UNIVERSIDADE FEDERAL DO PARANÁ

GUILHERME YUDI TERAJIMA

CONCEPÇÃO DE UM ASSISTENTE VIRTUAL PARA AUXÍLIO NA TOMADA DE DECISÃO TÉCNICA EM AMBIENTES FABRIS

CURITIBA

2025

GUILHERME YUDI TERAJIMA

CONCEPÇÃO DE UM ASSISTENTE VIRTUAL PARA AUXÍLIO NA TOMADA DE DECISÃO TÉCNICA EM AMBIENTES FABRIS

Monografia apresentada ao curso de Graduação em Engenharia Mecânica, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Bacharel em Engenharia Mecânica.

UFPR Supervisor: Profª Dr. Pablo Deivid Valle

CURITIBA

2025

UNIVERSIDADE FEDERAL DO PARANÁ

PARECER Nº **6/2025/UFPR/R/TC/DEMEC**
PROCESSO Nº 23075.041715/2025-19
INTERESSADO: GUILHERME YUDI TERAJIMA

TERMO DE APROVAÇÃO

Título: CONCEPÇÃO DE UM ASSISTENTE VIRTUAL PARA AUXÍLIO NA TOMADA DE DECISÃO TÉCNICA EM AMBIENTES FABRIS

Autor: GUILHERME YUDI TERAJIMA

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de bacharel em Engenharia Mecânica. Aprovado pela seguinte banca examinadora:

Prof. Pablo Deivid Valle (UFPR/DEMEC) – Orientador

Prof. Harrison Lourenço Corrêa (UFPR/DEMEC)

Eng. Mateus Rigo Franco (Mestrando/PPGEM)

Curitiba, 14 de julho de 2025

Documento assinado eletronicamente por **PABLO DEIVID VALLE**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 18/07/2025, às 09:54, conforme art. 1º, III, "b", da Lei 11.419/2006.

Documento assinado eletronicamente por **MATEUS RIGO FRANCO**, **Usuário Externo**, em 18/07/2025, às 10:55, conforme art. 1º, III, "b", da Lei 11.419/2006.

Documento assinado eletronicamente por **HARRISON LOURENCO CORREA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/07/2025, às 17:16, conforme art. 1º, III, "b", da Lei 11.419/2006.

A autenticidade do documento pode ser conferida aqui informando o código verificador **7953919** e o código CRC **F1D07D89**.

Dedico este trabalho à minha família, que me apoiou e me deu confiança durante minha jornada.

Também dedico aos meus amigos, que sempre me auxiliam e trazem alegria.

AGRADECIMENTOS

O desenvolvimento deste trabalho contou com a ajuda de diversas pessoas, dentre as quais agradeço:

A Deus, pela minha saúde, pelo meu desenvolvimento pessoal e por me auxiliar a ultrapassar meus obstáculos.

Aos meus pais, Sandra Yoko e Eduardo Shindi, que sempre se esforçaram diariamente, mesmo nas mais difíceis condições, para proporcionar o melhor para mim e meus irmãos.

Aos meus avós, Setsuko, Hatsuko, Koqui e Koichi, pelos ensinamentos morais e éticos que passaram durante minha vida.

Aos meus amigos Leonardo Romano, Eduardo Tulio, João Egri, Luiza Simioni, Leonardo Canova, Maria Kmiecik, Mariane Depetriz, Yuri Salvi, Lucas Ma Famg, Michel Hilgemberg e Rafael Paraguassu, que me trouxeram grande alegria em toda essa jornada.

Ao professor Pablo Deivid, por ter sido meu orientador e ter desempenhado tal função com dedicação e amizade.

Aos meus colegas e amigos de trabalho Renato Lufchitz, Samantha Wasilewski, Pedro Mazanek, Thiago Daroz Pinheiro, Alexandre Neves, André Souza, Catarina Theodorovitz, Ciro Domingos e Fernanda Knopki, pelos ensinamentos e momentos de descontração.

A todos aqueles que contribuíram, de alguma forma, para a realização deste trabalho.

“It’s only when we truly know and understand that we have a limited time on earth - and that we have no way of knowing when our time is up - that we will then begin to live each day to the fullest, as if it was the only one we had."

(Elisabeth Kubler-Ross)

RESUMO

Novas tecnologias vêm sendo implementadas em ambientes industriais com o objetivo de aumentar a eficiência operacional, reduzir falhas técnicas e melhorar a rastreabilidade de processos. Nesse contexto, o uso de sistemas inteligentes, como assistentes virtuais baseados em *inteligência artificial (IA)*, surgiu como uma solução promissora para apoiar a tomada de decisão técnica em tempo real. No entanto, a integração entre ferramentas de automação e modelos de linguagem natural ainda enfrenta desafios técnicos, como a adaptação a contextos industriais específicos e a confiabilidade na geração de respostas. Este estudo apresenta a concepção de um sistema composto por dois módulos independentes: um conjunto de automações desenvolvido com a ferramenta Microsoft Power Automate, responsável por executar comandos técnicos, e um assistente virtual baseado em linguagem natural, construído em Python com uso de bibliotecas como *LangChain*, *FAISS* e *OpenAI*. O sistema permite que operadores interajam por meio da plataforma Microsoft Teams, selecionando comandos via *Adaptive Cards* ou enviando perguntas abertas para o assistente. Para avaliar a funcionalidade, foram simulados três fluxos automáticos (ajuste de acesso, registro de falhas e envio para auditoria), além da implementação de um modelo de diagnóstico técnico treinado com base documental da linha de montagem. Os resultados mostraram alta taxa de sucesso nas interações com o sistema, bem como clareza e assertividade nas respostas do módulo de IA, com potencial para aplicação real em linhas produtivas. A combinação das tecnologias se mostrou viável e eficaz para ambientes fabris, promovendo maior automação, autonomia operacional e suporte técnico em tempo real.

Palavras-chave: Inteligência Artificial. IA. Machine Learning. Robotic Process Automation. RPA. Processamento de Linguagem Natural. Automação de Processos. Assistente Virtual. Diagnóstico Técnico. Microsoft Power Automate. Indústria 4.0

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIAÇÕES E ACRÔNIMOS

API	Application Programming Interface
CSV	Comma-Separated Values
DL	Deep Learning
ERP	Enterprise Resource Planning
FAISS	Facebook AI Similarity Search
GPT	Generative Pre-trained Transformer
HTTP	Hypertext Transfer Protocol
IA	Inteligência Artificial
JSON	JavaScript Object Notation
LLaMA	Large Language Model Meta AI
MA	Machine Learning
MES	Manufacturing Execution System
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
OCR	Optical Character Recognition
PCA	Principal Component Analysis
POPs	Procedimentos Operacionais Padrão
REST	Representational State Transfer
RPA	Robotic Process Automation
SCADA	Supervisory Control and Data Acquisition
SVM	Support Vector Machine
UI	User Interface
XML	eXtensible Markup Language
web	World Wide Web

SUMÁRIO

1 INTRODUÇÃO

O crescente uso de tecnologias de machine learning e inteligência artificial na indústria tem despertado grande interesse das empresas, devido à necessidade de otimização de processos e desempenho superior em termos de custo, manutenção, agilidade e retorno. As IA's têm se destacado em diversas aplicações de engenharia, tais como computacional, simulações, automação, chatbots, e diversas outras áreas fabris. Estas tecnologias estão em grande crescência e ainda em desenvolvimento. A precisão das ferramentas de linguagem natural (NLP), geração e análise de imagens, por exemplo, estão em uma escala de avanço exponencial, e trazem possibilidades de aplicação em ramos além da engenharia, como análises de imagens médicas e auxílio na eficácia de diagnósticos técnicos através de chatbots.

A IA, com suas capacidades avançadas de aprendizado de máquina e análise de dados, possibilita a otimização de processos complexos e a tomada de decisões mais precisas (RUSSELL; NORVIG, 2021). Ainda que o tema traga discusões pertinentes a respeito de ética, as tecnologias de IA abrem portas para aplicações em todos os ramos sociais. Paralelamente, a RPA automatiza tarefas repetitivas e baseadas em regras, liberando os recursos humanos para atividades de maior valor agregado (AGGARWAL; GOEL, 2021). A combinação de IA e RPA tem mostrado grande potencial em diversas áreas, como finanças, saúde, manufatura e atendimento ao cliente. Especificamente, a integração de IA com RPA pode aumentar significativamente a eficiência operacional e a precisão, automatizando processos que exigem análise e interpretação de dados complexos (AGUIRRE; RODRIGUEZ, 2017).

Segundo estudo conduzido por Willcocks, Lacity e Craig (2017), a adoção de RPA e IA nas empresas tem aumentado de forma consistente, refletindo a busca por maior eficiência e insights orientados por dados (WILLCOCKS et al., 2017).

Entre as tecnologias emergentes, os sistemas de IA generativa, como os modelos de linguagem avançados, têm revolucionado a forma como as empresas interagem com clientes e processam informações. Esses sistemas são capazes de compreender e gerar texto de maneira quase indistinguível de um ser humano, abrindo novas possibilidades para automação em áreas como atendimento ao cliente, suporte técnico e criação de conteúdo (BROWN et al., 2020).

No entanto, a implementação de IA e RPA enfrenta desafios significativos, incluindo a necessidade de qualificação da força de trabalho, reorganização das atividades produtivas e questões éticas emergentes associadas à sua adoção. A falta de padrões e diretrizes claras sobre o uso responsável da IA permanece uma barreira

para sua disseminação em larga escala (BRYNJOLFSSON; MCAFEE, 2014).

Uma abordagem promissora para superar esses desafios é a adoção de frameworks de governança de IA que assegurem a transparência, responsabilidade e justiça nos sistemas automatizados. Segundo Floridi (2018), a implementação de políticas robustas de governança pode mitigar riscos e garantir que os benefícios da IA e da RPA sejam amplamente distribuídos (FLORIDI et al., 2018).

Portanto, o futuro da IA e RPA é promissor, mas depende de um equilíbrio cuidadoso entre inovação tecnológica e considerações éticas. As empresas que conseguirem integrar essas tecnologias de maneira eficaz estarão bem posicionadas para liderar em um mercado cada vez mais competitivo e dinâmico.

2 OBJETIVOS

2.1 OBJETIVOS GERAIS

O principal objetivo deste trabalho é desenvolver um assistente virtual (chatbot) com capacidade de processamento de linguagem natural, ou seja, capaz de compreender e interpretar comandos em linguagem humana, apto a receber documentações técnicas e armazenar informações relevantes. A partir dos dados técnicos inseridos, o chatbot poderá ser utilizado para a análise de falhas e o direcionamento de diagnósticos técnicos. Além disso, o sistema contará com integração a plataformas de automação, possibilitando o acesso a outros sistemas sempre que necessário. O projeto será estruturado com base na definição de um modelo de linguagem — uma arquitetura computacional treinada para compreender e gerar textos — e no processamento de dados. Também será realizada uma análise de viabilidade para a utilização de interfaces de programação de aplicações (APIs, na sigla em inglês), que são conjuntos de padrões que permitem a integração entre sistemas, com o objetivo de identificar a solução mais adequada às necessidades do projeto.

2.2 OBJETIVOS ESPECÍFICOS

1. Deselvolver um sistema de automação de processos de sistemas e páginas *web*
2. Desenvolver e/ou aplicar um modelo de processamento de linguagem natural (NLP);
3. Treinar o modelo escolhido com documentações técnicas diversas, a fim de realizar diagnósticos de falhas;
4. Apresentar de forma independente as capacidades de automação técnica e de diagnóstico inteligente por linguagem natural, destacando sua aplicabilidade futura integrada;
5. Analisar a viabilidade de implementação do resultado final.

3 REVISÃO BIBLIOGRÁFICA

3.1 INTELIGÊNCIA ARTIFICIAL

A área da ciência da computação conhecida como inteligência artificial (IA) estuda e propõe a criação de dispositivos computacionais capazes de reproduzir características do intelecto humano, como perceber, raciocinar, resolver problemas e tomar decisões. Ao aprender e se adaptar aos dados, os sistemas de IA podem melhorar ao longo do tempo. Segundo Nilsson (1998), uma definição comum de IA é "a atividade dedicada a tornar as máquinas inteligentes"(NILSSON, 1998).

Por meio do uso de algoritmos e processamento de dados, a IA principalmente tenta reproduzir as funções cognitivas humanas. A capacidade de analisar conjuntos de dados complexos e identificar padrões é uma característica importante da IA, que pode ser usada em uma variedade de aplicações, como análises de imagens e diagnósticos técnicos. O potencial para aproveitar as capacidades da IA em várias indústrias cresce exponencialmente como resultado do avanço dessas tecnologias. Isso reformulará a maneira como interagimos com a tecnologia e com o mundo ao nosso redor (BRYNJOLFSSON; MCAFEE, 2014).

3.2 O SURGIMENTO DA INTELIGÊNCIA ARTIFICIAL

A origem da Inteligência Artificial (IA) remonta às primeiras tentativas de compreender e formalizar processos cognitivos humanos por meio de modelos computacionais. Desde o período pós-Segunda Guerra Mundial, pesquisadores das áreas de matemática, lógica e engenharia passaram a explorar a possibilidade de construir máquinas capazes de simular o raciocínio humano. Com os avanços na eletrônica e na ciência da computação, a década de 1950 consolidou-se como marco inicial da IA como campo de estudo.

O termo "inteligência artificial"foi oficialmente cunhado por John McCarthy durante a Conferência de Dartmouth, em 1956, considerada o ponto de partida da pesquisa formal na área (MCCARTHY et al., 2006). Desde então, a IA tem evoluído significativamente, passando de sistemas baseados em regras simples para modelos probabilísticos e, mais recentemente, arquiteturas baseadas em redes neurais profundas.

Essa trajetória foi impulsionada por três pilares fundamentais: a disponibilidade de grandes volumes de dados, o aumento da capacidade computacional e o desenvolvimento de algoritmos cada vez mais sofisticados. Tais avanços permitiram que

a IA deixasse de ser um conceito puramente teórico e se tornasse uma ferramenta aplicada em diferentes setores, como diagnóstico médico, análise financeira, veículos autônomos e controle industrial.

No setor produtivo, a aplicação da IA tem promovido ganhos em eficiência, qualidade e rastreabilidade, ao permitir análises preditivas e automação inteligente de processos. Contudo, seu avanço também levanta questionamentos éticos e técnicos, relacionados à autonomia das máquinas, privacidade dos dados e impacto sobre o trabalho humano. Compreender os marcos históricos e as bases conceituais da IA é, portanto, essencial para avaliar seu papel transformador na sociedade contemporânea.

3.3 DO TESTE DE TURING AOS PRIMEIROS PROGRAMAS DE IA

O teste de Turing, proposto por Alan Turing em 1950, é considerado uma das primeiras formulações teóricas a questionar a possibilidade de inteligência nas máquinas. Publicado no artigo “Computing Machinery and Intelligence”, o teste consiste em avaliar se uma máquina é capaz de produzir respostas indistinguíveis das humanas em uma conversa escrita com um avaliador humano (TURING, 1950). Embora o teste tenha limitações como critério definitivo de inteligência, ele representou um marco ao deslocar o debate da viabilidade técnica para a análise do comportamento observável.

Inspirados por essa provocação conceitual, pesquisadores começaram a desenvolver programas que tentavam simular o pensamento humano. Entre os primeiros sistemas notáveis estão o Logic Theorist, desenvolvido por Allen Newell e Herbert Simon em 1956, capaz de provar teoremas matemáticos; e o General Problem Solver (GPS), criado em 1957 para simular a resolução de problemas por meio de regras heurísticas (RUSSELL; NORVIG, 2021; NEWELL; SIMON, 1956; NEWELL et al., 1959).

Esses programas foram os precursores da chamada IA simbólica, que buscava representar o conhecimento por meio de regras explícitas e estruturas lógicas. Embora limitados pelas capacidades computacionais da época, tais sistemas lançaram as bases para o desenvolvimento de linguagens de programação especializadas, como LISP, e abriram espaço para aplicações mais complexas nas décadas seguintes (MCCARTHY, 1960).

Além da vertente técnica, a inteligência artificial começou a ser debatida em outros domínios, como o jurídico e o sociológico. Alguns autores, como Aldama et al. (ALDAMA et al., 2022), exploram a representação cultural e a atribuição de personalidade jurídica a sistemas de IA, especialmente em narrativas de ficção científica. Outros, como Sugiyama et al. (SUGIYAMA et al., 2021), discutem a busca por simulação emocional e autenticidade em tecnologias que interagem com seres humanos. Embora essas abordagens estejam mais relacionadas ao campo das humanidades, elas enriquecem

o debate sobre os limites, possibilidades e implicações da IA no cotidiano.

3.4 ARQUITETURA GERAL DA INTELIGÊNCIA ARTIFICIAL

A inteligência artificial (IA) pode ser compreendida como um campo multidisciplinar, cuja arquitetura geral é sustentada por três pilares centrais, sendo eles os dados, o algoritmos e a capacidade computacional. A eficácia de um sistema inteligente depende do equilíbrio entre a qualidade dos dados, a sofisticação dos métodos utilizados para interpretá-los e o desempenho dos dispositivos que os processam (RUSSELL; NORVIG, 2021). Historicamente, duas grandes abordagens se destacaram: a inteligência artificial simbólica e a conexionista.

A IA simbólica, também conhecida como “boa e velha IA” (Good Old-Fashioned Artificial Intelligence — GOFAI), baseia-se na representação explícita do conhecimento por meio de regras lógicas e inferências formais. Nesse modelo, o sistema armazena fatos e opera sobre eles utilizando mecanismos dedutivos, como silogismos e cadeias lógicas. Sistemas especialistas, muito utilizados nas décadas de 1980 e 1990, são exemplos clássicos dessa abordagem. Apesar de sua clareza e previsibilidade, esses sistemas são limitados em ambientes com alta variabilidade ou ambiguidade contextual, como linguagem natural (NILSSON, 1998).

Em contrapartida, a IA conexionista foi inspirada no funcionamento do cérebro humano e ganhou protagonismo com o avanço das redes neurais artificiais. Essa abordagem trabalha com representações distribuídas e aprendizado por exemplos, dispensando regras explícitas. As conexões entre unidades computacionais são ajustadas com base em padrões extraídos de grandes volumes de dados. Em vez de seguir instruções definidas manualmente, o sistema é treinado para identificar relações e padrões por conta própria. As redes neurais artificiais — e, em especial, as redes profundas (deep learning) — tornaram-se a base dos sistemas de IA mais avançados da atualidade, especialmente na interpretação de linguagem natural, visão computacional e reconhecimento de padrões complexos (GOODFELLOW et al., 2016).

A escolha entre abordagens simbólicas e conexionistas depende do contexto da aplicação. Em alguns casos, soluções híbridas são adotadas, combinando o raciocínio lógico da IA simbólica com a capacidade de generalização das redes neurais (RUSSELL; NORVIG, 2021).

Além da arquitetura conceitual, o funcionamento da IA moderna depende fortemente da relação entre três componentes fundamentais: os dados, os algoritmos e a infraestrutura computacional. Os dados alimentam os sistemas e fornecem os exemplos sobre os quais os modelos aprenderão. Os algoritmos definem a lógica de como esse aprendizado ocorre — sejam eles baseados em aprendizado supervisionado,

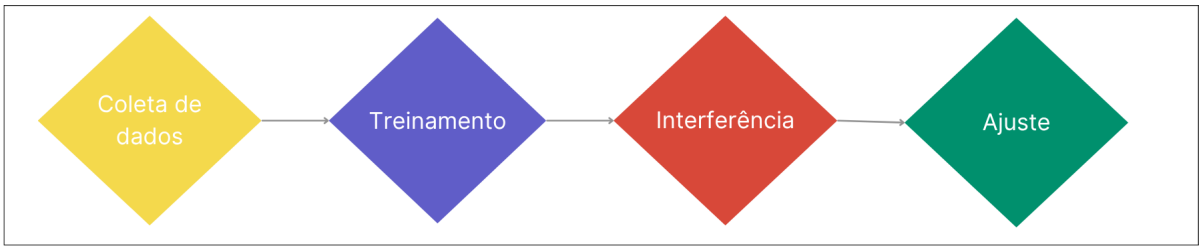
não supervisionado ou por reforço. Já a infraestrutura computacional, especialmente com o uso de GPUs e TPUs, viabiliza o treinamento de modelos em larga escala com tempos de resposta aceitáveis (SUTTON; BARTO, 2018; LECUN et al., 2015).

O processo de desenvolvimento de um modelo de IA segue um ciclo básico composto por quatro etapas principais: coleta de dados, treinamento, inferência e ajuste. Inicialmente, dados são reunidos e organizados de forma a representar o fenômeno que se deseja modelar. Durante a etapa de treinamento, o modelo é alimentado com esses dados e ajusta seus parâmetros internos para minimizar o erro entre a saída prevista e a esperada (RUSSELL; NORVIG, 2021; SUTTON; BARTO, 2018).

Na fase de inferência, o modelo já treinado é utilizado para fazer previsões ou tomar decisões com base em novos dados, que não foram vistos durante o treinamento. Com o tempo, o desempenho do modelo pode ser degradado por mudanças nos dados reais (fenômeno conhecido como *drift*), exigindo ajustes periódicos — como reentrenamento, regularização ou ampliação da base de conhecimento (GAMA et al., 2014).

Esse ciclo, embora conceitualmente simples, é complexo em sua implementação prática, demandando cuidados com viés de dados, sobreajuste, generalização e validação dos resultados (DOMINGOS, 2012; LECUN et al., 2015). A Figura 1 ilustra esse processo de forma esquemática.

FIGURE 1 – CICLO DE VIDA DE UM MODELO IA.



SOURCE: O AUTOR, 2024

Conforme argumentam Russell e Norvig (RUSSELL; NORVIG, 2021), o sucesso de um sistema inteligente não depende apenas da sofisticação do algoritmo, mas de sua capacidade de aprender com dados representativos e de se adaptar a variações do ambiente. Essa característica torna a IA uma ferramenta poderosa para aplicações industriais, em que o volume e a diversidade dos dados aumentam continuamente.

3.5 APRENDIZADO DE MÁQUINA E REDES NEURAIS: FUNDAMENTOS DO DESENVOLVIMENTO DA IA

As redes neurais artificiais representam um dos principais avanços da Inteligência Artificial moderna, sendo amplamente utilizadas em tarefas como classificação de imagens, reconhecimento de voz, detecção de anomalias e processamento de linguagem natural. Inspiradas em aspectos da neurociência, essas redes são compostas por unidades denominadas neurônios artificiais, que simulam o comportamento de seus equivalentes biológicos (GOODFELLOW et al., 2016).

Um neurônio artificial é, essencialmente, uma função matemática que recebe múltiplas entradas (inputs), aplica um conjunto de pesos a essas entradas, soma os resultados e então processa essa soma através de uma função de ativação. A saída obtida é transmitida para os neurônios da próxima camada da rede. Esse modelo simples, conhecido como Perceptron, foi introduzido por Rosenblatt na década de 1950 e representa a base conceitual das redes mais complexas atuais (ROSENBLATT, 1958).

A estrutura básica de uma rede neural é organizada em três tipos principais de camadas: a camada de entrada, que recebe os dados iniciais; uma ou mais camadas ocultas, onde ocorrem os principais cálculos e transformações; e a camada de saída, que gera o resultado final. Quanto maior o número de camadas ocultas e de neurônios por camada, maior é a profundidade da rede — característica que define as chamadas redes neurais profundas (deep neural networks) (GOODFELLOW et al., 2016).

Cada conexão entre neurônios é associada a um peso, que determina a importância relativa da entrada no processamento. Além dos pesos, cada neurônio possui um parâmetro chamado *bias*, responsável por ajustar o ponto de ativação da função. O conjunto desses parâmetros (pesos e bias) é o que define o comportamento da rede e precisa ser ajustado durante o processo de treinamento (GOODFELLOW et al., 2016).

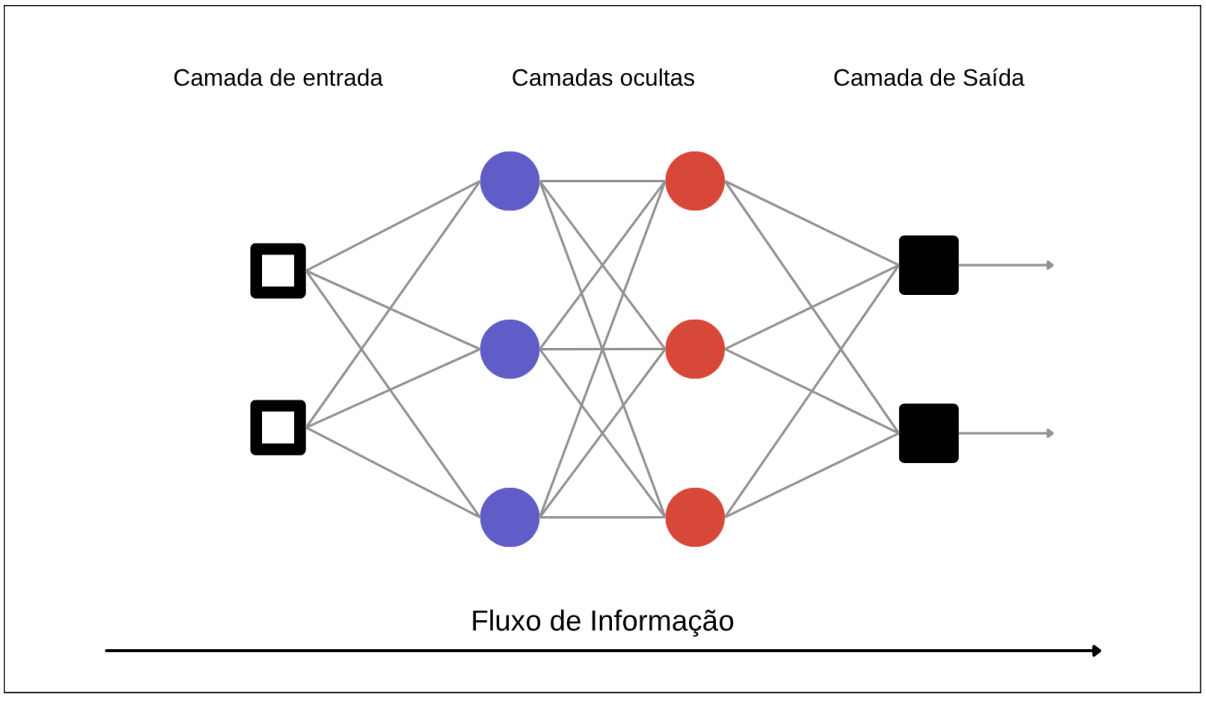
A função de ativação desempenha um papel crucial nesse processo, pois introduz não linearidade ao sistema, permitindo que a rede resolva problemas complexos que não podem ser descritos apenas por relações lineares. As funções mais comuns incluem a ReLU (Rectified Linear Unit), que retorna zero para valores negativos e o valor original para positivos, e a Sigmoid, que limita a saída entre 0 e 1. A escolha da função de ativação influencia diretamente na capacidade de generalização e na velocidade de convergência do modelo (NWANKPA et al., 2018).

Durante o treinamento da rede, um algoritmo de otimização ajusta iterativamente os pesos e bias com o objetivo de minimizar o erro entre a saída prevista e a saída real. Esse processo é conhecido como retropropagação (*backpropagation*), uma

técnica introduzida formalmente por Rumelhart, Hinton e Williams (RUMELHART et al., 1986) que calcula o gradiente do erro em relação a cada parâmetro da rede e atualiza os pesos no sentido de reduzir esse erro. O método de otimização mais utilizado é o Gradiente Descendente Estocástico (SGD – Stochastic Gradient Descent), embora variações como Adam e RMSprop também sejam amplamente aplicadas em redes mais profundas (KINGMA; BA, 2014).

Para ilustrar esse funcionamento, pode-se considerar o exemplo da classificação de imagens em um sistema de inspeção industrial. Suponha-se que uma câmera capte imagens de um componente mecânico e que o objetivo seja determinar, a partir dessas imagens, se há presença de rachaduras. A rede recebe como entrada os valores dos pixels da imagem (normalmente convertidos em tons de cinza), realiza sucessivas transformações nas camadas ocultas e, por fim, retorna uma probabilidade de que a imagem corresponda a uma peça defeituosa. Com milhares de exemplos rotulados (com ou sem falhas), a rede aprende a identificar padrões visuais característicos de anomalias (LECUN et al., 2015).

FIGURE 2 – DIAGRAMA DE ARQUITETURA DE REDE NEURAL.



SOURCE: O AUTOR, 2024

Esse tipo de aplicação mostra como as redes neurais têm se tornado ferramentas valiosas na indústria, auxiliando não apenas em tarefas operacionais, mas também na tomada de decisão técnica. No entanto, para que esse processo ocorra de forma eficiente, é necessário garantir a disponibilidade de uma base de dados representativa,

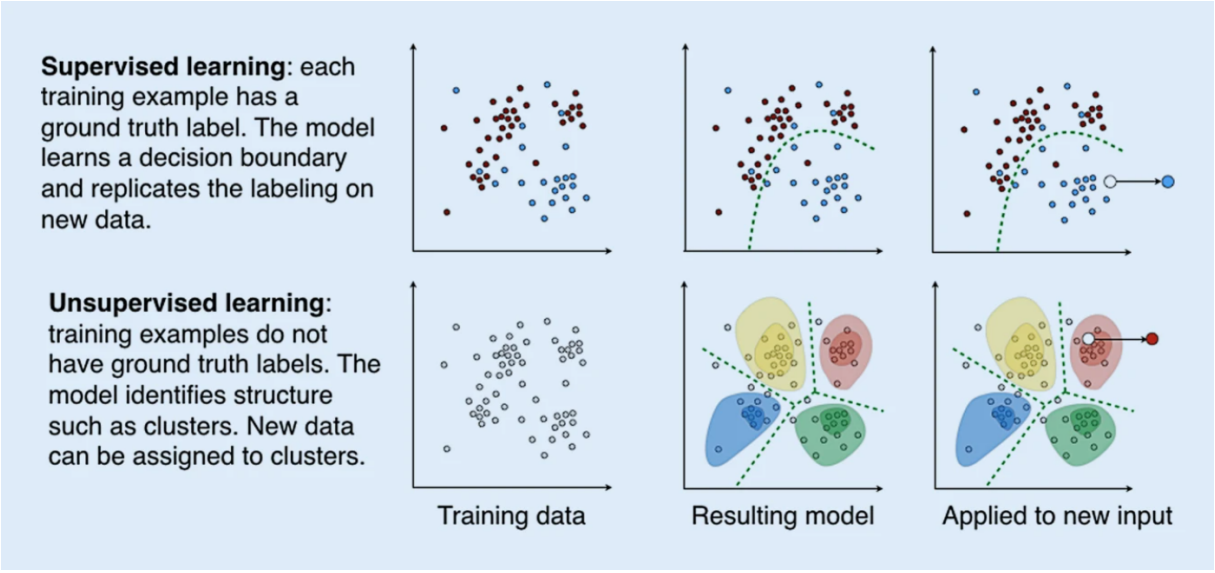
bem como recursos computacionais adequados para o treinamento e a validação do modelo.

O desempenho da rede depende de diversos fatores, como a quantidade e a qualidade dos dados de entrada, o número de camadas e neurônios, a função de ativação escolhida e os parâmetros de treinamento. Além disso, é fundamental realizar etapas de validação cruzada e regularização para evitar o sobreajuste (overfitting), que ocorre quando o modelo memoriza os dados de treinamento, mas falha ao generalizar para novos casos (GOODFELLOW et al., 2016; DOMINGOS, 2012).

Autores como Goodfellow, Bengio e Courville (GOODFELLOW et al., 2016) destacam que redes neurais profundas — compostas por várias camadas ocultas — são especialmente eficazes em tarefas onde a representação hierárquica dos dados é relevante. Isso inclui aplicações com imagens, sinais temporais e linguagem natural, o que justifica seu uso crescente em sistemas industriais e de suporte técnico.

Além das redes neurais, a Inteligência Artificial moderna é impulsionada por diferentes formas de aprendizado de máquina, classificadas de acordo com a forma como o algoritmo interage com os dados. Três abordagens principais se destacam, sendo elas o aprendizado supervisionado, o não supervisionado e por reforço.

FIGURE 3 – MODELO DE APRENDIZADO SUPERVISIONADO E NÃO SUPERVISIONADO.



SOURCE: Adaptado de Langs et al. (LANGS et al., 2018), disponível sob licença CC BY 4.0.

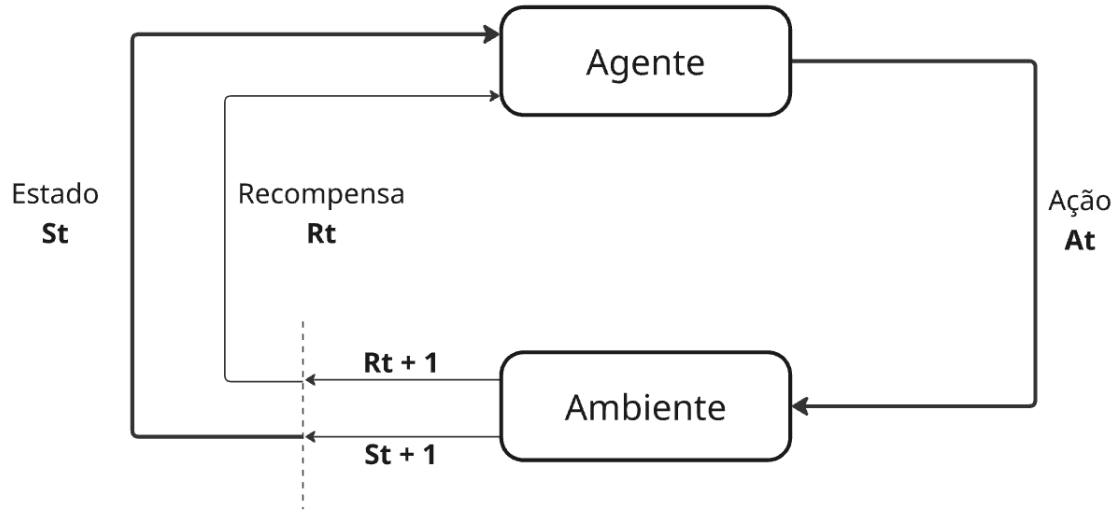
O aprendizado supervisionado ocorre quando o modelo é treinado com base em um conjunto de dados rotulado, ou seja, onde cada entrada está associada a uma saída esperada. A tarefa do algoritmo é encontrar padrões que relacionem essas entradas às suas respectivas saídas, de forma que, ao receber novos dados, seja

capaz de prever corretamente o resultado. Essa abordagem é amplamente utilizada em problemas de regressão e classificação, sendo exemplos típicos a previsão de falhas em equipamentos e a identificação de anomalias em sensores. Entre os algoritmos mais conhecidos desse grupo destacam-se a Regressão Linear, Regressão Logística, Support Vector Machines (SVM), Árvores de Decisão e Redes Neurais Multicamadas (MLP – Multi-Layer Perceptron) (HASTIE et al., 2009).

No aprendizado não supervisionado, por outro lado, os dados fornecidos ao modelo não contêm rótulos. O objetivo do algoritmo é descobrir estruturas ou agrupamentos ocultos dentro dos dados, sem uma saída explícita para guiar o processo. Essa técnica é utilizada em tarefas de agrupamento (clustering), detecção de padrões e redução de dimensionalidade. Um exemplo clássico de aplicação industrial seria o agrupamento de padrões de consumo de energia em diferentes turnos de uma fábrica. Entre os algoritmos mais comuns estão o K-Means, DBSCAN, Hierarchical Clustering e PCA (Principal Component Analysis) (MURPHY, 2012).

O aprendizado por reforço, por sua vez, representa uma abordagem distinta, em que o agente aprende por meio da interação com o ambiente. A cada ação tomada, ele recebe um feedback na forma de recompensa ou penalidade, e ajusta sua política de comportamento com o objetivo de maximizar o retorno acumulado ao longo do tempo. Essa técnica tem sido aplicada com sucesso em problemas de controle, como robótica industrial, planejamento de produção e sistemas autônomos. Um algoritmo amplamente estudado nesse contexto é o Q-Learning, que permite ao agente aprender a melhor sequência de ações com base em uma matriz de estados e recompensas (SUTTON; BARTO, 2018).

FIGURE 4 – FLUXOGRAMA DE MODELO Q-LEARNING.



SOURCE: Adaptado de Sutton e Barto (SUTTON; BARTO, 2018).

Neste modelo, o *agente* é responsável por tomar decisões com base em um determinado *estado* S_t do ambiente. A cada passo de tempo, o agente executa uma *ação* A_t, que afeta o ambiente e resulta em uma *recompensa* R_t e uma transição para um novo *estado* S_{t+1}.

O algoritmo *Q-learning* visa aprender uma política ótima que maximize a recompensa acumulada ao longo do tempo. Para isso, é utilizada uma função de valor de ação $Q(s, a)$, que estima o valor de realizar a ação a no estado s. A atualização dessa função segue a equação:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

em que:

- α é a taxa de aprendizado,
- γ é o fator de desconto (representando a importância da recompensa futura),
- r_t é a recompensa imediata recebida após a ação a_t,
- $\max_{a'} Q(s_{t+1}, a')$ representa a estimativa do valor da melhor ação no próximo estado.

O agente, então, aprende por tentativa e erro, explorando o ambiente e ajustando sua política com base nos valores de $Q(s, a)$, mesmo sem conhecimento prévio do modelo do ambiente (SUTTON; BARTO, 2018). Essa abordagem tem sido aplicada com sucesso em jogos, controle autônomo de robôs e otimização de processos industriais adaptativos.

As três abordagens são complementares e cada uma possui vantagens específicas dependendo do problema em questão. Enquanto o aprendizado supervisionado é indicado para tarefas com dados históricos bem definidos, o aprendizado não supervisionado é útil em fases exploratórias, e o reforço é adequado para situações onde a resposta ótima só pode ser obtida após múltiplas interações com o sistema (MURPHY, 2012; SUTTON; BARTO, 2018).

3.6 MODELOS PRÉ-TREINADOS E APRENDIZADO TRANSFERIDO

O desenvolvimento recente da Inteligência Artificial tem sido fortemente impulsionado pelo uso de modelos pré-treinados em larga escala. Essa abordagem, conhecida como aprendizado transferido (*transfer learning*), consiste em utilizar modelos previamente treinados em bases de dados extensas para resolver novos problemas, com pouca ou nenhuma necessidade de recomeçar o treinamento do zero (RUDER, 2019).

Um dos principais motivos para essa prática é o alto custo computacional associado ao treinamento de modelos complexos. Por exemplo, o treinamento de um modelo como o GPT-3 envolve centenas de bilhões de parâmetros e exige semanas de processamento em infraestruturas distribuídas (BROWN et al., 2020). Ao reutilizar modelos já treinados, torna-se possível aplicar o conhecimento previamente adquirido a contextos específicos com um custo consideravelmente reduzido.

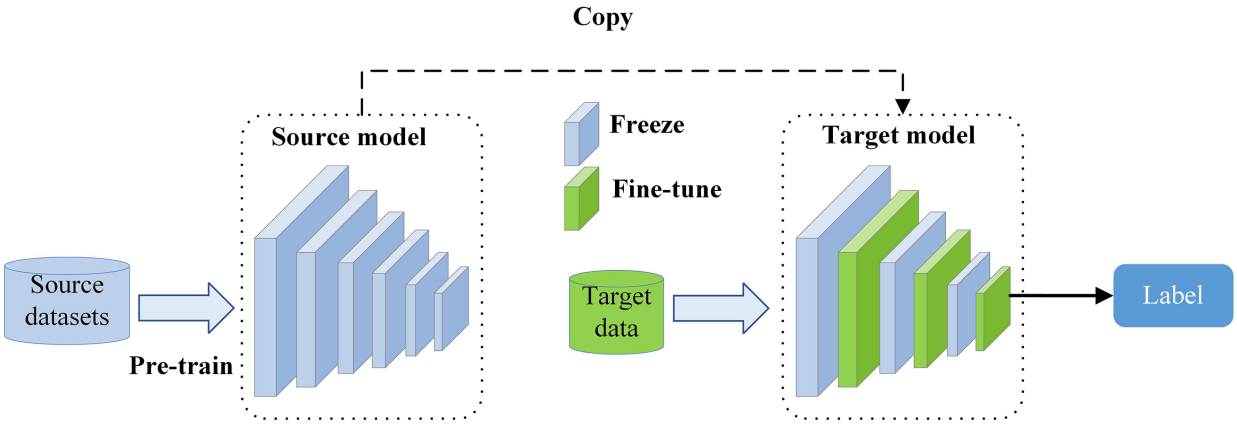
Esses modelos são normalmente ajustados para tarefas específicas por meio de um processo chamado *fine-tuning*, no qual o modelo recebe novos exemplos de um domínio restrito e ajusta seus parâmetros com base nesse novo contexto. Isso permite que o modelo generalista torne-se mais sensível às particularidades do ambiente em que será aplicado. O *fine-tuning* pode variar em intensidade: desde pequenos ajustes nas camadas finais da rede até o reprocessamento completo com nova base de dados (HOWARD; RUDER, 2018).

Um componente essencial para o funcionamento desses modelos são os *embeddings*, que representam elementos complexos — como palavras, frases ou até imagens — em vetores de números reais de dimensão fixa. Esses vetores capturam propriedades semânticas dos dados, permitindo que relações de similaridade, contexto e hierarquia sejam aprendidas (MIKOLOV et al., 2013). Por exemplo, em um espaço vetorial bem treinado, os vetores associados às palavras “motor” e “virabrequim” estarão mais próximos do que os vetores de “motor” e “óculos”.

Essa representação vetorial torna possível que modelos como o GPT-3.5 ou o BERT compreendam e gerem linguagem natural com fluência, mesmo em contextos técnicos. Esses modelos utilizam arquiteturas chamadas *transformers*, introduzidas por Vaswani et al. (VASWANI et al., 2017), que se diferenciam por seu mecanismo de atenção, capaz de avaliar a relevância de diferentes partes de uma entrada textual simultaneamente. Com isso, o modelo pode atribuir pesos dinâmicos às palavras, considerando seu papel dentro de uma sentença e em relação ao contexto global do texto.

O uso de modelos pré-treinados com aprendizado transferido tem sido particularmente útil em contextos com bases de dados limitadas ou com necessidade de rápida adaptação. Em aplicações industriais, permite a construção de sistemas de diagnóstico técnico, assistentes virtuais e classificadores de falhas com custo e tempo de desenvolvimento reduzidos, sem perda de precisão.

FIGURE 5 – REPRESENTAÇÃO GRÁFICA DO PROCESSO DE FINE TUNING EM UM DOMÍNIO DE FONTE ÚNICA.



SOURCE: https://doi.org/10.7717/peerjcs.2107/fig-1

3.7 DEFINIÇÃO DE IA, SUA LÓGICA E ESTRUTURA FUZZY

A lógica fuzzy (*fuzzy logic*), ou lógica difusa, é uma extensão da lógica clássica proposta por Lotfi Zadeh em 1965, cujo objetivo é lidar com situações de incerteza e imprecisão presentes em problemas do mundo real (ZADEH, 1965). Diferentemente da lógica booleana tradicional, que trabalha com valores binários estritos (0 ou 1), a lógica fuzzy permite a atribuição de graus de verdade, expressos por valores contínuos no intervalo entre 0 e 1. Esse modelo se mostra útil na representação de conceitos vagos ou subjetivos, como “temperatura ligeiramente alta” ou “pressão moderada”, oferecendo maior aderência à linguagem natural humana e à variabilidade dos sistemas físicos.

No contexto da inteligência artificial, a lógica fuzzy é aplicada na modelagem de sistemas em que os dados são incertos, incompletos ou ruidosos, tornando-se uma alternativa valiosa para o raciocínio aproximado em ambientes de controle, previsão e diagnóstico (ROSS, 2004). O uso de funções de pertinência (*membership functions*) e operadores fuzzy permite o tratamento de variáveis linguísticas e a construção de regras do tipo “SE...ENTÃO”, conferindo ao sistema capacidade de tomada de decisão mesmo diante de ambiguidade.

A integração entre lógica fuzzy e redes neurais artificiais resulta em sistemas neuro-fuzzy, capazes de combinar a aprendizagem adaptativa das redes com a representação flexível da lógica fuzzy. Essa fusão oferece um arcabouço robusto para reconhecimento de padrões, classificação e controle em sistemas complexos e dinâmicos (KOSKO, 1992). Em aplicações industriais, esses sistemas têm sido empregados no controle de processos, previsão de falhas, sistemas especialistas e na construção de agentes autônomos.

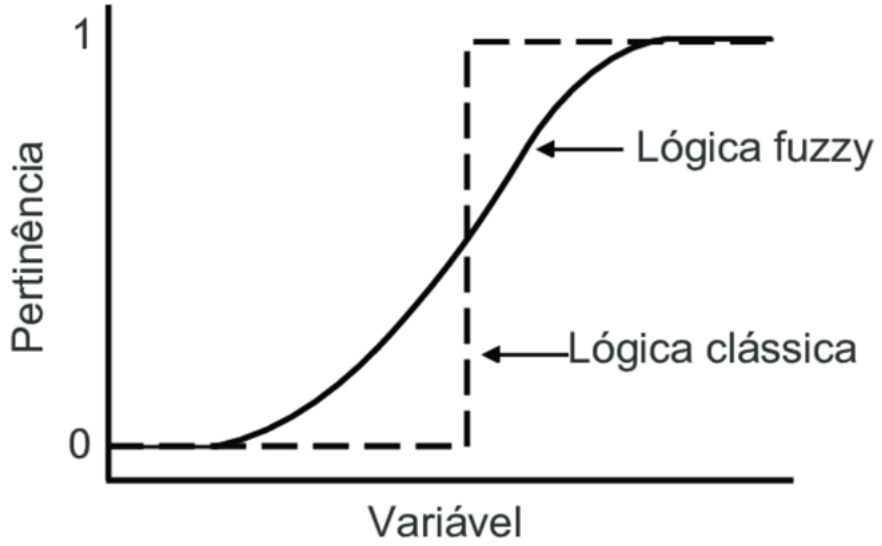
A literatura apresenta diferentes arquiteturas para sistemas híbridos, como o

modelo ANFIS (*Adaptive Neuro-Fuzzy Inference System*), que utiliza estruturas de redes neurais para otimizar os parâmetros de um sistema fuzzy de inferência. Tais modelos têm se destacado pela capacidade de realizar inferência lógica com base em dados históricos e adaptar-se automaticamente a novos cenários (MENDEL, 1995).

Na área de diagnóstico técnico, como ilustrado neste trabalho, a lógica fuzzy pode ser empregada para atribuir pesos ou graus de confiança a sintomas relatados e sugerir ações corretivas com base em múltiplas evidências incompletas ou imprecisas. Além disso, ao ser combinada com técnicas de aprendizado de máquina, como *clustering* e algoritmos evolutivos, a lógica fuzzy potencializa a criação de sistemas de apoio à decisão mais resilientes e interpretáveis.

A adoção de uma lógica fuzzy, quando combinada a mecanismos conexionistas, permite não apenas a melhoria da acurácia em tarefas específicas, mas também a incorporação de conhecimento humano de forma transparente, por meio de regras linguísticas. Dessa forma, a lógica fuzzy não apenas amplia a capacidade de representação da IA, mas também aproxima o raciocínio computacional da lógica humana, contribuindo para sistemas mais flexíveis, explicáveis e confiáveis.

FIGURE 6 – DIAGRAMA CONCEITUAL DE UM SISTEMA FUZZY COM ENTRADA, BASE DE REGRAS E INFERÊNCIA.



SOURCE: O AUTOR, 2024

3.8 AUTOMAÇÃO DE PROCESSOS ROBÓTICOS

A Automação de Processos Robóticos, ou *Robotic Process Automation* (RPA), representa uma abordagem tecnológica voltada à automação de tarefas repetitivas e baseadas em regras, tradicionalmente realizadas por operadores humanos em siste-

mas digitais. Trata-se de uma tecnologia que emprega robôs de software — comumente denominados *bots* — para imitar a interação humana com interfaces gráficas, como clicar em botões, preencher formulários, mover arquivos e acessar sistemas corporativos. A proposta da RPA é substituir o esforço humano em tarefas rotineiras, reduzindo erros operacionais e aumentando a produtividade organizacional (AGGARWAL; GOEL, 2021).

Embora frequentemente associada à inteligência artificial (IA), a RPA se diferencia por sua natureza determinística. Os *bots* não aprendem ou adaptam seu comportamento de forma autônoma, mas sim executam sequências pré-definidas de comandos, conforme instruções estabelecidas durante o processo de desenvolvimento do fluxo. Enquanto a IA busca simular capacidades cognitivas humanas como raciocínio e aprendizado, a RPA opera com foco na eficiência operacional, sem inferência ou análise contextual complexa (BHATTACHARYYA et al., 2023).

Ainda assim, ambas as tecnologias podem ser combinadas em soluções conhecidas como *RPA inteligente* ou *hyperautomation*, nas quais algoritmos de IA são integrados à lógica da RPA para permitir tomada de decisão com base em dados dinâmicos. Essa sinergia é especialmente útil em processos que exigem, além da execução de tarefas, análise contextual, interpretação de documentos ou classificação de informações não estruturadas (KUTUKOV et al., 2023).

Na prática, a RPA tem sido aplicada com sucesso em áreas como auditoria contábil, gestão de documentos, atendimento ao cliente, e ambientes industriais. Segundo Czarnecki e Fettke (CZARNECKI; FETTKE, 2021), os benefícios tangíveis da adoção da RPA incluem redução de custos operacionais, aumento da acurácia e melhoria da rastreabilidade dos processos. Em ambientes fabris, o uso da RPA pode auxiliar na integração entre sistemas legados, atualização automática de registros em bancos de dados, e execução de tarefas administrativas repetitivas, como geração de relatórios de produção.

A literatura recente aponta que a RPA também se mostra promissora em setores educacionais e de auditoria. Em instituições de ensino, a tecnologia tem sido utilizada para automatizar atividades administrativas, melhorando a experiência dos alunos e reduzindo a sobrecarga dos funcionários (KUTUKOV et al., 2023). Já na auditoria, a RPA permite a execução automática de verificações contábeis e fiscais, com ganho significativo em agilidade e conformidade regulatória (FIGUEROA MORENO et al., 2023).

No escopo deste trabalho, a RPA foi explorada por meio da plataforma Microsoft Power Automate, que oferece suporte nativo à criação de fluxos de automação em ambientes Windows e web. Sua proposta *low-code* facilita a implementação de tarefas automatizadas sem a necessidade de programação tradicional, tornando-a acessível a

usuários de perfil técnico intermediário (MACDONALD, 2020). O uso dessa ferramenta permitiu simular, de forma prática e controlada, aplicações reais de RPA no contexto da linha de montagem de motores, como alteração de acessos de usuários, registro de peças não conformes e envio automático de relatórios para auditoria.

Com o avanço das tecnologias de integração e a crescente digitalização dos processos industriais, a RPA tende a ocupar um papel cada vez mais relevante na transição para a Indústria 4.0. Sua capacidade de replicar ações humanas com rapidez e precisão, aliada à possibilidade de integração com sistemas cognitivos, torna essa abordagem uma das mais promissoras para ganhos de produtividade e confiabilidade em operações repetitivas.

3.9 COMUNICAÇÃO ENTRE SISTEMAS: PROTOCOLOS *HTTP* E *WEBHOOKS*

A comunicação entre sistemas heterogêneos é um elemento central na construção de soluções automatizadas, especialmente em arquiteturas distribuídas e ambientes fabris integrados. A padronização dessa comunicação é viabilizada por protocolos amplamente adotados, entre os quais se destaca o *Hypertext Transfer Protocol* (HTTP), utilizado como base para requisições e respostas entre clientes e servidores.

O protocolo HTTP define um conjunto de métodos para o tratamento de dados e operações, sendo os principais: *GET* (consulta de dados), *POST* (envio ou criação de dados), *PUT* (atualização) e *DELETE* (remoção de dados). Esses métodos são empregados por sistemas que seguem a arquitetura REST (*Representational State Transfer*), descrita por Fielding (FIELDING, 2000), a qual estabelece princípios para interfaces web escaláveis e modulares.

Em conjunto com o HTTP, formatos de dados como JSON (*JavaScript Object Notation*) e XML (*eXtensible Markup Language*) são utilizados para estruturar informações de forma padronizada, legível e leve. A predominância do JSON em sistemas modernos se deve à sua simplicidade e compatibilidade com múltiplas linguagens de programação e plataformas de integração (OETIKER et al., 2020).

No contexto da automação de processos, especialmente aqueles baseados em eventos, destaca-se o uso de *webhooks* como mecanismos eficientes de notificação assíncrona. Um webhook é, essencialmente, um endereço de escuta (URL) configurado para receber chamadas HTTP sempre que um evento específico ocorre em outro sistema. Diferentemente das consultas periódicas (*polling*), nas quais o cliente verifica constantemente o servidor em busca de atualizações, os webhooks operam de forma reativa: o servidor envia a notificação de forma imediata assim que o evento ocorre, acionando fluxos automatizados em tempo real (CORPORATION, 2023).

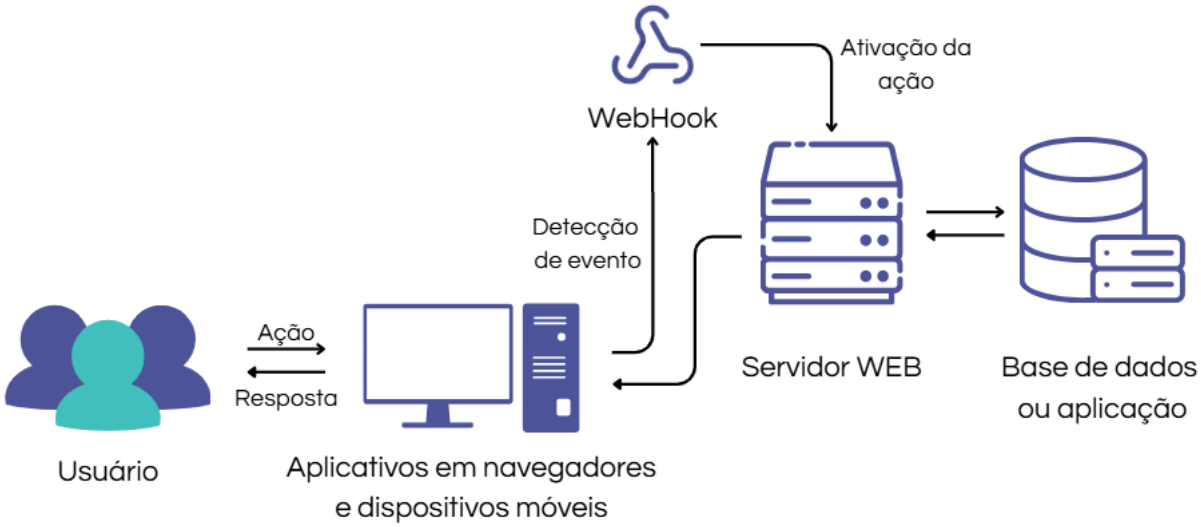
Em soluções como o Microsoft Power Automate, os webhooks são utilizados

para acionar fluxos sempre que um gatilho externo — como o envio de uma mensagem no Microsoft Teams — é detectado. Essa estrutura permite que a lógica automatizada responda dinamicamente a ações dos usuários, sem necessidade de supervisão contínua ou verificação manual. A arquitetura orientada a eventos (*Event-Driven Architecture*) viabiliza a construção de sistemas reativos e escaláveis, com capacidade de integrar diferentes componentes de maneira modular e eficiente.

Segundo Bhattacharyya et al. (BHATTACHARYYA et al., 2023), esse modelo de integração permite a construção de sistemas inteligentes com elevada interoperabilidade, sendo essencial para aplicações industriais baseadas em múltiplas fontes de dados e em decisões descentralizadas. Lacity e Willcocks (LACITY; WILLCOCKS, 2014) destacam ainda o uso de webhooks na redução de latência e aumento da confiabilidade em sistemas de resposta automatizada, como em ambientes educacionais e de monitoramento. Tais contribuições são aplicáveis a contextos industriais com requisitos semelhantes de responsividade.

O uso de webhooks também contribui para a rastreabilidade e segurança de processos, pois permite o registro de eventos em tempo real, associando cada ação a um identificador de origem, data e horário. Com isso, é possível auditar as interações entre sistemas e diagnosticar falhas de forma mais precisa.

FIGURE 7 – FLUXO ESQUEMÁTICO DE ATIVAÇÃO DE UMA WEBHOOK.



SOURCE: O AUTOR, 2024

3.10 FERRAMENTAS DE AUTOMAÇÃO

A evolução das ferramentas de automação tem desempenhado um papel estratégico na transformação digital de processos industriais e corporativos. Diversas plataformas vêm sendo desenvolvidas com o propósito de automatizar fluxos de trabalho repetitivos, reduzir o esforço manual e aumentar a precisão das operações. Essas soluções se dividem entre abordagens baseadas em codificação tradicional e aquelas voltadas para o paradigma *low-code*, com destaque para ferramentas orientadas a blocos de ação.

No cenário atual, destaca-se a ferramenta *Microsoft Power Automate*, já discutida neste trabalho, que oferece suporte tanto a automações baseadas em navegador quanto a fluxos locais executados em ambiente desktop. A sua proposta de codificação visual permite que usuários não especializados programem ações complexas com base em uma lógica condicional estruturada em fluxogramas. Segundo MacDonald (MACDONALD, 2020), a plataforma representa um avanço relevante no contexto de Robotic Process Automation (RPA), pela sua integração nativa com o ecossistema Microsoft e suporte a webhooks, conectores e fluxos híbridos.

Além do Power Automate, outras plataformas têm ganhado destaque em aplicações industriais. O *UiPath*, por exemplo, oferece um ambiente robusto voltado à automação de processos em larga escala, com recursos avançados para reconhecimento de padrões, OCR (reconhecimento óptico de caracteres) e integração com sistemas ERP. De acordo com Czarnecki e Fettke (CZARNECKI; FETTKE, 2021), essa ferramenta tem sido amplamente adotada por empresas que buscam combinar RPA com Inteligência Artificial para atingir maior grau de automação cognitiva.

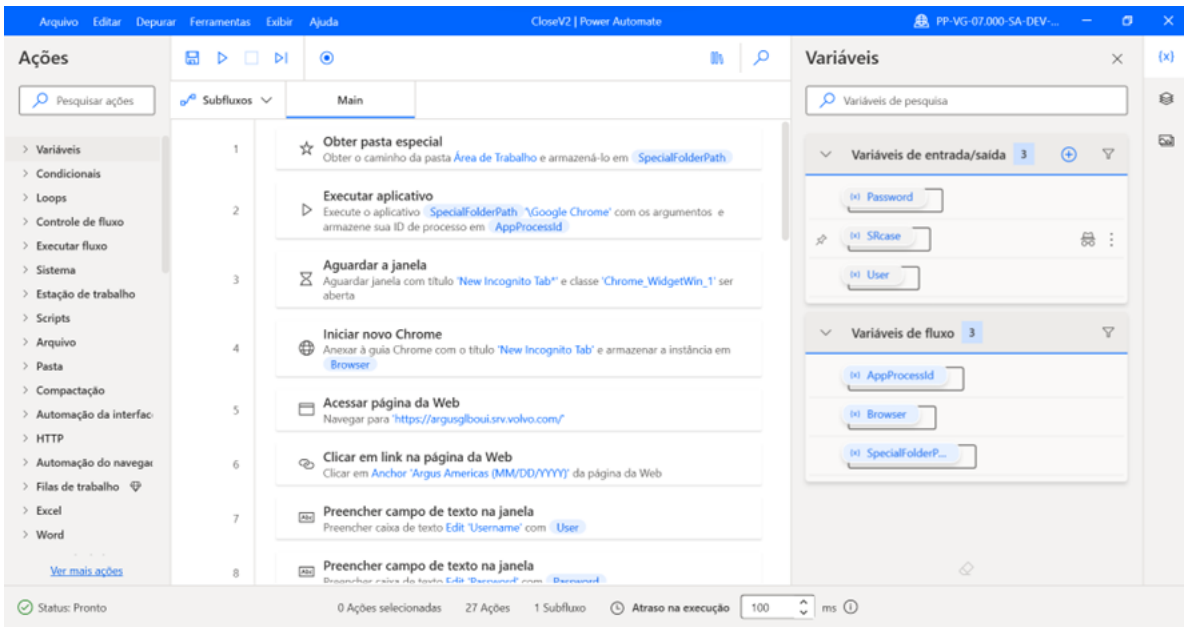
Outra plataforma relevante é o *KNIME*, que se diferencia por permitir automações de processos analíticos e fluxos de dados, sendo amplamente utilizado em áreas como ciência de dados e modelagem preditiva. Sua arquitetura modular e extensível, baseada em blocos, facilita a prototipação de soluções integradas com bibliotecas de aprendizado de máquina e serviços externos, tornando-o uma ferramenta versátil para aplicações industriais e acadêmicas (BERTHOLD et al., 2008).

Embora linguagens de programação como Python, Java e C possam ser utilizadas para desenvolver rotinas de automação, seu uso requer maior nível de especialização técnica. Em contrapartida, as ferramentas *low-code* oferecem ganhos substanciais em agilidade e manutenção, especialmente em ambientes com múltiplas tarefas e usuários com perfil não desenvolvedor. Esse modelo democratiza o acesso à automação, permitindo que engenheiros, analistas e operadores configurem soluções sem depender de equipes de TI dedicadas.

Dessa forma, a escolha da ferramenta de automação mais adequada depende

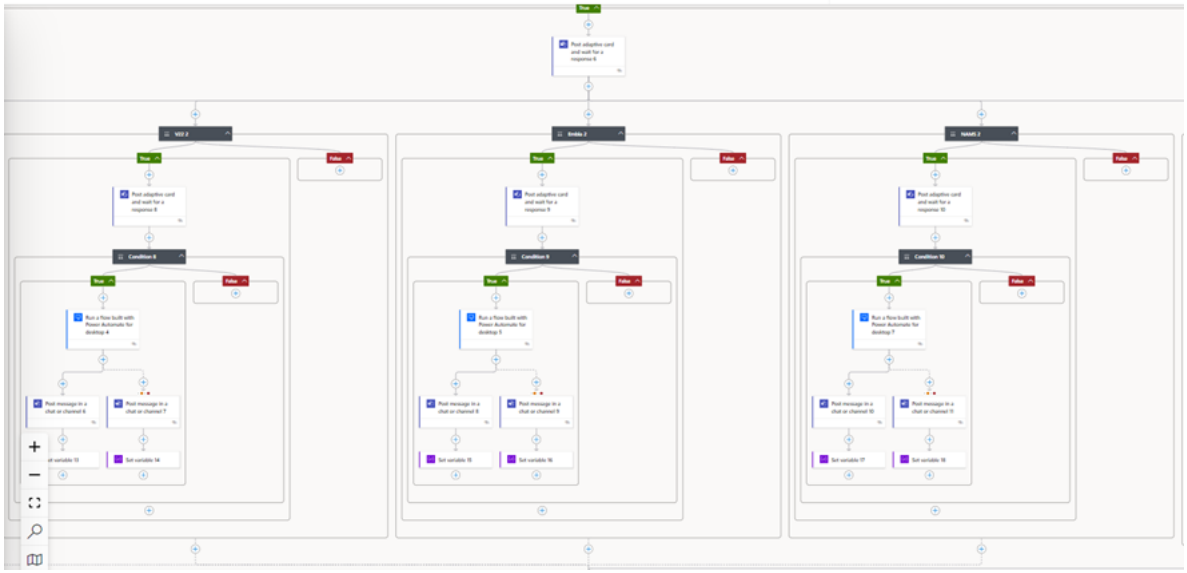
do contexto operacional, do grau de integração desejado com outros sistemas e da complexidade dos fluxos a serem implementados. Seja por meio de soluções proprietárias robustas ou ferramentas de código aberto, o avanço das plataformas de automação representa um eixo central na estratégia de digitalização da indústria contemporânea.

FIGURE 8 – EXEMPLO DE FLUXO DE AUTOMAÇÃO EM POWER AUTOMATE DESKTOP



SOURCE: O AUTOR, 2024

FIGURE 9 – EXEMPLO DE FLUXO DE AUTOMAÇÃO EM POWER AUTOMATE DESKTOP



SOURCE: O AUTOR, 2024

4 METODOLOGIA

4.1 TECNOLOGIAS UTILIZADAS

O desenvolvimento do sistema proposto neste trabalho fundamentou-se na integração de ferramentas voltadas à automação de processos, comunicação corporativa e inteligência artificial aplicada ao suporte técnico. As tecnologias escolhidas pertencem, em sua maioria, ao ecossistema Microsoft, em virtude da facilidade de integração entre plataformas, disponibilidade de conectores nativos, compatibilidade com ambientes corporativos e aderência ao modelo de desenvolvimento *low-code*. Complementarmente, foram utilizadas bibliotecas em linguagem Python para a construção de um módulo de conversação autônomo baseado em modelos de linguagem natural.

4.1.1 Microsoft Power Automate

O Microsoft Power Automate é uma plataforma para automação de fluxos de trabalho que permite a criação de rotinas digitais baseadas em lógica condicional, utilizando uma interface gráfica intuitiva. No contexto deste projeto, o Power Automate foi empregado para implementar três automações distintas, acionadas a partir de mensagens enviadas por usuários no Microsoft Teams. A ferramenta possibilita a execução de ações como envio de e-mails, manipulação de listas no SharePoint e execução de fluxos em agentes locais, sendo adequada para simulações de processos industriais automatizados.

Sua utilização justifica-se pela capacidade de estruturar fluxos modulares e determinísticos, com rápida capacidade de prototipação, além da ampla compatibilidade com sistemas de informação utilizados em ambientes organizacionais.

4.1.2 Microsoft Teams

O Microsoft Teams é uma plataforma de comunicação corporativa que integra recursos de mensagens instantâneas, videoconferência, compartilhamento de arquivos e conectividade com serviços automatizados. Neste trabalho, o Teams foi adotado como ponto de entrada do sistema de automação, permitindo a comunicação entre o usuário e o assistente por meio de comandos enviados em linguagem natural ou por meio de *cards* interativos.

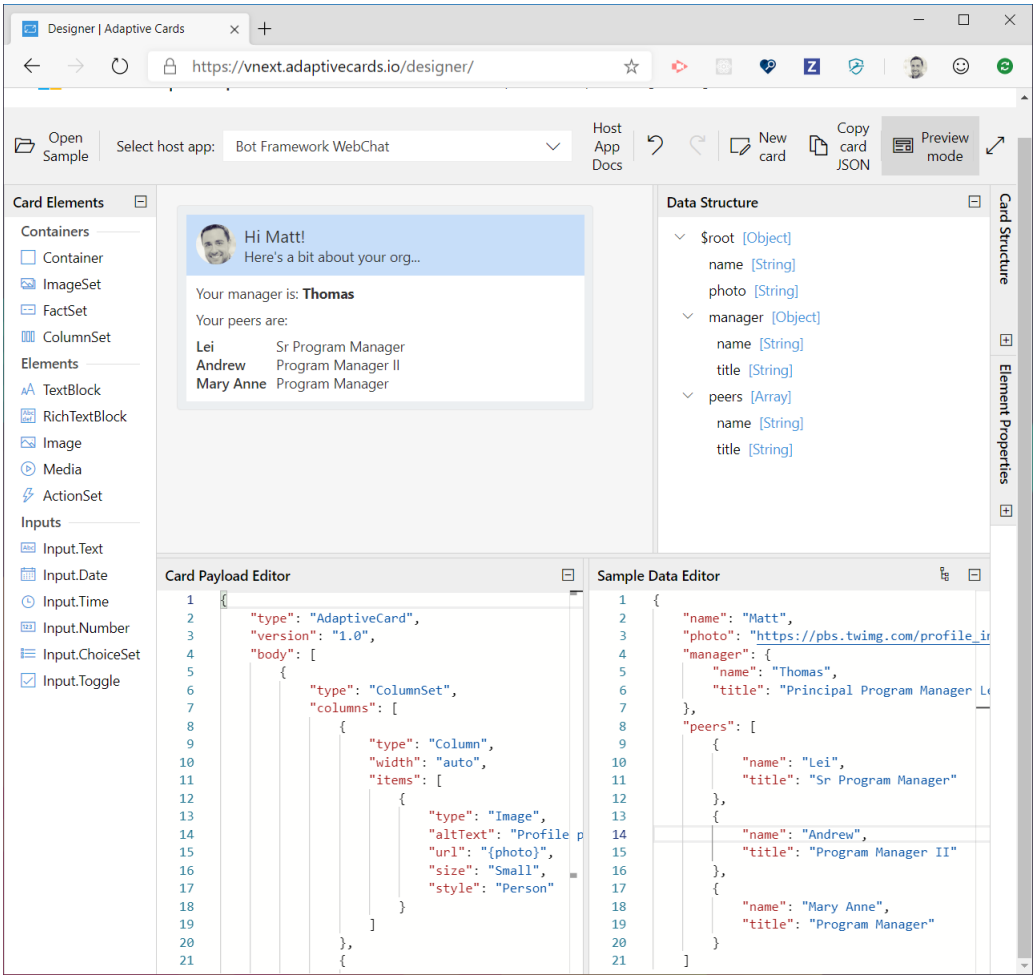
A ferramenta suporta o uso de elementos visuais dinâmicos, como menus, listas e botões, viabilizados pela estruturação de *Adaptive Cards*, o que facilita a interação com usuários não técnicos em um ambiente familiar e acessível.

4.1.3 Adaptive Cards

Os *Adaptive Cards* consistem em componentes de interface definidos em formato JSON, desenvolvidos para apresentar informações estruturadas e interativas em plataformas como Microsoft Teams, Outlook e outras soluções Microsoft. No projeto em questão, foram utilizados para fornecer menus de seleção ao operador, registrar comandos de execução e exibir mensagens de retorno.

A adoção dessa tecnologia decorreu de sua integração nativa ao Teams, sua flexibilidade no design de elementos interativos e sua compatibilidade com plataformas baseadas em nuvem.

FIGURE 10 – EXEMPLO DE *ADAPTIVE CARD*



SOURCE: Microsoft (2025)

4.1.4 Ambiente de Armazenamento e Dados

A entrada e a persistência de dados no sistema de automação foram implementadas por meio de uma lista hospedada no Microsoft SharePoint, utilizada como

repositório para o registro de peças não conformes. Essa estrutura simula a coleta de informações em um ambiente fabril, permitindo o armazenamento de dados como descrição da falha, operador responsável, estação envolvida e data da ocorrência.

Para as demais automações, os dados trafegam apenas dentro do fluxo do Power Automate, sem persistência em banco de dados formal. A escolha por um ambiente simulado e controlado visa simplificar o escopo e manter o foco na demonstração das funcionalidades do sistema proposto, sem necessidade de integração a sistemas industriais reais ou fontes externas complexas.

4.1.5 Bibliotecas e Serviços em Python

Para o desenvolvimento do módulo de diagnóstico baseado em linguagem natural, foram empregadas bibliotecas em Python, com destaque para a Streamlit, LangChain, FAISS e OpenAI. O objetivo deste módulo é simular uma aplicação de conversação técnica, capaz de receber dúvidas dos usuários e responder com base em uma base documental previamente construída.

Cada uma das bibliotecas desempenha uma função específica: a Streamlit é responsável pela construção da interface gráfica da aplicação; a LangChain permite a estruturação do fluxo conversacional com suporte a recuperação de documentos; a FAISS é utilizada para a indexação vetorial dos documentos técnicos; e a API da OpenAI é responsável pela geração das respostas, por meio do modelo GPT-3.5-turbo.

4.2 ESTRUTURAÇÃO DO SISTEMA DE AUTOMAÇÃO

A estruturação do sistema de automação desenvolvido neste trabalho baseia-se em uma arquitetura orientada a eventos, na qual comandos enviados via Microsoft Teams disparam fluxos de trabalho no Power Automate. O sistema é dividido em três blocos funcionais independentes, correspondentes às automações propostas: (i) alteração de acesso de usuários, (ii) registro de peças não conformes e (iii) envio de informações técnicas para auditoria. As automações propostas visam mostrar a versatilidade e aplicabilidade em um sistema fabril real.

O fluxo é ativado por uma mensagem recebida no canal do Teams, que contém a solicitação do operador registrada por meio de um *Adaptive Card*. A mensagem é interpretada por meio de ações de análise JSON e variáveis condicionais, as quais determinam qual bloco do fluxo será executado. A lógica foi estruturada de forma modular, utilizando blocos condicionais (*if/else*) para avaliar a escolha do usuário e direcionar o fluxo conforme a ação requerida.

A automação de alteração de acesso simula o acionamento de um fluxo local,

em uma máquina dedicada online, responsável por modificar permissões atribuídas a um usuário em ambiente corporativo. A automação de não conformidade insere dados em uma lista do SharePoint, configurada para armazenar ocorrências técnicas simuladas. Já a automação de auditoria organiza os dados fornecidos pelo operador e os encaminha por e-mail a um destinatário predeterminado.

Essa separação por blocos facilita a manutenção, a escalabilidade e a possibilidade de adaptação do sistema a novos cenários. A abordagem utilizada também possibilita a reutilização da estrutura para diferentes contextos fabris, alterando apenas os parâmetros de entrada e a lógica específica de cada operação.

4.3 MÓDULO DE DIAGNÓSTICO BASEADO EM LINGUAGEM NATURAL

O módulo de diagnóstico conversacional foi concebido com o objetivo de simular a aplicação de modelos de linguagem natural no suporte técnico a operadores de linha de produção. Trata-se de uma aplicação desenvolvida de forma independente, a partir de bibliotecas Python, com interface implementada por meio do Streamlit.

A metodologia aplicada contemplou a construção de uma base textual contendo falhas técnicas, causas prováveis e recomendações associadas a diferentes estações da linha de montagem de motores. Esse conteúdo foi armazenado localmente em um arquivo de texto, estruturado de forma a possibilitar sua posterior indexação e recuperação semântica.

A aplicação utiliza a biblioteca LangChain para organizar o fluxo de consulta e resposta, vinculando o conteúdo técnico à solicitação do usuário. O conteúdo é processado com apoio da biblioteca FAISS, que permite a recuperação das informações mais relevantes com base em embeddings gerados pela API da OpenAI. A resposta é então produzida pelo modelo GPT-3.5-turbo e apresentada ao operador em três seções: (i) resumo técnico da dúvida, (ii) contexto explicativo e (iii) recomendação de ação.

Para fins de rastreabilidade e análise posterior, todas as interações são registradas localmente em um arquivo ‘.csv‘, contendo data, pergunta e resposta. Essa funcionalidade permite a avaliação do uso do sistema e pode subsidiar melhorias no conteúdo técnico e na lógica de resposta.

Por fim, destaca-se que a estrutura do módulo permite sua futura integração a plataformas corporativas, como o Microsoft Teams, ou a substituição da base textual por documentos técnicos mais extensos. A modularidade e a simplicidade da arquitetura adotada contribuem para a viabilidade de adaptação a contextos reais de aplicação em ambientes fabris.

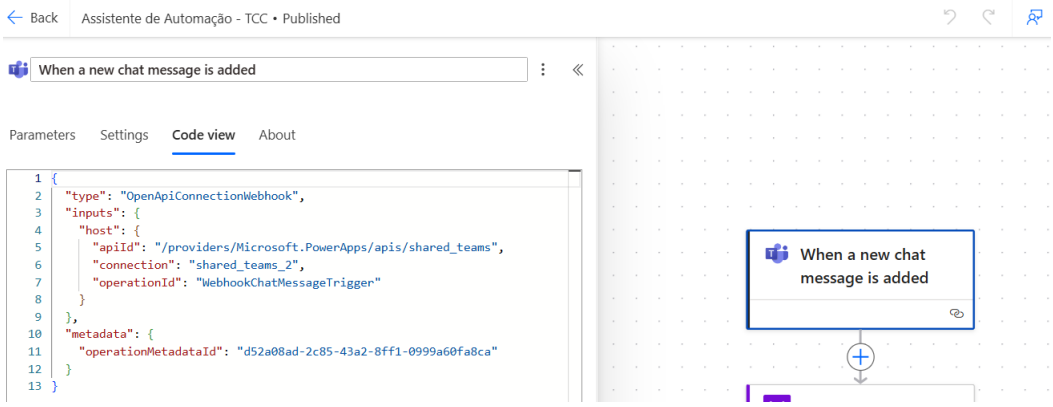
5 DESENVOLVIMENTO

O sistema proposto foi estruturado em dois módulos independentes: (i) um conjunto de automações construídas na plataforma Microsoft Power Automate, responsável pela execução de tarefas técnicas simuladas, e (ii) um módulo de diagnóstico baseado em linguagem natural, desenvolvido em linguagem Python. Embora operem separadamente, ambos foram concebidos para atuar de forma complementar, simulando o funcionamento de um assistente técnico virtual com capacidade de executar ações operacionais e responder a dúvidas técnicas.

5.1 DESENVOLVIMENTO DO SISTEMA DE AUTOMAÇÃO

A lógica de automação foi implementada integralmente no Power Automate, utilizando um fluxo principal acionado por mensagens enviadas no Microsoft Teams. O gatilho do fluxo consiste em um conector do tipo *Webhook*, que monitora continuamente as mensagens recebidas em um canal específico. Quando um operador envia uma solicitação por meio de uma mensagem, o fluxo é disparado automaticamente, iniciando a análise da mensagem recebida.

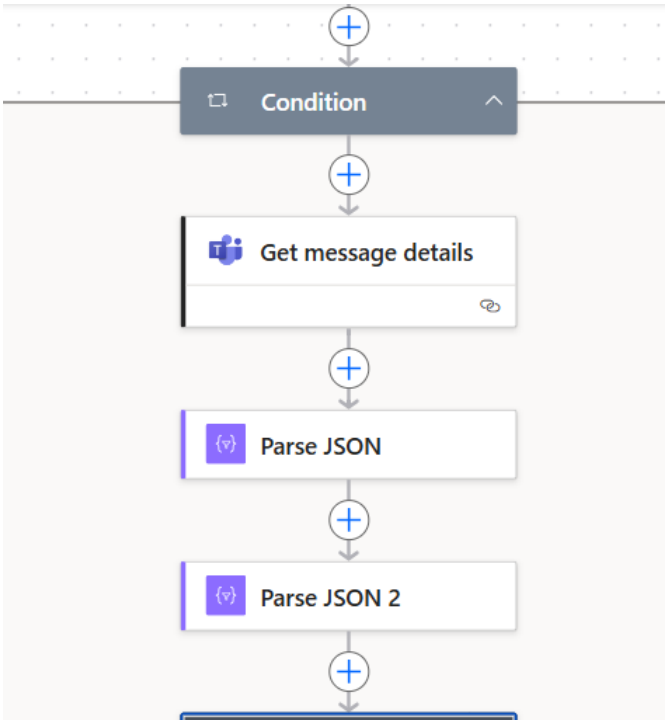
FIGURE 11 – EVENTO DE ATIVAÇÃO DO FLUXO DE AUTOMAÇÃO.



SOURCE: O AUTOR, 2025

A primeira etapa do fluxo é a extração dos dados contidos na mensagem do gatilho, estruturados em formato JSON. Essa estrutura é analisada com o uso da ação “Parse JSON”, que permite acessar as informações da mensagem e identificar se o comando de ativação, neste caso */help*, foi enviado. Se identificado o comando de ativação, um card é enviado para que o usuário faça a escolha da automação. Posteriormente, uma variável de controle é definida com base na escolha realizada no menu de seleção do card, determinando qual das automações será executada.

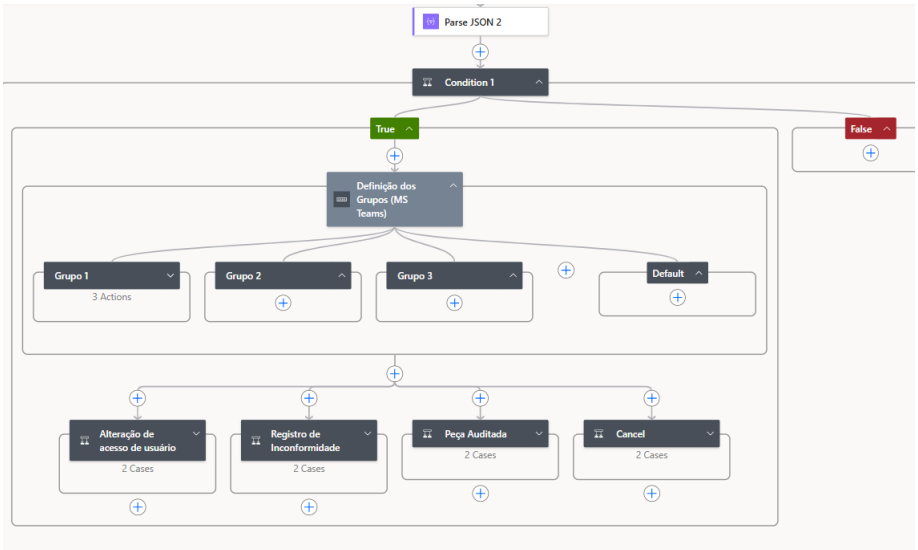
FIGURE 12 – ETAPA DE EXTRAÇÃO DE DADOS DA MENSAGEM DE GATILHO.



SOURCE: O AUTOR, 2025

Para garantir clareza e organização, a lógica do fluxo foi estruturada com o uso de blocos condicionais do tipo “Switch case”, nos quais cada ramificação corresponde a uma ação distinta. Essa abordagem modular permite a manutenção isolada de cada funcionalidade, além de facilitar a inclusão de novas automações futuras.

FIGURE 13 – BLOCOS DE DECISÃO DE GRUPOS E AUTOMAÇÕES.



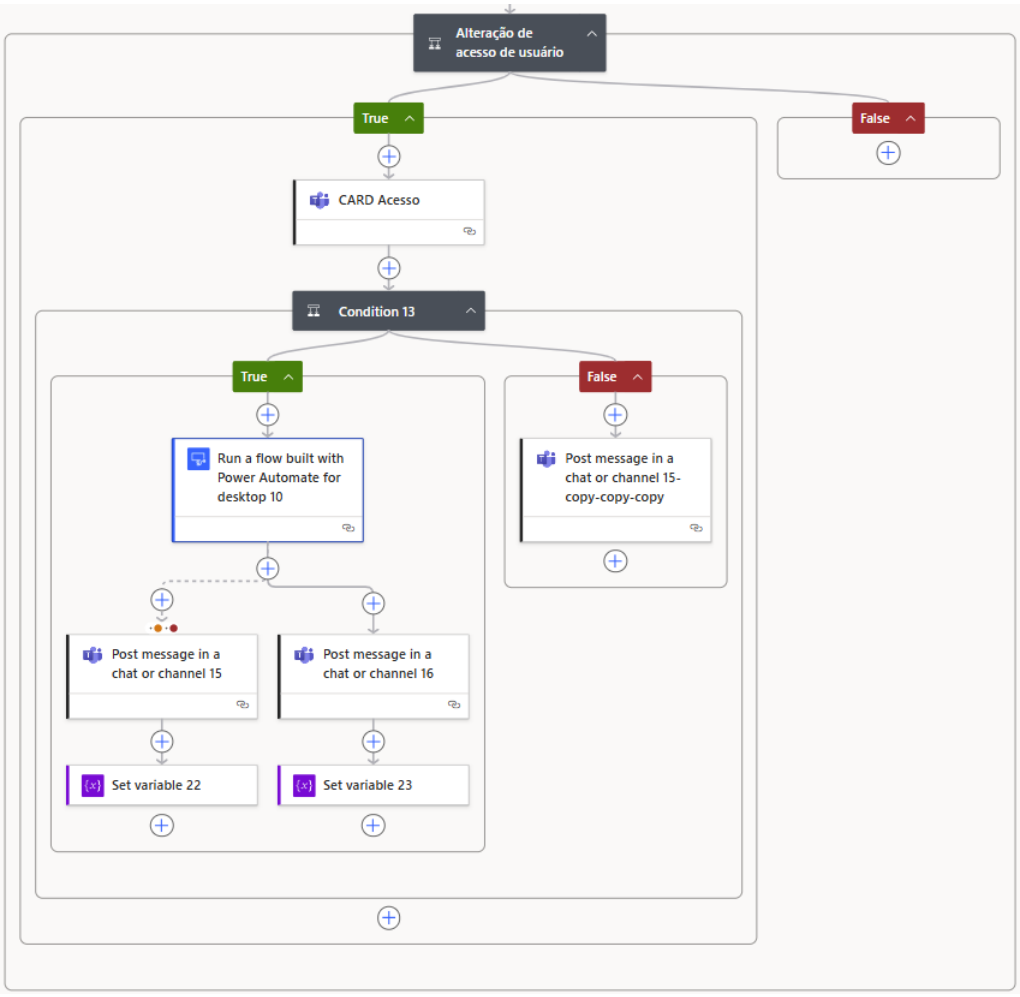
SOURCE: O AUTOR, 2025

5.1.1 Automação 1: Alteração de Acesso de Usuário

Essa automação simula o ajuste de permissões de um operador em um sistema de programação de ECU, a partir de uma situação técnica em que falhas de gravação indicam a necessidade de reconfiguração do perfil de acesso. O fluxo obtém o nome do operador, o sistema afetado e o novo nível de acesso requerido, inseridos previamente no Adaptive Card.

Após a validação dos dados, o fluxo aciona um segundo fluxo local via conector “Desktop Flow”, executado por meio do Power Automate for Desktop, em uma máquina dedicada, simulando a interação com a interface de um sistema corporativo. Em seguida, uma mensagem de retorno é enviada ao usuário no Teams, confirmando a execução e listando os parâmetros aplicados.

FIGURE 14 – BLOCO DE AÇÕES PARA AUTOMAÇÃO DE LIBERAÇÃO DE ACESSO.



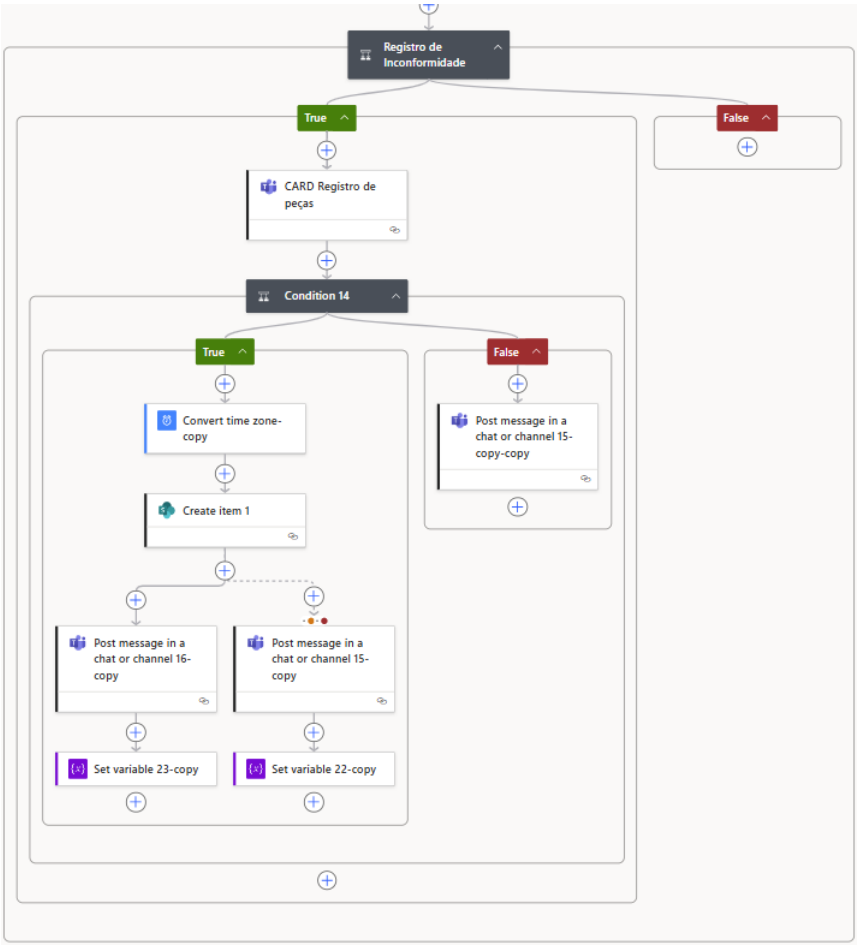
SOURCE: O AUTOR, 2025

5.1.2 Automação 2: Registro de Peça Não Conforme

Esta automação foi projetada para registrar, de forma padronizada, peças identificadas com falhas durante a linha de produção. O usuário informa no Adaptive Card o código da peça, o tipo de falha, a estação de ocorrência e o número do lote.

Esses dados são armazenados diretamente em uma lista personalizada no Microsoft SharePoint, estruturada com colunas específicas para cada campo. A adição é realizada por meio do conector “Create item”, que interage com a lista remota. O sistema retorna ao usuário uma confirmação visual contendo os dados registrados e o link de acesso à lista de controle. Essa automação visa simular o processo de rastreabilidade e controle de qualidade amplamente utilizado em ambientes industriais reais.

FIGURE 15 – BLOCO DE AÇÕES PARA AUTOMAÇÃO DE PEÇAS COM INCONFORMIDADE.



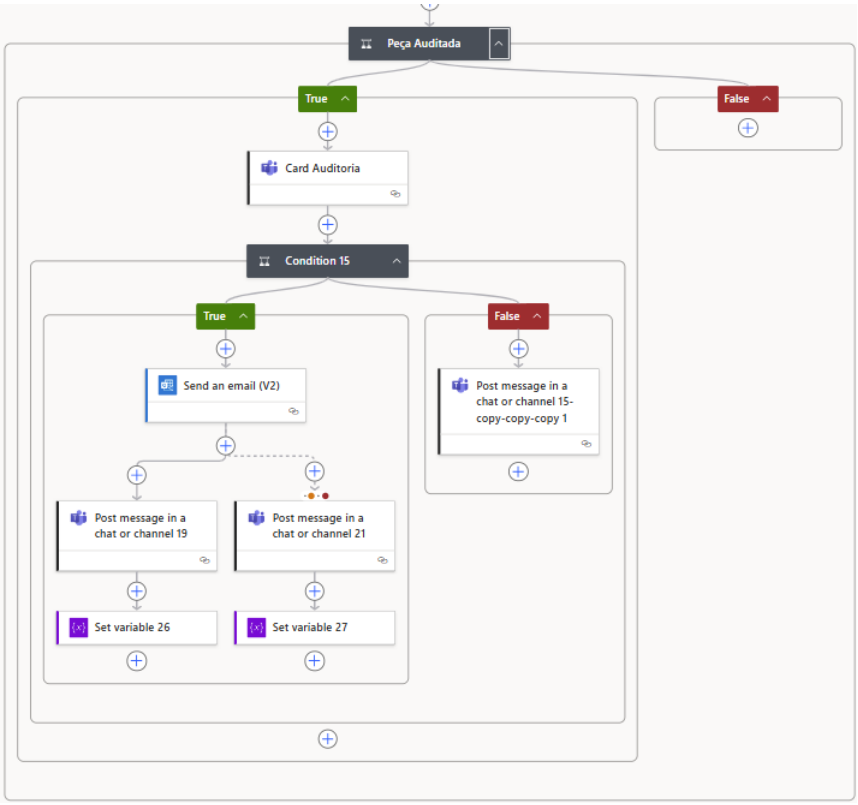
SOURCE: O AUTOR, 2025

5.1.3 Automação 3: Envio de Informações para Auditoria

A terceira automação realiza o envio automatizado de dados técnicos para fins de auditoria. A simulação parte de um cenário em que uma peça crítica precisa ser avaliada por um auditor técnico, mesmo que não tenha sido reprovada formalmente.

O operador insere informações detalhadas sobre a peça e sua condição, e o fluxo organiza essas informações em um corpo de e-mail padronizado, utilizando a ação “Send an email (V2)”. O e-mail é enviado ao destinatário previamente cadastrado e contém os dados em formato tabular, com destaque para lote, estação, operador e comentário técnico. A mensagem de confirmação é enviada de volta ao Teams, indicando que o envio foi concluído com sucesso.

FIGURE 16 – BLOCO DE AÇÕES PARA AUTOMAÇÃO DE AUDITORIA.

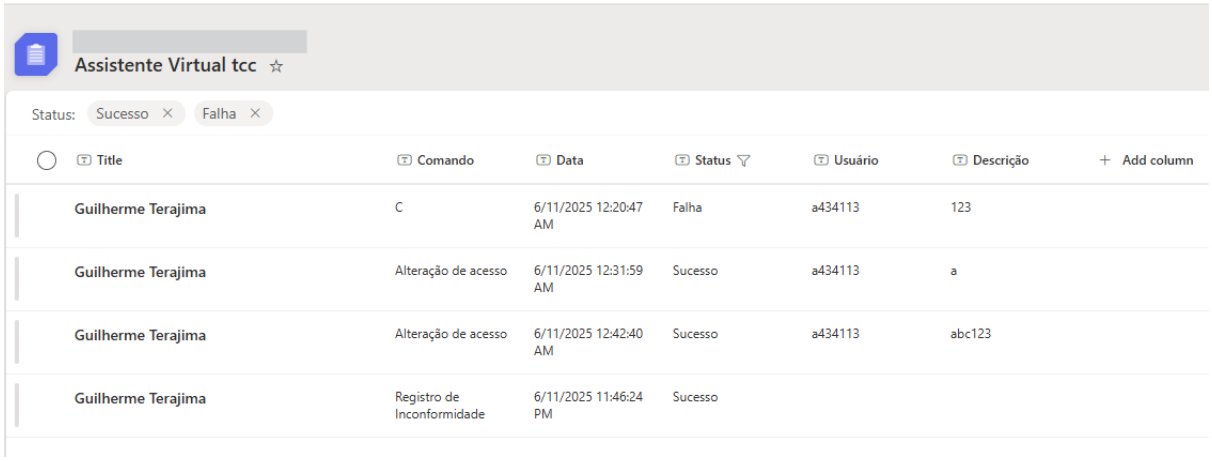


SOURCE: O AUTOR, 2025

5.1.4 Considerações sobre a Estrutura do Fluxo

Para registro de utilização e rastreabilidade, foram definidas variáveis condicionais ao final de cada automação, que definem se a ação foi executada com sucesso ou falha. Estas informações são registradas em uma planilha da Microsoft List, contendo registros de nome do usuário que solicitou a ação, horário e data, justificativa da automação e outras informações relevantes.

FIGURE 17 – REGISTRO DE UTILIZAÇÃO.



SOURCE: O AUTOR, 2025

5.2 DESENVOLVIMENTO DO MÓDULO DE DIAGNÓSTICO BASEADO EM IA

O módulo de diagnóstico técnico foi desenvolvido em linguagem Python, com o objetivo de permitir a análise e resposta a dúvidas técnicas com base em um corpus textual simulado, representativo de um ambiente de montagem de motores. A interface da aplicação foi implementada com o uso da biblioteca Streamlit, que possibilita a construção de aplicações web leves, com entrada de texto e exibição dinâmica de resultados.

5.2.1 Estrutura do Assistente

A estrutura da aplicação contempla as seguintes etapas:

- Carregamento da base de conhecimento textual, armazenada localmente em um arquivo `.txt`, contendo falhas, sintomas e ações recomendadas organizadas por estação de trabalho;
- Indexação semântica da base com uso da biblioteca FAISS, a partir de embeddings gerados pela API da OpenAI;
- Definição de um *prompt* orientado, especificando a estrutura esperada da resposta (resumo, contexto, ação);
- Processamento da pergunta do usuário e recuperação do conteúdo mais relevante com LangChain;

- Geração da resposta com o modelo GPT-3.5-turbo, exibida ao usuário com formatação visual adequada.

A base foi elaborada com conteúdo técnico realista, abrangendo estações como instalação do bloco motor, acoplamento do virabrequim, inserção dos pistões e teste de estanqueidade. Cada item inclui uma descrição do sintoma, a causa provável e a ação recomendada.

5.2.2 Detalhamento do Código

O código da aplicação foi organizado em estrutura modular, com separação clara entre carregamento de dados, configuração de embeddings, definição de prompt e exibição da interface. O carregamento da base textual é feito com o uso da classe `TextLoader`, disponibilizada pela biblioteca LangChain. O conteúdo é processado em lotes, preservando a estrutura por estação de trabalho e mantendo a coerência semântica entre sintomas e recomendações.

Para transformar os textos em representações vetoriais, são utilizados os *embeddings* da OpenAI, gerados com a classe `OpenAIEmbeddings`. Cada parágrafo ou conjunto de registros é convertido em um vetor de 1536 dimensões. Esses vetores são armazenados localmente em um índice FAISS, criado com o tipo `IndexFlatL2`, que aplica distância euclidiana simples na recuperação dos dados. A escolha por este tipo de índice visou garantir precisão na busca, priorizando desempenho em bases de tamanho reduzido.

O modelo de recuperação de contexto e geração de resposta foi construído com o uso da classe `RetrievalQA`, também do LangChain. A configuração utilizada definiu `k=4` como o número de documentos mais similares a serem buscados no índice FAISS para cada pergunta feita pelo usuário. Esses documentos são concatenados e utilizados como base de conhecimento para geração da resposta final. O modelo GPT-3.5-turbo é acessado via API da OpenAI, com `temperature=0.2`, de forma a garantir respostas determinísticas e tecnicamente consistentes.

A definição do comportamento da IA foi refinada por meio de um *prompt* personalizado, orientando a estrutura da resposta. O modelo é instruído a produzir três blocos bem definidos: resumo técnico, contexto e solução recomendada. Exemplo do trecho de prompt:

> “Baseado na dúvida do operador e na documentação abaixo, forneça uma resposta estruturada com:
>
> 1. **Resumo técnico do problema**
> 2. **Contexto técnico e causas possíveis**

3. **Solução recomendada** ”

Adicionalmente, a aplicação conta com um mecanismo de histórico automatizado, por meio do qual cada pergunta submetida, a resposta gerada e a data/hora da interação são registradas em um arquivo `historico.csv`. Essa abordagem permite reconstituir sessões passadas, monitorar o uso da ferramenta e auditar a rastreabilidade das decisões.

5.2.3 Interface e Apresentação das Respostas

A interface da aplicação foi construída com a biblioteca Streamlit, utilizando o componente `st.markdown()` com elementos HTML para formatar a resposta. Cada bloco é apresentado com destaque visual, como cor de fundo, espaçamento e ícones, tornando a leitura mais agradável e objetiva. O layout da interface foi definido de forma responsiva, com título, campo de entrada de texto e painel de resposta, simulando uma experiência semelhante à de assistentes comerciais de IA.

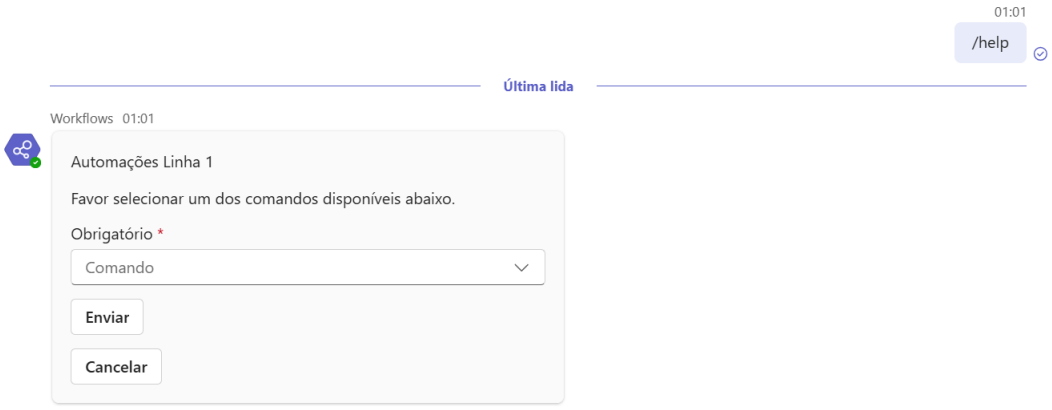
O comportamento da aplicação foi testado com diversas entradas, representando falhas plausíveis na linha de montagem simulada. As respostas demonstraram coerência técnica, estrutura adequada e alinhamento com o conteúdo previamente indexado, evidenciando o funcionamento esperado do sistema.

A aplicação foi desenvolvida com foco em modularidade e reusabilidade, o que viabiliza a futura integração com plataformas corporativas como Microsoft Teams ou Power Automate, mediante autorização para uso de conectores externos.

6 RESULTADOS E DISCUSSÕES

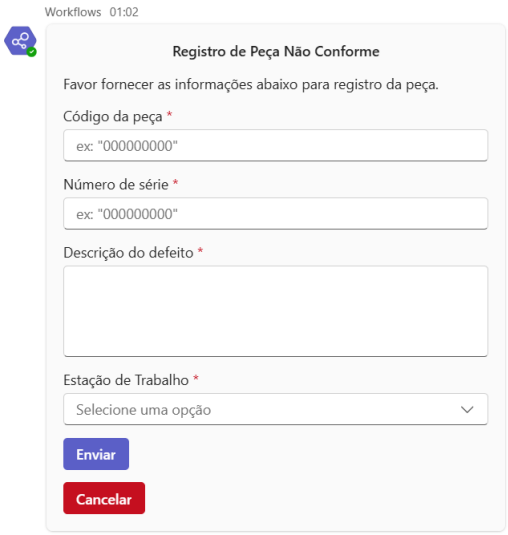
A avaliação do sistema desenvolvido foi conduzida em ambiente controlado, simulando o uso por operadores em um cenário fabril fictício, baseado na linha de montagem de motores. O objetivo foi verificar a funcionalidade dos fluxos de automação implementados no Power Automate e a consistência das respostas fornecidas pelo módulo de diagnóstico baseado em linguagem natural.

FIGURE 18 – DEMONSTRAÇÃO DE USO DO ASSISTENTE DE AUTOMAÇÃO. ETAPA DE ATIVAÇÃO E ESCOLHA DE COMANDO.



SOURCE: O AUTOR, 2025

FIGURE 19 – DEMONSTRAÇÃO DE USO DO ASSISTENTE DE AUTOMAÇÃO. ETAPA DE PREENCHIMENTO E ENVIO DE INFORMAÇÕES.



SOURCE: O AUTOR, 2025

No módulo de automação, os três fluxos implementados foram acionados com dados de teste representando situações operacionais plausíveis. A automação de alteração de acesso respondeu de forma consistente aos comandos enviados, confirmando corretamente os parâmetros informados e simulando a execução do processo localmente. A automação de registro de peças não conformes foi validada por meio da inclusão de diferentes entradas na SharePoint List, demonstrando a persistência adequada dos dados e a exibição clara das informações no Teams. Por fim, a automação de envio de informações técnicas por e-mail demonstrou bom desempenho na formatação da mensagem, no envio automático e na recepção pelo destinatário, respeitando o modelo definido.

Complementarmente, foi possível observar, com base em dados reais de um ambiente de produção corporativo, que fluxos desenvolvidos em Power Automate e implementados desde janeiro de 2025 totalizaram 2176 execuções em um período de seis meses. Destas, aproximadamente 82% foram concluídas com sucesso, conforme apresentado na Figura. Considerando que cada execução representa uma economia média de cinco minutos de trabalho manual, estima-se uma economia total superior a 149 horas de atividade humana no período analisado. Este dado reforça o impacto positivo da adoção de automações em atividades operacionais rotineiras.

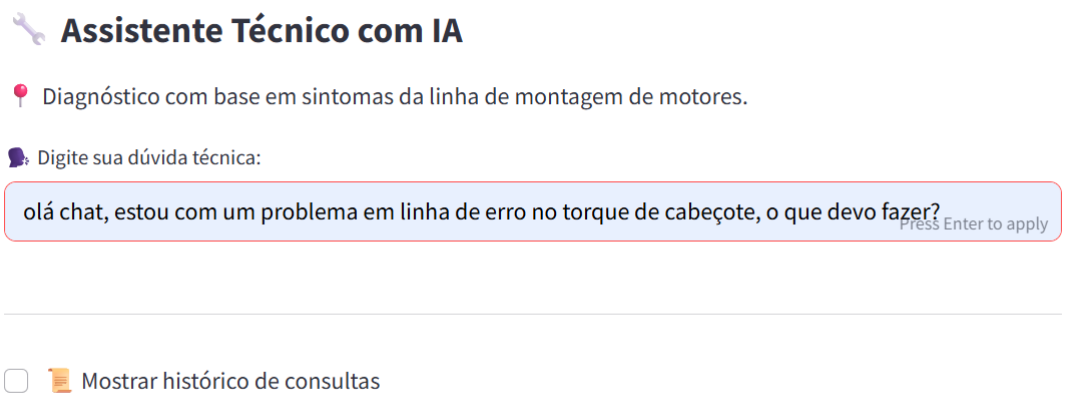
FIGURE 20 – TAXA DE SUCESSO DO ASSISTENTE DE AUTOMAÇÃO EM AMBIENTE DE PRODUÇÃO.

SOURCE: O AUTOR, 2025

No módulo de diagnóstico baseado em linguagem natural, optou-se por restringir o sistema a consultas com até 300 caracteres e respostas com limite de 500 tokens, a fim de garantir desempenho estável da API e compatibilidade com o modelo de apresentação no navegador. Em testes realizados, observou-se que perguntas genéricas ou fora do escopo resultam em respostas neutras e instruções para revisão

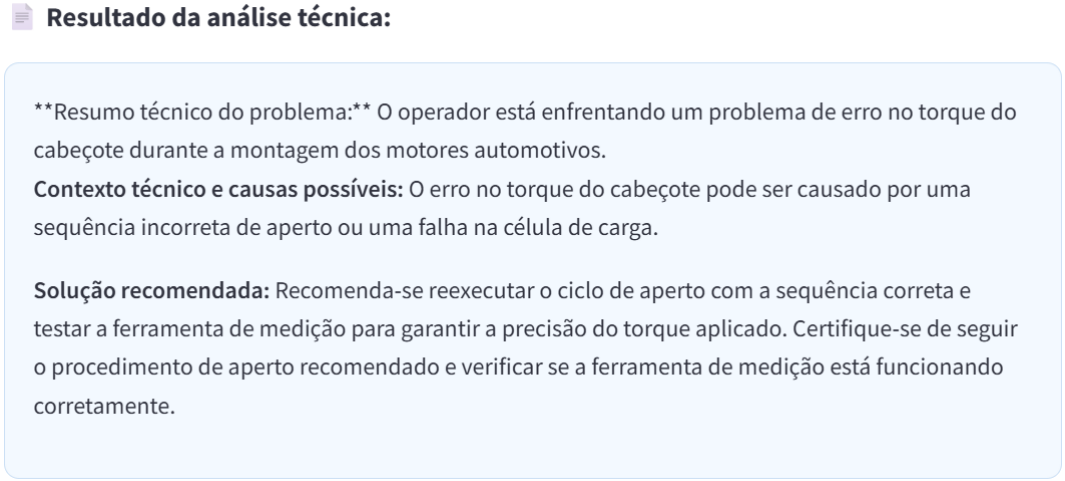
da base, como forma de mitigar alucinações comuns em modelos de linguagem.

FIGURE 21 – DEMONSTRAÇÃO DE USO DO ASSISTENTE DE DIAGNÓSTICO.



SOURCE: O AUTOR, 2025

FIGURE 22 – DEMONSTRAÇÃO DE USO DO ASSISTENTE DE DIAGNÓSTICO. RESPOSTA GERADA.



SOURCE: O AUTOR, 2025

O uso do FAISS garante boa performance local e preserva a privacidade dos dados, visto que nenhuma informação confidencial é armazenada em servidores externos. Todo o conteúdo textual permanece no ambiente do operador, sendo enviados à API da OpenAI apenas os trechos já registrados na base e a pergunta do usuário — ambos em linguagem neutra e sem dados pessoais. Essa abordagem foi escolhida para garantir conformidade com os princípios da Lei Geral de Proteção de Dados (LGPD).

Além disso, a modularidade da arquitetura permite que o índice FAISS seja substituído futuramente por um banco de dados em nuvem ou uma instância de vetor persistente, como Pinecone ou Weaviate, caso haja necessidade de escalabilidade. Da mesma forma, o modelo GPT pode ser trocado por alternativas open-source, como LLaMA ou Mistral, se desejado.

Apesar do êxito no desenvolvimento dos dois módulos — o sistema de automação via Power Automate e o assistente de diagnóstico baseado em linguagem natural —, a unificação completa entre essas funcionalidades em uma única interface de comunicação não foi viabilizada nesta etapa do projeto. Essa limitação decorreu de restrições associadas ao modelo de licenciamento corporativo do Power Automate utilizado, o qual impede a integração direta com serviços externos via API, como a plataforma da OpenAI, além de restrições complementares ligadas à conformidade com a Lei Geral de Proteção de Dados (LGPD).

A integração entre os dois módulos, embora não implementada neste trabalho, apresenta um potencial significativo. A convergência entre automação estruturada por fluxos determinísticos e a capacidade interpretativa de modelos de linguagem natural poderia permitir uma interação mais fluida e versátil com o usuário, reduzindo a necessidade de formulários manuais e aumentando a autonomia da aplicação. Além disso, a centralização do atendimento técnico em um único canal simplificaria o treinamento de operadores e a manutenção do sistema, ampliando sua aplicabilidade em ambientes industriais reais.

7 CONCLUSÃO

Este trabalho teve como objetivo a concepção e implementação de um assistente técnico virtual voltado ao apoio à tomada de decisão em ambientes fabris, com ênfase em processos automatizados e no uso de linguagem natural para consulta de dados técnicos. A proposta foi desenvolvida a partir da combinação de duas abordagens complementares: a automação de tarefas por meio da plataforma Microsoft Power Automate e a utilização de modelos de linguagem de última geração para diagnóstico técnico baseado em texto.

Durante a fase de desenvolvimento, foram implementados dois módulos independentes. O primeiro consistiu na criação de fluxos automatizados acionados a partir do Microsoft Teams, capazes de simular ações operacionais como a alteração de permissões de acesso, o registro de peças não conformes e o envio de dados técnicos para auditoria. O segundo módulo, baseado em uma aplicação desenvolvida em Python com uso da biblioteca Streamlit, foi estruturado para receber descrições de falhas formuladas em linguagem natural, interpretá-las e retornar uma resposta estruturada com base em uma base técnica de conhecimento previamente construída.

Os testes realizados demonstraram que ambos os módulos são funcionalmente viáveis, mesmo em ambiente simulado. O sistema de automação apresentou comportamento estável, com retorno imediato ao operador e estrutura modular facilmente expansível. Já o módulo de diagnóstico baseado em IA revelou-se capaz de interpretar descrições técnicas e sugerir soluções adequadas, respeitando os limites da base de conhecimento fornecida.

Além disso, dados reais de uso de fluxos automatizados no Power Automate ao longo de seis meses apontaram para 2176 execuções, com uma taxa de sucesso de 82% e uma economia estimada de mais de 149 horas de esforço operacional. Esses números reforçam a aplicabilidade prática e o impacto positivo que soluções baseadas em automação podem trazer aos processos industriais.

Apesar das limitações enfrentadas, como a impossibilidade de integrar diretamente os dois módulos devido a restrições de licenciamento e políticas de governança de dados, o projeto demonstrou que é possível construir soluções tecnológicas alinhadas à realidade operacional de ambientes industriais. A separação entre os módulos não comprometeu os objetivos do trabalho, mas ressaltou a necessidade de estratégias flexíveis de desenvolvimento quando operando em contextos organizacionais restritivos.

O trabalho abre espaço para diversas possibilidades de evolução, como a integração dos módulos em um único ambiente, o uso de bases técnicas mais amplas e

a conexão direta com sistemas industriais reais. Dessa forma, conclui-se que a proposta desenvolvida não apenas atende aos requisitos definidos, mas também oferece uma base sólida para futuros projetos voltados à digitalização e à automação cognitiva em linhas de produção.

8 TRABALHOS FUTUROS

Apesar dos resultados positivos obtidos no desenvolvimento e validação dos módulos apresentados neste trabalho, diversas oportunidades de aprimoramento e expansão funcional podem ser exploradas em projetos futuros, tanto em nível técnico quanto estratégico.

Uma das possibilidades mais promissoras reside na **integração efetiva entre os módulos de automação e de inteligência artificial**. A unificação das interfaces, atualmente separadas por restrições de licença e governança, permitiria que o operador realizasse diagnósticos técnicos e acionasse fluxos automatizados dentro de um mesmo canal conversacional, como o Microsoft Teams. Para viabilizar essa integração, seria necessário o desbloqueio de conectores personalizados no ambiente corporativo e a autorização para utilização segura da API da OpenAI ou de modelos alternativos.

Outra vertente de continuidade envolve a **expansão da base de conhecimento técnico** utilizada pelo assistente virtual. A base atual, embora suficiente para validação, é limitada a um conjunto simulado de falhas. Em cenários industriais reais, documentos como manuais técnicos, instruções operacionais padronizadas (POPs), registros históricos de falhas e bases de dados de manutenção corretiva poderiam ser integrados ao modelo por meio de técnicas de *document retrieval*, enriquecendo o repertório técnico da IA.

Adicionalmente, considera-se viável a implementação de uma **camada de tratamento de linguagem natural mais robusta**, com suporte a múltiplos idiomas e compreensão de linguagem coloquial frequentemente utilizada por operadores. O uso de modelos open source com hospedagem local, como LLaMA ou Mistral, poderia ser investigado como alternativa à dependência de provedores externos.

Também é recomendável o desenvolvimento de **métricas quantitativas de desempenho** para o assistente virtual, com indicadores como acurácia da resposta, taxa de concordância técnica entre IA e especialistas humanos, tempo médio de atendimento e número de interações por estação. Tais métricas seriam úteis para auditorias internas e para avaliação contínua da qualidade do sistema.

No módulo de automação, a ampliação dos fluxos pode abranger integrações com sistemas industriais reais (ERP, MES, SCADA), permitindo ações como a abertura automática de ordens de manutenção, a solicitação de peças de reposição ou o bloqueio de estações com falha. Essa integração com o chão de fábrica representaria um passo significativo rumo à automação cognitiva em processos industriais.

Por fim, sugere-se a realização de **testes de usabilidade e validação em

ambiente fabril real**, com operadores, supervisores e engenheiros de processo. A coleta de feedback qualitativo e quantitativo permitiria ajustar tanto a interface quanto a lógica de decisão, promovendo a adaptação do sistema à cultura operacional e à dinâmica específica de cada linha de produção.

Dessa forma, os caminhos para evolução da solução proposta são amplos e tecnicamente viáveis, com potencial de gerar ganhos substanciais em confiabilidade, agilidade e autonomia na tomada de decisão em ambientes industriais complexos.

REFERÊNCIAS

AGGARWAL, A.; GOEL, R. **RPA and AI: Transforming the Enterprise**. [S.l.]: McGraw-Hill Education, 2021. Citado 2 vezes nas páginas 11, 26.

AGUIRRE, S.; RODRIGUEZ, M. **Automation, Robotics, and the Future of Work**. [S.l.: s.n.], 2017. Acesso em: 23 jun. 2025. Disponível em: https://www2.deloitte.com/insights/us/en/focus/signals-for-strategists/automation-robotics-and-future-of-work.html. Citado 1 vez na página 11.

ALDAMA, F. L. et al. **AI and Personhood in Popular Culture**. [S.l.: s.n.], 2022. Fonte não especificada, revisar ou substituir. Citado 1 vez na página 15.

BERTHOLD, M. R.; CEBRON, N.; DILL, F.; GABRIEL, T. R.; KÖTTER, T.; MEINL, T.; OHL, P.; THIEL, K.; WISWEDEL, B. KNIME: The Konstanz Information Miner. In: PREISACH, C.; BURKHARDT, H.; SCHMIDT-THIEME, L.; DECKER, R. (Ed.). **Studies in Classification, Data Analysis, and Knowledge Organization**. [S.l.]: Springer, 2008. P. 319–326. DOI: 10.1007/978-3-540-78246-9_38. Citado 1 vez na página 29.

BHATTACHARYYA, S.; BANERJEE, J. S.; DE, D. **Confluence of Artificial Intelligence and Robotic Process Automation**. [S.l.]: Springer, 2023. Citado 2 vezes nas páginas 26, 28.

BROWN, T. B.; MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J.; DHARIWAL, P.; NEELAKANTAN, A.; SHYAM, P.; SASTRY, G.; ASKELL, A. et al. Language Models are Few-Shot Learners. **arXiv preprint arXiv:2005.14165**, 2020. Acesso em: 23 jun. 2025. Disponível em: https://arxiv.org/abs/2005.14165. Citado 2 vezes nas páginas 11, 23.

BRYNJOLFSSON, E.; MCAFEE, A. **The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies**. New York: W. W. Norton & Company, 2014. Citado 2 vezes nas páginas 12, 14.

CORPORATION, M. **Microsoft Power Automate, Teams and Adaptive Cards Documentation**. Acesso em: 23 jun. 2025. 2023. Disponível em: https://learn.microsoft.com/en-us/power-automate/. Citado 1 vez na página 27.

CZARNECKI, C.; FETTKE, P. **Robotic Process Automation**. [S.l.]: Walter de Gruyter, 2021. Citado 2 vezes nas páginas 26, 29.

DOMINGOS, P. A Few Useful Things to Know About Machine Learning. **Communications of the ACM**, v. 55, n. 10, p. 78–87, 2012. DOI: 10.1145/2347736.2347755. Citado 2 vezes nas páginas 17, 20.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. 2000. Tese (Doutorado) – University of California, Irvine. Acesso em: 23 jun. 2025. Disponível em: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. Citado 1 vez na página 27.

FIGUEROA MORENO, J.; ÁLVAREZ, A. M.; SOTO, C. Robotic Process Automation in Financial and Tax Audit: A Systematic Review. **Journal of Accounting and Taxation**, v. 15, n. 2, p. 24–34, 2023. DOI: 10.5897/JAT2023.0578. Citado 1 vez na página 26.

FLORIDI, L.; COWLS, J.; BELTRAMETTI, M.; CHATILA, R.; CHAZERAND, H.; DIGNUM, V.; LUETGE, C.; MADELIN, R.; PAGALLO, U.; ROSSI, F. et al. AI4People—An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations. **Minds and Machines**, v. 28, n. 4, p. 689–707, 2018. Acesso em: 23 jun. 2025. DOI: 10.1007/s11023-018-9482-5. Disponível em: https://doi.org/10.1007/s11023-018-9482-5. Citado 1 vez na página 12.

GAMA, J.; ŽLIOBAITĖ, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. A Survey on Concept Drift Adaptation. **ACM Computing Surveys**, v. 46, n. 4, p. 1–37, 2014. DOI: 10.1145/2523813. Citado 1 vez na página 17.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. Citado 6 vezes nas páginas 16, 18, 20.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. 2. ed. New York: Springer, 2009. ISBN 978-0387848570. Citado 1 vez na página 21.

HOWARD, J.; RUDER, S. Universal Language Model Fine-tuning for Text Classification. In: PROCEEDINGS of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). [S.l.: s.n.], 2018. P. 328–339. DOI: 10.18653/v1/P18-1031. Disponível em: https://aclanthology.org/P18-1031. Citado 1 vez na página 23.

KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. **arXiv preprint arXiv:1412.6980**, 2014. Acesso em: 23 jun. 2025. Disponível em: https://arxiv.org/abs/1412.6980. Citado 1 vez na página 19.

KOSKO, B. **Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence**. [S.l.]: Prentice Hall, 1992. Citado 1 vez na página 24.

KUTUKOV, A.; SYSOEV, A.; BONDARENKO, D. Robotic Process Automation in Education and Audit: Review and Prospects. **Procedia Computer Science**, v. 218, p. 1123–1130, 2023. DOI: 10.1016/j.procs.2023.01.141. Citado 2 vez na página 26.

LACITY, M. C.; WILLCOCKS, L. P. Business Process Automation and the Role of Webhooks. **MIS Quarterly Executive**, v. 13, n. 3, p. 143–156, 2014. Acesso em: 23 jun. 2025. Citado 1 vez na página 28.

LANGS, G.; RÖHRICH, S.; HOFMANNINGER, J.; PRAYER, D.; PAN, J.; BISCHOF, H. Machine Learning: From Radiomics to Discovery and Routine. **Der Radiologe**, v. 58, p. 1–6, 2018. Imagem sob licença CC BY 4.0. Acesso em: 23 jun. 2025. DOI: 10.1007/s00117-018-0407-3. Disponível em: https://commons.wikimedia.org/wiki/File:Machine_learning_-_from_radiomics_to_discovery_and_routine.png. Citado 0 vez na página 20.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep Learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015. DOI: 10.1038/nature14539. Citado 3 vezes nas páginas 17, 19.

MACDONALD, M. **RPA and Microsoft Power Automate**. [S.l.]: Apress, 2020. Citado 2 vezes nas páginas 27, 29.

MCCARTHY, J. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I. **Communications of the ACM**, v. 3, n. 4, p. 184–195, 1960. Citado 1 vez na página 15.

MCCARTHY, J.; MINSKY, M.; ROCHESTER, N.; SHANNON, C. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955. **AI Magazine**, v. 27, n. 4, p. 12–14, 2006. Republicação do memorando original de 1955. DOI: 10.1609/aimag.v27i4.1904. Citado 1 vez na página 14.

MENDEL, J. M. Fuzzy Logic Systems for Engineering: A Tutorial. **Proceedings of the IEEE**, v. 83, n. 3, p. 345–377, 1995. Citado 1 vez na página 25.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient Estimation of Word Representations in Vector Space. **arXiv preprint arXiv:1301.3781**, 2013. Acesso em: 23 jun. 2025. Disponível em: https://arxiv.org/abs/1301.3781. Citado 1 vez na página 23.

MURPHY, K. P. **Machine Learning: A Probabilistic Perspective**. Cambridge, MA: MIT Press, 2012. ISBN 978-0262018029. Citado 2 vezes nas páginas 21, 22.

NEWELL, A.; SHAW, J. C.; SIMON, H. A. Report on a General Problem-Solving Program. **Proceedings of the International Conference on Information Processing**, p. 256–264, 1959. Citado 1 vez na página 15.

NEWELL, A.; SIMON, H. A. The Logic Theory Machine: A Complex Information Processing System. In: 9. PROCEEDINGS of the IRE. [S.l.: s.n.], 1956. v. 44, p. 1555–1564. Citado 1 vez na página 15.

NILSSON, N. J. **Artificial Intelligence: A New Synthesis**. San Francisco: Morgan Kaufmann, 1998. Citado 2 vezes nas páginas 14, 16.

NWANKPA, C.; IJOMAH, W.; GACHAGAN, A.; MARSHALL, S. **Activation Functions: Comparison of Trends in Practice and Research for Deep Learning**. [S.l.: s.n.], 2018. arXiv preprint arXiv:1811.03378. Acesso em: 23 jun. 2025. Disponível em: https://arxiv.org/abs/1811.03378. Citado 1 vez na página 18.

OETIKER, T.; PARTL, H.; HYNA, I.; SCHLEGL, E. **The Not So Short Introduction to LaTeX2 (versão com comentários sobre web APIs)**. [S.l.: s.n.], 2020. Capítulo sobre comunicação web e formatos de dados. Acesso em: 23 jun. 2025. Citado 1 vez na página 27.

ROSENBLATT, F. The Perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, n. 6, p. 386–408, 1958. Citado 1 vez na página 18.

ROSS, T. J. **Fuzzy Logic with Engineering Applications**. [S.l.]: John Wiley & Sons, 2004. Citado 1 vez na página 24.

RUDER, S. **Neural Transfer Learning for Natural Language Processing**. [S.l.: s.n.], 2019. PhD Thesis, National University of Ireland. Acesso em: 23 jun. 2025. Disponível em: https://ruder.io/thesis/. Citado 1 vez na página 22.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Representations by Back-propagating Errors. **Nature**, v. 323, p. 533–536, 1986. DOI: 10.1038/323533a0. Citado 1 vez na página 19.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 4. ed. [S.l.]: Pearson, 2021. Citado 6 vezes nas páginas 11, 15–17.

SUGIYAMA, S. et al. **Technological Affect and the Authenticity Paradox**. [S.l.: s.n.], 2021. Fonte não especificada, revisar ou substituir. Citado 1 vez na página 15.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. 2. ed. [S.l.]: MIT Press, 2018. Citado 5 vezes nas páginas 17, 21, 22.

TURING, A. M. Computing Machinery and Intelligence. **Mind**, v. 59, n. 236, p. 433–460, 1950. Citado 1 vez na página 15.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention Is All You Need. **arXiv preprint arXiv:1706.03762**, 2017. Citado 1 vez na página 23.

WILLCOCKS, L. P.; LACITY, M. C.; CRAIG, A. Robotic Process Automation: Strategic Transformation Lever for Shared Services. **Journal of Information Technology Teaching Cases**, London School of Economics, v. 7, n. 2, p. 17–28, 2017. Disponível em: https://eprints.lse.ac.uk/71146/. Acesso em: 23 jun. 2025. DOI: 10.1057/s41266-016-0016-9. Citado 1 vez na página 11.

ZADEH, L. A. Fuzzy Sets. **Information and Control**, v. 8, n. 3, p. 338–353, 1965. Citado 1 vez na página 24.

APÊNDICE 1 – CÓDIGO-FONTE DO ASSISTENTE VIRTUAL

1.1 SCRIPT PRINCIPAL EM PYTHON

```
# chat_tecnico.py

import streamlit as st
from langchain_community.document_loaders import TextLoader
from langchain.vectorstores import FAISS
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.chat_models import ChatOpenAI
from langchain.chains import RetrievalQA
from langchain.prompts import PromptTemplate
import os
import csv
import pandas as pd
from datetime import datetime

# Estilo e configura  o da p gina
st.set_page_config(page_title="Assistente T cnico IA", page_icon="    "
    )
st.markdown("""
    <style>
        .response-box {
            background-color: #f3f9ff;
            padding: 1.2em;
            border-radius: 10px;
            border: 1px solid #cce0f5;
            margin-top: 1em;
        }
        .section-title {
            font-weight: bold;
            font-size: 18px;
            margin-top: 1em;
        }
        .header-text {
            font-size: 24px;
            font-weight: bold;
            margin-bottom: 0.5em;
        }
    </style>
""", unsafe_allow_html=True)

st.markdown('<div class="header-text">     Assistente T cnico com IA</
    div>', unsafe_allow_html=True)
```

```
st.write("Diagn stico com base em sintomas da linha de montagem de motores.")

# Configura  o da chave OpenAI
os.environ["OPENAI_API_KEY"] = "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"

# Carregamento da base t cnica
try:
    loader = TextLoader("base_tecnica.txt", encoding="utf-8")
    docs = loader.load()
except Exception as e:
    st.error(f"Erro ao carregar base_tecnica.txt: {e}")
    st.stop()

# Banco vetorial
db = FAISS.from_documents(docs, OpenAIEmbeddings())

# Prompt estruturado
template = """
Voc     um assistente t cnico especializado em montagem de motores automotivos.

Com base no conte do t cnico abaixo e na pergunta feita, elabore uma resposta com a seguinte estrutura:

1. **Resumo t cnico do problema**
2. **Contexto t cnico e causas poss veis**
3. **Solu  o recomendada**

Base de conhecimento:
{context}

D vida do operador:
{question}
"""

prompt = PromptTemplate(input_variables=["context", "question"], template=template)

qa = RetrievalQA.from_chain_type(
    llm=ChatOpenAI(model_name="gpt-3.5-turbo"),
    chain_type="stuff",
    retriever=db.as_retriever(),
    chain_type_kwargs={"prompt": prompt}
```

```
80 )
81
82 # Entrada do usu rio
83 user_input = st.text_input("        Digite sua d vida t cnica:")
84
85 if user_input:
86     with st.spinner("     Analisando sua d vida..."):
87         resposta = qa.run(user_input)
88         data_hora = datetime.now().strftime("%d/%m/%Y %H:%M")
89         with open("historico.csv", mode="a", newline="", encoding="utf-8
               ") as file:
90             writer = csv.writer(file)
91             writer.writerow([data_hora, user_input, resposta])
92
93     st.markdown('<div class="section-title">     Resultado da an lise
           t cnica:</div>', unsafe_allow_html=True)
94     st.markdown(f'<div class="response-box">{resposta}</div>',
           unsafe_allow_html=True)
95
96     # --- Coleta de feedback integrada ---
97     with st.expander("     Avalie esta resposta"):
98         estrelas = st.slider("Nota (1  5 )", 1, 5, 3)
99         comentario = st.text_input("Coment rio (opcional):")
100
101         if st.button("Enviar feedback"):
102             with open("feedback.csv", mode="a", newline="", encoding="
                    utf-8") as f:
103                 writer = csv.writer(f)
104                 writer.writerow([data_hora, user_input, resposta,
                        estrelas, comentario])
105             st.success("    Feedback registrado com sucesso.")
106
107 # --- Hist rico de feedbacks ---
108 with st.expander("     Ver hist rico de intera  es com avalia  o"):
109     try:
110         with open("feedback.csv", encoding="utf-8") as f:
111             reader = csv.reader(f)
112             dados = list(reader)
113
114         if not dados:
115             st.info("Nenhum feedback registrado ainda.")
116         else:
117             df = pd.DataFrame(dados, columns=["Data/Hora", "Pergunta", "
                    Resposta", "Nota", "Coment rio"])
118             df["Nota"] = pd.to_numeric(df["Nota"], errors='coerce').
                    fillna(0).astype(int)
```

```
119             st.dataframe(df[["Data/Hora", "Pergunta", "Nota", "
                    Coment rio"]].sort_values("Data/Hora", ascending=False),
                     use_container_width=True)
120     except FileNotFoundError:
121         st.info("Nenhum feedback encontrado ainda.")
```

Listing 1.1 – Script principal