

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Alfredo Vasconcelos de Andrade

Comparativo de Plataformas para Orquestração de Agentes Inteligentes: n8n vs. Crew AI na Sumarização de Tickets

CAMPINA GRANDE - PB
2025

A553c Andrade, Alfredo Vasconcelos.
Comparativo de plataformas para orquestração de agentes inteligentes: n8n vs. Crew AI na sumarização de tickets / Alfredo Vasconcelos Andrade. - Campina Grande, 2025.
8 f. : il. color.

Monografia (Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2025.
"Orientação: Prof. Dr. Rohit Gheyi."
Referências.

1. Inteligência Artificial. 2. N8n. 3. Crew AI. 4. Sistemas Multiagentes. 5. LLMS. 6. Tickets. I. Gheyi, Rohit. II. Título.

CDU 004.8(04)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA ITAPUANA SOARES DIAS GONÇALVES CRB-15/93

Alfredo Vasconcelos de Andrade

Comparativo de Plataformas para Orquestração de Agentes Inteligentes: n8n vs. Crew AI na Sumarização de Tickets

Trabalho de Conclusão de Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Rohit Gheyi

CAMPINA GRANDE - PB
2025

Alfredo Vasconcelos de Andrade

Comparativo de Plataformas para Orquestração de Agentes Inteligentes: n8n vs. Crew AI na Sumarização de Tickets

Trabalho de Conclusão de Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Rohit Gheyi
Orientador – UASC/CEEI/UFCG

Salatiel Dantas Silva
Examinador – UASC/CEEI/UFCG

Ricardo Santos de Oliveira
Examinador – UASC/CEEI/UFCG

Trabalho aprovado em: 22 de setembro de 2025

CAMPINA GRANDE - PB

ABSTRACT

Inefficient support ticket management in technology companies can lead to significant delays in customer responses, increased operational costs, and overload of support teams, directly affecting customer satisfaction and retention. With the growing volume of data and the need for quick responses, automation and intelligent analysis of these tickets become essential to optimize resources and reduce bottlenecks. This work proposes the creation of a multi-agent system to summarize Zendesk tickets stored in AWS S3 and generate managerial reports, leveraging Large Language Models (LLMs), including but not limited to Claude 3.5 Sonnet. For the implementation, two distinct platforms were evaluated: n8n, an open-source workflow automation tool that adopts the low-code/no-code paradigm and enables the orchestration of AI agents through interconnected nodes; and Crew AI, a Python-based framework that allows the creation and coordination of intelligent agents with greater programmatic control and architectural flexibility. The evaluation was conducted by the same person responsible for the development in both platforms, ensuring consistency in the analysis of criteria such as ease of development, flexibility, maintainability, and suitability for using LLMs in real-world scenarios. The results showed that n8n stands out for its fast development and visual interface, being more suitable for prototyping and direct automations, while Crew AI is better suited for complex architectures — here understood as those involving multiple agents, MCPs, and requiring more specialized algorithms to perform tasks. It is worth noting that Zendesk and S3 were chosen because they are widely used solutions in the market, but the proposed system is not restricted to them. It is concluded that the choice of the most appropriate platform depends directly on the specifications of the demand, such as workflow complexity, available implementation time, and the required level of technical control.

RESUMO

A gestão ineficiente de *tickets* de suporte em empresas de tecnologia pode gerar atrasos significativos nas respostas aos clientes, aumento de custos operacionais e sobrecarga das equipes de atendimento, impactando diretamente a satisfação e a retenção de clientes. Diante do crescimento do volume de dados e da necessidade de respostas rápidas, a automatização e a análise inteligente desses tickets tornam-se essenciais para otimizar recursos e reduzir gargalos. Este trabalho propõe a criação de um sistema multiagente para sumarizar tickets do Zendesk, armazenados no AWS S3, e gerar relatórios gerenciais, utilizando Large Language Models (LLMs), incluindo mas não se limitando ao Claude 3.5 Sonnet. Para a implementação, foram avaliadas duas plataformas distintas: o n8n, ferramenta de automação de fluxo de trabalho de código aberto que adota o paradigma low-code/no-code e permite orquestrar agentes de inteligência artificial por meio de nós interligados; e o Crew AI, framework baseado em Python que possibilita a criação e coordenação de agentes inteligentes com maior controle programático e flexibilidade arquitetural. A avaliação foi conduzida pela mesma pessoa responsável pelo desenvolvimento em ambas as plataformas, garantindo uniformidade na análise dos critérios considerados, que incluíram facilidade de desenvolvimento, flexibilidade, manutenibilidade e adequação ao uso com LLMs em cenários reais. Os resultados indicaram que o n8n se destaca pela rapidez de desenvolvimento e interface visual, sendo mais adequado para prototipagem e automações diretas, enquanto o Crew AI é mais apropriado para arquiteturas complexas — entendidas aqui como aquelas que envolvem múltiplos agentes, MCPs e demandam algoritmos mais especializados para o cumprimento das tarefas. Ressalta-se que a escolha do Zendesk e do S3 ocorreu por serem soluções amplamente utilizadas no mercado, mas o sistema proposto não é restrito a essas ferramentas. Conclui-se que a melhor plataforma depende diretamente das especificações da demanda, como complexidade do fluxo, tempo disponível para implementação e nível de controle técnico exigido.

Conteúdo

Conteúdo 1
Abstract 2
Resumo 2
1 Introdução 2
2 Fundamentação Teórica 3
2.1 Sistemas Multiagentes 3
2.2 Large Language Models 3
2.3 Crew AI: Um Framework para Orquestração de Agentes Inteligentes 3
2.4 n8n: Automação e Orquestração Low-Code/No-Code 3
3 Metodologia 4
3.1 Desenho Experimental 4
3.2 Ferramentas e Dados 4
3.3 Implementação das Soluções 4
3.4 Métricas de Avaliação e Comparação 4
3.5 Análise dos Resultados 4
4 Implementação e Arquitetura 4
4.1 Implementação e Arquitetura com Crew AI 4
4.2 Implementação e Arquitetura com n8n 5
5 Resultados e Discussão 6
5.1 Contexto dos Dados de Teste 6
5.2 Comparação Baseada nas Métricas de Avaliação 6
5.2.1 Facilidade de Desenvolvimento 6
5.2.2 Flexibilidade e Customização 6
5.2.3 Manutenibilidade 7
5.3 Discussão Geral e Cenários de Aplicação 7
5.4 Ameaças à Validade 7
6 Conclusões 7
Agradecimentos 8
Referências 8

Comparativo de Plataformas para Orquestração de Agentes Inteligentes: n8n vs. Crew AI na Sumarização de Tickets

Alfredo Vasconcelos de Andrade
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba - Brasil
alfredo.andrade@ccc.ufcg.edu.br

Rohit Gheyi
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Paraíba - Brasil
rohit@computacao.ufcg.edu.br

ABSTRACT

Inefficient support ticket management in technology companies can lead to significant delays in customer responses, increased operational costs, and overload of support teams, directly affecting customer satisfaction and retention. With the growing volume of data and the need for quick responses, automation and intelligent analysis of these tickets become essential to optimize resources and reduce bottlenecks. This work proposes the creation of a multi-agent system to summarize Zendesk tickets stored in AWS S3 and generate managerial reports, leveraging Large Language Models (LLMs), including but not limited to Claude 3.5 Sonnet. For the implementation, two distinct platforms were evaluated: n8n, an open-source workflow automation tool that adopts the low-code/no-code paradigm and enables the orchestration of AI agents through interconnected nodes; and Crew AI, a Python-based framework that allows the creation and coordination of intelligent agents with greater programmatic control and architectural flexibility. The evaluation was conducted by the same person responsible for the development in both platforms, ensuring consistency in the analysis of criteria such as ease of development, flexibility, maintainability, and suitability for using LLMs in real-world scenarios. The results showed that n8n stands out for its fast development and visual interface, being more suitable for prototyping and direct automations, while Crew AI is better suited for complex architectures — here understood as those involving multiple agents, MCPs, and requiring more specialized algorithms to perform tasks. It is worth noting that Zendesk and S3 were chosen because they are widely used solutions in the market, but the proposed system is not restricted to them. It is concluded that the choice of the most appropriate platform depends directly on the specifications of the demand, such as workflow complexity, available implementation time, and the required level of technical control.

Resumo

A gestão ineficiente de *tickets* de suporte em empresas de tecnologia pode gerar atrasos significativos nas respostas aos clientes, aumento de custos operacionais e sobrecarga das equipes de atendimento, impactando diretamente a satisfação e a retenção de clientes. Diante do crescimento do volume de dados e da necessidade de respostas rápidas, a automatização e a análise inteligente desses *tickets* tornam-se essenciais para otimizar recursos e reduzir gargalos. Este trabalho propõe a criação de um sistema multiagente para sumarizar *tickets* do Zendesk, armazenados no AWS S3, e gerar relatórios gerenciais, utilizando Large Language Models (LLMs), incluindo, mas não se limitando ao Claude 3.5 Sonnet. Para a implementação, foram avaliadas duas plataformas distintas: o n8n, ferramenta de automação de fluxo de trabalho de código aberto que adota o paradigma low-code/no-code e permite orquestrar agentes de inteligência artificial por meio de nós interligados; e o Crew AI, framework baseado em Python que possibilita a criação e coordenação de agentes inteligentes com maior controle programático e flexibilidade arquitetural. A avaliação foi conduzida pela mesma pessoa responsável pelo desenvolvimento em ambas as plataformas, garantindo uniformidade na análise dos critérios considerados, que incluíram facilidade de desenvolvimento, flexibilidade, manutenibilidade e adequação ao uso com LLMs em cenários reais. Os resultados indicaram que o n8n se destaca pela rapidez de desenvolvimento e interface visual, sendo mais adequado para prototipagem e automações diretas, enquanto o Crew AI é mais apropriado para arquiteturas complexas — entendidas aqui como aquelas que envolvem múltiplos agentes, MCPs e demandam algoritmos mais especializados para o cumprimento das tarefas. Ressalta-se que a escolha do Zendesk e do S3 ocorreu por serem soluções amplamente utilizadas no mercado, mas o sistema proposto não é restrito a essas ferramentas. Conclui-se que a melhor plataforma depende diretamente das especificações da demanda, como complexidade do fluxo, tempo disponível para implementação e nível de controle técnico exigido.

PALAVRAS-CHAVE

Inteligência Artificial, n8n, crew ai, sistemas multiagentes, LLMs, Tickets.

1 Introdução

A gestão de *tickets* em empresas de tecnologia enfrenta desafios crescentes, como alto volume e necessidade de respostas rápidas. Sistemas tradicionais, ainda fortemente dependentes de intervenção humana, geram gargalos e ineficiências que afetam tanto a experiência do cliente quanto a visão gerencial, resultando em atrasos no atendimento, aumento de custos operacionais e perda de oportunidades de melhoria de processos [3, 16]. Esses impactos comprometem a satisfação e a retenção de clientes, além de demandar mais recursos humanos para lidar com tarefas repetitivas.

Avanços em Inteligência Artificial, especialmente os *Large Language Models* (LLMs), oferecem novas possibilidades para a análise textual [4, 9], mas ainda são insuficientes para lidar com a complexidade da sumarização e geração de relatórios gerenciais, que

Os autores retêm os direitos, ao abrigo de uma licença Creative Commons Atribuição CC BY, sobre todo o conteúdo deste artigo (incluindo todos os elementos que possam conter, tais como figuras, desenhos, tabelas), bem como sobre todos os materiais produzidos pelos autores que estejam relacionados ao trabalho relatado e que estejam referenciados no artigo (tais como códigos fonte e bases de dados). Essa licença permite que outros distribuam, adaptem e evoluam seu trabalho, mesmo comercialmente, desde que os autores sejam creditados pela criação original.

exigem múltiplas etapas de processamento e coordenação de tarefas [5]. Por exemplo, em um cenário corporativo, é necessário coletar dados brutos de diferentes sistemas, padronizar formatos, extrair informações relevantes, sintetizar os achados em linguagem natural e, por fim, estruturar um relatório coerente para apoio à decisão. Cada uma dessas etapas envolve desafios técnicos distintos, como a heterogeneidade das fontes de dados, risco de perda de contexto durante a sumarização e a necessidade de manter precisão e clareza na geração final do documento. Neste trabalho, a tarefa de sumarização é explorada como um **cenário experimental**, servindo de base para a comparação entre diferentes ferramentas de orquestração e automação.

Para enfrentar esses desafios, este trabalho propõe um sistema multiagente (SMA), constituído por um conjunto de agentes autônomos e cooperativos capazes de executar tarefas específicas e interagir para alcançar objetivos comuns [17]. Nesse contexto, agentes especializados realizam coleta, análise, sumarização e criação de relatórios com informações como frequência de tópicos e recorrência de problemas [15].

O foco do estudo é comparar as plataformas Crew AI e n8n para o desenvolvimento e orquestração de sistemas de sumarização de *tickets*. A Crew AI [8] — baseada em Python — é adequada para cenários complexos com LLMs e tomada de decisão distribuída [14], enquanto o n8n [11] é uma solução low-code/no-code que facilita a integração e acelera o desenvolvimento. A avaliação foi conduzida por meio de experimentação prática, desenvolvendo o mesmo sistema em ambas as plataformas e considerando critérios como facilidade de desenvolvimento, flexibilidade, manutenibilidade e adequação ao uso com LLMs em cenários reais.

Os resultados indicam que o n8n se destaca para fluxos mais diretos e protótipos rápidos, enquanto o Crew AI é mais indicado para arquiteturas complexas e personalizadas. Conclui-se que a escolha ideal entre as plataformas depende das especificações da demanda, como complexidade do fluxo, tempo disponível para implementação e necessidade de controle técnico.

O restante deste trabalho está organizado da seguinte maneira: uma fundamentação teórica sobre sistemas multiagentes e LLMs na Seção 2, seguida de metodologia e descrição da implementação nas Seções 3 e 4, análise comparativa dos resultados na Seção 5 e, por fim, conclusões com limitações e perspectivas futuras na Seção 6.

2 Fundamentação Teórica

A presente seção estabelece o arcabouço teórico que sustenta a pesquisa, detalhando os conceitos essenciais para a compreensão do desenvolvimento e da comparação de sistemas multiagentes na sumarização de *tickets* do Zendesk. Serão abordados desde os fundamentos dos Sistemas Multiagentes e o papel dos LLMs, até as características arquitetônicas e funcionais das plataformas Crew AI e n8n, ferramentas centrais desta análise comparativa.

2.1 Sistemas Multiagentes

Sistemas Multiagentes (SMAs) são uma área da inteligência artificial que estuda a interação e coordenação de múltiplos agentes autônomos para resolver problemas complexos [17]. Um agente é uma entidade autônoma que percebe seu ambiente e age para alcançar objetivos, caracterizado por autonomia, reatividade, proatividade e sociabilidade [13].

SMAs são vantajosos em cenários que exigem distribuição de tarefas, flexibilidade e robustez. A arquitetura multiagente permite a modularização de responsabilidades, onde cada agente se especializa em uma função, como coleta, transformação, análise de dados ou geração de relatórios [15]. Essa abordagem é altamente relevante para automatizar o fluxo de trabalho de atendimento ao cliente, delegando etapas a agentes especializados para maior eficiência.

2.2 Large Language Models

LLMs são modelos de inteligência artificial treinados em vastos volumes de dados textuais, capazes de compreender, gerar e interagir com a linguagem natural. Baseados em arquiteturas de transformadores, realizam tarefas de Processamento de Linguagem Natural (PLN) como tradução, resumo, classificação e resposta a perguntas [9, 4]. Modelos como GPT e BERT demonstram o potencial dos LLMs em capturar nuances semânticas e contextuais [12, 10].

A relevância dos LLMs reside em diversas capacidades, sendo a principal delas a de gerar texto coeso e gramaticalmente correto, mesmo que por vezes apresentem alucinações. Entre essas capacidades, destaca-se a habilidade de resumir grandes volumes de texto, fundamental para a gestão de *tickets* de suporte. Um LLM pode processar o conteúdo completo de um ticket e extrair seus pontos mais importantes, automatizando a criação de resumos concisos para relatórios gerenciais e oferecendo uma visão rápida e acionável. Neste trabalho, essa capacidade de sumarização é explorada como um **cenário experimental**, permitindo avaliar e comparar diferentes ferramentas de orquestração e automação.

2.3 Crew AI: Um Framework para Orquestração de Agentes Inteligentes

Crew AI é um framework Python de código aberto para criar e gerenciar sistemas multiagentes baseados em LLMs. Ele permite que desenvolvedores construam equipes de "agentes" autônomos que colaboram para atingir objetivos complexos, dividindo tarefas e compartilhando informações [8]. O framework define componentes chave: agentes, que são entidades autônomas com papel, histórico (personalidade/contexto) e objetivo, podendo ter ferramentas para interagir com o ambiente externo; tarefas, que são ações específicas para os agentes realizarem; e o "Crew", que é o grupo de agentes e tarefas, definindo o processo de interação para alcançar o objetivo final.

A Crew AI oferece flexibilidade e controle programático, sendo ideal para lógicas de negócio complexas e personalização profunda, como análise de dados, tomada de decisão distribuída, sumarização e geração de relatórios a partir de dados de atendimento.

2.4 n8n: Automação e Orquestração Low-Code/No-Code

n8n é uma ferramenta de automação de fluxo de trabalho de código aberto, operando no paradigma low-code/no-code. Permite criar automações complexas com pouca ou nenhuma escrita de código, usando uma interface visual de "arrastar e soltar" para conectar "nós" que representam aplicações, serviços ou funções [11]. Seu objetivo principal é integrar sistemas diversos e automatizar processos, tornando a orquestração de tarefas acessível.

Sua arquitetura baseia-se em uma vasta biblioteca de integrações pré-construídas e inclui nós dedicados a funcionalidades de inteligência artificial, como interação com LLMs. Embora o n8n

não defina "agentes" exatamente como a Crew AI, ele fornece os recursos necessários para orquestrar agentes de IA por meio da composição e encadeamento de nós. É possível criar fluxos onde cada nó representa um agente com função específica (por exemplo, um "agente leitor" para extração de dados, um "agente sumarizador" com LLM e um "agente gerador de relatório"), permitindo que trabalhem de forma coordenada. Essa modularidade, aliada à capacidade de integrar APIs e serviços de IA, possibilita ao n8n construir sistemas que operam de forma semelhante a um ambiente multiagente, tornando-o um concorrente relevante na comparação de abordagens.

3 Metodologia

Este trabalho apresenta a comparação entre n8n e Crew AI no desenvolvimento de um sistema multiagente voltado à sumarização de *tickets* do Zendesk, com base em dados armazenados em um bucket S3, e geração de relatórios. A Seção 3.1 apresenta o desenho experimental, a Seção 3.2 detalha as ferramentas e dados utilizados, a Seção 3.3 contém a implementação do caso de uso em ambas as plataformas e, ao final, a Seção 3.4 descreve as métricas de avaliação aplicadas.

3.1 Desenho Experimental

O experimento foi estruturado para comparar diretamente as capacidades de n8n e Crew AI, seguindo os mesmos requisitos funcionais. As etapas incluem:

- **Coleta de dados:** uso de dados sintéticos representativos de chats do Zendesk.
- **Leitura e pré-processamento:** ambas as plataformas acessam os dados diretamente do S3.
- **Sumarização:** aplicação do mesmo LLM (Claude 3.5 Sonnet) com prompts padronizados para garantir comparabilidade.
- **Geração de relatórios:** agregação dos resumos e de informações extraídas (ex: tópicos, sentimentos, recorrência) para compor um relatório gerencial.
- **Saída do relatório:** armazenamento final em formato acessível (JSON, CSV ou PDF) ou via API.

3.2 Ferramentas e Dados

Para as plataformas de comparação, utilizamos a Crew AI (v0.114.0, implementada em Python) e o n8n (v1.84.3, self-hosted). O AWS S3 foi o serviço de armazenamento de dados dos *tickets*. Para a sumarização, empregamos o Claude 3.5 Sonnet, acessado via Amazon Bedrock [1], configurado de forma consistente em ambas as plataformas.

Durante o desenvolvimento e os testes iniciais, utilizamos dados sintéticos para os *tickets* do Zendesk. No entanto, para uma validação mais robusta, realizamos uma comparação adicional com dados reais de transcrições de chat de uma empresa de jogos mobile. Embora não possamos disponibilizar esses dados reais devido a questões de confidencialidade e privacidade, os resultados obtidos foram promissores e reforçam a eficácia das soluções propostas.

3.3 Implementação das Soluções

A implementação para cada plataforma (n8n e Crew AI) foi realizada pelo autor seguindo os passos definidos no Desenho Experimental:

- No Crew AI, definiu-se agentes especializados (ex: *BucketReaderAgent* para S3, *ChatSummarizerAgent* para LLM, *ReportGeneratorAgent*), criação de ferramentas (tools) para cada agente (e.g., leitura de S3, interação com API do Amazon Bedrock) e a orquestração do *crew* para a colaboração eficiente no fluxo de sumarização e geração de relatórios.
- No n8n, construiu-se um workflow visual com nós (nodes) apropriados para conexão e leitura de dados do AWS S3, interação com a API do Amazon Bedrock (via Claude 3.5 Sonnet) para sumarização, transformação e agregação de dados para o relatório, e saída final do relatório.

3.4 Métricas de Avaliação e Comparação

A comparação entre n8n e Crew AI foi baseada nas seguintes métricas qualitativas. Na facilidade de desenvolvimento, esta avaliação considerou a curva de aprendizado, o tempo total de implementação e a complexidade do código/workflow. Flexibilidade e customização, esta métrica foi avaliada pela capacidade de adaptação da lógica de sumarização, integração de novos LLMs, pela extensibilidade para integrar novos componentes e pela possibilidade de desenvolver conectores ou ferramentas customizadas. Por fim, manutenibilidade foi utilizada para a avaliação da facilidade de depuração, atualização e suporte das soluções implementadas ao longo do tempo.

3.5 Análise dos Resultados

Os dados coletados a partir da implementação e testes de ambas as plataformas foram submetidos a uma análise comparativa e crítica. A metodologia de análise se baseou na avaliação das métricas qualitativas e quantitativas definidas na Seção 3.4. Para cada métrica, como facilidade de desenvolvimento, flexibilidade e manutenibilidade, os resultados observados foram sistematizados e discutidos em profundidade, destacando os pontos fortes e fracos de cada plataforma. A discussão subsequente aborda, de forma conclusiva, em quais cenários e para quais tipos de projetos cada ferramenta se mostra mais adequada, considerando aspectos técnicos e de negócios.

4 Implementação e Arquitetura

Esta seção detalha a implementação do sistema de sumarização e geração de relatórios de *tickets*, conforme definido na metodologia, e discute as escolhas arquiteturais específicas para cada plataforma: Crew AI e n8n. O objetivo é apresentar como as funcionalidades foram configuradas para processar os dados dos chats de atendimento do Zendesk a partir do AWS S3, sumarizá-los com o Claude 3.5 Sonnet via Amazon Bedrock — modelo escolhido por ser o mais atualizado e avançado disponível até a execução deste trabalho — e, por fim, gerar os relatórios. As descrições visam fornecer clareza sobre os passos técnicos, as configurações e as características arquiteturais que fundamentam cada solução, servindo como base para a posterior análise comparativa.

4.1 Implementação e Arquitetura com Crew AI

A implementação da solução de sumarização e geração de relatórios foi realizada com o Crew AI (v0.114.0), que adota uma arquitetura de sistemas multiagentes intrinsecamente modular e orientada a tarefas. A estrutura central do Crew AI se baseia em Agentes, Ferramentas (Tools) para interação com o ambiente externo, Tarefas a serem executadas e um Crew para orquestrar o processo colaborativo. Na concepção arquitetural, cada agente é definido com um

role (papel), um *backstory* (histórico/contexto) e um *goal* (objetivo), permitindo sua especialização e autonomia. As ferramentas são abstrações para funcionalidades externas, permitindo que os agentes executem ações sem se preocupar com os detalhes de implementação. As tarefas definem o que precisa ser feito, e o Crew gerencia o fluxo de trabalho e a comunicação entre os agentes. A arquitetura pode ser observada na **Figura 1**.

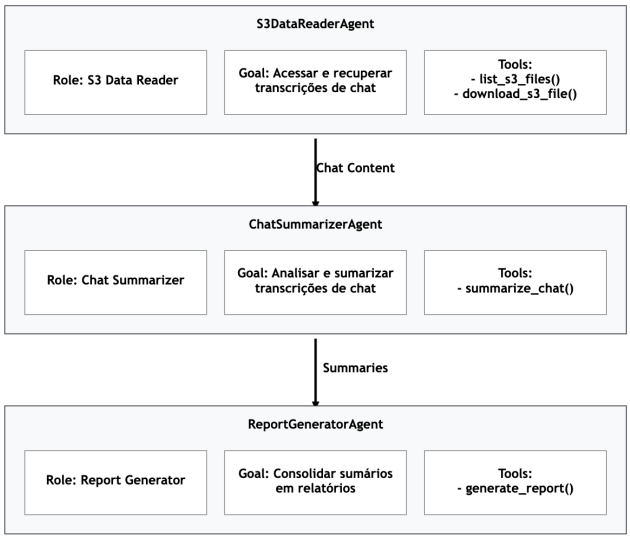


Figura 1: Arquitetura de orquestração Crew AI, demonstrando a interação sequencial entre os agentes *S3DataReaderAgent, ChatSummarizerAgent* e *ReportGeneratorAgent* para processamento e geração do relatório em JSON.

Para este trabalho, foram implementados três tipos principais de agentes:

Agente Leitor de Dados (S3DataReaderAgent) . Projetado como o ponto de entrada de dados, este agente possui uma ferramenta que encapsula a funcionalidade de listar e baixar arquivos de transcrições de chat do Zendesk diretamente de um bucket AWS S3. Arquiteturalmente, a ferramenta usa o `boto3` (SDK da AWS para Python) para chamadas seguras à API do S3, com autenticação via variáveis de ambiente para acesso com privilégio mínimo. O Código 1 apresenta a configuração do agente.

Código 1: Configuração do Agente Leitor de Dados

```
s3_data_reader = Agent(
    role='S3 Data Reader',
    goal='Acessar e recuperar transcricoes de chat do AWS S3 de
          forma eficiente e segura',
    backstory="""Voce e um agente especializado em leitura de
          dados do AWS S3.
    Sua funcao e garantir o acesso seguro e eficiente aos arquivos
           de transcricao
    de chat armazenados no bucket S3.""",
    tools=[s3_tools.list_s3_files, s3_tools.download_s3_file],
    verbose=True
)
```

Agente Sumarizador de Chats (ChatSummarizerAgent) . Este agente é o componente de inteligência central, recebendo as transcrições completas dos chats do `S3DataReaderAgent`. Sua ferramenta é configurada para interagir com a API do Amazon Bedrock, invocando especificamente o modelo Claude 3.5 Sonnet. O Código 2 apresenta a configuração do agente.

Código 2: Configuração do Agente Sumarizador de Chats

```
chat_summarizer = Agent(
    role='Chat Summarizer',
    goal='Analisar e sumarizar transcricoes de chat de forma
          precisa e relevante',
    backstory="""Voce e um agente especializado em analise e
          sumarizacao de conversas
    de atendimento. Sua funcao e extrair os pontos mais
          importantes e gerar resumos
    concisos e uteis usando o Claude 3.5 Sonnet.""",
    tools=[bedrock_tools.summarize_chat],
    verbose=True
)
```

Agente Gerador de Relatórios (ReportGeneratorAgent) . Este agente finaliza o pipeline, compilando os resumos e outras informações relevantes dos *tickets* em um único objeto JSON. O Código 3 apresenta a configuração do agente.

Código 3: Configuração do Agente Gerador de Relatórios

```
report_generator = Agent(
    role='Report Generator',
    goal='Consolidar sumarios de chat em relatorios gerenciais
          estruturados',
    backstory="""Voce e um agente especializado em geracao de
          relatorios.
    Sua funcao e compilar os resumos dos chats em um formato
          estruturado
    e util para analise gerencial.""",
    tools=[report_tools.generate_report],
    verbose=True
)
```

A orquestração (`crew`) entre esses agentes foi configurada com um processo sequencial, garantindo que as tarefas sejam executadas em ordem e que a saída de um agente seja a entrada do próximo.

4.2 Implementação e Arquitetura com n8n

A implementação com n8n (v1.84.3) foi concebida como um *workflow* visual, utilizando sua natureza *low-code/no-code* para automatizar e orquestrar o processo de sumarização e geração de relatórios. A arquitetura(**Figura 2**). do n8n é baseada em um modelo de fluxo de eventos e dados, onde componentes modulares chamados “nós” são conectados para formar pipelines.

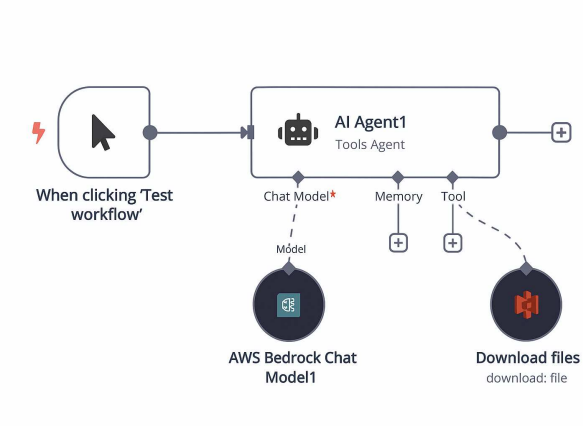


Figura 2: Visão geral do workflow no n8n: nó *AI Agent* centralizando a lógica de orquestração, conectado ao *AWS Bedrock Chat Model* e à *Tool* de *Download files* (S3).

O *workflow* consiste nos seguintes componentes principais:

- **Nó de Início:** Aciona o *workflow* manualmente para testes.
- **Nó AI Agent:** Encapsula toda a lógica de leitura, sumarização, agrupamento e geração de relatórios.

O *prompt* principal utilizado pelo AI Agent é o seguinte:

```
Fetch the Parquet file from S3, using the tool that already knows
      the file. This file contains multiple Zendesk tickets; each
      row represents a single ticket and includes at least a
      timestamp and a description or subject of the issue.
Group the tickets by day using the timestamp field. For each group
      , analyze and cluster the ticket descriptions to detect
      common patterns, topics, or recurring requests.
For each day, generate a detailed natural-language summary that
      includes:
The date
Identified categories of issues, grouped by theme (e.g., Account
      Access, Billing, Technical Issues)
For each category:
A brief summary of the type of issue
Examples of ticket content (can be paraphrased)
An estimate of frequency or how prominent this issue was that day
Any observed trends (e.g., increase from previous day, sudden
      appearance, etc.)
Do not just count identical descriptions. Instead, cluster
      semantically similar tickets (e.g., password reset = "forgot
      password", "can't log in", etc.).
Use structured bullet points and natural human language. Be
      concise, but informative and specific.
At the end, provide a summary of evolving trends across the
      analyzed days, including any noticeable spikes, declines, or
      new themes that emerged.
```

O prompt principal utilizado pelo AI Agent segue o padrão de prompting por passo-a-passo, guiando o modelo de forma lógica e sequencial: primeiro, instruindo-o a buscar e agrupar os dados, depois a analisar e agrupar semanticamente as descrições e, por fim, a gerar os resumos e o relatório final. Além disso, o prompt inclui instruções claras e restrições explícitas, como a diretriz de "não apenas contar descrições idênticas", mas sim agrupar tickets semanticamente similares, o que força uma análise mais profunda e menos superficial. Ele também estabelece um formato de saída desejado, solicitando o uso de "marcadores estruturados e linguagem humana natural", e oferece exemplos e formato explícitos para garantir que o resultado seja claro e preciso. Essa combinação de técnicas demonstra como o prompt atua como um orquestrador inteligente, usando suas ferramentas para entregar uma análise complexa e de alta qualidade.

Ao final, o relatório consolidado em formato JSON é retornado como *output* direto do nó AI Agent.

5 Resultados e Discussão

Esta seção apresenta e discute os resultados da implementação do sistema de sumarização e geração de relatórios de *tickets*, utilizando Crew AI e n8n. A análise foca nas métricas de facilidade de desenvolvimento, flexibilidade e manutenibilidade.

5.1 Contexto dos Dados de Teste

Para os experimentos, utilizamos um conjunto de dados sintéticos de transcrições de chat do Zendesk, que simularam um cenário de atendimento real, variando em complexidade textual e volume de conversas. Cada ticket continha um ID, timestamp e a transcrição completa do chat. Isso permitiu testar as soluções de forma controlada.

5.2 Comparação Baseada nas Métricas de Avaliação

Para determinar a melhor solução para o sistema, avaliamos as duas ferramentas - Crew AI e n8n - em três métricas-chave: Facilidade de Desenvolvimento, Flexibilidade e Customização, e Manutenibilidade. A análise a seguir detalha o desempenho de cada uma nessas áreas críticas.

5.2.1 Facilidade de Desenvolvimento Com Crew AI, o desenvolvimento exigiu conhecimento sólido em Python e na sua estrutura orientada a objetos para a definição de agentes, ferramentas e tarefas. A curva de aprendizado inicial foi moderada, especialmente na concepção da lógica multiagente e na integração das ferramentas para S3 e Amazon Bedrock. O tempo de desenvolvimento foi de aproximadamente 7 dias, e a complexidade do código, embora modular, demanda maior familiaridade com paradigmas de programação.

Já no n8n, a implementação se beneficiou da interface low-code/no-code, resultando em uma curva de aprendizado consideravelmente mais rápida para montar o workflow visual e configurar o nó AI Agent. O tempo de desenvolvimento foi de aproximadamente 2 dias, significativamente menor. A complexidade do workflow é visualmente reduzida, pois o nó AI Agent encapsula grande parte da lógica, facilitando a compreensão geral do fluxo, mesmo que a depuração de ferramentas personalizadas possa aumentar a complexidade.

Em síntese, o n8n demonstrou clara vantagem em rapidez e facilidade de desenvolvimento para o escopo do projeto, sendo ideal para prototipagem ágil. O Crew AI, por outro lado, oferece controle mais granular e flexibilidade ilimitada para desenvolvedores Python, sendo mais adequado para sistemas que demandam arquiteturas multiagentes complexas e customização profunda. A escolha entre as duas ferramentas impacta diretamente o tempo de colocação em produção e a expertise técnica necessária da equipe.

5.2.2 Flexibilidade e Customização O Crew AI demonstrou alta capacidade de adaptação e extensibilidade. Sua natureza programática permite controle total sobre a lógica de cada agente, a interação entre eles e a integração com qualquer biblioteca Python ou API. Alterar prompts, adicionar ferramentas personalizadas ou mudar o

formato do relatório é uma questão de modificar o código Python, oferecendo liberdade completa para requisitos específicos.

O n8n ofereceu boa capacidade de adaptação por meio de seus nós configuráveis e, principalmente, pela capacidade de injetar prompts complexos no nó AI Agent. A adição de novas ferramentas externas (como para outros serviços de S3 ou outros LLMs/provedores de LLM) é relativamente simples através de nós predefinidos ou conectores. A extensibilidade para comportamentos muito específicos pode ser atingida com o desenvolvimento de ferramentas customizadas ou nós de código JavaScript, que exigem maior conhecimento técnico.

Em resumo, o Crew AI oferece customização ilimitada, ideal para cenários altamente específicos ou em constante evolução que demandam intervenção direta no código. O n8n, por sua vez, equilibra flexibilidade com facilidade de uso, sendo robusto para a maioria dos casos de uso por meio de sua interface visual, mas podendo exigir contorno ou desenvolvimento customizado para requisitos muito particulares.

5.2.3 Manutenibilidade A manutenibilidade de uma solução Crew AI depende da qualidade do código Python. Ferramentas de depuração de código Python são padrão e robustas, e a modularidade dos agentes e ferramentas auxilia na depuração de partes específicas do sistema. Atualizações do framework podem exigir pequenas adaptações de código.

No n8n, a manutenibilidade dos workflows é facilitada pela interface visual, que permite rápida compreensão do fluxo de dados e identificação de problemas. A depuração visual dos nós e a capacidade de reexecutar passos específicos são grandes vantagens. As atualizações da plataforma são geralmente gerenciadas pelo próprio n8n. No entanto, para lógicas complexas encapsuladas no nó AI Agent (principalmente no prompt e na interação com ferramentas), a depuração pode depender mais da observação dos logs do LLM e do output do nó, exigindo uma compreensão aprofundada do comportamento do prompt.

O n8n tem uma vantagem visual na manutenibilidade para equipes que priorizam a visualização do fluxo e a depuração interativa. O Crew AI oferece as ferramentas robustas de depuração de um ambiente de desenvolvimento tradicional e o controle direto sobre o código. A escolha dependerá do perfil da equipe de manutenção e da complexidade da lógica implementada.

5.3 Discussão Geral e Cenários de Aplicação

A análise comparativa entre Crew AI e n8n revela que ambas as plataformas são capazes de construir um sistema eficiente para a sumarização de *tickets* do Zendesk e geração de relatórios, mas se destacam em diferentes contextos.

Para prototipagem rápida e automações visuais com IA consolidada, o n8n se mostrou superior em termos de facilidade e rapidez de desenvolvimento, especialmente com o poder do nó AI Agent para encapsular lógicas complexas de IA. Sua interface low-code/no-code permite que automações complexas com IA sejam implementadas por equipes com menos proficiência em programação, ideal para validação de conceitos ou automação de processos internos onde a visualização do fluxo é prioritária e a lógica de agente pode ser concentrada.

Para arquiteturas distribuídas de agentes e controle programático detalhado, o Crew AI oferece uma flexibilidade e um controle programático inigualáveis. Para projetos que exigem a definição de múltiplos agentes com interações complexas e autônomas, customização profunda do comportamento de cada agente, ou a integração com sistemas de software existentes em um ambiente de desenvolvimento Python, o Crew AI é a escolha mais robusta. Ele é mais adequado para engenheiros de IA e desenvolvedores que precisam de total controle sobre cada camada da arquitetura e um modelo distribuído de colaboração de agentes.

A escolha entre Crew AI e n8n dependerá, portanto, da proficiência técnica da equipe, da complexidade e customização exigidas pelo sistema, e da prioridade entre agilidade de desenvolvimento e controle total sobre a arquitetura. Os testes com dados sintéticos permitiram validar a funcionalidade e as diferenças arquitetônicas, e a comparação com dados reais, embora não publicável devido à confidencialidade, reforçou a aplicabilidade e os resultados promissores de ambas as abordagens no contexto prático.

5.4 Ameaças à Validade

Este estudo apresenta algumas ameaças à validade que devem ser consideradas. Primeiramente, a utilização de dados sintéticos pode não refletir completamente a complexidade e variabilidade de dados reais de *tickets* do Zendesk, podendo impactar a generalização dos resultados. Em segundo lugar, a experiência prévia da equipe com Python e ferramentas low-code/no-code pode ter influenciado a percepção de facilidade de desenvolvimento, introduzindo viés de conhecimento. Por fim, a avaliação das métricas, como flexibilidade e manutenibilidade, envolveu análises qualitativas e subjetivas, o que pode afetar a consistência das conclusões. Ademais, o fato de essas métricas subjetivas terem sido avaliadas por apenas uma pessoa deve ser considerado uma ameaça adicional à validade. Essas limitações devem ser ponderadas na interpretação dos resultados e em estudos futuros.

6 Conclusões

Este trabalho comparou criticamente o Crew AI e o n8n no desenvolvimento de um sistema multiagente para sumarizar transcrições de chat do Zendesk e gerar relatórios, demonstrando a viabilidade de ambas as plataformas para automação inteligente, cada uma com características distintas para diferentes cenários.

A análise prática revelou que o n8n se destacou pela facilidade e rapidez de desenvolvimento, com sua abordagem low-code/no-code e interface visual intuitiva. Seu nó AI Agent mostrou-se eficaz em consolidar múltiplas responsabilidades. O Crew AI, por outro lado, exigiu maior proficiência em Python, mas ofereceu controle total e flexibilidade ilimitada para customizações complexas e integração profunda.

Em termos de manutenibilidade, o n8n se beneficiou de sua representação visual, facilitando a depuração para diversas equipes. O Crew AI, com sua base em Python, ofereceu ferramentas de depuração robustas e modularidade para isolar problemas, sendo preferível para equipes com expertise em desenvolvimento de software.

Em suma, a escolha entre Crew AI e n8n para orquestração de agentes inteligentes deve equilibrar a velocidade de prototipagem e desenvolvimento (n8n) com a necessidade de controle granular e flexibilidade programática (Crew AI). O n8n é ideal para automações rápidas e gestão visual, enquanto o Crew AI é mais robusto para sistemas de IA complexos com lógicas personalizadas.

Este estudo contribui com uma análise prática e detalhada de duas abordagens distintas (low-code vs. framework programático) para sistemas multiagentes com IA generativa, auxiliando desenvolvedores e gestores na escolha tecnológica.

Como trabalhos futuros, sugere-se expandir a pesquisa com: (i) implementação em ambientes de produção para análise de escalabilidade e custo; (ii) inclusão de métricas como usabilidade para não-programadores e segurança; e (iii) exploração de outros frameworks ou plataformas low-code/no-code com capacidades de IA.

Agradecimentos

Primeiramente, minha profunda gratidão a Deus, por me guiar e abençoar em cada passo desta jornada. Sinto-me imensamente grato à minha família – pais, irmão, avós, tias e tios – por todo o apoio incondicional e o constante incentivo aos meus estudos. Agradeço também a todos os professores que contribuíram para minha formação acadêmica, e em especial ao Professor Rohit Gheyi, meu orientador, cujo apoio e orientação foram fundamentais para meu crescimento profissional. Por fim, minha sincera gratidão aos meus amigos, que tornaram essa jornada acadêmica mais leve e inspiradora, oferecendo ajuda e suporte em todos os momentos.

Referências

[1] [n. d.] Amazon Bedrock. https://aws.amazon.com/bedrock/. Acesso em: 12 ago. 2025. ().

[2] [n. d.] Amazon Simple Storage Service (S3). https://aws.amazon.com/s3/. Acessado em 2025-08-12. ().

[3] Mary Jo Bitner, Bernard H. Booms e Mary S. Tetreault. 1990. The Service Encounter: Diagnosing Favorable and Unfavorable Incidents. *Journal of Marketing*, 54, 1, 71–84.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell e et al. 2020. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901.

[5] Erik Cambria. 2016. Affective Computing and Sentiment Analysis. *IEEE Intelligent Systems*, 31, 2, 102–107.

[6] [n. d.] Claude 3.5 Sonnet. https://www.anthropic.com/index/claude. Acessado em 2025-08-12. ().

[7] [n. d.] Claude 3.5 Sonnet. https://www.anthropic.com/. Acesso em: 12 ago. 2025. ().

[8] [n. d.] Crew AI Platform. https://crew.ai. Accessed: 2025-08-12. ().

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee e Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee e Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. Em *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: Human language technologies, Volume 1 (Long and short papers)*, 4171–4186.

[11] [n. d.] n8n Workflow Automation. https://n8n.io. Accessed: 2025-08-12. ().

[12] Alec Radford, Karthik Narasimhan, Tim Salimans e Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *Technical report, OpenAI*.

[13] Stuart Russell e Peter Norvig. 2020. *Artificial Intelligence: A Modern Approach*. (4th edição). Pearson, Harlow, UK.

[14] Yoav Shoham e Kevin Leyton-Brown. 2008. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, Cambridge, UK.

[15] Peter Stone e Manuela Veloso. 2000. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots*, 8, 3, 345–383.

[16] Wil van der Aalst. 2016. *Process Mining: Data Science in Action*. Springer, Heidelberg, Germany.

[17] Michael Wooldridge. 2021. *An Introduction to MultiAgent Systems*. (3rd edição). Wiley, Chichester, UK.