Relato de Experiência

Automação do Onboarding Corporativo com Agente Conversacional Baseado em IA: Um Relato de Experiência

Corporate Onboarding Automation with an AI-Based Conversational Agent: An Experience Report

Vinicius de Castro Coutinho [Pesquisador independente | *viniciuscoutinho_@hotmail.com*]

Pesquisador independente, Fortaleza, Ceará, Brasil.

Resumo. O *onboarding*, processo de integração de novos colaboradores, costuma gerar um volume grande de dúvidas repetidas que acabam caindo sobre a equipe de Recursos Humanos (RH). Este artigo relata o desenvolvimento de um agente conversacional baseado em Inteligência Artificial (IA), integrado ao *Slack*, para responder a essas dúvidas. O agente foi construído e testado no cenário de uma empresa hipotética, chamada aqui de Empresa E: uma organização de tecnologia com 85 colaboradores, nove departamentos e trabalho híbrido, cujos dados foram todos criados para este estudo. A solução roda na plataforma de automação *n8n* e usa o modelo de linguagem *Gemini 3.5 Flash*, do Google, uma base de conhecimento estruturada com os departamentos da empresa e uma memória que guarda o histórico recente de cada conversa. A arquitetura segue uma versão simplificada do padrão RAG (*Retrieval-Augmented Generation*): em vez de busca vetorial, o conhecimento da empresa fica disponível ao agente como uma ferramenta que ele consulta quando precisa. Nos testes, feitos em ambiente de desenvolvimento, o agente respondeu corretamente a perguntas sobre responsáveis por setor, processos, prazos e contatos, e manteve o contexto em conversas com várias trocas de mensagens. As principais limitações são a base de conhecimento mantida manualmente e a falta de escalada automática, isto é, de um mecanismo que encaminhe a uma pessoa do RH as perguntas que o agente não consegue responder.

Abstract. Employee *onboarding* usually generates a large volume of repetitive questions that end up on the Human Resources (HR) team. This paper reports the development of an Artificial Intelligence (AI) conversational agent integrated with *Slack* to answer these questions. The agent was built and tested in the scenario of a hypothetical company, referred to here as Company E: a technology organization with 85 employees, nine departments and hybrid work, whose data was entirely created for this study. The solution runs on the *n8n* automation platform and uses Google's *Gemini 3.5 Flash* language model, a structured knowledge base describing the company's departments, and a memory that keeps the recent history of each conversation. The architecture follows a simplified version of the RAG (Retrieval-Augmented Generation) pattern: instead of vector search, company knowledge is exposed to the agent as a tool it calls when needed. In tests run in a development environment, the agent correctly answered questions about department heads, processes, deadlines and contacts, and kept context across multi-turn conversations. The main limitations are the manually maintained knowledge base and the lack of automatic escalation, that is, a mechanism that forwards to an HR person the questions the agent cannot answer.

Palavras-chave: onboarding; agente conversacional; inteligência artificial; automação de processos; slack; n8n

Keywords: onboarding; conversational agent; artificial intelligence; process automation; slack; n8n

Recebido/Received: 13 07 2026 • **Aceito/Accepted:** 07 10 2026 • **Publicado/Published:** 15 10 2026

Publicado sob a Licença Pública Internacional Creative Commons Atribuição 4.0 (CC BY 4.0)

1 Introdução

Quem entra numa empresa nova passa as primeiras semanas perguntando. Com quem falo para pedir férias? Quando sai o contracheque? Quem libera o acesso a tal sistema? Essas dúvidas são normais e fazem parte do *onboarding*, o processo pelo qual a organização ajuda o novo colaborador a se ajustar ao trabalho, às pessoas e às regras da casa [Bauer, 2010]. A literatura sobre socialização organizacional mostra que esse ajuste tem efeito concreto: colaboradores que entendem com clareza o seu papel e sabem a quem recorrer tendem a ter mais desempenho e satisfação, e menos intenção de sair [Bauer *et al*., 2007].

O problema é que boa parte dessas perguntas se repete a cada nova contratação, e quase todas acabam no RH. A equipe responde a mesma coisa várias vezes, perde tempo que poderia ir para atividades mais estratégicas, e o novo colaborador fica esperando uma resposta que já estava escrita em algum lugar [Wandhe, 2026].

Os *chatbots* não são uma ideia nova para esse tipo de demanda [Adamopoulou and Moussiades, 2020], mas durante muito tempo dependeram de regras e respostas fixas, que quebravam quando a pergunta vinha formulada de um jeito diferente. Os Modelos de Linguagem de Grande Escala (*Large Language Models*, LLMs) mudaram esse quadro, porque conseguem entender perguntas em linguagem natural e produzir respostas contextualizadas [Brown *et al*., 2020; Zhao

Tabela 1. Resumo da base de conhecimento fictícia da Empresa E.

Departamento	Exemplos de informação cadastrada
RH	Contracheque até o 5º dia útil; férias pedidas com 30 dias de antecedência; alterações cadastrais feitas pelo RH
Financeiro	Reembolso pago em até 5 dias úteis após aprovação; despesas acima de R$ 1.000 exigem dupla aprovação
Compras	Fluxo: formulário interno, aprovação da liderança, aprovação do Financeiro, envio ao fornecedor
TI	Pedidos via *Slack* ou sistema interno; resposta em até 4 horas úteis
Demais setores	Comercial, Marketing, Operações, Jurídico e Suporte: responsável, e-mail, atribuições e dúvidas comuns

et al., 2023]. No RH, as revisões mais recentes apontam o atendimento a dúvidas rotineiras como uma das aplicações mais promissoras desses modelos [Sakib *et al*., 2024], e uma pesquisa da IBM indica que 38% dos líderes de RH já estavam implementando IA generativa nas suas operações [IBM, 2025].

Usar um LLM sozinho, porém, não resolve. O modelo não conhece os processos internos de uma empresa específica e, quando não sabe a resposta, pode inventar uma que soa plausível, fenômeno chamado de alucinação [Ji *et al*., 2023]. A técnica mais usada para contornar isso é a geração aumentada por recuperação (*Retrieval-Augmented Generation*, RAG), em que o sistema busca informações numa base de conhecimento da própria organização e as entrega ao modelo junto com a pergunta [Lewis *et al*., 2020; Gao *et al*., 2024]. Assim, o modelo responde a partir de dados confiáveis em vez de depender só do que aprendeu no treinamento. Neste trabalho foi adotada uma versão simplificada desse padrão, explicada em detalhe nas Seções 3.2 e 4.

Este artigo relata a construção de um agente conversacional de *onboarding* integrado ao *Slack*, ferramenta de comunicação corporativa usada por muitas empresas no dia a dia [Slack Technologies, 2025]. O agente foi montado na plataforma de automação *n8n* [n8n GmbH, 2025], usa um modelo da família *Gemini*, do Google [Gemini Team, Google, 2023], e foi testado num cenário simulado, descrito a seguir.

1.1 O cenário: a Empresa E

Para desenvolver e testar a solução sem expor dados de uma organização real, foi criada uma empresa hipotética, chamada aqui de **Empresa E**. O nome foi escolhido de propósito por ser genérico, para evitar qualquer confusão com empresas existentes. Todos os dados descritos nesta seção foram inventados para o estudo.

A Empresa E atua com tecnologia, automação e soluções digitais, foi fundada em 2018, tem 85 colaboradores e trabalha em modelo híbrido, com parte da equipe presencial e parte remota. Sua estrutura tem nove departamentos com responsáveis definidos: Recursos Humanos, Financeiro, Compras, Tecnologia da Informação (TI), Comercial, Marketing, Operações, Jurídico e Atendimento ao Cliente, além da Diretoria. A comunicação interna acontece no *Slack*.

Para cada departamento foram cadastrados, com dados fictícios, o nome do responsável, um e-mail de contato, as atribuições do setor, informações importantes (prazos e regras internas) e exemplos de dúvidas comuns. Os e-mails usam o domínio `empresa-e.example`, que é reservado para exemplos e não pertence a ninguém. A Tabela 1 resume o conteúdo dessa base.

A situação simulada é a de um colaborador recém-contratado que, nos primeiros dias, recorre ao *Slack* para tirar dúvidas que normalmente iriam para o RH. Em vez de procurar alguém da equipe, ele escreve para o agente, que responde no mesmo canal.

1.2 Objetivos

O objetivo geral deste trabalho é relatar o desenvolvimento de um agente conversacional baseado em IA, integrado ao *Slack*, capaz de responder de forma autônoma às dúvidas rotineiras de novos colaboradores durante o *onboarding*.

Os objetivos específicos são:

- estruturar uma base de conhecimento com as informações de cada departamento da empresa;
- construir, em uma plataforma *low-code*, um fluxo que receba as mensagens do *Slack*, consulte essa base e devolva a resposta ao colaborador;
- manter o contexto de cada conversa, para que o colaborador possa fazer perguntas de continuação;
- testar o agente com perguntas típicas de *onboarding* e discutir o que funcionou, as limitações e os próximos passos.

As contribuições do trabalho são a descrição detalhada de uma arquitetura *low-code* reproduzível para esse tipo de agente, o relato das decisões de implementação e uma discussão honesta sobre o alcance e as limitações da abordagem.

1.3 Estrutura do artigo

A Seção 2 discute trabalhos relacionados e posiciona esta proposta em relação a eles. A Seção 3 apresenta os conceitos usados na solução. A Seção 4 descreve como o agente foi construído e testado. A Seção 5 traz os resultados e as limitações, e a Seção 6 fecha o artigo com as conclusões e os trabalhos futuros.

2 Trabalhos Relacionados

Wandhe [2026] analisa o uso de *chatbots* com IA no *onboarding* e argumenta que eles reduzem a carga administrativa do RH e tornam a integração mais acessível e personalizada para o novo colaborador. O trabalho é conceitual: discute benefícios e cuidados, mas não descreve uma implementação.

Sakib *et al*. [2024] fazem uma revisão crítica sobre *chatbots* baseados em LLMs na gestão de pessoas. Os autores apontam ganhos de eficiência, principalmente no atendimento a perguntas frequentes, e chamam atenção para questões de privacidade, viés e confiabilidade das respostas. Também é um trabalho de revisão, sem um sistema construído.

No lado técnico, Gao *et al*. [2024] reúnem as principais arquiteturas de RAG, desde a versão mais simples, que recupera trechos de documentos e os passa ao modelo, até variações com etapas de reescrita da pergunta e reordenação dos resultados. Esse *survey* é a base conceitual para a escolha feita aqui, de uma recuperação estruturada no lugar da busca vetorial.

Na indústria, o caso mais conhecido é o AskHR, da IBM [IBM, 2024]. Segundo a empresa, o sistema processou mais de 11,5 milhões de interações em 2024, resolveu 94% das perguntas sem intervenção humana e reduziu em 75% os chamados abertos para o RH desde 2016. É uma solução de

grande porte, construída com uma equipe e uma infraestrutura que não estão ao alcance de empresas menores.

Sobre a plataforma usada, Amir e Atif [2026] avaliaram o *n8n* num estudo de caso de pequena empresa. Os autores implementaram um fluxo de processamento de *leads*, que armazena os dados recebidos, envia um e-mail de confirmação e dispara notificações em tempo real, e compararam 20 execuções manuais com 25 execuções automatizadas em condições controladas. O tempo médio caiu de 185,35 para 1,23 segundos, e a taxa de erro foi de 5% no processo manual para zero no automatizado. É importante deixar claro que esse estudo não envolve LLMs nem agentes conversacionais: ele serve aqui como evidência de que o *n8n* é uma plataforma viável para automatizar processos em organizações pequenas, e não como validação do agente proposto.

A Tabela 2 compara esses trabalhos com a solução deste artigo.

Olhando o conjunto, os trabalhos de RH discutem o potencial dos *chatbots*, mas não mostram como construir um. O caso da IBM mostra que funciona, mas numa escala distante da realidade da maioria das empresas. E o estudo com *n8n* mostra que a plataforma serve para automação, mas sem IA. Este trabalho fica no meio desse caminho: descreve, passo a passo, um agente de *onboarding* com LLM que pode ser montado por uma pessoa só, em uma ferramenta *low-code*, sem infraestrutura própria. O que ele não oferece é uma avaliação quantitativa em produção, ponto discutido na Seção 5.

3 Referencial Teórico

Esta seção apresenta os quatro conceitos em que a solução se apoia. Primeiro, os LLMs, que são o motor do agente. Depois, o padrão RAG, que define como o agente acessa o conhecimento da empresa. Em seguida, as plataformas *low-code*, onde o fluxo foi construído. Por fim, a memória conversacional, que permite ao agente lembrar do que já foi dito.

3.1 Modelos de Linguagem de Grande Escala (LLMs)

Os LLMs são redes neurais treinadas com enormes volumes de texto para prever a próxima palavra de uma sequência. Quase todos se baseiam na arquitetura *Transformer*, proposta por Vaswani *et al*. [2017], que usa um mecanismo de atenção para relacionar cada palavra com todas as outras do texto, o que permite capturar contexto de longo alcance.

Quando esses modelos passaram a ter bilhões de parâmetros, surgiu uma característica importante para este trabalho: eles conseguem executar tarefas novas apenas a partir de instruções e exemplos escritos no próprio pedido, sem precisar de um novo treinamento [Brown *et al*., 2020]. Na prática, isso significa que basta descrever ao modelo o seu papel, por exemplo "você é um assistente de *onboarding*", e fornecer as informações necessárias para que ele passe a atuar nessa função. O texto que define esse papel é chamado de *prompt* de sistema. Zhao *et al*. [2023] apresentam um panorama completo da evolução desses modelos.

O *Gemini* é uma família de modelos multimodais desenvolvida pelo Google, capaz de lidar com texto, imagem, áudio e código [Gemini Team, Google, 2023]. Neste trabalho foi usada a versão *Gemini 3.5 Flash*, uma variante mais leve e rápida da família, escolhida principalmente por ter uma API com camada gratuita durante o desenvolvimento e por responder bem em português.

Os LLMs têm limitações conhecidas. A principal, para um assistente corporativo, é a alucinação: o modelo pode gerar uma resposta fluente e convincente, mas incorreta [Ji *et al*., 2023]. Ele também não sabe nada sobre a empresa além do que for informado. As duas limitações levam diretamente ao próximo conceito.

3.2 Arquitetura RAG (*Retrieval-Augmented Generation*)

O RAG foi proposto por Lewis *et al*. [2020] como forma de combinar o conhecimento que o modelo aprendeu no treinamento com informações buscadas em uma fonte externa no momento da pergunta. O funcionamento básico tem três passos: o sistema recebe a pergunta, recupera da base de conhecimento os trechos mais relevantes e entrega esses trechos ao modelo junto com a pergunta, para que a resposta seja gerada a partir deles [Gao *et al*., 2024].

Na versão mais comum, os documentos são divididos em pedaços e convertidos em vetores numéricos (*embeddings*), e a recuperação é feita por similaridade entre o vetor da pergunta e os vetores dos pedaços. Gao *et al*. [2024] chamam essa abordagem de RAG ingênuo (*naive RAG*) e descrevem variantes mais elaboradas. As vantagens em relação a usar o modelo sozinho são respostas mais fiéis aos dados da organização, menos alucinação e a possibilidade de atualizar o conhecimento sem treinar o modelo de novo.

Neste trabalho foi usada uma variante mais simples. A base da Empresa E é pequena e já está organizada por departamento, então não há necessidade de busca vetorial. Ela foi escrita como um objeto estruturado e oferecida ao agente como uma ferramenta: quando precisa de uma informação, o agente chama a ferramenta e recebe a base. A ideia de que o modelo decida sozinho quando usar ferramentas externas foi explorada por Schick *et al*. [2023] e, combinada com etapas de raciocínio, por Yao *et al*. [2023], no padrão conhecido como ReAct. O componente de agente do *n8n* segue essa linha. A troca é clara: o desenho fica mais simples e previsível, mas não escala para bases grandes ou documentos sem estrutura, como discutido na Seção 5.

3.3 Automação de Processos com Plataformas *Low-Code*

Plataformas *low-code* são ambientes em que se constrói *software* principalmente por meio de interfaces visuais e configuração, com pouca ou nenhuma programação [Bock and Frank, 2021]. A proposta é permitir que pessoas sem formação aprofundada em desenvolvimento criem aplicações e automações, e que equipes técnicas entreguem mais rápido. Sahay *et al*. [2020] compararam várias dessas plataformas e mostram que elas variam bastante em recursos, mas compartilham a ideia de montar a lógica com componentes prontos ligados entre si. Di Ruscio *et al*. [2022] relacionam essa abordagem com a engenharia dirigida por modelos, já conhecida na área de *software*.

O *n8n* é uma plataforma desse tipo, voltada para auto-

Tabela 2. Comparação entre os trabalhos relacionados e a solução proposta.

Trabalho	Tipo	Usa LLM	Foco
Wandhe [2026]	Conceitual	Sim (discussão)	*Chatbots* no *onboarding*
Sakib *et al*. [2024]	Revisão	Sim (discussão)	LLMs na gestão de pessoas
Gao *et al*. [2024]	*Survey*	Sim	Arquiteturas RAG
IBM AskHR [IBM, 2024]	Caso industrial	Sim	Atendimento de RH em larga escala
Amir e Atif [2026]	Estudo de caso	Não	Automação de fluxos com *n8n*
Este trabalho	Relato de experiência	Sim	Agente de *onboarding low-code* no *Slack*, montado por uma pessoa e sem infraestrutura própria

mação de fluxos de trabalho [n8n GmbH, 2025]. Cada fluxo é formado por nós, e cada nó executa uma ação: receber um evento, aplicar uma condição, chamar uma API, rodar um trecho de código. A plataforma tem conectores prontos para centenas de ferramentas, incluindo o *Slack*, e oferece nós de IA construídos sobre a biblioteca *LangChain* [Chase, 2022], que facilitam montar agentes com modelo de linguagem, memória e ferramentas. Foi essa combinação que tornou possível construir o agente deste trabalho sem escrever uma aplicação do zero.

3.4 Memória Conversacional em Agentes de IA

Um LLM, por si só, não guarda nada entre uma chamada e outra. Cada mensagem é processada isoladamente, então, sem algum mecanismo de memória, o agente não saberia a que se refere uma pergunta como "e qual o e-mail dele?". A memória é considerada um dos componentes centrais de agentes baseados em LLMs, ao lado do planejamento e do uso de ferramentas [Wang *et al*., 2024; Zhang *et al*., 2024].

A forma mais simples de memória é guardar as últimas mensagens da conversa e reenviá-las ao modelo a cada nova pergunta. Essa estratégia é chamada de memória de janela deslizante (*window buffer memory*), porque só os últimos N turnos ficam guardados e os mais antigos vão sendo descartados. Ela é barata e fácil de implementar, mas tem um limite: o que sai da janela é esquecido. Existem abordagens mais sofisticadas, como a de Packer *et al*. [2023], que gerencia diferentes níveis de memória de forma parecida com um sistema operacional. Há ainda um detalhe que justifica não usar janelas muito grandes: Liu *et al*. [2024] mostraram que os modelos tendem a aproveitar pior as informações que ficam no meio de contextos longos. Para conversas curtas de *onboarding*, a janela deslizante foi considerada suficiente.

4 Metodologia

Este trabalho é um relato de experiência de pesquisa aplicada. O problema que motivou a solução, a sobrecarga do RH com dúvidas repetitivas no *onboarding*, é real e bem documentado na literatura [Wandhe, 2026; Sakib *et al*., 2024]. Para estudá-lo sem depender dos dados de uma organização específica, o agente foi construído e testado numa simulação: a Empresa E, apresentada na Seção 1.1, com dados inteiramente fictícios. Esse tipo de abordagem, em que se constrói um artefato para um problema real e se avalia o artefato num ambiente controlado antes de levá-lo a uma organização, é comum na pesquisa em sistemas de informação [Hevner *et al*., 2004].

O trabalho foi feito em quatro etapas: (1) montagem da base de conhecimento da Empresa E; (2) construção do fluxo no *n8n*; (3) escrita do *prompt* de sistema; e (4) testes no *Slack*. As subseções a seguir descrevem cada uma.

4.1 Arquitetura e componentes do agente

O agente é um fluxo do *n8n* com sete nós, mostrado na Figura 1. A linha principal tem quatro nós, que levam a mensagem do *Slack* até o agente e devolvem a resposta. Os outros três se ligam ao nó central de agente e fornecem a ele o modelo de linguagem, a memória e a base de conhecimento. A Figura 2 mostra o mesmo fluxo como ele aparece na interface do *n8n*, já com os nós configurados.

O caminho de uma mensagem, do envio pelo colaborador até a resposta, passa pelos seguintes nós.

***Slack Trigger*.** É o ponto de entrada. Fica escutando os eventos do *workspace* do *Slack* e, a cada mensagem nova, captura o texto, o identificador do canal e a referência da *thread*, se houver.

Filtro de identidade (*If*). Verifica quem enviou a mensagem. Como o agente também escreve no *Slack*, as respostas dele geram novos eventos que voltariam para o próprio fluxo e criariam um laço infinito de respostas. O filtro compara o identificador do remetente com o do bot e só deixa passar mensagens enviadas por pessoas.

***AI Agent*.** É o centro da solução. Usa o componente de agente do *n8n*, construído sobre o *LangChain* [Chase, 2022]. Recebe o texto da mensagem, segue as instruções do *prompt* de sistema (Seção 4.2) e decide, a cada pergunta, se precisa consultar a base de conhecimento. Trabalha com três recursos conectados a ele:

- **modelo de linguagem**: o *Gemini 3.5 Flash*, que interpreta a pergunta e escreve a resposta;
- ***Simple Memory***: a memória de janela deslizante, configurada para guardar os últimos 10 turnos. A chave da memória é o identificador do canal do *Slack*, então cada canal tem o seu próprio histórico e uma conversa não se mistura com outra;
- ***Code Tool***: a ferramenta que devolve a base de conhecimento da Empresa E, escrita como um objeto JavaScript (Seção 4.2).

Enviar mensagem. Publica a resposta gerada pelo agente no mesmo canal de onde veio a pergunta e, quando a pergunta foi feita numa *thread*, responde dentro dela, para manter a conversa organizada.

Um exemplo ajuda a visualizar o funcionamento. Um colaborador novo escreve no canal: "Como faço para pedir férias?". O *Slack Trigger* captura a mensagem, o filtro confirma que ela veio de uma pessoa e o agente a recebe. O modelo percebe que precisa de informação da empresa, chama a *Code Tool*, encontra no departamento de RH a regra das férias e

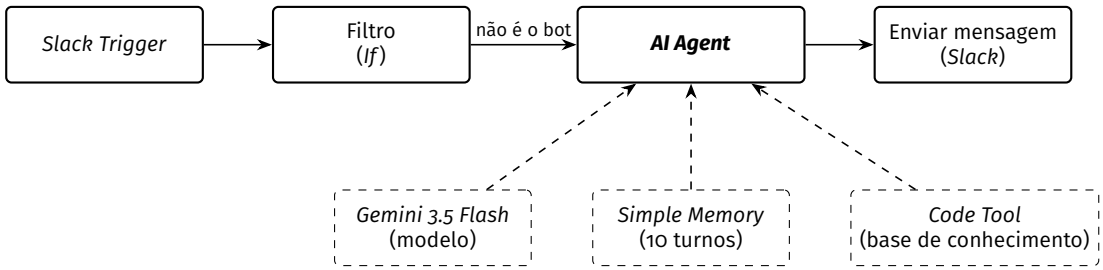


Figura 1. Fluxo do agente no *n8n*. As setas contínuas mostram o caminho da mensagem; as tracejadas, os recursos conectados ao agente.

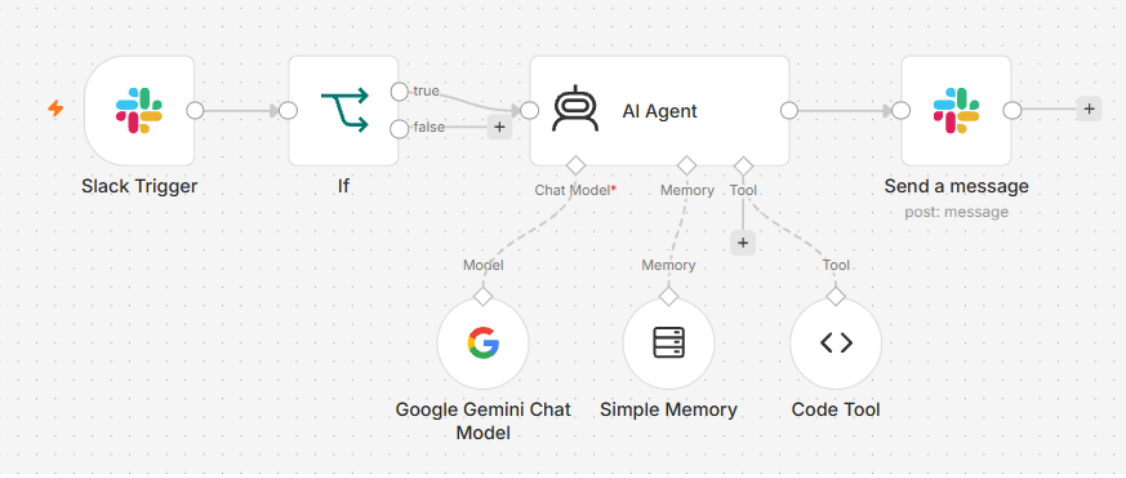


Figura 2. O fluxo implementado, como aparece no editor do *n8n*. Os nós de modelo, memória e ferramenta se conectam ao *AI Agent* pelas entradas *Chat Model*, *Memory* e *Tool*.

Código 1: *Prompt* de sistema do agente.

```
Voc  um assistente interno responsvel por apoiar o onboarding de
     novos colaboradores. Seu papel  esclarecer dvidas sobre
     setores, processos, contatos e rotinas da empresa. Sempre
     que for perguntado a pessoa responsvel por algum setor,
     informe a partir das informaes abaixo. Para responder as
     dvidas utilize as informaes encontradas no n de Code Tool.
```

escreve a resposta, que o último nó publica no canal. A pergunta e a resposta ficam guardadas na memória daquele canal. Se, em seguida, o colaborador perguntar "e com quem eu falo sobre isso?", o agente usa o histórico para entender que "isso" se refere às férias e informa o responsável pelo RH.

4.2 *Prompt* de sistema e base de conhecimento

O comportamento do agente é definido por dois elementos: o *prompt* de sistema, que diz ao modelo qual é o seu papel, e a base de conhecimento, que diz o que ele sabe sobre a empresa.

O *prompt* de sistema é o texto que o modelo recebe antes de qualquer mensagem do usuário. Ele funciona como uma descrição de função: diz quem o agente é, o que deve fazer e onde buscar as informações. O *prompt* usado está no Código 1. Ele foi mantido curto de propósito e tem três partes: o papel (assistente interno de *onboarding*), o escopo (setores, processos, contatos e rotinas) e uma instrução explícita para buscar as respostas na *Code Tool*. Essa última instrução é a mais importante, porque é ela que leva o modelo a responder com base nos dados da empresa e não no que ele "acha" que sabe.

A base de conhecimento foi escrita como um objeto JavaScript dentro da *Code Tool*. A escolha por um objeto estruturado, e não por um texto corrido, tem dois motivos. O primeiro é que cada informação fica num lugar previsível (departamento, responsável, e-mail, atribuições), o que facilita tanto a consulta pelo modelo quanto a manutenção por quem atualiza a base. O segundo é que, com uma base desse tamanho, cabe enviar o conteúdo inteiro ao modelo de uma vez, sem precisar dividir em pedaços nem gerar *embeddings*.

O objeto tem quatro blocos: dados gerais da empresa; missão, visão e valores; a lista de departamentos; e o detalhamento de cada departamento. O Código 2 mostra um trecho com os dados gerais e o departamento de RH. Os outros oito departamentos seguem a mesma estrutura, com campos extras quando faz sentido, como o fluxo de aprovação em Compras e as regras de reembolso no Financeiro (Tabela 1).

O campo `duvidasComuns` merece um comentário. Ele não é usado para comparar perguntas de forma literal. Serve como exemplo para o modelo de que tipo de pergunta cada departamento resolve, o que ajuda o agente a escolher o setor certo quando a pergunta vem escrita de outro jeito.

4.3 Procedimento de teste

Os testes foram feitos em ambiente de desenvolvimento, num *workspace* do *Slack* criado para o estudo e conectado ao fluxo do *n8n*. O autor fez o papel de um colaborador recém-contratado da Empresa E e enviou ao agente perguntas de quatro tipos:

1. **responsáveis e contatos**: quem cuida de cada setor e qual o e-mail;
2. **processos**: como pedir férias, como solicitar uma compra, como pedir reembolso;

Tabela 3. Exemplos representativos das perguntas de teste e da informação esperada na resposta, segundo a base de conhecimento.

Tipo	Pergunta enviada	Informação esperada na resposta
Contato	Quem é o responsável pelo RH?	Vitor Frota, rh@empresa-e.example
Contato	Com quem eu falo sobre LGPD?	Jurídico, Fernanda Almeida
Processo	Preciso comprar um monitor, como faço?	Formulário interno, aprovação da liderança, aprovação do Financeiro, envio ao fornecedor
Processo	Como peço reembolso de uma despesa?	Formulário com comprovantes enviado ao Financeiro
Prazo	Quando sai meu contracheque?	Até o 5º dia útil
Prazo	Abri um chamado na TI, quanto tempo demora?	Até 4 horas úteis
Continuação	(logo após a anterior) E qual o e-mail deles?	ti@empresa-e.example

Código 2: Trecho da base de conhecimento (*Code Tool*).

```
const onboarding = {
  empresa: {
    nome: "Empresa E",
    segmento: "Tecnologia, Automao e Solues Digitais",
    numeroColaboradores: 85,
    modeloTrabalho: "Hbrido (presencial e remoto)"
  },
  departamentos: {
    rh: {
      nome: "Recursos Humanos",
      responsavel: "Vitor Frota",
      email: "rh@empresa-e.example",
      atribuicoes: ["Admisso e desligamento",
        "Emisso de contracheque", "Gesto de frias",
        "Administrao de benefcios"],
      informacoesImportantes: [
        "Contracheque disponibilizado at o 5 dia til",
        "Frias solicitadas com 30 dias de antecedncia"],
      duvidasComuns: [
        "Com quem falo para solicitar meu contracheque?",
        "Como fao para pedir frias?"]
    }
    // demais departamentos seguem a mesma estrutura
  }
};
return onboarding;
```

3. **prazos e regras**: quando sai o contracheque, prazo de reembolso, tempo de resposta da TI;
4. **continuação**: perguntas que só fazem sentido com o contexto anterior, como "e qual o e-mail dele?", para verificar a memória.

As perguntas foram escritas de formas variadas, e não apenas copiadas do campo `duvidasComuns`, para observar se o agente lidava com formulações diferentes. Cada resposta foi conferida manualmente contra a base de conhecimento. Como se trata de um relato de experiência, a avaliação foi qualitativa: o critério foi verificar se a resposta estava correta e de acordo com a base, se o agente usava o contexto da conversa e se a resposta fazia sentido para quem perguntou. A Tabela 3 mostra exemplos representativos das perguntas de cada tipo.

5 Resultados

Nos testes, o agente respondeu corretamente às perguntas sobre responsáveis por setor, processos de solicitação, prazos e contatos de e-mail, sempre de acordo com o que estava cadastrado na base. Perguntas escritas de forma diferente da prevista no campo `duvidasComuns`, como "preciso comprar um monitor, como faço?", também foram encaminhadas ao setor certo, o que mostra a vantagem de usar um LLM em vez de regras fixas.

A memória funcionou como esperado. Em conversas com várias trocas, o agente entendeu perguntas de continuação, como pedir o e-mail "deles" logo depois de uma pergunta sobre a TI, sem que o colaborador precisasse repetir o assunto. Como a memória é separada por canal, conversas em canais diferentes não interferiram entre si.

É preciso ser claro sobre o alcance desses resultados. Os testes mostram que a solução funciona tecnicamente: o fluxo recebe a pergunta, consulta a base, responde corretamente e mantém o contexto. Eles não medem o impacto num ambiente real, como redução de chamados ao RH ou satisfação dos colaboradores, porque não houve implantação numa empresa com usuários reais. Os benefícios a seguir são, portanto, os que o desenho da solução permite esperar, apoiados no que a literatura relata para sistemas parecidos:

- **disponibilidade contínua**: o agente responde a qualquer hora, sem depender da agenda da equipe de RH;
- **respostas padronizadas**: como a resposta vem sempre da mesma base, a informação não muda de acordo com quem responde;
- **menos interrupções ao RH**: dúvidas rotineiras podem ser resolvidas sem envolver a equipe, como relatam Sakib *et al*. [2024] e, em escala bem maior, o caso AskHR [IBM, 2024];
- **manutenção simples**: para mudar uma informação, basta editar o objeto na *Code Tool*, sem alterar o resto do fluxo.

Quanto ao custo, a solução usa a camada gratuita da API do *Gemini* e o *Slack*, que muitas empresas já têm. O *n8n* pode ser instalado em servidor próprio. Para uma empresa do porte da Empresa E, isso torna o custo de operação baixo, embora ele cresça com o volume de mensagens se for necessário passar para um plano pago do modelo.

5.1 Limitações

O trabalho tem limitações que precisam ser consideradas.

(i) **Base mantida à mão.** Toda a informação está num objeto escrito manualmente. Se ninguém atualizar quando um responsável muda de cargo ou uma regra muda, o agente vai passar informação errada com toda a segurança.

(ii) **Sem escalada automática.** Quando o agente não encontra a resposta na base, não há um caminho para levar a dúvida a uma pessoa do RH. O ideal seria que ele reconhecesse que não sabe e abrisse um chamado ou mencionasse alguém da equipe na conversa.

(iii) **Memória curta.** A janela de 10 turnos é suficiente para perguntas rápidas, mas conversas mais longas perdem o começo.

(iv) **Não escala para bases grandes.** Enviar a base inteira ao modelo funciona com nove departamentos, mas não com

centenas de documentos, manuais ou políticas em texto corrido. Nesse caso seria necessário um RAG completo, com busca vetorial [Gao *et al*., 2024].

(v) **Avaliação limitada.** Os testes foram feitos por uma única pessoa, num ambiente simulado e sem métricas quantitativas.

6 Conclusões e Trabalhos Futuros

Este trabalho relatou o desenvolvimento de um agente conversacional baseado em IA para apoiar o *onboarding* de novos colaboradores, integrado ao *Slack* e construído no *n8n*. No cenário simulado da Empresa E, o agente respondeu corretamente a dúvidas sobre responsáveis, processos, prazos e contatos e manteve o contexto ao longo das conversas. Os testes indicam que a solução é tecnicamente viável, de baixo custo e simples de manter, e que pode ser montada por uma pessoa só, sem desenvolver uma aplicação do zero.

A principal lição do processo foi que, para uma base de conhecimento pequena e bem organizada, a versão simplificada de RAG resolve o problema sem a complexidade da busca vetorial. A estrutura da base, separada por departamento e com exemplos de dúvidas, contribuiu tanto quanto o modelo para a qualidade das respostas. Por outro lado, a experiência deixou claro que um agente que não sabe dizer "não sei" e não tem para quem passar a pergunta é um risco num ambiente corporativo.

Como trabalhos futuros, pretende-se: (i) implementar a escalada automática, para que perguntas sem resposta na base sejam encaminhadas ao RH; (ii) incorporar busca vetorial (RAG completo), para lidar com bases maiores e documentos não estruturados; e (iii) implantar a solução numa organização real e medir o impacto com métricas de uso, como volume de perguntas respondidas sem intervenção humana e redução de chamados ao RH.

Declarações complementares

Contribuições dos autores

Este é um trabalho de autor único. VCC contribuiu em todos os aspectos do trabalho: Conceituação, Metodologia, Software, Validação, Análise Formal, Redação (rascunho original) e Redação (revisão e edição), conforme a Taxonomia CRediT (https://credit.niso.org/).

Conflitos de interesse

O autor declara que não há conflitos de interesse.

Disponibilidade de artefatos de pequisa

O arquivo JSON do fluxo n8n e a base de conhecimento utilizada neste estudo serão disponibilizados mediante solicitação. Todos os dados utilizados são fictícios e foram criados exclusivamente para fins acadêmicos.

Uso de Inteligência Artificial

A plataforma n8n e o modelo Gemini 3.5 Flash fazem parte do sistema descrito neste artigo. O Claude (Anthropic), usado por meio do Claude Code em outubro de 2026, auxiliou na redação e na revisão do texto, na verificação das referências em bases bibliográficas e na formatação em LaTeX. O autor revisou todo o conteúdo e é responsável por ele.

Outras informações relevantes

Este estudo não envolveu participantes humanos, dados pessoais reais ou informações sensíveis. Todos os perfis de colaboradores e dados organizacionais utilizados, incluindo o nome Empresa E, são inteiramente fictícios e foram criados exclusivamente para fins acadêmicos.

Referências

Adamopoulou, E. and Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2:100006. DOI: 10.1016/j.mlwa.2020.100006.

Amir, A. R. and Atif, S. M. (2026). Evaluating workflow automation efficiency using n8n: A small-scale business case study. arXiv preprint arXiv:2602.01311.

Bauer, T. N. (2010). *Onboarding New Employees: Maximizing Success*. SHRM Foundation, Alexandria, VA.

Bauer, T. N., Bodner, T., Erdogan, B., Truxillo, D. M., and Tucker, J. S. (2007). Newcomer adjustment during organizational socialization: A meta-analytic review of antecedents, outcomes, and methods. *Journal of Applied Psychology*, 92(3):707–721. DOI: 10.1037/0021-9010.92.3.707.

Bock, A. C. and Frank, U. (2021). Low-code platform. *Business & Information Systems Engineering*, 63(6):733–740. DOI: 10.1007/s12599-021-00726-8.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., *et al*. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Chase, H. (2022). LangChain. `https://github.com/langchain-ai/langchain`.

Di Ruscio, D., Kolovos, D., de Lara, J., Pierantonio, A., Tisi, M., and Wimmer, M. (2022). Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling*, 21(2):437–446. DOI: 10.1007/s10270-021-00970-2.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., and Wang, H. (2024). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997.

Gemini Team, Google (2023). Gemini: A family of highly capable multimodal models. arXiv preprint arXiv:2312.11805.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1):75–105. DOI: 10.2307/25148625.

IBM (2024). Transforming HR support with agentic AI: IBM AskHR. IBM Case Studies.

IBM (2025). What are chatbots for HR? IBM Think Topics.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38. DOI: 10.1145/3571730.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173. DOI: 10.1162/tacl_a_00638.

n8n GmbH (2025). n8n documentation. `https://docs.n8n.io`.

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., and Gonzalez, J. E. (2023). MemGPT: Towards LLMs as operating systems. arXiv preprint arXiv:2310.08560.

Sahay, A., Indamutsa, A., Di Ruscio, D., and Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. In *46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 171–178. DOI: 10.1109/SEAA51224.2020.00036.

Sakib, M. N. *et al*. (2024). The ChatGPT and the future of HR: A critical review on the benefits and challenges of AI chatbots in human resource management. *Multidisciplinary Reviews*, 7(8). DOI: 10.31893/multire.2024136.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551.

Slack Technologies (2025). Slack developer docs. `https://docs.slack.dev`.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Wandhe, P. (2026). AI chatbots as enablers of effective employee onboarding. *SSRN Electronic Journal*. DOI: 10.2139/ssrn.6099951.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., *et al*. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345. DOI: 10.1007/s11704-024-40231-1.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Zhang, Z., Bo, X., Ma, C., Li, R., Chen, X., *et al*. (2024). A survey on the memory mechanism of large language model based agents. arXiv preprint arXiv:2404.13501.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., *et al*. (2023). A survey of large language models. arXiv preprint arXiv:2303.18223.