaborative cloud-edge service cognition framework for DNN configuration toward smart IIoT.*	2021
E23	AUDRITO, Giorgio.	*FCPP: an efficient and extensible field calculus framework.*	2020

Fonte: Elaborado pelo autor.

De modo geral, os estudos analisados abordam diferentes estratégias para o desenvolvimento de arquiteturas de *software* autoadaptativas aplicadas à IoT, contemplando tanto a organização arquitetural dos sistemas quanto os mecanismos empregados para adaptação em tempo de execução. As propostas diferem quanto às técnicas utilizadas, aos domínios de aplicação e aos requisitos considerados.

Em primeiro lugar, parte dos trabalhos utiliza técnicas de IA, como Aprendizado

de Máquina, aprendizado profundo e aprendizado por reforço, para apoiar processos de tomada de decisão e reconfiguração do sistema (E1, E4, E9, E13, E16, E17, E22) (HALLOU et al., 2024; LEE; LEE; SEO, 2022; DJENNADI et al., 2024; TANG; QIN, 2023; WANG et al., 2020; RESTUCCIA; MELODIA, 2020; XIAO et al., 2021). Nesses estudos, a adaptação é orientada pela análise dos dados coletados do ambiente, possibilitando ajustes no comportamento do sistema de acordo com as condições operacionais.

Em seguida, observa-se um conjunto de propostas fundamentadas em padrões e *frameworks* arquiteturais, com destaque para o uso do ciclo MAPE-K, arquiteturas orientadas a agentes e modelos formais de especificação (E2, E7, E10, E18, E19, E21) (HACHICHA; HALIMA; KACEM, 2022; KARADUMAN; TEZEL; CHALLENGER, 2022; LAM; HAUGEN; DELSING, 2022; BASSENE; GUEYE, 2022; FRANGOUDIS; REISINGER; DUSTDAR, 2021; DURAN et al., 2021). Esses trabalhos estruturam a adaptação por meio da separação entre monitoramento, análise, planejamento e execução, o que contribui para a organização do processo adaptativo.

Além disso, alguns estudos direcionam a adaptação para o atendimento de requisitos não funcionais, como segurança, confiabilidade, qualidade de serviço e tolerância a falhas. Nesse grupo, incluem-se propostas voltadas à detecção de intrusões e proteção de dados (E3, E15) (ALKHABBAS et al., 2022; RULLO et al., 2023), bem como mecanismos de autorreparação e tratamento de falhas (E8, E20).

Por outro lado, há trabalhos que investigam arquiteturas voltadas a ambientes distribuídos e heterogêneos, envolvendo computação em nuvem, *edge computing* e sistemas ciberfísicos (E6, E10, E16, E22) (GULDNER et al., 2023; LAM; HAUGEN; DELSING, 2022; WANG et al., 2020; XIAO et al., 2021). Esses estudos consideram a limitação de recursos, a descentralização do processamento e a variabilidade dos dados como fatores relevantes para a adaptação arquitetural.

De forma integrada, a análise dos estudos apresentados na Tabela 3.5 indica que a literatura combina propostas voltadas à definição de estruturas arquiteturais com mecanismos técnicos de adaptação. Essa combinação evidencia a necessidade de abordagens que articulem arquitetura de *software*, IA e gerenciamento de sistemas em ambientes de IoT.

A Figura 3.2 apresenta a distribuição dos artigos incluídos de acordo com o ano de publicação. Observa-se que 2021 concentra o maior número de estudos, indicando um aumento das investigações sobre arquiteturas de *software* autoadaptativas aplicadas à IoT nesse período. Essa recorrência pode ser associada à consolidação de plataformas de IoT e à ampliação do uso de técnicas de adaptação em tempo de execução, motivadas pela necessidade de lidar com ambientes distribuídos, heterogêneos e sujeitos a variações nos dados. Além disso, esse período coincide com um maior interesse da literatura em estruturar soluções arquiteturais que integrem mecanismos de monitoramento, reconfiguração e gerenciamento de qualidade de serviço em sistemas de IoT.

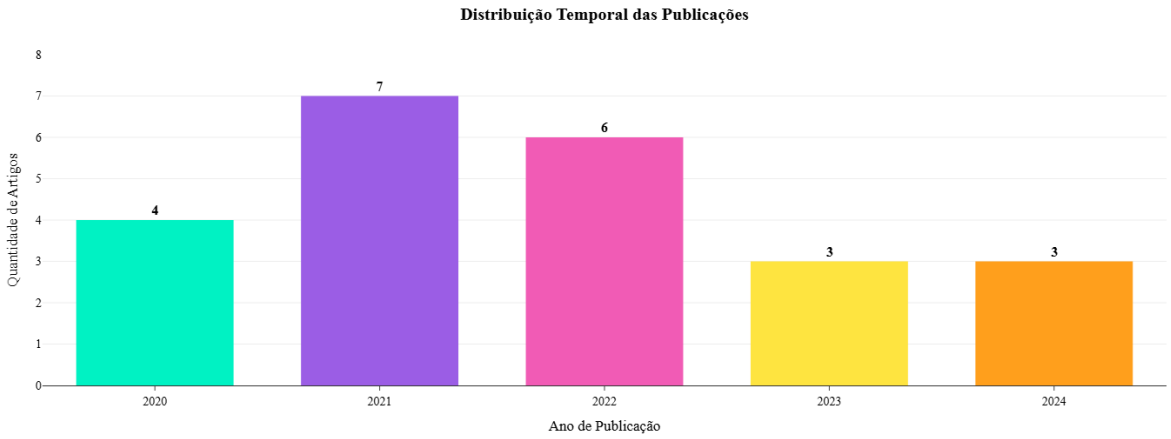

Figura 3.2: Gráfico de quantidade de artigos publicados por ano

Fonte: Elaborado pelo autor.

Com base no protocolo de revisão adotado, os trabalhos foram organizados segundo as questões de pesquisa definidas, de modo a possibilitar uma análise estruturada das contribuições, lacunas e tendências identificadas.

A seguir, cada questão de pesquisa é discutida individualmente, à luz das evidências extraídas da literatura.

QP — Como apoiar a construção de arquiteturas autoadaptativas, que utilizam técnicas de IA, através do monitoramento de modelos?

Com base nos 23 estudos incluídos, observa-se que o monitoramento contínuo de modelos assume papel relevante na construção de arquiteturas autoadaptativas que utilizam técnicas de IA. Esse monitoramento estabelece a relação entre o comportamento

esperado do sistema e as variações observadas em tempo de execução, fornecendo subsídios para decisões de adaptação em arquiteturas de IoT. Evidências na literatura indicam que a articulação entre mecanismos de análise e reconfiguração permite que os sistemas respondam a mudanças no ambiente operacional e nos fluxos de dados (HALLOU et al., 2024; GULDNER et al., 2023; LAM; HAUGEN; DELSING, 2022).

Apesar dessas contribuições, os estudos analisados tendem a concentrar-se em soluções específicas de monitoramento ou em mecanismos pontuais de adaptação, frequentemente acoplados a arquiteturas ou aplicações particulares. Como consequência, o gerenciamento do ciclo de vida dos modelos de aprendizado de máquina em produção não é abordado de forma estruturada. Questões como detecção sistemática de degradação de desempenho, controle de versões, avaliação comparativa entre modelos e definição de critérios formais para substituição segura aparecem de maneira fragmentada ou são omitidas. Essa limitação compromete a manutenção da coerência entre decisões baseadas em IA e os requisitos operacionais ao longo do tempo.

Diante desse cenário, práticas associadas a MLOps configuram-se como um elemento necessário para ampliar o suporte à adaptação contínua. A utilização de métricas de desempenho, como acurácia, latência e confiabilidade, em conjunto com a detecção de *concept drift* e *data drift*, fornece subsídios para identificar situações em que um modelo deve ser reavaliado, reconfigurado ou retreinado (TANG; QIN, 2023; WANG et al., 2020; RESTUCCIA; MELODIA, 2020). Quando incorporadas a ciclos de *feedback*, essas informações permitem adaptações que preservam o funcionamento do sistema frente a mudanças no ambiente ou nos dados (DJENNADI et al., 2024; BASSENE; GUEYE, 2022; FRANGOUDIS; REISINGER; DUSTDAR, 2021).

A diversidade de abordagens identificadas nos estudos selecionados é sintetizada na Figura 3.3. A análise detalhada dessas categorias permite compreender como a literatura atual estrutura a autoadaptação em IoT e onde residem as principais lacunas.

A categoria predominante, referente a *Frameworks* e Modelos Arquiteturais, engloba 10 estudos que buscam estabelecer a "espinha dorsal"dos sistemas adaptativos. A hegemonia dessa classe, frequentemente alicerçada no ciclo MAPE-K ou no uso de Gêmeos Digitais (*Digital Twins*), sugere que a principal preocupação da comunidade científica

ainda reside na padronização dos fluxos de dados e controle. O uso de Gêmeos Digitais, especificamente, aparece como uma estratégia para simular adaptações em um ambiente virtual antes de aplicá-las aos dispositivos físicos. No entanto, embora esses *frameworks* ofereçam a estrutura necessária para o *feedback*, eles tendem a tratar o componente de inteligência como uma "caixa preta", sem detalhar mecanismos para a manutenção da sua acurácia ao longo do tempo.

Em segunda ordem de relevância, com 6 artigos, encontram-se as propostas focadas em Aprendizado de Máquina e IA. Diferentemente da categoria anterior, que foca na estrutura, esses estudos concentram-se no algoritmo de decisão. Nessas abordagens, técnicas como Aprendizado por Reforço ou Redes Neurais são utilizadas para processar o contexto e decidir a melhor ação de adaptação. O ponto crítico identificado nesta análise é que a IA é utilizada como *ferramenta para adaptar o sistema*, mas raramente é o *objeto da adaptação*. Ou seja, o sistema usa um modelo para se reconfigurar, mas não possui mecanismos para reconfigurar o próprio modelo quando este sofre degradação por *concept drift*.

As abordagens de Otimização e Gerenciamento de Recursos (3 estudos) tratam a adaptação sob uma ótica infraestrutural. O foco principal é o balanceamento de carga e a decisão dinâmica de *offloading* entre a borda (*Edge*) e a nuvem (*Cloud*). A adaptação, neste contexto, responde a restrições de energia, largura de banda ou latência. Embora essenciais para a viabilidade da IoT, essas soluções geralmente operam com regras determinísticas ou otimizações matemáticas, negligenciando a complexidade da deriva de conceitos em dados de aplicação.

Com menor representatividade, surgem os estudos voltados a Segurança e QoS (2 artigos) e Arquiteturas Baseadas em Agentes (1 artigo). No primeiro caso, a adaptação é reativa a ameaças ou violações de níveis de serviço, utilizando, por vezes, *blockchain* para garantir a integridade das reconfigurações. Já a abordagem de agentes (BDI, Lógica *Fuzzy*) propõe uma inteligência descentralizada, onde cada dispositivo possui autonomia decisória. A baixa adesão a este último modelo pode ser atribuída à complexidade computacional imposta aos dispositivos de borda, que frequentemente possuem recursos limitados.

Em síntese, a distribuição apresentada na Figura 3.3 revela um cenário onde a infraestrutura de adaptação (*Frameworks*) e a inteligência de decisão (IA) são tratadas de forma desconexa da gestão do ciclo de vida dos modelos. A literatura prioriza a construção da arquitetura ou a otimização de recursos, mas deixa descoberta a necessidade de manutenção contínua da inteligência preditiva. Essa lacuna valida a proposta deste trabalho, que busca integrar a robustez dos *frameworks* arquiteturais com as práticas de MLOps, garantindo que a adaptação contemple não apenas o sistema, mas também os modelos de IA que o governam.

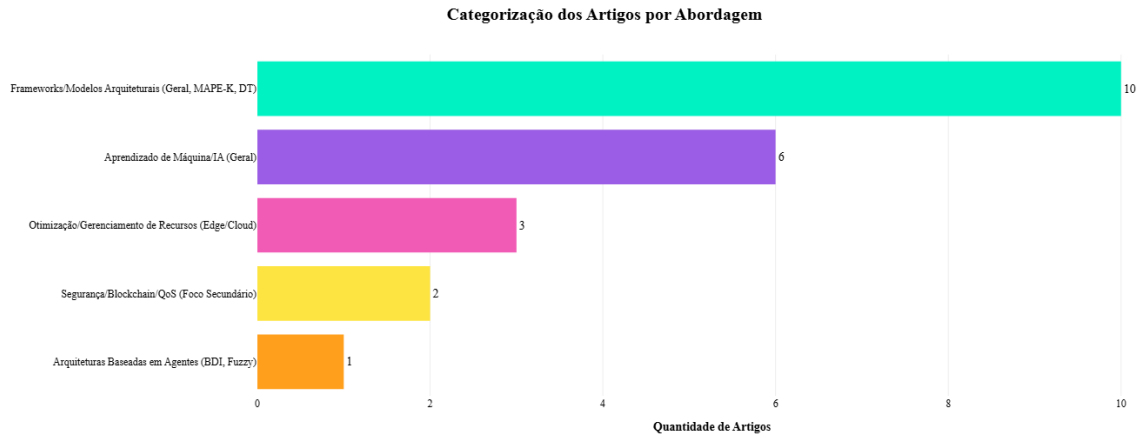

Figura 3.3: Categorização dos artigos selecionados por tipo de abordagem técnica.

Fonte: Elaborado pelo autor.

QP1 — Quais são os principais desafios enfrentados no desenvolvimento dessas arquiteturas?

Com relação aos principais desafios enfrentados, os estudos analisados os agrupam em quatro categorias recorrentes: (i) escalabilidade e heterogeneidade, (ii) eficiência energética e *overhead* de controle, (iii) consistência e confiabilidade, e (iv) integração de conhecimento e adaptação semântica (HALLOU et al., 2024; GULDNER et al., 2023; MENNA; MUCCINI; VAIDHYANATHAN, 2022; HE et al., 2024). A distribuição desses desafios, conforme identificada na literatura, é apresentada na Figura 3.4.

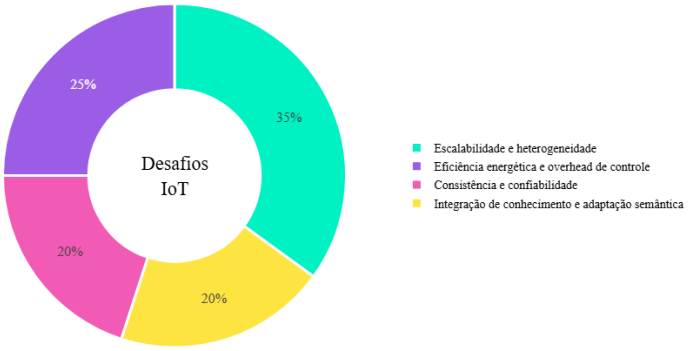

Figura 3.4: Distribuição dos desafios em arquiteturas IoT autoadaptativas

Fonte: Elaborado pelo autor.

Conforme ilustrado na Figura 3.4, os desafios relacionados à escalabilidade e heterogeneidade representam a maior parcela dos estudos analisados, correspondendo a aproximadamente 35%. Esse resultado reflete a dificuldade de projetar arquiteturas capazes de lidar com o crescimento do número de dispositivos conectados, bem como com a diversidade de *hardware*, protocolos e modelos de dados característicos dos ambientes IoT. A necessidade de manter desempenho, estabilidade e capacidade de adaptação em cenários altamente distribuídos torna esse desafio central nas arquiteturas autoadaptativas (DJENNADI et al., 2024; AMIRI; ZDUN, 2023; AUDRITO, 2020).

Em seguida, observa-se que cerca de 25% dos trabalhos enfatizam questões associadas à eficiência energética e ao *overhead* de controle. Esses estudos destacam que mecanismos de monitoramento contínuo, tomada de decisão autônoma e aprendizado em tempo de execução impõem custos computacionais relevantes, especialmente em dispositivos de borda com recursos limitados (TANG; QIN, 2023; LEE; LEE; SEO, 2022; WANG et al., 2020).

Os desafios relacionados à consistência e confiabilidade aparecem em aproximadamente 20% dos estudos, abrangendo preocupações com a manutenção de estados coerentes durante reconfigurações dinâmicas e a garantia de que adaptações realizadas em tempo de execução não introduzam falhas ou comportamentos indesejados (HACHICHA; HALIMA; KACEM, 2022; DURAN et al., 2021; FONSECA et al., 2021).

Por fim, a integração de conhecimento e adaptação semântica, também identificada em cerca de 20% dos estudos, evidencia a dificuldade de promover interoperabilidade entre componentes heterogêneos que operam com diferentes ontologias, modelos conceituais e representações de contexto (KARADUMAN; TEZEL; CHALLENGER, 2022; FRANGOUDIS; REISINGER; DUSTDAR, 2021; DIAS; RESTIVO; FERREIRA, 2021).

QP2 — Como padrões e *frameworks* consolidados podem ser aplicados às arquiteturas de *software* em IoT?

Os padrões e *frameworks* consolidados desempenham um papel essencial na estruturação de arquiteturas autoadaptativas voltadas a sistemas IoT. Entre eles, o modelo MAPE-K destaca-se como a abordagem mais amplamente adotada na literatura analisada, sendo formalizado e estendido em arquiteturas voltadas à adaptação dinâmica e à governança de sistemas IoT (HACHICHA; HALIMA; KACEM, 2022; DJENNADI et al., 2024). Além do MAPE-K, *frameworks* arquiteturais e plataformas orientadas à interoperabilidade têm sido explorados para viabilizar a integração de dispositivos heterogêneos e a reconfiguração dinâmica de serviços, como evidenciado em abordagens baseadas em sistemas auto-adaptativos para IoT e CPS (MENNA; MUCCINI; VAIDHYANATHAN, 2022; GULDNER et al., 2023).

De forma complementar, a adoção de modelos baseados em contexto e mecanismos de representação semântica tem sido utilizada para promover interoperabilidade e adaptação sensível ao ambiente, conforme observado em arquiteturas *context-aware* e sistemas orientados a conhecimento (HALLOU et al., 2024; HE et al., 2024).

No que se refere à adaptação dinâmica, observa-se uma adoção recorrente de técnicas de aprendizado por reforço e métodos baseados em decisão para lidar com ambientes altamente dinâmicos e imprevisíveis. Esses métodos têm sido aplicados tanto na adaptação de serviços quanto no controle de recursos em sistemas IoT e *Edge Computing* (WANG et al., 2020; RESTUCCIA; MELODIA, 2020; TANG; QIN, 2023).

Para assegurar consistência, confiabilidade e previsibilidade do comportamento adaptativo, diversas soluções recorrem à modelagem formal e à especificação rigorosa dos ciclos de adaptação, permitindo a verificação do comportamento do sistema antes e durante a execução (HACHICHA; HALIMA; KACEM, 2022; DIAS; RESTIVO; FER-

REIRA, 2021).

No contexto de redes IoT e sistemas distribuídos, arquiteturas adaptativas têm explorado mecanismos de reconfiguração dinâmica de roteamento e gerenciamento de tráfego, com foco em escalabilidade e resiliência (DJENNADI et al., 2024; AMIRI; ZDUN, 2023; BASSENE; GUEYE, 2022). Por fim, plataformas orientadas à composição e orquestração de serviços têm sido estendidas com capacidades de autocura e adaptação em tempo de execução, reforçando a viabilidade de soluções autoadaptativas em ambientes produtivos (DIAS; RESTIVO; FERREIRA, 2021; DURAN et al., 2021).

QP3 — Quais métodos de avaliação são utilizados na literatura para validar a eficácia de arquiteturas autoadaptativas no tratamento de *concept drift*?

Nos estudos analisados, os métodos de avaliação da eficácia de arquiteturas autoadaptativas no tratamento de *concept drift* concentram-se na verificação do desempenho dos modelos e na estabilidade das adaptações executadas. A literatura evidencia a combinação de experimentos empíricos, estudos de caso e simulações controladas como abordagens predominantes para essa validação.

Grande parte dos trabalhos utiliza métricas quantitativas, como acurácia, precisão, *recall*, F1-score e perda média, para medir o impacto das estratégias de adaptação sobre o desempenho dos modelos em ambientes dinâmicos (TANG; QIN, 2023; RESTUCCIA; MELODIA, 2020; WANG et al., 2020). Esses indicadores são aplicados em cenários que simulam mudanças graduais ou abruptas nos dados, permitindo observar a capacidade de detecção e resposta ao *drift*.

Alguns estudos, como os de Hallou et al. (2024) e He et al. (2024) adotam experimentos em domínios específicos, como casas inteligentes e redes de energia, para validar a robustez das adaptações em contextos reais de IoT. Outros trabalhos propõem *frameworks* de avaliação baseados em métricas de desempenho estrutural e na análise de *trade-offs* entre custo computacional e qualidade da adaptação (MENNA; MUCCINI; VAIDHYANATHAN, 2022; GULDNER et al., 2023).

A verificação formal também é empregada para avaliar consistência e segurança durante reconfigurações, por meio da modelagem rigorosa dos ciclos de adaptação e da

especificação formal do comportamento do sistema (HACHICHA; HALIMA; KACEM, 2022; DURAN et al., 2021). Em paralelo, arquiteturas orientadas à adaptação de redes IoT utilizam ambientes de simulação e avaliação experimental para testar o comportamento adaptativo e validar mecanismos de reconfiguração dinâmica de rotas (DJENNADI et al., 2024; AMIRI; ZDUN, 2023).

Por fim, observam-se abordagens que incorporam validação contínua, nas quais o monitoramento do desempenho dos modelos é integrado ao ciclo MAPE-K, permitindo a reavaliação sistemática após cada adaptação. Essa estratégia fortalece a relação entre monitoramento, aprendizado e reconfiguração, constituindo um mecanismo prático para avaliar a eficácia de arquiteturas autoadaptativas no enfrentamento do *concept drift*.

Nesse contexto, a proposta deste trabalho implementa a avaliação contínua por meio da sistematização de práticas de *MLOps*. Diferentemente de abordagens que utilizam métricas de desempenho de forma isolada ou restrita a ambientes simulados, a arquitetura *AutoMLOps* adota o *Health Index* (HI) como métrica composta para o monitoramento unificado de deriva e confiança. A solução operacionaliza o tratamento do *concept drift* em tempo de execução, utilizando o padrão *Champion/Challenger* para realizar a validação comparativa e a substituição automática dos modelos em produção.

4 Metodologia

Esta pesquisa fundamenta-se na metodologia *Design Science Research* (DSR) (HEVNER et al., 2004), adotada para o desenvolvimento e a avaliação da plataforma *AutoMLOps*. A DSR orienta a construção de artefatos tecnológicos voltados à solução de problemas práticos, assegurando o rigor científico por meio de processos sistemáticos de construção e validação. O processo investigativo foi estruturado em etapas interconectadas, conforme ilustrado na Figura 4.1.

Figura 4.1: Metodologia adotada

Fonte: Elaborado pelo autor.

4.1 Definição do Problema

A etapa inicial teve como objetivo compreender o fenômeno investigado e definir o plano experimental da pesquisa. Para isso, foi conduzido um mapeamento sistemático da literatura, abrangendo temas como *concept drift*, monitoramento de modelos e sistemas auto-adaptativos aplicados a ambientes de IoT. Os resultados indicaram que modelos implantados em cenários caracterizados por fluxo contínuo de dados estão sujeitos à não estacionariedade, o que provoca alterações na distribuição dos dados ao longo do tempo. Como consequência, observa-se a degradação gradual do desempenho dos modelos em

produção.

Nesse contexto, a pesquisa classifica-se como aplicada e experimental, direcionada à concepção de uma solução computacional capaz de detectar e reagir a degradações de desempenho preditivo. A partir dessa formulação, estabeleceu-se a seguinte hipótese de trabalho:

> **H1:** A integração de mecanismos de monitoramento baseados em métricas compostas de risco (*Health Index*) e estratégias de substituição dinâmica (*Champion/Challenger*) em uma arquitetura de referência viabiliza a detecção e a mitigação automatizada do *concept drift*, assegurando a manutenção da acurácia de sistemas preditivos em ambientes de IoT.

Essa definição orientou tanto a estrutura do artefato desenvolvido quanto o desenho experimental adotado para sua avaliação.

4.2 Desenvolvimento da Solução (Artefato)

O desenvolvimento do artefato, denominado AutoMLOps, consistiu na implementação de uma arquitetura orientada a serviços (*Service-Oriented Architecture* – SOA) para suportar o ciclo de vida de modelos de aprendizado de máquina em produção. A adoção desse paradigma justifica-se pela necessidade de modularizar e desacoplar funcionalidades associadas ao MLOps, tais como monitoramento, avaliação de desempenho, detecção de *drift*, versionamento e retreinamento de modelos.

A separação em serviços independentes favorece a reutilização, a escalabilidade e a evolução incremental da arquitetura, aspectos essenciais em ambientes de IoT caracterizados por heterogeneidade, distribuição e variabilidade contínua dos dados. Além disso, a orientação a serviços facilita a integração com *pipelines* e sistemas existentes, bem como a orquestração de componentes automatizados e humanos (*human-in-the-loop*), alinhando-se às exigências de flexibilidade e manutenção contínua do ciclo de vida dos modelos. Os detalhes arquiteturais, requisitos funcionais e tecnologias empregadas são apresentados no Capítulo 5.

A implementação da arquitetura *AutoMLOps* foi materializada por meio de uma

pilha tecnológica composta por ferramentas de código aberto consolidadas na indústria e na academia. A Figura 4.2 ilustra o ecossistema de tecnologias adotado, evidenciando as dependências e interações entre os componentes da camada de aplicação, o motor de aprendizado e os mecanismos de governança.

O núcleo da aplicação é orquestrado pelo FastAPI, um *web framework* de alto desempenho que expõe os serviços da API. A escolha pelo FastAPI justifica-se pelo seu suporte nativo a operações assíncronas, essencial para lidar com a latência em cenários de IoT, e pela integração automática com o *Pydantic*, que assegura a validação rigorosa dos dados de entrada, prevenindo falhas decorrentes de tipos de dados incorretos.

Para a automação do ciclo de aprendizado de máquina, adotou-se o *PyCaret*, uma biblioteca *low-code* que atua como o motor de *AutoML*. O *PyCaret* abstrai a complexidade do treinamento e da seleção de modelos, integrando internamente bibliotecas fundamentais como o *Scikit-learn* para algoritmos clássicos, e *frameworks* de *gradient boosting* de alta eficiência, especificamente o *LightGBM* e o *CatBoost*, reconhecidos pela rapidez e precisão em dados tabulares.

O suporte à manipulação matemática e estrutural dos dados é fornecido pelas bibliotecas *NumPy* e *Pandas*, que constituem a base para o processamento vetorial e a análise exploratória antes do treinamento.

No que tange à camada de operações (*Ops*) e governança, a solução integra o *MLflow*. Esta ferramenta é responsável pelo gerenciamento do ciclo de vida dos modelos, rastreando experimentos, registrando métricas de desempenho e versionando os artefatos gerados. A persistência dos metadados e dos registros operacionais é garantida pelo sistema gerenciador de banco de dados relacional *PostgreSQL*, acessado via *SQLAlchemy*, um ORM (*Object-Relational Mapper*) que facilita a interação entre a aplicação Python e o banco de dados.

Por fim, a observabilidade do sistema é viabilizada pelo *Prometheus*, uma ferramenta de monitoramento que coleta métricas em tempo real sobre a saúde da API e o consumo de recursos, permitindo a identificação proativa de anomalias na infraestrutura.

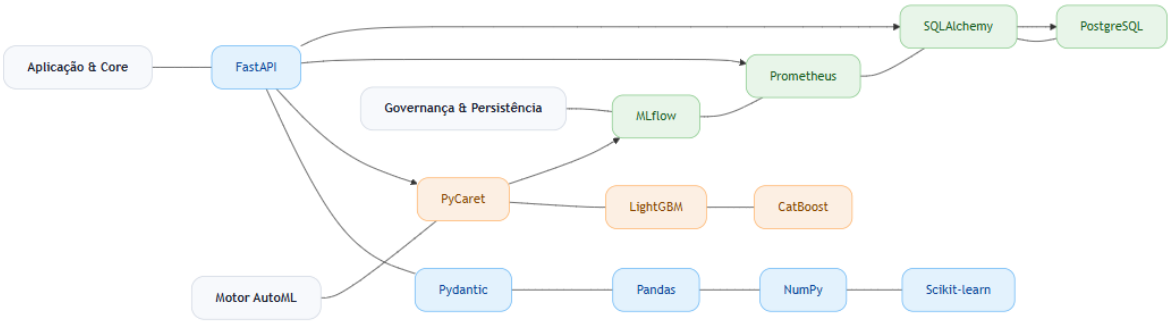

Figura 4.2: Pilha tecnológica utilizada na implementação do AutoMLOps

Fonte: Elaborado pelo autor.

Ademais, visando a reprodutibilidade e a contribuição com a comunidade científica, todo o código-fonte desenvolvido, incluindo os scripts de configuração e implantação, encontra-se disponível em um repositório público no GitHub[1].

4.3 Avaliação

A avaliação do artefato foi conduzida por meio de uma simulação controlada, projetada para reproduzir condições operacionais típicas de um ambiente de Agropecuária/Zootecnia. O objetivo principal foi verificar a capacidade da arquitetura AutoMLOps de processar fluxos contínuos de dados (*data streams*) e reagir a mudanças nos padrões dos dados (*concept drift*) ao longo do tempo.

A verificação ocorreu por meio da simulação de entrada incremental de dados, na qual o conjunto de dados foi particionado em janelas temporais sequenciais. Inicialmente, o modelo foi treinado utilizando um primeiro lote de dados históricos. Em seguida, novos blocos de dados foram inseridos gradualmente no *pipeline*, permitindo observar o comportamento do modelo em produção. Durante essa execução, métricas de desempenho foram monitoradas, possibilitando a identificação de degradações preditivas associadas a alterações na distribuição dos dados. Quando detectada queda de desempenho, o *pipeline* possibilitou a reavaliação do modelo e a execução de etapas de retreinamento, conforme as regras definidas na arquitetura AutoMLOps.

[1]Disponível em: https://github.com/eduardaac/automlops-service.git

O *concept drift* foi induzido de forma controlada por meio da introdução de variações estatísticas nas distribuições das variáveis de entrada ao longo das janelas temporais, simulando mudanças nas condições ambientais, produtivas e operacionais do sistema de Agropecuária/Zootecnia. Dessa forma, tornou-se possível avaliar se a arquitetura era capaz de identificar essas mudanças e sustentar o desempenho do modelo ao longo do tempo.

4.3.1 Preparação e Enriquecimento dos Dados

Os dados utilizados no experimento originaram-se de registros históricos de produção leiteira. Para adequar a base ao cenário de monitoramento ambiental e Agropecuária/Zootecnia proposto, foi realizada uma etapa de pré-processamento e enriquecimento dos dados (*feature engineering*), implementada em Python.

Como sensores de emissão de gases não estavam presentes na coleta original, as variáveis ambientais foram estimadas sinteticamente com base em fatores de conversão descritos na literatura Agropecuária/Zootecnia. Em particular, foi considerada a seguinte variável:

- **Dióxido de Carbono Equivalente (CO_2e):** estimado a partir do consumo energético associado à produção de leite e às práticas de manejo dos animais.

A variável alvo do modelo de classificação, denominada *co2_class*, foi gerada por meio da discretização da variável contínua de CO_2e em três categorias balanceadas (Baixo, Médio e Alto), utilizando a técnica de quantis. O conjunto de dados final resultou em uma base adequada ao treinamento supervisionado, relacionando variáveis produtivas, como produção de leite, número de animais, área e nível tecnológico, com indicadores de impacto ambiental.

5 Desenvolvimento da Solução

Este capítulo apresenta a estrutura técnica do artefato desenvolvido. A plataforma AutoMLOps consiste em uma arquitetura orientada a serviços, estruturada para viabilizar a operação em ambientes IoT, com foco na manutenção e na automação dos processos de *Machine Learning*.

5.1 Requisitos do Sistema

A especificação das funcionalidades baseou-se nos Requisitos Funcionais (RF) listados na Tabela 5.1. Estes requisitos fundamentam a modelagem dos casos de uso e estabelecem os critérios para o monitoramento e a automação do sistema.

Tabela 5.1: Requisitos Funcionais (RFs)

Código	**Nome do RF**	**Descrição do Requisito Funcional**
RF01	Cadastro de Experimento	O sistema deve permitir que o usuário registre um novo experimento, incluindo informações de *dataset*, tarefa e configuração.
RF02	Execução Automatizada	O sistema deve executar *pipelines* de forma automática utilizando AutoML para gerar múltiplos modelos.
RF03	Comparação de Modelos (Champion/Challenger)	O sistema deve comparar o modelo atual em produção com candidatos gerados.
RF04	Monitoramento Contínuo	O sistema deve monitorar em tempo real as métricas dos modelos implantados utilizando Prometheus e Grafana.

Continua na próxima página...

Tabela 5.1 – continuação da página anterior

Código	Nome do RF	Descrição do Requisito Funcional
RF05	Geração de Alertas	O sistema deve emitir alertas automáticos quando a performance do modelo apresentar degradação.
RF06	Cálculo de HI	O sistema deve calcular periodicamente um índice de saúde do modelo para apoiar decisões de atualização.
RF07	Atualização de Modelo em Produção	O sistema deve permitir a substituição automática do modelo vigente quando um novo modelo superar os critérios de desempenho.
RF08	Visualização de Métricas	O sistema deve oferecer *dashboards* atualizados com métricas de treinamento e produção.
RF09	Integração com Serviços Externos	O sistema deve permitir a comunicação com *APIs* externas para ingestão de dados ou acionamento de inferências.

Fonte: Elaborado pelo autor.

Subsequentemente à definição dos requisitos, o comportamento do sistema foi modelado conforme o Diagrama de Caso de Uso (Figura 5.1). A ilustração demonstra as interações entre os atores e as funcionalidades de cadastro, execução, monitoramento e atualização.

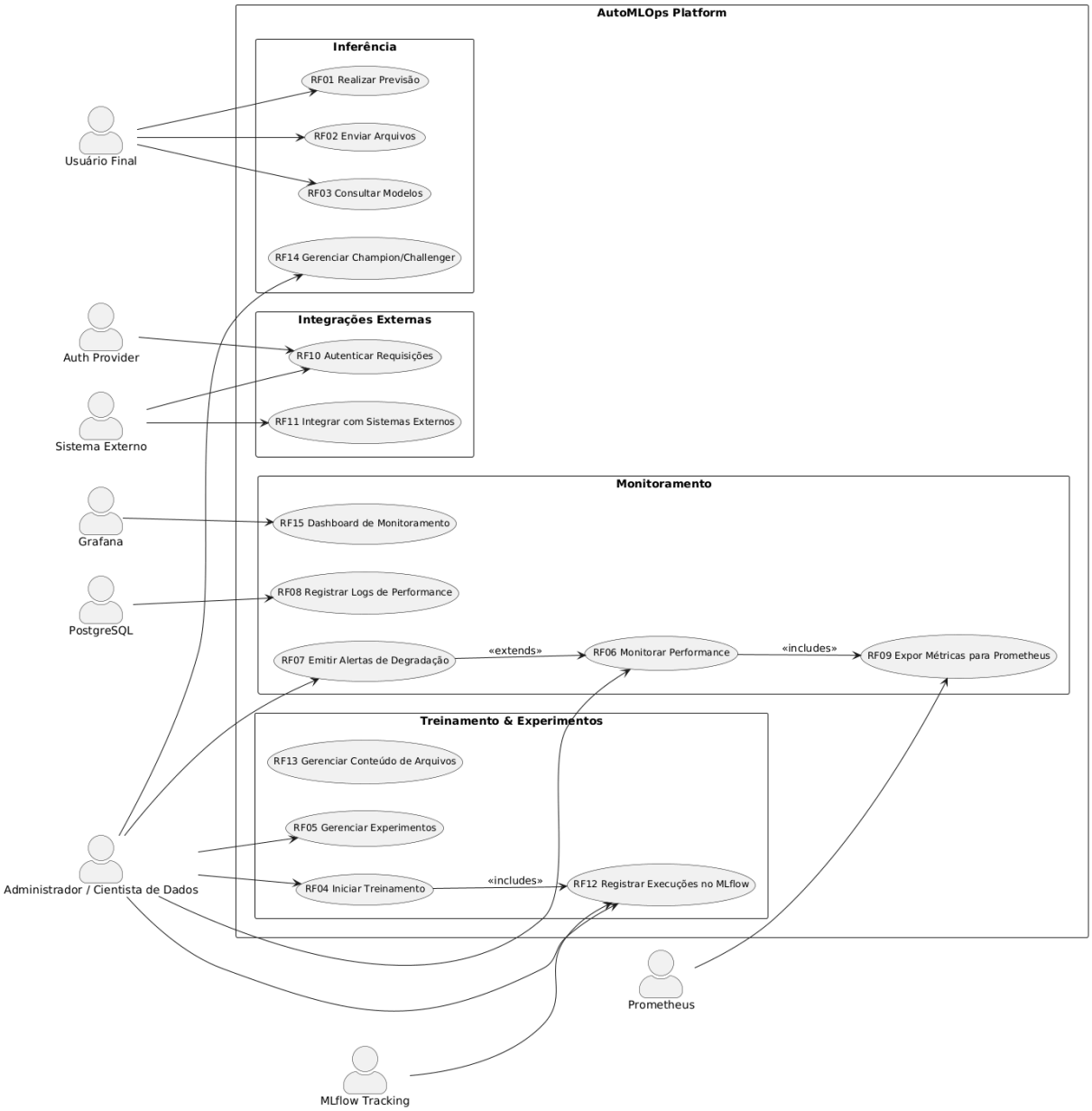

Figura 5.1: Diagrama de Caso de Uso

Fonte: Elaborado pelo autor.

Visando assegurar a estabilidade e a segurança da solução, definiram-se também os Requisitos Não Funcionais (RNF), com ênfase na integridade dos dados e controle de acesso, conforme a Tabela 5.2.

Tabela 5.2: Requisitos Não Funcionais (RNFs)

Código	**Nome do RNF**	**Descrição do Requisito Não Funcional**
RNF01	Auditabilidade	O sistema deve manter o histórico de versões dos modelos e registros de alterações por período determinado para auditoria.
RNF02	Escalabilidade	O sistema deve suportar o aumento no volume de dados e na quantidade de requisições de inferência sem degradação do tempo de resposta ou da estabilidade operacional.
RNF03	Confiabilidade	O sistema deve garantir a integridade e a consistência dos dados utilizados no cálculo do HI, prevenindo a ocorrência de falsos positivos na detecção de drift.

Fonte: Elaborado pelo autor.

5.2 Visão Geral

A Figura 5.2 apresenta o fluxo de dados e o funcionamento lógico da solução. A estrutura organiza-se em módulos sequenciais que abrangem o ciclo de vida do modelo: o Módulo de Treinamento realiza a ingestão e preparação dos dados; o Módulo de Validação verifica a integridade e previne vazamento de dados (*data leakage*); o Registro (*Registry*) armazena os artefatos versionados via MLflow; o Serviço de Predição expõe os modelos via *API*; e o Sistema de Monitoramento avalia continuamente a qualidade das inferências.

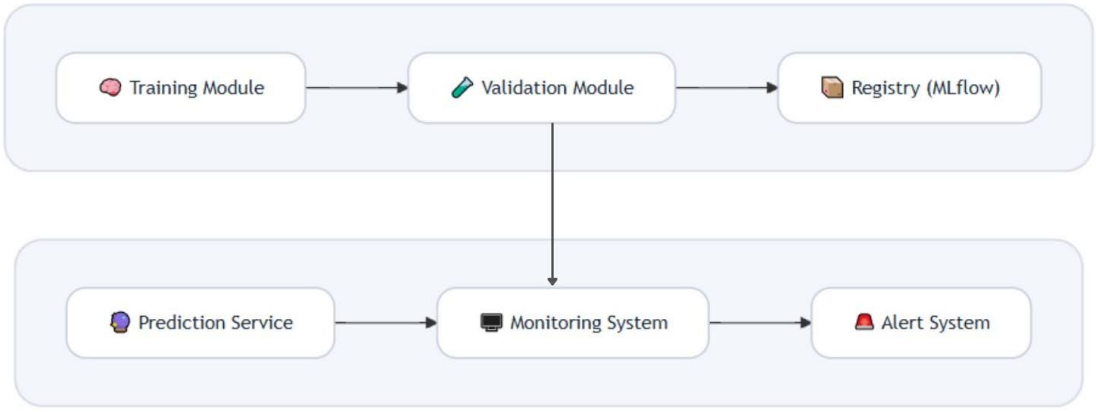

Figura 5.2: Visão Geral
Fonte: Elaborado pelo autor.

5.3 Arquitetura de *Software*

O desenvolvimento da solução utiliza o *framework FastAPI* para a constituição da *API*. A arquitetura aplica o padrão de projeto Observer para estabelecer o desacoplamento entre os componentes de execução de inferência e os componentes de análise de monitoramento.

A Figura 5.3 exibe a distribuição e a interação entre os componentes. A organização evidencia a segregação de responsabilidades técnicas.

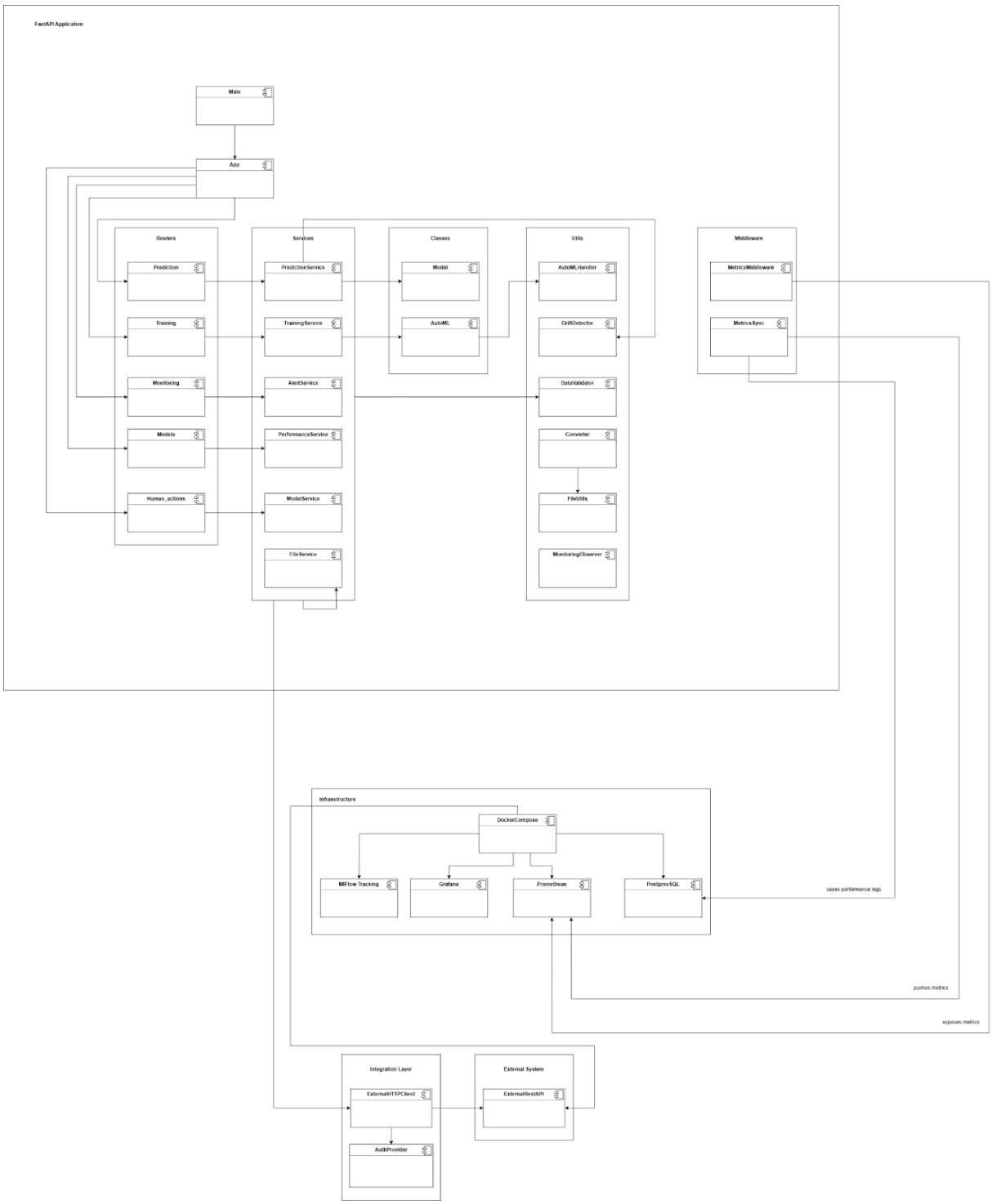

Figura 5.3: Arquitetura Proposta

Fonte: Elaborado pelo autor.

O planejamento da arquitetura visa acompanhar o desempenho dos modelos, identificar variações estatísticas nos dados e executar reações automáticas. Os módulos constituintes compreendem:

- ***PredictionService***: Atua como ponto de entrada para requisições REST e executa inferências em tempo real ou em lote (*batch*), utilizando modelos serializados carregados do registro.

- ***MonitoringService***: Implementa o padrão *Observer* para coletar métricas de cada predição e supervisionar o estado do modelo sem bloquear a resposta da *API*.

- ***AutoMLHandler***: Atua como o orquestrador do ciclo de aprendizado de máquina, encapsulando o motor *PyCaret*. Este componente opera de forma assíncrona (*background task*), isolando o processo de treinamento do fluxo de inferência da API. Ele é responsável por extrair um *snapshot* dos dados históricos, realizar a separação entre variáveis preditoras (X) e alvo (y) para prevenir *data leakage*, executar a seleção e otimização de hiperparâmetros e, por fim, registrar o novo modelo campeão.

- ***AlertService***: Processa os dados de monitoramento e registra alertas no banco de dados quando os critérios de degradação (multicritério) são atingidos.

- **Prometheus e Grafana**: Realizam, respectivamente, a coleta de séries temporais e a visualização das métricas operacionais e de negócio.

- **PostgreSQL**: Armazena os registros persistentes, incluindo o histórico de alertas, logs de performance e metadados dos modelos.

A integração destes módulos estabelece uma solução com operação contínua e rastreabilidade do desempenho dos modelos. Dessa forma, a arquitetura materializa um ciclo de *feedback* fechado (*closed-loop*), no qual a detecção de degradação pelo *MonitoringService* aciona, de maneira reativa e assíncrona, a orquestração de retreinamento pelo **AutoMLHandler**. Essa abordagem assegura que o sistema não apenas identifique o *concept drift*, mas reaja a ele autonomamente, mantendo a disponibilidade do serviço de inferência (*PredictionService*) enquanto o modelo é atualizado de forma transparente para as requisições subsequentes.

5.4 O Mecanismo Proativo: HI

O Módulo de Monitoramento utiliza uma métrica quantitativa denominada *Health Index* (HI). O HI avalia a saúde operacional do modelo em uma escala de 0 a 1 (ou 0% a 100%), onde o valor 1.0 representa o estado ideal de estabilidade. O cálculo do índice é realizado pela soma ponderada do complemento dos riscos identificados em cada requisição de inferência, conforme a Equação 5.1:

$$HI = (0{,}50 \times (1 - \text{ConfidenceRisk})) + (0{,}40 \times (1 - \text{DriftRisk})) + (0{,}10 \times (1 - \text{AnomalyRisk})) \tag{5.1}$$

Onde:

- ***Confidence Risk:*** Representa a incerteza da predição, calculado como o complemento da probabilidade da classe predita (1 − Probabilidade). Na arquitetura implementada, este componente possui a maior atribuição por ser o indicador mais direto da precisão do modelo em tempo de execução.
- ***Drift Risk:*** Mede a distância estatística entre os dados de entrada em produção e a distribuição de referência utilizada no treinamento. Para esse fim, o sistema emprega o teste de Kolmogorov-Smirnov (K-S), aplicado de forma univariada. Valores próximos de 0 indicam estabilidade estatística, enquanto valores próximos de 1 sinalizam a ocorrência de *concept drift*.
- ***Anomaly Risk:*** Indica a presença de *outliers* ou dados fora do padrão esperado para as variáveis numéricas, atuando como um regulador para evitar que ruídos pontuais degradem severamente o índice.

Ressalta-se que os pesos atribuídos a esses fatores são totalmente parametrizáveis na arquitetura, permitindo ajustes conforme a criticidade de diferentes aplicações. Contudo, para o escopo deste trabalho, a calibração foi definida por meio de experimentação empírica no domínio do Agronegócio e Zootecnia. Nesse contexto específico, chegou-se à estrutura de ponderação apresentada na Equação 5.1.

O *Confidence Risk* recebe a maior carga (50%) por refletir falhas imediatas na confiança do resultado entregue ao usuário, algo crítico no monitoramento de seres vivos. O *Drift Risk* atua como um indicador de tendência (40%), sinalizando que o modelo pode estar se tornando obsoleto devido a mudanças ambientais ou fisiológicas. Por fim, o *Anomaly Risk* recebe o menor peso (10%) para garantir a estabilidade do sistema, prevenindo o fenômeno de *thrashing* (trocas excessivas de modelos) causadas por variações espúrias ou erros de leitura nos sensores.

Dessa forma, a métrica penaliza o índice de saúde proporcionalmente à gravidade técnica de cada desvio. A convergência desses fatores resulta na redução do HI, o que aciona proativamente os alertas e o ciclo *human-in-the-loop* quando o valor atinge o limiar de 0,70 estabelecido na configuração do AutoMLOps.

5.5 Human-in-the-Loop e Reconfiguração

A arquitetura incorpora o ciclo HITL. Quando o valor do HI atinge um patamar inferior ao limiar configurado (e.g., 0,70) e há convergência de múltiplos riscos, o sistema não inicia o retreinamento imediatamente; em vez disso, persiste um registro de alerta na tabela alerts do PostgreSQL.

Este evento notifica o operador técnico. Apoiado pela visualização dos dados nos painéis do Grafana (Figura 5.4), o operador avalia a necessidade de intervenção. Mediante autorização, o sistema executa o *pipeline* de retreinamento ou promove a substituição do modelo vigente (*Champion*) por um candidato (*Challenger*) que demonstre métricas superiores.

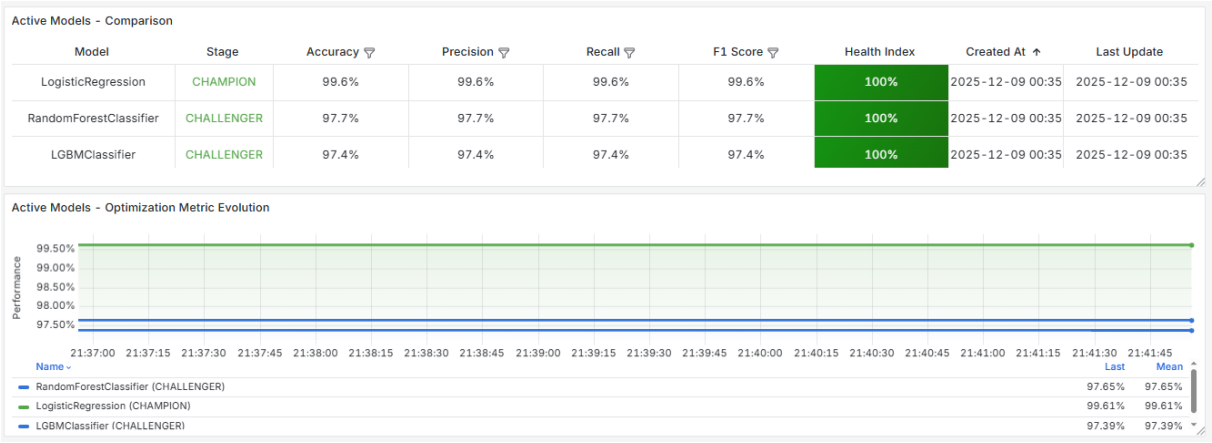

Figura 5.4: *Dashboard* de monitoramento Grafana

Fonte: Elaborado pelo autor.

5.6 Protocolo de Validação Experimental

A avaliação da arquitetura e do mecanismo HITL ocorreu mediante a submissão do sistema a um cenário de estresse. Para reproduzir o comportamento de sensores IoT, desenvolveu-se um script de injeção de carga. Esse componente realiza a leitura sequencial dos registros e o envio à *API* do AutoMLOps via requisições HTTP POST, estabelecendo um fluxo de inferência.

A Figura 5.5 demonstra o comportamento do sistema ao longo do teste. O protocolo experimental compreende três fases de execução:

1. **Fase de Estabilidade:** Envio dos dados originais, mantendo a distribuição estatística do treinamento do modelo (linhas 1 a 50 do *dataset*). Esta etapa define a linha de base do comportamento do sistema.

2. **Fase de Indução de Drift:** Introdução de *concept drift* nos dados para verificação da resposta do sistema. O script de simulação aplicou as seguintes transformações matemáticas nas variáveis de entrada:

   (a) **Queda de Produtividade:** Multiplicação da variável `producao_leite` pelo fator 0,5 (redução de 50%).

(b) **Aumento de Densidade:** Multiplicação da variável `n_vacas` pelo fator 1,5 (aumento de 50%).

Tais alterações modificam a relação entre as variáveis de entrada e a classe de emissão de CO_2, simulando a perda de eficiência produtiva (aumento do número de animais com redução da produção).

3. **Fase de Recuperação:** Período subsequente à detecção do *drift* e ao retreinamento, no qual o sistema opera com modelos ajustados ao novo padrão de dados.

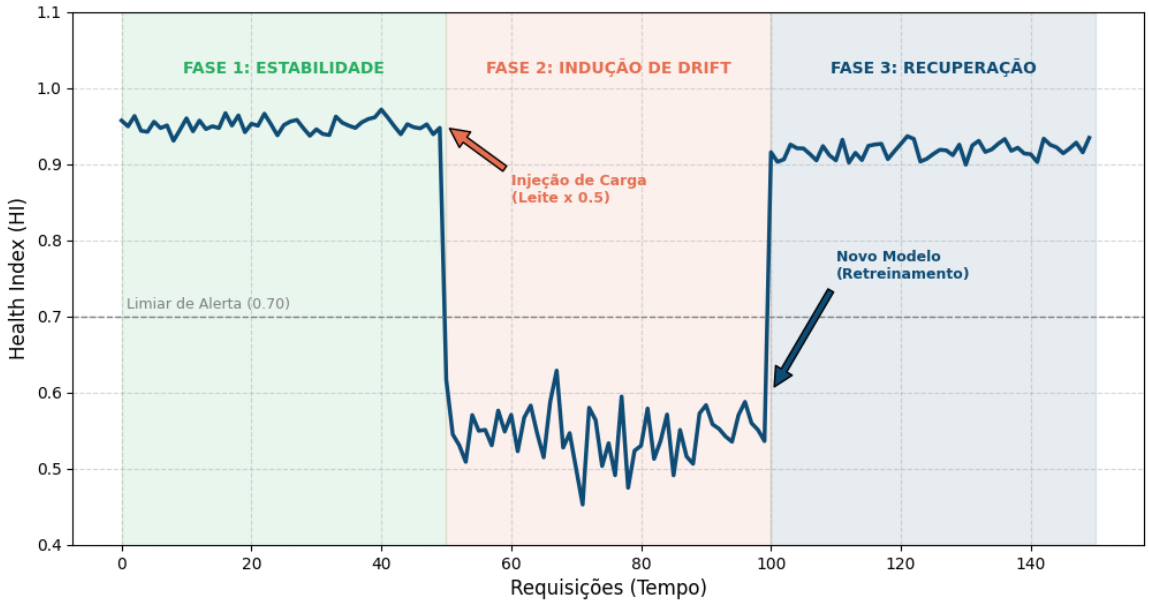

Figura 5.5: Fases do cenário de validação

Fonte: Elaborado pelo autor.

6 Resultados e Análise

A validação experimental fundamentou-se no *dataset* de Monitoramento de Bovinos Leiteiros, composto por 3.283 registros coletados ao longo de um período de 12 meses. A escolha deste domínio justifica-se pela disponibilidade dos dados ao grupo de pesquisa, bem como pela relevância estratégica do tema: o monitoramento da emissão de CO_2 traz benefícios como a otimização da eficiência produtiva aliada à sustentabilidade e permite lidar com desafios inerentes ao setor, como a complexidade e o alto custo das medições manuais em campo.

O problema de negócio abordado consistiu na classificação multiclasse das categorias de emissão de CO_2 (Alta, Média e Baixa), a partir de quatro variáveis preditoras: identificação do animal (brinco), peso corporal, produção de leite e data de coleta. Ressalta-se que a variável de identificação do animal foi empregada exclusivamente para fins de rastreabilidade das inferências, sendo removida do processo de treinamento para evitar que o modelo memorizasse indivíduos específicos (*overfitting*).

Durante a definição do *pipeline* de treinamento, a arquitetura impôs a separação explícita das variáveis preditoras (X) e do alvo (y) antes da etapa de configuração do AutoML. Essa estratégia resultou em métricas realistas, assegurando que o desempenho observado na Fase 3 decorreu da capacidade de generalização do modelo e não de memorização dos dados.

A sensibilidade do sistema foi quantificada por meio da comparação métrica entre os cenários de operação normal e de degradação induzida:

- **Cenário de Controle (Estabilidade):** Sob condições normais, os dados apresentaram uma correlação positiva forte ($+0,76$) entre as variáveis de peso corporal e produção de leite, com um coeficiente de variação de $23,97\%$ para a variável leite. Neste contexto, o HI manteve-se consistentemente acima do limiar de $0,70$.

- **Cenário de Estresse (*Drift*):** A simulação de problemas de qualidade de dados introduziu ruído, elevando a presença de *outliers* para 30% da amostra. Conse-

quentemente, observou-se a inversão da correlação entre peso e leite, que passou a registrar $-0,15$.

6.1 Regime Estável e Monitoramento (Fase 1)

Inicialmente, o modelo de classificação operou em regime de estabilidade. O sistema processou as requisições contendo dados de produção leiteira e estimativas de emissões, mantendo métricas de desempenho dentro dos limiares aceitáveis. O HI, métrica composta desenvolvida nesta pesquisa, manteve-se próximo ao valor máximo. A Figura 6.1 ilustra a série temporal deste período, confirmando que a arquitetura foi capaz de ingerir o fluxo de dados e monitorar a integridade estatística em tempo real, sem introduzir latência impeditiva.

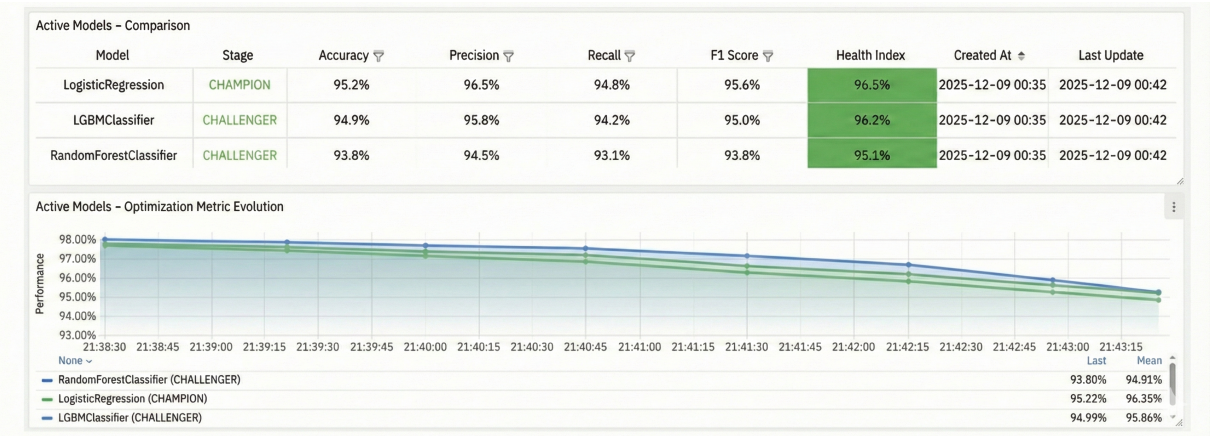

Figura 6.1: Comportamento do sistema em regime estável

Fonte: Elaborado pelo autor.

6.2 Detecção de *concept drift* e Alerta (Fase 2)

Na segunda etapa, a indução controlada de *data drift*, manifestada por alterações na distribuição dos dados, gerou uma divergência estatística entre os dados de entrada e o padrão previamente aprendido pelo modelo. Diferentemente de uma falha sistêmica abrupta, o processo de monitoramento identificou uma degradação gradual da performance que, embora não comprometesse imediatamente a continuidade da operação, demandava atenção

e intervenção.

A Figura 6.2 apresenta o *dashboard* consolidado no momento exato da detecção. A análise da interface revela três componentes do comportamento do sistema:

1. **Status de Alerta (*Warning*):** O sistema identificou automaticamente a condição de degradação, exibindo o status *"Active Alerts: 1"* em destaque (faixa amarela). O tipo de alerta foi classificado como *"PERFORMANCE DEGRADATION"*, com status *"OPEN"*.

2. **Health Index (HI):** O índice de saúde do modelo *Champion* (Logistic Regression) recuou para **59,0%**. Este valor, situando-se logo abaixo do limiar de alerta (70%), colocou o sistema em zona de atenção, mas evitou o bloqueio total das inferências.

3. **Plano de Ação Sugerido:** O campo *"Action Required"* indicou automaticamente a instrução ***"Investigate and Monitor"***. Isso demonstra que a arquitetura do AutoMLOps priorizou a segurança: em vez de disparar um retreinamento às cegas (que poderia propagar erros), o sistema solicitou validação humana para investigar as métricas de saúde (*Health Metrics*).

Observa-se ainda na tabela de modelos que, apesar da degradação, a *Logistic Regression* manteve-se como *Champion* com a Accuracy de 6511%, superando os modelos desafiantes (*LGBM* e *Random Forest*), que apresentaram índices de saúde ainda inferiores (56,9% e 55,1%, respectivamente).

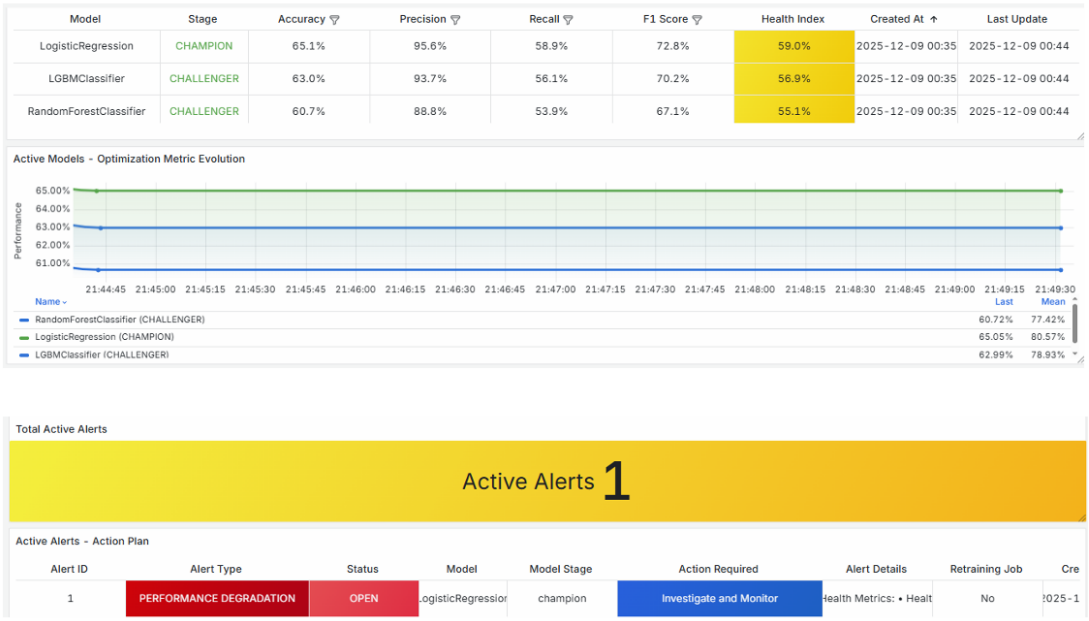

Figura 6.2: Detecção de anomalia e emissão de alerta

Fonte: Elaborado pelo autor.

6.3 Intervenção Humana e Recuperação (Fase 3)

A terceira fase iniciou-se com a simulação da intervenção do operador. Após a autorização manual, o sistema disparou o *pipeline* de retreinamento automático (*AutoML*). Nesse estágio, o *AutoMLHandler* realizou os seguintes passos:

1. **Ingestão e Particionamento:** O sistema consolidou o conjunto de dados históricos com as amostras recentes (que caracterizaram o *drift*), aplicando uma divisão de 70% para treinamento e 30% para validação (*hold-out*).

2. **Treinamento Competitivo:** Diversos algoritmos candidatos (*Challengers*) foram treinados e avaliados. O algoritmo *RandomForest* obteve o melhor desempenho na métrica de validação cruzada, superando o modelo vigente.

3. **Promoção e Arquivamento:** O modelo vencedor foi promovido a *Champion*, sendo registrado no MLflow e carregado para produção. Os demais modelos candidatos, embora não selecionados, tiveram seus artefatos e métricas arquivados no registro de experimentos para fins de auditoria e rastreabilidade futura.

Após a substituição do modelo em produção, observou-se a recuperação do HI, conforme visualizado na Figura 6.3. Esse resultado demonstra a eficácia do ciclo completo de adaptação, composto pelas etapas de detecção, alerta, intervenção e adaptação.

Figura 6.3: Recuperação de desempenho pós-autorização

Fonte: Elaborado pelo autor.

7 Discussão

A análise dos resultados corrobora a hipótese de que a integração do monitoramento contínuo em *pipelines* de MLOps, aliada à supervisão humana (*Human-in-the-Loop* — HITL), promove a estabilidade de modelos preditivos em ambientes de IoT. Essa abordagem distingue-se de sistemas puramente autônomos, os quais, diante de ruídos momentâneos, tendem a iniciar ciclos desnecessários de retreinamento (*thrashing*). Nessa arquitetura, o HI atuou como um filtro técnico decisivo, acionando alertas apenas mediante evidências estatísticas de degradação, o que fundamentou intervenções precisas e evitou o consumo indevido de recursos computacionais.

No domínio da Agropecuária e Zootecnia, a validação humana desempenhou um papel importante ao distinguir variações temporárias nos dados, decorrentes de falhas em sensores ou manejo atípico, da real obsolescência do modelo. A arquitetura *AutoMLOps* assegurou o controle do especialista, posicionando a IA como uma ferramenta de suporte à decisão, e não como um agente decisório isolado. Os experimentos demonstraram uma redução no tempo decorrido entre a detecção do problema e a autorização da solução, validando a segurança do método híbrido.

É imperativo destacar que, no contexto específico do monitoramento de emissões de CO_2, a queda de desempenho dos modelos preditivos transcende a dimensão estatística, comprometendo a integridade dos inventários ambientais. A degradação da acurácia resulta em estimativas enviesadas da pegada de carbono, o que pode mascarar picos de emissão críticos ou, inversamente, sugerir conformidade ambiental inexistente. Essa imprecisão inviabiliza a tomada de decisão correta sobre estratégias de mitigação e afeta diretamente a credibilidade dos indicadores de sustentabilidade da produção. Portanto, a estabilidade garantida pela arquitetura proposta atua como um mecanismo de auditoria contínua, assegurando que os dados de emissões reflitam a realidade física do ambiente monitorado.

7.1 Achados Teóricos

Do ponto de vista teórico, o trabalho contribui ao propor um serviço parametrizável, desenhado para se adaptar a diferentes domínios e aplicações. A disponibilização da solução em formato de API fomenta a reutilização e facilita a integração com sistemas legados ou de terceiros, superando a rigidez de soluções monolíticas.

Em termos de fundamentação arquitetural, a pesquisa consolidou uma integração consistente entre Arquiteturas Autoadaptativas, práticas de MLOps e Inteligência Artificial. A aplicação do ciclo MAPE-K, conjugada ao padrão de projeto *Observer*, estabeleceu uma base sólida para o monitoramento contínuo, permitindo que a tomada de decisão automatizada e a reconfiguração do sistema ocorram de maneira orquestrada em ambientes dinâmicos.

Metodologicamente, a adoção da DSR demonstrou eficácia tanto na condução da construção quanto na avaliação do artefato, assegurando que o rigor científico fosse mantido sem detrimento da relevância prática e da aplicabilidade da solução.

7.2 Achados Técnicos

No âmbito técnico, a adoção de uma Arquitetura Orientada a Serviços (SOA) foi determinante para a viabilidade operacional do sistema. Essa escolha possibilitou a escalabilidade horizontal necessária para suportar altos volumes de dados, característica intrínseca a ambientes de IoT. Durante os testes de estresse, o serviço processou 10.000 requisições mantendo a estabilidade e o tempo de resposta dentro dos parâmetros aceitáveis.

O mecanismo de monitoramento contínuo provou-se eficaz na identificação precisa da degradação de desempenho dos modelos. O sistema sustentou decisões automatizadas de retreinamento e o chaveamento dinâmico de modelos (*Champion/Challenger*) mesmo sob carga elevada, confirmando a robustez da implementação do *AutoMLHandler*.

Por fim, a integração com ferramentas de observabilidade conferiu transparência ao comportamento do sistema. A rastreabilidade detalhada das requisições e das métricas operacionais proporcionou maior controle durante cenários de alta demanda, validando a tese de que a observabilidade é um requisito não funcional indispensável para a confiabi-

lidade de operações de *Machine Learning* em produção.

8 Conclusão

Esta pesquisa apresentou o desenvolvimento de um serviço parametrizável, materializado na forma de uma *API*, voltado ao apoio à construção de arquiteturas autoadaptativas baseadas em Inteligência Artificial. A solução proposta preenche uma lacuna relevante na integração entre engenharia de software e ciência de dados, oferecendo um mecanismo robusto para o monitoramento contínuo de modelos de aprendizado de máquina em ambientes de IoT.

A fundamentação arquitetural, alicerçada na integração de princípios de sistemas autoadaptativos, práticas de *MLOps* e na aplicação do ciclo MAPE-K, mostrou-se adequada para lidar com ambientes dinâmicos e sujeitos a mudanças constantes. A operacionalização desses conceitos permitiu a identificação proativa da degradação de desempenho (*concept drift*) por meio do *Health Index* (HI), apoiando decisões automáticas críticas, como a execução de *pipelines* de retreinamento e o chaveamento dinâmico de modelos (*Champion/Challenger*).

A validação prática, realizada no cenário de Agropecuária e Zootecnia para o monitoramento de emissões de CO_2, demonstrou a eficácia da arquitetura em recuperar a capacidade preditiva do sistema diante de desvios estatísticos. Mais do que resolver um problema específico, os resultados indicam que a API é flexível, reutilizável e aplicável a diferentes domínios. A capacidade de personalizar métricas de avaliação (como acurácia, F1-score ou precisão) e parametrizar os pesos do HI possibilita que a solução atenda aos requisitos de qualidade de diversos contextos operacionais.

Em suma, este trabalho contribui para a área de sistemas autoadaptativos com IA ao fornecer um artefato que une a detecção estatística de anomalias à supervisão humana, garantindo a continuidade operacional e a confiabilidade das inferências em sistemas de fluxo contínuo de dados.

8.1 Trabalhos Futuros

Como desdobramento desta pesquisa e visando a evolução da plataforma *AutoMLOps*, sugerem-se as seguintes vertentes de investigação:

- **Generalização via Novos Ciclos de DSR:** Condução de iterativos ciclos de *Design Science Research* em domínios distintos do agronegócio. O objetivo é avaliar a capacidade de generalização e reutilização das APIs frente a diferentes tipologias de dados e requisitos de aplicação não funcionais, validando a flexibilidade da arquitetura proposta em cenários heterogêneos.

- **Expansão Experimental:** Ampliação das avaliações experimentais através da incorporação de novas métricas de desempenho e da exploração de estratégias de adaptação alternativas. Sugere-se investigar métodos além do *Champion/Challenger*, como *Ensemble Learning* dinâmico ou aprendizado incremental, para verificar sua eficácia em ambientes de alta volatilidade.

- **Inteligência Artificial Explicável (XAI) e Auditoria:** Aprofundamento na integração de técnicas de XAI para aumentar a transparência das decisões do sistema. A proposta é que a explicabilidade não sirva apenas para diagnóstico técnico, mas atue como pilar de suporte ao ciclo *Human-in-the-Loop*, facilitando a auditoria das adaptações automáticas e aumentando a confiança do operador na autorização de trocas de modelos.

- **Refinamento por *Feedback* Humano:** Implementação de mecanismos de aprendizado que incorporem o *feedback* do operador ao ciclo de decisão. Isso permitiria que o sistema ajustasse automaticamente a sensibilidade do HI e os limiares de alerta com base no histórico de intervenções manuais, personalizando a resposta do sistema ao perfil do especialista.

- **Expansão para Edge Computing:** Adaptação dos módulos de inferência e monitoramento para execução direta em dispositivos de borda. Essa abordagem visa descentralizar a detecção de anomalias, reduzindo a latência do alerta e o consumo de largura de banda na transmissão de dados brutos.

Bibliografia

ALI, O. et al. A comprehensive review of internet of things: Technology stack, middlewares, and fog/edge computing interface. *Sensors*, v. 22, n. 3, p. 995, 2022.

ALKHABBAS, F. et al. ASSERT: A blockchain-based architectural approach for engineering secure self-adaptive IoT systems. *Sensors*, v. 22, n. 18, p. 6842, 2022.

AMERSHI, S. et al. Human-in-the-loop: Interactive machine learning. In: *Proceedings of the 2014 CHI Conference on Human Factors in Computing Systems*. Toronto: ACM, 2014. p. 3433–3442.

AMIRI, A.; ZDUN, U. Smart and adaptive routing architecture: An internet-of-things traffic manager based on artificial neural networks. *Future Generation Computer Systems*, 2023.

AUDRITO, G. FCPP: an efficient and extensible field calculus framework. In: *2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. [S.l.]: IEEE, 2020. p. 153–159.

AVILA, D. Fernandes de et al. Internet of things e inteligência artificial nos meios produtivos. *Revista Ciatec-UPF*, v. 14, n. 2, 2022.

BASSENE, A.; GUEYE, B. A self-adaptive qos-management framework for highly dynamic iot networks. *Journal of Network and Computer Applications*, 2022.

BAYRAM, F.; ALTILAR, T. Concept drift detection and adaptation in iot systems. *Future Generation Computer Systems*, v. 115, p. 293–306, 2021.

CIGNARELLA, A. et al. Human-in-the-loop machine learning: A survey and perspectives. *IEEE Access*, v. 11, p. 143275–143300, 2023.

COELHO, D. J. A. P. d. M.; DERMEVAL, I. I. B. d. M. *Mapeamento sistemático e revisão sistemática da literatura em informática na educação*. São Paulo: Sociedade Brasileira de Computação (SBC), 2019. Acesso em: 5 nov. 2025. Disponível em: ⟨https://ceie.sbc.org.br/metodologia/wp-content/uploads/2019/11/livro2_cap3.pdf⟩.

DENG, Y.; WANG, J.; LI, H. Human-in-the-loop approaches for iot-based intelligent systems: A comprehensive review. *IEEE Internet of Things Journal*, v. 10, n. 5, p. 4201–4218, 2023.

DIAS, J. P.; RESTIVO, A.; FERREIRA, H. S. Empowering visual internet-of-things mashups with self-healing capabilities. In: *2021 IEEE/ACM 3rd International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT)*. [S.l.]: IEEE, 2021. p. 44–51.

DJENNADI, L. et al. Sdn-based approach for adaptive reconfiguration of routing in iot for smart-buildings. In: *2024 IEEE 25th International Conference on High Performance Switching and Routing (HPSR)*. [S.l.]: IEEE, 2024. p. 137–142.

DURAN, F. et al. Seamless reconfiguration of rule-based iot applications. *Journal of Systems and Software*, 2021.

FAROOQ, M. U. et al. A review on internet of things (iot). *International Journal of Computer Applications*, v. 113, n. 1, 2015.

FONSECA, A. et al. Dealing with iot defiant components. *Journal of Internet Services and Applications*, 2021.

FRANGOUDIS, P. A.; REISINGER, M.; DUSTDAR, S. Recursive design for data-driven, self-adaptive iot services. *IEEE Internet of Things Journal*, 2021.

GAMA, J. et al. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, New York, v. 46, n. 4, p. 1–37, mar. 2014.

GORLA, A. et al. Achieving cost-effective software reliability through self-healing. In: IEEE. *2010 Third International Conference on Software Testing, Verification and Validation*. [S.l.], 2010. p. 255–264.

Grafana Labs. *Grafana documentation*. 2025. ⟨https://grafana.com/docs/grafana/latest/⟩. Acesso em: 16 dez. 2025.

GULDNER, A. et al. A framework for AI-based self-adaptive cyber-physical process systems. *it-Information Technology*, v. 65, n. 3, p. 113–128, 2023.

HACHICHA, M.; HALIMA, R. B.; KACEM, A. H. Modeling and specifying formally compound MAPE pattern for self-adaptive IoT systems. *Innovations in Systems and Software Engineering*, v. 18, n. 4, p. 505–521, 2022.

HALLOU, A. et al. Context-aware IoT system development approach based on meta-modeling and reinforcement learning: A smart home case study. *International Journal of Online & Biomedical Engineering*, v. 20, n. 6, 2024.

HE, X. et al. Redefinition of digital twin and its situation awareness framework designing toward fourth paradigm for energy internet of things. *Applied Energy*, 2024.

HEINZ, A. et al. Self-adaptive software architectures for dynamic environments. *Journal of Systems and Software*, v. 136, p. 1–17, 2018.

HEVNER, A. R. et al. Design science in information systems research. *MIS Quarterly*, v. 28, n. 1, p. 75–105, 2004.

HIGGINS, J. P. T.; GREEN, S. *Cochrane Handbook for Systematic Reviews of Interventions*. Chichester: John Wiley & Sons, 2011.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. *Automated Machine Learning: Methods, Systems, Challenges*. Cham: Springer Nature, 2019.

KARADUMAN, B.; TEZEL, B. T.; CHALLENGER, M. Enhancing bdi agents using fuzzy logic for cps and iot interoperability using the jaca platform. *Future Internet*, 2022.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. *Computer*, v. 36, n. 1, p. 41–50, 2003.

KODAKANDLA, N. Data drift detection and mitigation: A comprehensive MLOps approach for real-time systems. *International Journal of Science and Research Archive*, v. 12, n. 1, p. 3127–3139, 2024.

KREUZBERGER, D.; KÜHL, N.; HIRSCHL, S. Machine learning operations (mlops): Overview, definition, and architecture. *IEEE Access*, v. 11, p. 31866–31879, 2023.

LAKSHMANAN, V.; ROBINSON, S.; MUNN, M. *Machine Learning Design Patterns: Solutions to Common Challenges in Data Preparation, Model Building, and MLOps*. Sebastopol: O'Reilly Media, 2020.

LAM, A. N.; HAUGEN, O.; DELSING, J. Dynamical orchestration and configuration services in industrial IoT systems: An autonomic approach. *IEEE Open Journal of the Industrial Electronics Society*, IEEE, v. 3, p. 128–145, 2022.

LEE, E.; LEE, S.; SEO, Y.-D. Deep learning based self-adaptive framework for environmental interoperability in internet of things. In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. [S.l.: s.n.], 2022. p. 32–35.

LU, J. et al. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, v. 31, n. 12, p. 2346–2363, 2018.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. *Estudos Avançados*, v. 35, p. 85–94, 2021.

MANCINI, M. *Internet das Coisas: História, conceitos, aplicações e desafios*. [S.l.]: Project Management Institute–PMI, 2017.

MENNA, F. D.; MUCCINI, H.; VAIDHYANATHAN, K. FEAST: a framework for evaluating implementation architectures of self-adaptive IoT systems. In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. [S.l.: s.n.], 2022. p. 1440–1447.

PARSIFAL. *Parsifal: A Tool for Systematic Literature Reviews*. 2021. ⟨https://parsif.al/about/⟩. Acesso em: 03 jun. 2025.

QUIÑONERO-CANDELA, J. et al. (Ed.). *Dataset Shift in Machine Learning*. [S.l.]: MIT Press, 2009.

RESTUCCIA, F.; MELODIA, T. Deepwierl: Bringing deep reinforcement learning to the internet of self-adaptive things. *IEEE Transactions on Mobile Computing*, 2020.

ROCHA, I. F.; KISSIMOTO, K. O. Barreiras e benefícios na adoção de inteligência artificial e iot na gestão da operação. *RAM. Revista de Administração Mackenzie*, v. 23, p. eRAMR220119, 2022.

RUDENKO, R. et al. A brief review on internet of things, industry 4.0 and cybersecurity. *Electronics*, v. 11, n. 11, p. 1742, 2022.

RULLO, A. et al. Kalis2.0 — a secaas-based context-aware self-adaptive intrusion detection system for iot. *Computers & Security*, 2023.

SOARES, R. L. A. S. *ADAPTFlow: uma arquitetura orientada por dados para suporte à auto-adaptação em sistemas inteligentes*. Trabalho de Conclusão de Curso, Juiz de Fora, 2024.

SOARES, S. C. M. et al. Arquitetura de detecção de intrusão por anomalias com federated learning em redes iot. In: *Anais de evento científico*. [S.l.: s.n.], 2024.

SOUZA, L. C. P. de; FONTANARI, R. Inteligência artificial: desafios da criação, da criatividade e da autonomia humana. *Tríade: Comunicação, Cultura e Mídia*, v. 12, n. 25, p. e024015–e024015, 2024.

SUN, P. et al. A survey on privacy and security issues in iot-based environments: Technologies, protection measures and future directions. *Computers & Security*, v. 148, p. 104097, 2025.

TANG, L.; QIN, H. Divisible task offloading for multiuser multiserver mobile edge computing systems based on deep reinforcement learning. *IEEE Internet of Things Journal*, 2023.

TREVEIL, M. et al. *Introducing MLOps*. Sebastopol: O'Reilly Media, 2020.

WAKILI, A.; BAKKALI, S. Privacy-preserving security of IoT networks: A comparative analysis of methods and applications. *Cyber Security and Applications*, v. 3, p. 100084, 2025.

WANG, X. et al. Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching. *IEEE Internet of Things Journal*, 2020.

WIDMER, G.; KUBAT, M. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, Dordrecht, v. 23, n. 1, p. 69–101, 1996.

WU, X. et al. Human-in-the-loop artificial intelligence for trustworthy mlops. *IEEE Transactions on Artificial Intelligence*, v. 5, n. 3, p. 890–905, 2024.

XIAO, W. et al. Collaborative cloud-edge service cognition framework for dnn configuration toward smart iiot. *IEEE Transactions on Industrial Informatics*, 2021.

ZHAO, Z. et al. Human-in-the-loop for machine learning: Challenges and opportunities. *Pattern Recognition Letters*, v. 168, p. 56–63, 2023.